University of Alberta

Mobile Collaborative Peer Learning in an Apprenticeship Context

by

Atmaram Mulliah 

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Edmonton, Alberta

Fall 2006

Your file Votre référence
ISBN: 978-0-494-22328-4
Our file Notre référence
ISBN: 978-0-494-22328-4

Abstract

Wireless devices like cell phones and PDAs are very popular and highly affordable these days because they provide users with instant access to various resources and convenient services with no restriction to a particular location and time. Instant messaging, calendar services, and Web browsing are among those features offered by mobile devices. There is a growing need for the use of hand-helds in the teaching and learning domains since the growing number of students in classrooms makes traditional teaching and interaction methods less fruitful. Challenges arise when developing m-learning applications, due to device constraints such as small screen size, limited memory capacity, low processing power, and intermittent wireless connections.

The goal of this research is to design a collaborative application using Java™ 2 Micro Edition platform (J2ME™) that medical students can use during their surgery rotation to maintain their portfolios and collaborate on their daily unique cases that they encounter.

Acknowledgement

I wish to extend my heartfelt thanks to my supervisor: Dr. Eleni Stroulia. Her invaluable help, ideas, and support throughout this research have really geared me in the right direction. All her help is very much appreciated and will always be remembered.

I would also like to extend many thanks to all my friends who supported me throughout my graduate studies.

Finally, I would like to thank my parents for all their prayers, love, and encouragement, while always reminding me that patience and commitment will help me succeed in my endeavors.

Atmaram Mulliah

June 2006

Edmonton, Alberta, Canada.

To my family, always very supportive

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ABC.....*Activity-Based Computing*

APIs.....*Application Programming Interfaces*

CDC.....*Connected Device Configuration*

CLDC.....*Connected Limited Device Configuration*

DNS.....*Domain Name Service*

GUI.....*Graphical User Interface*

HTTP.....*HyperText Transport Protocol*

IDE.....*Integrated Development Environment*

IP.....*Internet Protocol*

J2ME.....*Java 2 Micro Edition*

J2SE.....*Java 2 Standard Edition*

J2EE.....*Java 2 Enterprise Edition*

JAD.....*Java Application Descriptor*

JAR.....*Java Archive*

JNI.....*Java Native Interface*

JVM.....*Java Virtual Machine*

JXME.....*JXTA for J2ME*

JXTA.....*JuXTApose*

KVM.....*K Virtual Machine*

MIDP.....*Mobile Information Device Profile*

NAT.....*Network Address Translation*

P2P.....*Peer-to-Peer*

PDA.....*Personal Digital Assistant*

PDAP.....*Personal Digital Assistant Profile*

PDF.....*Portable Document Format*

PNG.....*Portable Network Graphics*

PKI.....*Public Key Infrastructure*

RMS.....*Record Management System*

RTT.....*Round-Trip Time*

SDCard.....*Secure Digital Card*

SMS.....*Short Message Service*

TCP.....*Transport Control Protocol*

UML.....*Unified Modeling Language*

UUID.....*Universally Unique Identifier*

WSDD.....*WebSphere Studio Device Developer*

WSDL.....*Web Service Description Language*

XML.....*eXtensible Markup Language*

Chapter 1 Introduction and motivation

Wireless mobile devices are becoming more affordable than ever and their everyday use has understandably become indispensable. As a consequence, various desktop-based applications are being migrated to the mobile platform. Compared to their desktop counterparts, these applications have to accommodate several constraints, which include different sets of interaction techniques, limited memory and processing power, and small screen size. Furthermore, wireless mobile learning applications also have to deal with the potential of intermittent wireless connectivity and low bandwidth. This chapter explains the motivation behind this research and summarizes what has been done within its realm.

1.1 Motivation

Many professional disciplines, such as law, engineering, medicine and computing science, require cohorts of students to go through an apprenticeship stage, during which, students attend "formal instruction" such as lectures and seminars while, at the same time, practicing incrementally difficult aspects of the discipline. From a pedagogical perspective, cohorts of peers encourage long-term collaborative peer learning, and various apprenticeship programs incorporate in their schedule different types of activities for cohorts to enable the peers to work together on solving problems.

Traditionally, it has been the apprentices' responsibility to maintain an integrated representation of their experience in a portfolio, which frequently also serves as the basis for the students' performance assessment. Mobile PDAs present a unique opportunity for enriching the apprenticeship experience by enabling students to access and update their portfolios in the context of their activities, thus enriching the portfolio entries with relevant activity details and informing their practice with their portfolio study notes.

Recently, there has been substantial effort towards mobile e-learning applications. A recent European project illustrates several potential usages of mobile devices for continued life-long learning and skills development. The high-level objective of the m-learning project (http://www.m-learning.org/projects.shtml) was to use mobile technologies, i.e., mobile phones and handheld computers, to enhance learning. The 3-year project started in 2001 to help young learners, who had not succeeded in the

education system. Recognizing that one of the few commonalities among young at-risk adults – with no education or training, no employment, sometimes not even a home – could be an inexpensive, portable mobile phone, the project developed materials to motivate learning and to help learners develop their skills. Several activities were undertaken in the context of the m-learning project. For example, SMS messages, short quizzes and downloadable cards were developed to communicate skills-related information to employed learners in particular sectors. Special m-blogs were used to motivate non-native language speakers to practice their English by writing captions for photos shared with their community. Finally, a combination of camera phones, picture messages and web sites was used to link learners from different countries engaging in a long-term collaborative project.

Mobile devices present a very interesting opportunity to support collaboration in a continuous and lightweight manner. At the simplest level, learners may have immediate access to tutors, through SMS. As the capabilities of devices increase, more sophisticated support, aimed at supporting particular pedagogical styles of teaching and learning, becomes possible.

1.2 Research problem and methodology

The overall set of requirements for the MediCol system developed in this thesis is embodied in the following "visionary scenario". A cohort of medical students and their mentor are using MediCol on their handheld PDAs during their surgery rotation. Each medical student maintains a portfolio recording all the activities that s/he indulges in. Additionally, students maintain materials such as e-books for their self-directed studies. The e-books and the various documented activities provide the basis of the students' interactions and collaboration. For example, students may discuss their activities with each other or they may annotate their portfolio entries with book passages. The collaboration process will essentially allow the medical students to ask questions on a range of medical cases that they encounter and receive answers, whereby a suitable method has to be designed for students to cross-reference all the related questions, follow-up questions and answers in a beneficial way. Furthermore, any question and answer can be annotated with the various documented activities from the portfolio and

passage excerpts from the e-books. As questions are posed to the group, the mentor can assign questions to peer students if the questions remain unaddressed for a long time. Of more importance, the medical students are expected to submit their portfolio records to their mentor for the evaluation of their daily contributions and performances. The objective of this thesis has been to develop MediCol, a mobile application for supporting the activities described above, and thus facilitating collaborative peer learning among a cohort of medical students in their surgery rotation.

The application is developed as a hybrid Peer-to-Peer (P2P) system, consisting of a static relay, responsible for core services including communications, and a set of mobile peers running on PDAs, supporting specific collaborative functionalities. It is based on the JXTA framework, which provides a set of protocols for handling discovery of new peers, resolution of a peer's identity and network address, and membership of peers in different groups. The application distinguishes between two types of peers, i.e., mentors and apprentices, and provides several functionalities in support of collaborative apprenticeship learning. First, it enables the peers to interact with each other, either with public-message broadcasting or with private chatting between two peers. Messages can be annotated with references to passages from textbooks that the peers may use for their rotation and with pointers to entries in their portfolio. Second, it supports a simple workflow among the apprentices and their mentor, i.e., a threaded question-and-answer forum, where the mentor can assign questions to individual students. Third, the application provides basic functionalities for management of the apprentice portfolio, i.e., adding and editing entries to a database, and the submission of completed portfolio entries for review by the mentor. The application distinguishes between archival data, such as submissions, which are stored at the relay and ephemeral data, such as conversations, which are stored locally at each peer.

The focus of this research is more on investigating and evaluating the relevance and appropriateness of the J2ME and JXME technologies for supporting the collaborative interactions that occur in group apprenticeship-learning contexts, an example of which is the surgery-rotation scenario. An earlier version of the application, called E-Logbook, has been used by medical students at the UoA. MediCol has still to be used by medical students. Instead, we have conducted a set of simulation-based experiments to evaluate

whether its run-time performance can realistically support the interactions of the student team during their surgery rotation. More specifically, we conducted the following three experiments.

(a) Experiment 1 – to investigate how the size of the group affects the response time (RTT) of the relay,

(b) Experiment 2 – to investigate the variation of the average data throughput of the JXTA unicast and propagate pipes with an increasing message size, and

(c) Experiment 3 – to model the sending rate (i.e., the rate at which a peer can broadcast/send messages infinitely to the group for every peer to receive the message successfully) with an increasing message and group sizes.

1.3 Contributions of this research

- A review of the literature on E-Learning and M-Learning – approximately 30 high-quality projects have been reviewed and are discussed in some detail in the related-work section;
- A collaborative system, MediCol, for the promotion of peer learning in apprenticeship settings;
- A set of experiments demonstrating the feasibility of the application's deployment in a realistic setting with very promising initial results;
- An initial assessment of the performance of JXTA on Java-enabled handhelds by using our collaborative apprenticeship-based application – to our knowledge, assessing the JXME performance is the first of its kind and has shown very promising results when JXME has been applied to the apprenticeship-learning setting.

1.4 Thesis outline

This thesis is organized as follows. Chapter 2 provides a description of some related research in areas such as e-learning, m-learning, collaboration, workflow-based applications, amongst others. Chapter 3 describes the JXTA software architecture by explaining its underlying concepts and showing the implications of P2P design. The

MediCol architecture is presented in Chapter 4. The latter sheds more light on the design and implementation of the MediCol application and all of its components. Besides, several tools offered by the J2ME Toolkit are used to analyze MediCol experimentally. Chapter 5 illustrates how the MediCol architectural implementation was tested by means of simulation-based experiments. A detailed description of the nature and analysis of these experiments is also provided. The contributions of this research, as well as the possible directions for future work are summarized in Chapter 6.

Chapter 2 E- and M- Learning

E-Learning has been receiving increasing attention as a research area in the recent past. Several different types of research questions are being currently investigated:

1) How can computers support traditional, classroom-based education?
2) How can computers be used to deliver learning modules at a distance, with either synchronous or asynchronous interactions among the students and the instructor?
3) How can computers, especially mobile devices, be used to enhance "regular" activities with potential for learning?
4) How can different pedagogical models of education, like peer-learning for example, benefit from computer support?

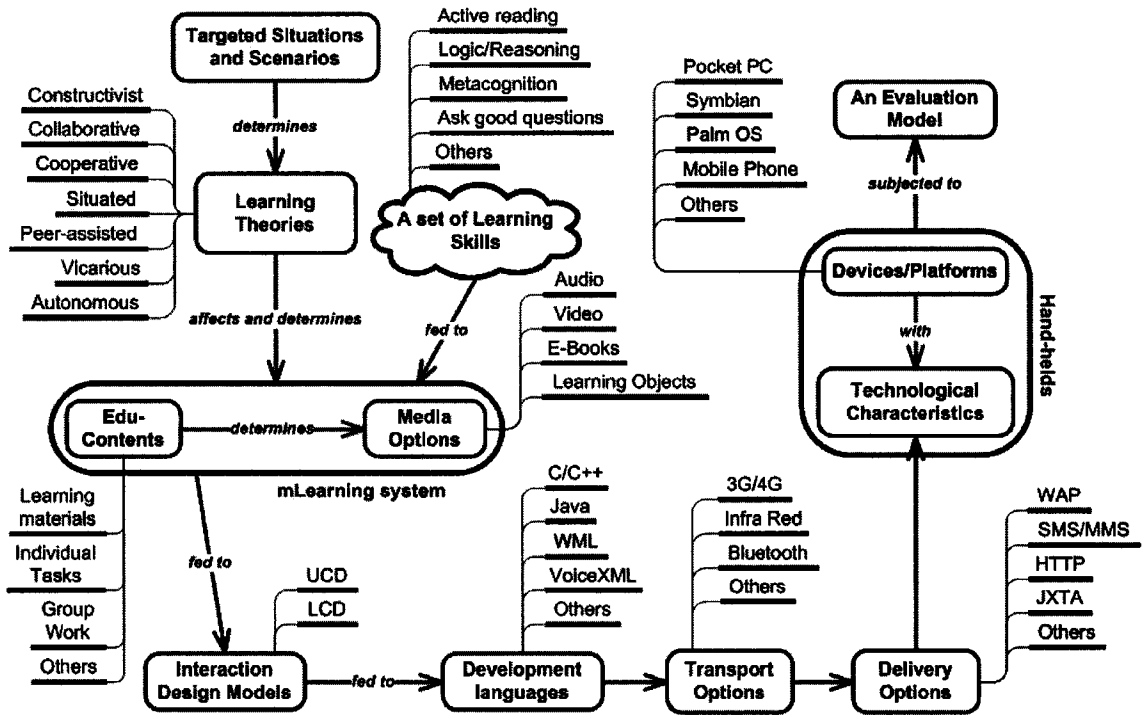


Figure 1: The E- and M-Learning Research Map

Various aspects of the E- and M-learning fields have received varying levels of attention from researchers. While some have concentrated more on the theory (learning theories and the learners' set of skills) and design (interaction design models and technological characteristics), others have placed more emphasis on the learning materials (contents to

be offered by the system) and media (the means through which the contents will be delivered).

Figure 1 summarizes the research focus of various researchers in the m-learning domain. Essentially, researchers start with a few scenarios to determine or specify how they want to integrate handhelds within a particular learning activity. These scenarios will then act as the driving force and will help choose a few learning theories that will be applicable to the selected scenarios. The m-learning system will be designed based on the selected scenarios and learning theories. Edu-contents (learning materials, autonomous tasks, project work) and media (audio, video, learning objects) form the crux of the m-learning system. Once the various educational contents, which will be offered by the application, are decided upon, the media through which such contents will be delivered are determined. A suitable interaction design model will then have to be finalized to model the way in which the learner will interact with the system. User-centered design (UCD) focuses more on making the system more user-friendly and easy to use, while learner-centered design (LCD) predicts learning materials to learners by relying on such information as what activities the learner has been doing in the past few days, the learner's interest, and his/her goals. Then, the focus shifts towards the technologies that will be used to develop the system. Any handheld (with operating system like Windows Mobile, Palm, Linux, Symbian or Mac) has specific technological characteristics such as development languages (each platform has a software development kit that should be used to develop and package the mobile application), transport option (such as 3G/4G networks and Bluetooth to transport m-learning contents), and delivery options (protocols such as WAP and JXTA to deliver the m-learning contents to the handhelds). Once the m-learning system is developed and fully functional, it is subjected to an evaluation model. Although, the latter has received very little attention from researchers, they are quickly trying to come up with sound evaluation models to better evaluate any m-learning system. Finally, any e- or m-learning system relies on a set of learning skills (such as read/participate actively, reason properly, ask good questions) that learners are expected to possess.

The MediCol application targets apprenticeship-based scenarios; uses the collaborative, cooperative, and vicarious learning theories; offers both autonomous and collaborative

group tasks that are delivered through such media as e-books and portfolio; has been implemented using Java (J2ME) to target platform-independence; uses 3G/4G wireless networks for transferring m-learning contents; uses JXTA as the underlying protocol to deliver HTTP-based binary messages among peer students and the mentor; and finally, MediCol can run on any mobile platform as long as the platform has a KVM installed on it.

2.1 Collaborative E-Learning applications

The subtle difference between collaborative and cooperative learning theories has caused confusion among many researchers. Collaborative learning theory is based on the collaboration among students and teachers [CSG05]. In this approach, teachers are not the ones guiding the learning process. Instead, they contribute to knowledge building by sharing their experiences like any other member. On the other hand, cooperative learning theory relies on the teacher as the one who trains students to learn concepts through the use of content materials and activities [CSG05]. Both approaches are enriched by the use of computers in order to support collaboration and distribution of content materials.

Quite recently, there has been substantial effort towards collaborative e-learning applications, with and without the use of JXTA. Following is a review of a few JXTA-based applications. Public Discussion Forum (PDF), a database-backed forum system, has been deployed as a means for fostering the exchange of class-related information between students and instructors [HD02]. Questions regarding lecture materials, assignments, projects and exams represent the bulk of discussions in the PDF system. PDF, originally designed as a client-server system, was redesigned and implemented as a JXTA-based P2P system to reduce hardware commitments and increase the scalability as well as availability of forum services. Hydra [Zua05], a lightweight and SCORM-based P2P e-learning architecture, makes use of a fast native XML database to enable basic query, retrieval and download service using SCORM (Sharable Content Object Reference Model) metadata.

On the other hand, there are a few interesting non-JXTA e-learning applications. One of them is OpenSeminar, which facilitates the structured compilation of open and on-line

resources among a group of collaborators (often, course instructors) who share a common area of interest [RSY+05]. It is a customizable web-based platform that enables a group of instructors to collaborate on materials for similar courses by sharing links to the open contents.

2.1.1 Collaborative data handling

The ABC architecture [BBP+05] distinguishes between accountable and ephemeral events. The former are state events which the collaborative system needs to keep track of, while the latter are events existing for only a short period of time and need not be saved or interpreted in some context. In the same respect, the nature of the exchanged data is an important aspect of any mobile collaborative application. All data involved have to be classified as either long-term or limited life-spanning [LLK04]. The truth is it may not always be necessary to discard temporary data. Such data may be saved, but later overwritten when it becomes sufficiently stale with negligible utility [BCM+05]. The reason behind saving limited life-spanning data at least until they become stale, is that people tend to be less committed and concerned about what they say when they know that the data they produce are ephemeral [HSH93]. Hence, such data should be saved, albeit for a short period of time, to make its users more responsible. For example, chat logs are simple ephemeral contexts; a workspace with shared resources and distributed data is a persistent one [BB03].

Any collaborative application must provide some sort of floor control that supports the rules by which an activity is usually managed [BBP+05]. Therefore, the floor control must assign the proper control functions to each peer according to the role played in the floor control model. For instance, the peers in a collaborative application may have a student's or instructor's role. Therefore, the application should provide the appropriate control functions (e.g., submission of assignments by students and marking of assignments by professor) based on the peer's role.

The work in this thesis is in the realm of the traditional chat application. However, the latter is often unstructured and full of ambiguities [MOM96][SCB00]. These can be eliminated through the use of the turn-taking structure of human conversation and

threaded chat [SCB00]. This has, so far, not been explored on the mobile platform and presents an interesting challenge.

2.1.2 Workflow-based applications

Workflow management provides the opportunity to enhance the efficiency of an organization or an educational institution by automating collaborative work activities over distributed environments [JHS+99]. Earlier, most workflow management systems supported users performing work activities only in the office environments. Nevertheless, the growing popularity and maturity of mobile computing have made the computerization and automation of such activities a reality. A workflow management system supports the definition and execution of a coordinated set of activities. Therefore, an application that can be modeled and executed through such a set of activities is called a workflow application. We have tried to model MediCol as a workflow application, particularly the conversational protocols and the submission procedure. Essentially, the submission procedure is broken down into a number of activities that interact together to achieve the students' objective of remotely getting their portfolio entries signed off by their mentor. In this sense, students submit their portfolio records and the JXTA relay logs the arrival of a new submission. After the relay forwards the submission to the mentor, the students get an acknowledgement confirming the successful receipt of their submissions by their mentor. Otherwise, the relay advises the students through a message to submit again. The mentor signs off the submissions and the relay logs the reviews before forwarding them to the appropriate students. Therefore, the various components that interact together to achieve the remote signing-off objective are submission "archiver" (residing at the relay), the JXME peers, and the relay itself (which acts as both a notifier and submission manager).

2.2 M-Learning

Handhelds are as common as stethoscopes at dozens of U.S Medical schools today [Fal02] since nearly one-fifth of U.S's 125 medical colleges require their third- and fourth-year medical students to use mobile devices. These medical schools are leading the way in the deployment of handheld computers and wireless technologies to monitor

closely the students' performance, enhance student-educator interactions, improve course management, and ensure that students have the latest information as they move between classrooms, hospitals, libraries, and clinics.

M-Learning projects appear to have mushroomed in the past 2-3 years. The question then becomes: What are the advantages of m-learning over traditional learning settings and e-learning? Several answers come to mind and the following are the most important ones.

- *Convenience*: Students can access and study their learning materials anytime, anywhere.
- *Fun*: Many m-learning applications adopt the guise of console games (edu-games) to engage the learners.
- *Collaboration*: Lightweight communication protocols, like SMS and chat, make collaboration and peer learning a very natural activity in the m-learning context.

The domain of m-learning systems abounds with a variety of research and deployed systems, which we classify around a few general themes and review in the forthcoming sub-sections.

2.2.1 In support of classroom learning

As with the earlier generation of e-learning systems, handhelds have been used to sustain traditional classroom-based instruction, by supporting instructors to manage their course contents and by enabling students to flexibly interact with their instructors, TAs, and with each other.

MAST [SCC03] is a mobile information system aimed at supporting instructors' activities such as creation, retrieval and updating of course materials and management of students' registration, lab assignments, and course attendance. Instructors must access the course repository while the network is available; then they can use MAST to update it virtually; once the instructor's mobile is back online, the course repository is updated.

ActiveClass [RST+03][GSB+04] was designed as an enabler for the interaction between the instructor and the students within the classroom. It allows students to post questions anonymously and students can rate question(s) of interest to urge the professor for

quicker attention. Students provide feedback by rating the professor's answers. Everyone can view the list of questions and their corresponding ratings. Similarly, ConcertStudeo [DWN03] aims at integrating handhelds into face-to-face learning and combine the benefits of both. It uses wirelessly-connected PDAs to support brainstorming, quizzes, polling, and an electronic blackboard in the context of workshops and classroom sessions. Handhelds have also been used to help students prepare for tests by delivering information and practice quizzes to the students' mobile phones [WKG+03]. They can thus access a set of review questions and the complemented instructional feedback. A comparable system, VUR [BMN+03], is reportedly used by 83 lecturers and 3100 students enhancing the lecture experience with messaging, discussion, and course-content access. Handhelds with WAP browsers are used to access course websites and are informed about new materials and postings through SMS notifications.

mediaBoard [CS04] enables learners to collaborate using text and annotated picture messaging as they work together to construct a shared website or repository that can be used as a diary or mobile web-blog. Interactive Logbook [BBC+04] also provides easy access to teaching materials and 'mini applications' such as internet browsing, instant messaging, freehand notes taking and a peer-to-peer whiteboard. The system keeps a log of the user's activities, which can later be used as a learning diary or portfolio.

Mercier et al. [MDC+04] developed a web-based thin client accessible through handhelds to deliver interactive tests to students, gather their feedback, enable note taking, and engage them in conversations with teachers and peers. Complementary features like better monitoring tools for the teacher and event tracking are included.

mLab [MFC04] is a laboratory environment accessible from handhelds that enables students to control simulations remotely. Using a wireless device, students log in to the remote server where the simulation is carried out, and can start or stop a simulation. At any time, students can check the simulation's progress via their handhelds. Eventually, students are provided with the simulation results, which can then be compared with the teacher's reference results.

Finally, the INLET project [Pin04] demonstrated how just-in-time knowledge delivery through an SMS-searchable database can be used for contextualized learning of

introductory Greek during the Olympic Games in 2004.

2.2.2 Meta-cognition enablers

Besides supporting traditional classroom-based activities, e- and m-learning systems have explored means for engaging learners in reflection about their knowledge and other meta-cognitive activities.

Topic or concept maps have been recently proposed as languages for representing domain facts and meta-data. PiCoMap is a concept-mapping program part of the Hi-CE [Mif03] suite of Palm-based tools, designed for students aged between 7 and 14. Students working on a topic can first generate their own concept maps, which can then be sent through an IR port to another PDA for review.

The mCLT system [AGT+04] logs in students and shows them the list of subscribed courses. Students can then select a particular course to view its contexts. Each context is a discussion on a topic and resembles closely a thread in a discussion forum. However, using meta-cognitive identifiers (e.g., problem, comment, working theory, note, etc) makes it easy for students to identify the key concepts in the discussion. Thus, the system increases the participants' awareness of their fellow students' thinking [LVH+02], allowing them to participate more actively.

2.2.3 In the context of professional development

Because professionals are increasingly using handhelds to communicate while at work, we are also seeing many m-learning applications tailored for apprenticeship and lifelong learning for particular professions.

mLES [MG04] supports the management of emergencies through m-learning. It integrates a GIS (Geographic Information System) with a GPS (Global Positioning System), alarm communicating tools, and emergency data-analysis tools.

[MTFM03] includes applications designed to support the 'Banking, Insurance and Securities Market Law (BISML)' and 'Actuarial Statistics (AcStat)' courses in the context of the actuarial degree at Málaga University. The MOBIlearn generic architecture [LBS+03] was used to develop two components in the context of a traditional MBA

course [Fro04].

KLIV [BH03] enables peers to interact with each other, through the production and consumption of learning content. For example, nurses produce short video clips on best nursing practices, which are stored at and accessed from locations where they may be relevant in a hospital.

Many documentation activities exist in the public health sector, including management of patient records in physician's offices and hospitals. Data is usually captured on paper and later transferred to electronic records. This transcription step is both time-consuming and error-prone. Therefore, by providing health practitioners with handhelds for recording their data at the point of first interaction with patients, a new level of health support can be achieved. Students of medicine, nursing, and dietetics practicing in the wards were trained by the HISS project [CCC+04] to use handhelds connected through a WLAN to record patients' data using the Electronic Patient Record structure, which is suitable for handhelds. Data security and concurrency during simultaneous records editing are among the particularly interesting issues that arise in this context.

2.2.4 Enhancing the field-trip experience

Mobile applications have also been explored as a means for enhancing the students learning experience during field-trips by supporting activities such as taking and annotating photos, discussing samples or observations, and taking notes for future reference.

The RAFT project [HRS03] investigated the provision of a cooperative learning environment that spans the field-trip and the classroom. To ensure all participants are fully engaged in the event, various field-trip roles, e.g. field communicators, data gatherers, and archivists, are being explored.

[CJ05] reports on the use of PDAs during field-trips as an alternative to traditional paper-based workbooks. The goal was to determine how handhelds can be used to enhance and support learners during field-trips. Prior to the field-trip, various information files were prepared by the lecturer and transferred to the PDAs. During the field-trip, learners became cognitively engaged in the activities offered on the PDAs as they took notes on

their devices.

Finally, there are a few concrete examples of the use of handhelds during field-trips, which were carried out at the University of Tennessee [ALM05]. For example, food-science students used PDAs, cameras, and temperature probes to collect on-site data from grocery stores and restaurants for analysis at a later time.

2.2.5 Context awareness

Mobility gives rise to an interesting opportunity for applications, namely context sensitivity: as users roam through a networked space, the mobile application can use their locations to infer what their tasks might be and offer only information that can be relevant to these tasks.

The MOBIlearn system [LBS+03] deploys the generic Context-Awareness System (CAS) consisting of a set of software objects called *context features*. The latter relate to the learner's activities and device capabilities to derive a *context substate*, which is used to exclude irrelevant contents (e.g., high-resolution web pages that cannot be displayed on a PDA) and then rank the remaining contents based on how closely they match the learner's profile and device's features. The ranked set of contents is then *output* to the content-delivery subsystem. CAS's primary purpose is to perform intelligent matching between content meta-data and learners' context (like physical location, current activities, and goals) to make recommendations to learners about currently appropriate contents.

[BLS+04] describes the development of a test-bed to explore the issues involved in the m-learning scenario of a visit to an art gallery. During such visits, handheld users will view the contents related to particular objects that they move past. Similarly, the ActiveCampus Explorer [GSB+04] acts as a guide to students moving through campus, informing them about persons, locations, and events of interest in their vicinity. Also Moop [MF05], designed for primary-school pupils for the Symbian (S60) platform, supports them in analyzing their surroundings and communicating with each other. The pupils' tasks are based on their geographical locations and require creative problem-solving during which, observations and information are saved in the pupils' handhelds.

Generally, e-mail and forums are viewed as the main forms of asynchronous interactions.

However, 'post-its' can now join this list. By extending the concept of asynchronous interaction, 'post-its' can be attached to physical locations to enable learners to leave important messages for others in the context of the current location [FHO04].

2.2.6 Edutainment

Edu-games are fun and especially appealing to young learners, who can polish their team skills, social and communication skills, and resource-sharing skills through game playing.

Fabricatore [Mit03] created six Game-Boy games, which are reported to have been successfully tested with the help of teachers' supervision in classrooms of around 300 children aged between 6 and 8. The teachers observed both intentional and unintentional learning gains, including 'general improvements in terms of discipline, concentration, and eagerness to understand technological issues related to the games they were playing.'

The mobileGame project [GHS04] explores opportunities for supporting learning through an orientation mobile game in a university environment. The competitive element is based on hunting rules, whereby each group tries to catch a second group and equally, the former group is hunted by yet another group. The handheld device shows each group where its corresponding hunter and prey are located. Such a system is not following any m-learning philosophy, but it is useful for new university students to "learn" their neighborhoods.

Chapter 3 Infrastructure for Distributed-Applications Development

Distributed or P2P applications are better compared to their centralized counterparts for reasons such as hardware commitments and service availability. This chapter reviews thoroughly the JXTA software architecture, which can be used to build both desktop-based and mobile P2P applications.

3.1 The JXTA specification for P2P architectures

P2P applications generally must handle intermittent connectivity, dynamic IP addresses and an unstable network topology. Therefore, developers face a challenge in designing applications that are independent of the DNS and IP addressing mechanism [HD-Aug03]. This is the main objective of JXTA, which is a P2P networking framework, and is achieved by introducing an overlay or virtual network layer on top of the existing physical network along with its own addressing and routing mechanisms. Publishing, discovery, and exchange of advertisements, which are used to publish the existence of resources and services, are essential during the process of connecting to a JXTA network.

JXTA is referred to as a framework because it provides a collection of reusable classes that offer a set of services for the P2P domain. JXTA consists of three layers, which in turn have several interacting components. The JXTA architecture, as shown in Figure 2, illustrates the layers, which are described below [JJPG], and the various components.

- Platform layer or JXTA Core: encapsulates the fundamental primitives that are common to any P2P networking solution. These include important building blocks such as discovery, transport (handling of firewalls and NATs), creation of peers and peer groups, and associated security primitives.
- Services layer: provides network services, which are desirable in a P2P network but are not critical for its successful operation. Examples of such network services are searching and indexing, directory, storage systems, file sharing, distributed file systems, protocol translation, authentication, and PKI services.
- Applications layer: includes the implementation of integrated applications, such as P2P instant messaging, document and resource sharing, content management and

delivery systems, P2P Email systems, distributed auction systems, and the likes.

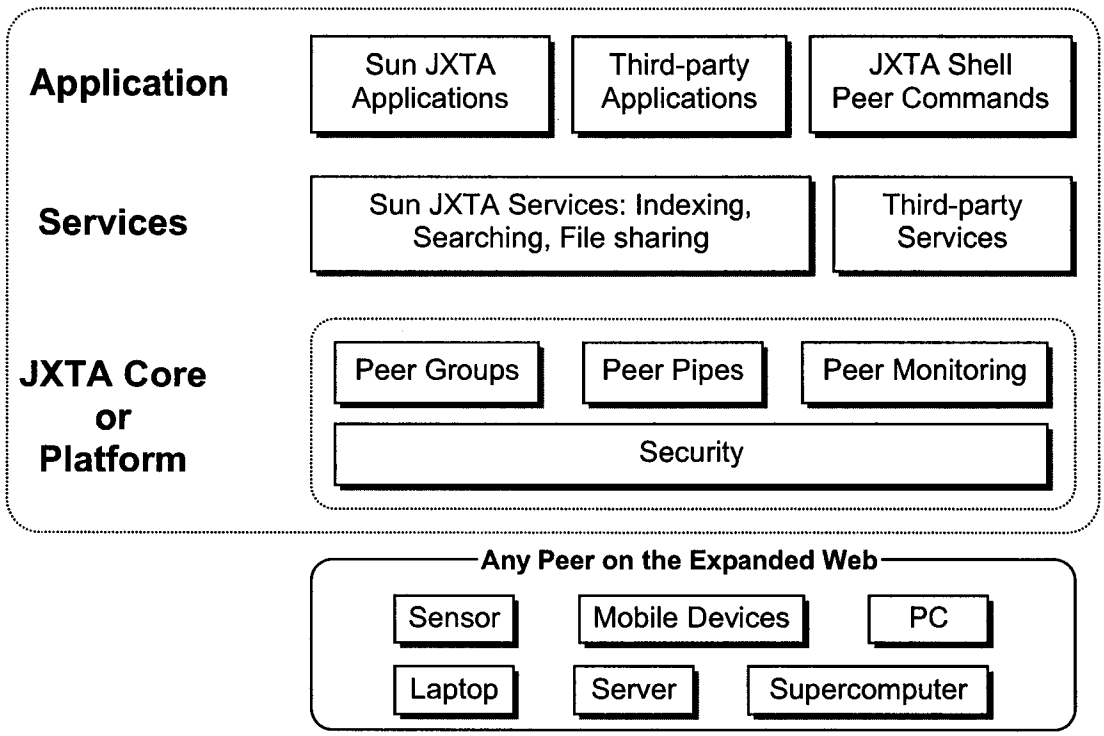


Figure 2: JXTA Software Architecture [Sul01]

3.1.1 JXTA Design objectives

The popularity of JXTA has been increasing constantly since its introduction. Viewed as one of the most versatile P2P networking framework, JXTA has several benefits. Among the most important ones are interoperability, support for network virtualization and dynamic reorganization capability, and fault resiliency [MPI-CJS].

Interoperability: JXTA is not an application; rather it is a set of specifications exhibiting P2P characteristics. By implementing these specifications in any programming language, JXTA peers can therefore cooperate with each other by exchanging XML messages.

Network virtualization: JXTA abstracts away the different transports and addressing schemes interconnecting the peers participating in the application and presents the peers with a virtual mesh network, in which every peer is connected to every other peer. This is achieved by an intelligent message-routing layer that uses a uniform addressing scheme

and very late binding, on top of multiple endpoint protocols. Essentially, each endpoint protocol becomes a driver for the virtualized JXTA network, mapping the virtualized network onto the physical one. A peer can generate an ID, which is human-readable and will be used by other peers to recognize him, for himself and join the application's virtual network immediately without contacting any registry such as DNS. This enables peers to join and leave the network at any time. Moreover, any change in the IP address following the movement of a peer from one domain to another, will not affect the application's behavior for that peer.

Fault resiliency: A peer is assigned a UUID by the JXTA Core layer, when it joins the network with a particular peer-generated ID. While joining the network, it must first initialize itself by discovering the local peers and their capabilities, the available peer groups and the services (e.g., discovery, membership, access, pipe, resolver, monitoring, and others) available to each peer group. It does so by submitting the appropriate 'search' requests to the relay (if it is a mobile device) or to the rendezvous. The relay, rendezvous, and different services are discussed thoroughly in the next few sub-sections. The new peer may also decide to join one or more groups and start using their services. Since peers join and leave the network dynamically throughout the lifetime of the application, they are essentially unreliable. The JXTA framework acts as a message-routing agent by rerouting messages on the fly, thus catering for a dynamically changing P2P network topology.

3.1.2 JXTA building blocks

The platform layer of the JXTA architecture consists of several fundamental building blocks that are characteristic of any P2P networking solution. The following sub-sections review the main ones in detail.

3.1.2.1 Peers

A JXTA network consists of a number of interconnected nodes, called peers. A peer is any networked device that implements one or more of the JXTA services/protocols. Peers can range from sensors, phones, and PDAs to PCs, servers, and supercomputers. Each peer, uniquely identified by the UUID assigned by the JXTA Core layer, operates

independently and asynchronously from all other peers. JXTA defines four types of peers [JJPG]:

- *Minimal edge peer:* it can send and receive messages, but neither routes messages nor caches advertisements for other peers. Devices with limited resources (e.g., a PDA or cell phone) fall in this category.
- *Full-featured edge peer:* it typically caches advertisements besides sending and receiving messages. Such a peer replies to discovery requests with information found in its cache, although it does not forward any discovery requests to other peers. Most peers belong to this category.
- *Rendezvous peer:* it resembles a full-featured peer and, in addition, it caches advertisements. Furthermore, rendezvous peers forward discovery requests to help other peers discover resources. While joining a peer group, a peer automatically seeks a rendezvous peer. If no rendezvous peer is found, it dynamically becomes one for that peer group. Each rendezvous peer, say X, maintains a list of other known rendezvous peers as well as those peers that are currently using X as a rendezvous. Only rendezvous peers that are a member of a peer group will see peer group specific search requests. Edge peers (minimal and full-featured) send search and discovery requests to rendezvous peers, which in turn forward requests that they cannot answer to other known rendezvous peers. This discovery process continues until either one peer has the answer or the request dies.
- *Relay peer:* it maintains information about the routes to other peers and routes messages to peers. A peer first seeks route information in its local cache. If it is not found, the peer sends queries to relay peers asking for route information. Relay peers also forward messages on behalf of peers that cannot directly address a recipient peer (e.g., peers behind NATs), thus bridging different physical and/or logical networks.

3.1.2.2 Peer groups

A peer can self-organize and join other peers to form one or more peer group(s), each identified by the UUID assigned by the JXTA Core layer [JJPG]. A peer group is a collection of peers that have agreed upon a common set of services. Examples of such

services are document sharing and chat applications. JXTA defines the following core set of peer group services or protocols [MPI-TJS]:

- *Discovery:* it enables peers to search within the scope of peer groups for resources such as peers, peer groups, pipes, and services. The rendezvous peer will handle the discovery requests.
- *Membership:* it accepts or rejects applications from peers who wish to establish membership with one or more peer groups. Such peers must first locate a current member and then request to join. The current set of members will jointly decide whether to accept or reject an application.
- *Access:* it validates requests made by one peer to another and acts as a security service for controlling access to services as well as resources within a peer group. It acts as some sort of security manager for the peer group.
- *Pipe:* it creates and manages pipe connections among the peer group members.
- *Resolver:* it allows peers to refer to other peers, peer groups, pipes, or services indirectly via an advertisement. The latter is bound at run-time to an implementation by the resolver.
- *Monitoring:* it gives one peer the privilege to monitor other members of the same peer group.

However, it is not mandatory that every peer group implements all of these services. A peer group can choose to implement only those services that it finds useful, while relying on the default *NetPeer* group to provide generic implementations of the remaining core services.

Peers cooperate and communicate to publish, discover, and invoke network services. They can publish multiple services and use the Discovery Protocol to discover other network services. The JXTA protocols recognize two levels of network services [JJPG]:

- *Peer services:* A peer service is accessible only on the peer that is publishing that service. The service will fail if that peer fails. Multiple instances of the service can be run on different peers, but each instance has to publish its own advertisement.

- *Peer group services:* A peer group service is composed of a collection of instances (potentially cooperating with each other) of the service running on multiple members of the peer group. If any one peer fails, the collective peer group service is not affected (assuming the service component is still available from another peer member). Peer group services are published as part of the peer group advertisement.

3.1.2.3 Advertisements

Services can be either pre-installed onto a peer or loaded from the network [JJPG]. In order to actually run a service, a peer has to locate an implementation suitable for his runtime environment. The process of finding, downloading, and installing a service from the network is similar to performing a search on the Internet for a Web page, retrieving the page, and then installing the required plug-in. All JXTA network resources - such as peers, peer groups, pipes, and services - are represented by an advertisement, which are language-neutral meta-data structures represented as XML documents. The JXTA protocols use advertisements to describe and publish the existence of network resources. Peers discover resources by searching for their corresponding advertisements, and may cache any discovered advertisements locally. Each advertisement is published with a lifetime specifying the availability of its corresponding resource. This enables the deletion of obsolete resources without requiring any centralized control. An advertisement can be republished (before the original advertisement expires) to extend the lifetime of the associated resource.

3.1.2.4 Pipes

JXTA pipes are an abstraction used for inter-peer communication. JXTA peers transmit messages through pipes, which are virtual channels consisting of input and output ends [JJPG]. The input end of a pipe refers to the receiver, while the output end of the pipe refers to the sender. Peers bind to each end of the pipe and when both ends are bound, messages can be transmitted along the pipe.

Pipes are not bound to the physical location, such as an IP address or a port. Instead, pipes have the unique UUID assigned by the JXTA Core layer, so that each peer can

carry its pipe with itself even when its physical network location changes. At runtime, a pipe end (input and output) is resolved to the endpoint address, which is a peer network interface (e.g., a TCP port and associated IP address), to which the peer is currently bound. By definition, pipes are asynchronous, unidirectional, and unreliable.

Pipes offer two main modes of communication, namely unicast and propagate, as illustrated in Figure 3.

- *Unicast pipe:* connects exactly two pipe endpoints together; the input end on one peer receives messages sent from the output end of another peer.
- *Propagate pipe:* connects one output end to multiple input ends; messages flow from the output end (the propagation source) to the input ends.

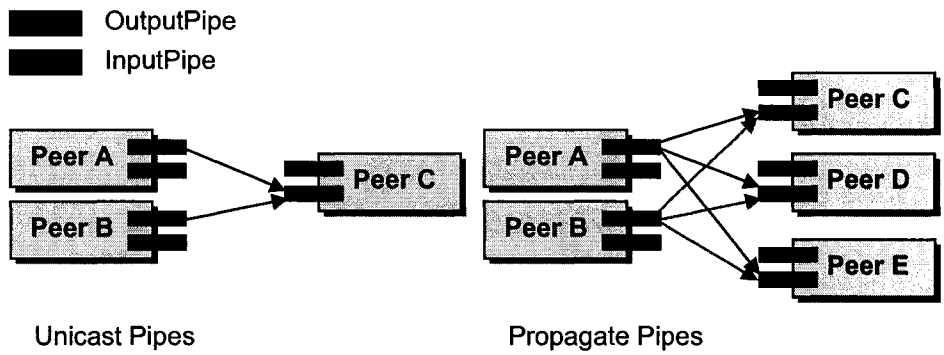


Figure 3: Unicast and Propagate Pipes [JJPG]

3.1.2.5 Messages

Messages, transmitted along pipes, are simple XML documents whose envelope contains routing, digest, and credential information [JJPG]. A message is an ordered sequence of named and typed contents called message elements. Thus, a message is essentially a set of name/value pairs and its content can be an arbitrary type (i.e., any type of payload).

There are two representations for messages, namely XML and binary. The latter is used by minimal edge peers because using an XML parser will exhaust the computationally constrained devices.

JXTA uses source-based routing whereby each message carries its routing information, which is a sequence of peers to traverse. The peers along the path may update this

information.

3.1.2.6 Resource identification

Peers, peer groups, pipes and other JXTA resources need to be uniquely identifiable. Consequently, a UUID assigned by the JXTA's Core layer uniquely identifies an entity and serves as a canonical way of referring to that entity [JJPG]. Uniform Resource Names (URNs) are used to express JXTA UUIDs. URNs, a form of Uniform Resource Identifiers (URIs), are intended to serve as persistent and location-independent resource identifiers. Like other forms of URIs, JXTA UUIDs are presented as text.

3.1.3 Performance of JXTA

The popularity and widespread use of JXTA have raised a few questions concerning its performance. This resulted in a few in-depth researches attempting to find convincing answers to the questions raised. Initially, [HD-Sep03] defined the following metrics, which are essential for an accurate and complete evaluation of the JXTA platform.

(1) *Message round-trip time*: evaluates the performance of JXTA pipes and the communication sub-system.

(2) *Pipe message throughput*: reflects the maximum number of messages that a JXTA pipe can transmit one-way without loss.

(3) *Rendezvous query throughput*: investigates the response time, message and query throughput, and advertisement cache management as part of the rendezvous' throughput.

(4) *Relay message throughput*: investigates the effect of an increasing number of NATs and firewalls on the relay's throughput.

The same researchers then drew the following conclusions regarding the performance considerations for JXTA.

- It is expected that a high performance penalty is associated with the layers of abstraction, especially in terms of latency and response delay. In addition, a peer may frequently perform some operations, which then aggregate to a large performance

cost. The most frequent operation would be sending a message and the second most frequent one would be obtaining a pipe advertisement.

- The nature of publishing may also introduce a delay. Particularly, JXTA allows for two types of publishing, local and remote. Local publishing puts the advertisement in the local cache, from where it is sent out when discovery queries arrive. In this case, it is up to the other peers to send a query to obtain the advertisement. In contrast, remote publishing sends out the advertisement to the rendezvous peer and other peers. Depending on the frequency of publishing and number of connected peers, this may turn out to be costly in terms of network traffic and processing on the peers and rendezvous.
- Obtaining a group membership may involve different delay costs, according to the security policies of the peer group.

A few other studies [HD-Aug03][Sei02][AHJ05] produced more insightful results on the performance of JXTA by unveiling some important and conclusive numbers as reference.

- It takes on average 10 seconds for a JXTA peer startup. The "startup" test measures the time it takes to initialize the JXTA platform on a single peer.
- A single HTTP relay costs the propagate pipe over 3 times the delay associated with IP multicast, while two relays cost 7 times.
- It is observed that the first 1000 round-trips of messages are processed slightly more slowly than the rest of the round-trips. This state may be attributed to the initial creation of objects required by the relay for processing the requests. Thus, a warm-up period of 1000 message acknowledgements is performed to ensure that the Just-In-Time (JIT) compiler does not influence the results during performance testing.

It is interesting to note here that – to our knowledge - there is little or no significant performance studies for the JXME (JXTA for J2ME) platform. The work in this thesis is also geared towards this direction and aims at throwing some light on the performance of JXME by using our collaborative apprenticeship-based application. The evaluation of the relay's behavior for large peer groups is necessary for a complete performance picture of JXTA and this is yet to be investigated [HD-Sep03]. We attempt to investigate this

feature too.

3.2 The J2ME Java Mobile platform

J2ME is quickly becoming a standard for mobile handheld devices and is being widely adopted as the platform for the development and delivery of various applications to mobile users. J2ME is Sun Microsystems' version of Java aimed at software development for machines with limited hardware resources such as PDAs, cell phones, and other consumer electronic and embedded devices. In this section, we will see why J2ME is clearly on the rise and becoming a front-runner in the mobile application development industry. This section will also describe the J2ME architecture briefly with respect to the Mobile Information Device Profile.

3.2.1 Importance of J2ME

With a significant increase in the number of mobile handheld users due to its affordability, the idea of using handheld in a collaborative apprenticeship scenario looks appealing. Besides this, the mobile industry is looking for better and quicker platforms for the development of device-independent mobile applications. J2ME seems to match these criteria because the Java programming language has the "write once, run anywhere" rule as its foundation. Therefore, some of the major benefits of using Java as the programming language for mobile application development as outlined by [ATR+01][Gig] are:

- Nature – Java is an object-oriented programming language, with better programming constructs and abstraction mechanisms than other tools and languages used for wireless software development.
- Popularity and support – Java has a very large programmer base worldwide. Moreover, many industries and tools support J2ME (e.g., Eclipse has a plug-in for J2ME/MIDP development, known as EclipseME).
- Platform-independence – Java has the ability to run the same bytecodes on top of any device's operating system as long as it has the appropriate JVM.
- User interface – Wireless Java technology offers a rich set of GUI widgets and

provides graphic capabilities for mobile devices.

- Faster development – Java codes can be developed more quickly and are easier to maintain than C/C++ codes.
- Security – Security is one of Java's main design goals and is built into the Java Sandbox model by including features such as the bytecode verifier and prevention of invalid random memory access.

The J2ME platform provides support for a range of low-end devices like cell phones and pagers, and high-end devices like set-top boxes and high-end PDAs. Because of a substantial increase in the number of mobile handheld users, mobile application developers will be more inclined towards adopting the J2ME platform. Some interesting facts and statistics outlined below can confirm the popularity status of J2ME:

- J2ME on handsets is supported by all major carriers and strongly backed by all major phone vendors (such as Motorola, Nokia, Palm Inc., Pocket PC, etc) [LG-WHY].
- It has been observed that the interest in Java as a platform for mobile handsets has grown significantly [McA02].
- J2ME on cell phones sells – [LG-WHY] shows that the cell phone and J2ME combination is a commercial success, with more than 94 million devices shipped worldwide since 2002 [LG-WHY].

These are only a few of the statistics available that show the emergence of J2ME as a standard for the fast growing wireless web industry [LG-WHY]. From these and many other statistics, it can be deduced that mobile users of and mobile research using J2ME-enabled devices are increasing rapidly, especially since this platform 'has positioned itself as the best solution for an extremely wide range of small devices' [LG-WHY]. As the J2ME platform becomes more and more mature over time, more applications (collaborative, gaming, and the like) will become available for this platform.

3.2.2 How does J2ME fit the Java Development Platform?

J2ME is part of the three main platforms available for the development and execution of Java or "write once, run anywhere" codes:

- Java 2 Enterprise Edition (J2EE) – enables the development of complex server-based applications.
- Java 2 Standard Edition (J2SE) – enables the development of applications for desktops and personal workstations.
- Java 2 Micro Edition (J2ME) – enables the development of mobile applications for limited footprint devices.

Figure 4 shows the relationship between these three platforms. Given that J2ME is derived from J2SE, many Java packages present in J2SE have been stripped so as to keep the platform small and suitable for its targeted devices. Even those J2SE packages that are present in J2ME do not contain all the classes present in the J2SE counterpart. J2ME provides its own packages for persistent storage, user interface widgets, and networking.

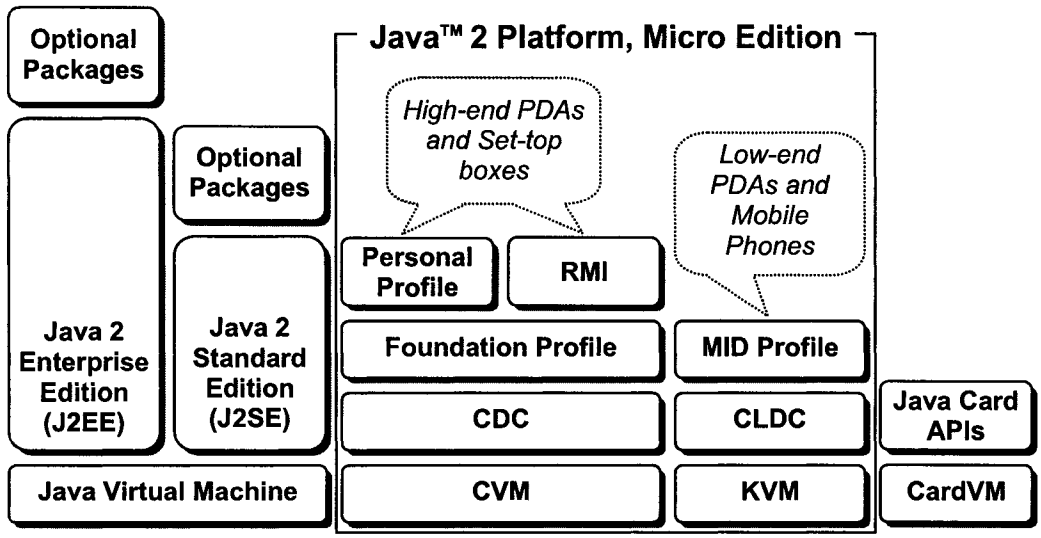


Figure 4: J2ME, J2SE and J2EE [Sul01]

3.2.3 Basics of the J2ME Architecture

We mentioned earlier that J2ME is Sun Microsystems' version of the Java runtime environment destined for mobile devices with limited hardware resources. The J2ME architecture has been designed to be both small and modular to cater for the varying level of diversity ranging from the low-end to the high-end devices. Consequently, to achieve modularity and scalability, the J2ME environment provides different JVMs to service the

diverse range of devices with different processor and memory capabilities. J2ME's modularity relies on *configurations* to maintain maximum portability, and on *profiles* to target specific devices [Gig].

Configurations are targeted towards a horizontal group of devices that have similar memory constraints, user interface requirements, network capabilities, among other such characteristics [Gig]. It is the minimum platform (VM and core Java classes) that will support a relatively broad range of similar devices (e.g., low- and medium-end PDAs, and cellphones could be placed in the same configuration because they have similar constraints). J2ME supports two configurations [LG-INTRO]: CDC (Connected Device Configuration), for capable devices such as set-top boxes and high-end PDAs; and CLDC (Connected Limited Device Configuration), for constrained devices such as low-end PDAs, cell phones, pagers, and smart cards.

On the other hand, profiles function on top of a configuration and will not work without the appropriate underlying configuration. Profiles target devices in a specific vertical market (e.g., the MIDP is part of the CLDC configuration and targets low-end cellphones, while the Foundation Profile is part of the CDC configuration and targets high-end PDAs) and contain the Java classes that focus on specific implementations such as user interface components and records management (i.e., where and how to store persistent data). In other words, profiles extend the functionality of or the availability of APIs to an application. On top of fundamental *profiles* like Foundation or MIDP, there can be optional profiles such as the PDA profile (to provide APIs for file connection and personal information management). An *optional profile* is a set of APIs providing support for additional behaviors that do not really belong to any specific configuration. For example, although a few applications may need the Bluetooth profile, which adds Bluetooth support to CLDC-based applications, others may not need it at all.

The MIDP resides on the CLDC, which functions on top of Sun Microsystems' KVM. The latter is a lightweight and compact Java Virtual machine designed for mobile devices that are small in size and limited in resources. Figure 5 below shows the overall relationship between the OS, KVM, CLDC, and MIDP in the J2ME environment.

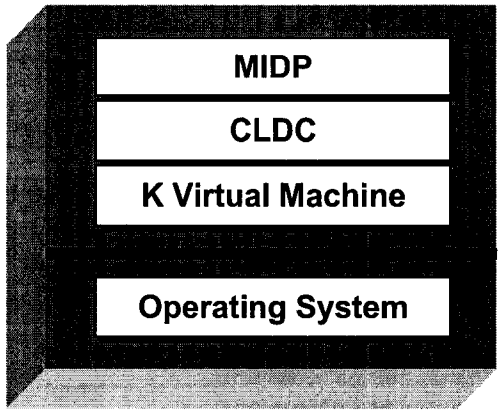


Figure 5: Java™ 2 Micro Edition MIDP Architecture

3.2.4 The MIDP Sandbox security model

The J2SE security mechanisms are not suitable for CLDC/MIDP-enabled devices due to the amount of memory required by these mechanisms [Mah02]. As a result, a few compromises were made while defining the security model for CLDC/MIDP. For simplicity, the focus is on two areas, namely:

- *Low-level KVM security*: Any MIDP application downloaded to the device and executed by the KVM must not cause any harm to the device on which it is running. In J2SE, the JVM's class verifier is responsible for rejecting invalid class files. The same should apply to the KVM; however, the class verification process is resource-expensive and time-consuming. Therefore, most of the verification work has been shifted from the device either to a desktop, where class files are compiled, or to a server from where the applications are downloaded. This off-device class verification process is called *preverification* and is performed by a *preverifier*. Essentially, the device is only responsible for running a few checks on the preverified class files to ensure their verification credibility and validity. The KVM rejects any invalid or non-preverified classes.
- *Application-level security*: The class verifier's security is limited to verifying that a given Java class file is valid but nothing more. Security threats, such as access to external resources (e.g., the file system), infrared devices, and network will go unnoticed by the verifier. In J2SE, access to such external resources is controlled by

the *SecurityManager*, *access controllers* and *security policies*. But, since this model is too memory-consuming to be included in CLDC/MIDP devices, the sandbox security model addresses application-level security issues.

The KVM provides a sandbox security model that is different from its J2SE counterpart since it does not control access through the SecurityManager or security policies. As a result, wireless Java applications can be run securely in a closed environment, called the sandbox, achieved by eliminating those features of the Java language that would pose security threats in the absence of the complete traditional Java security model. The three most significant features that were eliminated for security reasons are [Mah02] Java Native Interface (JNI), user-defined class loaders, and thread groups or daemon threads.

3.2.5 Data persistence in MIDP

In comparison with J2SE, J2ME/MIDP has very limited database options to cater for data persistence. Of major concerns, there are no APIs, like the *java.sql* package, in MIDP. The Record Management System (RMS) is one option. However, it cannot store large volumes of data and is inefficient for data retrieval due to the absence of indexing. To avoid these shortcomings, we had to consider MIDP-based third-party libraries to provide the conventional database functionalities in MediCol. To this end, the following solutions exist:

- CodeBase [CB-HSDP]: is a high-speed database engine that can be used to create robust database applications for constrained systems. However, CodeBase currently supports only devices with the Linux operating system.
- PointBase [PBM-D]: is a platform-independent relational database optimized to run on the J2ME/MIDP platform. It has an ultra-compact footprint (<45KB) that can be easily embedded within a mobile Java application. Nevertheless, PointBase is not available for evaluation and is quite expensive.
- DB2E [DB2E]: is a synchronization solution that extends enterprise data to mobile devices. Although it is platform-independent except for the Palm OS, its business logic is very resource-intensive. For instance, it first has to synchronize with a server to get all required data in a RMS and then manipulate the data on the device using

this RMS. After all modifications, the RMS is synchronized back into the server.

- Oracle Database Lite [ODL]: allows mobile users to access enterprise data even in the absence of a network connection. Additionally, it uses data synchronization to allow such users to reliably and securely exchange data with a corporate Oracle Database. It has the same shortcomings as DB2E despite being fully platform-independent.
- Sybase Anywhere Studio [SiAMD]: provides data management and enterprise data synchronization. However, it can currently generate databases only for CDC. It resembles Oracle Database Lite and DB2E; therefore the synchronization may prove very resource-intensive even if there were a package for CLDC.
- Mimer SQL Mobile [MSM]: is a small footprint relational DBMS for mobile devices. It supports standard SQL in multi-user mode. The size of data stored in the database is reduced significantly using very efficient compression techniques. Currently, it only supports platforms like Pocket PC and Symbian. Its main advantage is that it is available for free.

Based on factors like support and availability (price), suitability, and applicability in that order, Mimer SQL Mobile was used as our back-end database on the mobile device. Although, it can be argued that this database package is not fully platform-independent since it does not support the Linux and Palm OS, it has to be noted that finding a fully platform-independent database component is very difficult, nearly impossible, and quite expensive. Moreover, it does not use the synchronization mechanism.

3.2.6 Programming with J2ME

The MID Profile extends the CLDC, and thus inherits the CLDC Application Programming Interfaces (APIs). MIDP defines an application lifecycle model similar to the applet [Gig]. A MIDP application is referred to as a *MIDlet*. Similar to an applet extending the *java.applet.Applet* class, a MIDP application's entry point is a class X that extends the *javax.microedition.midlet.MIDlet* class. The latter defines abstract methods that X must implement; these methods are then called by the system to notify MIDlet X of its state changes. For instance, whenever X is launched, its *startApp* method is called. Clearly, this resembles the behaviour of applets. A group of MIDlets can be packaged

and installed on a device in the form of a MIDlet Suite, and can be removed only as a group. One or more MIDlets are packaged together into a suite, consisting of a standard JAR (Java archive) file and a separate file called a Java Application Descriptor (JAD). All the user-defined classes required by the suite's MIDlets must be in the JAR file, along with any other resources (like images or third-party JAR libraries) that the MIDlets use. The JAR file must also include a *manifest* with a number of MIDP-specific entries that describe all the MIDlets in the suite. The JAD file contains similar information, and is used by devices to obtain information about the suite prior to downloading and installing it. J2ME is a lightweight and compact version of J2SE, thus a programmer must be aware of the following important issues while developing J2ME-based applications.

- A growing number of devices from different manufacturers with support for J2ME have been found on the market; this implies that the programmer should be aware of what set of devices the application is targeting.
- J2ME-enabled devices have wireless networking, simple user interfaces and persistent storage for application-relevant data on the device. These properties may differ from one device to the other, and the developer can choose to take advantage of them in different ways. Additionally, mobile devices are always with the user, encouraging the development of mobile applications that can be customized to a user's needs.
- There are Java development tools available to mobile Java developers such as SDKs provided by many device manufacturers, and Integrated Development Environments (IDEs) like Eclipse and WSDD. EclipseME is an Eclipse plug-in that facilitates the development and testing of MIDP-based applications using Sun Microsystems' VM and emulators. WSDD is an IDE from IBM for developing MIDP-based applications and testing them on the IBM emulator, Palm simulators, or real devices (like Pocket PC handhelds and phones) using IBM's J9 KVM. Sun Microsystems provides the J2ME Wireless Toolkit that comes with various examples, CLDC/MIDP documentation, and a customizable environment emulating the behavior of different applications on a variety of devices.

3.3 JXME: JXTA on J2ME

JXTA is not only popular for the development of P2P applications for desktops and laptops, it is becoming more and more fashionable for resource-constrained devices like handhelds and sensors. For instance, JXME is the corresponding JXTA platform, albeit with minimal functionalities, for PDAs, mobile phones, and sensors. Such devices are becoming more affordable and are more likely to interoperate with each other in the absence of a coordinating authority such as a server [MM02]. Currently, JXME-based applications are developed using J2ME/MIDP, whose security model has some serious implications.

JXME is a lightweight and stripped-down version of the JXTA platform. To interoperate with each other while coping with the absence of servers, there is a need for a P2P technology like JXTA on top of the hardware abstraction level of IrDA and Bluetooth. This will result in JXME-enabled mobile peers, running on various platforms, to share data with each other and use the functions of other peers. For instance, with new communication technologies like Bluetooth and WLAN, a new level of health support can be achieved [MM02]. Peers, such as health professionals, patients, and supporting employees, can together form a special peer group, albeit with different access rights, to exchange patient records or other kinds of information. Since such information is often critical, the underlying P2P platform must be reliable. JXME ensures reliability and robustness of the system by eliminating the static IP address requirement for data transmission and enabling communication through firewalls and NATs. JXME-enabled peers rely on a JXTA relay configured to process the resource-intensive tasks that are usually performed by the rendezvous and relay peers.

3.4 Developing a collaborative JXTA application

This sub-section reviews the sequence of steps that should be followed while developing a collaborative JXME application. More specifically, we look at MediCol as an example application of this type to illustrate the development process and any important issues that need to be considered.

1) What needs to be downloaded, installed, and set-up?

The JXTA relay should first be configured properly to buffer incoming messages and relay or forward them to the addressed JXME clients. The latter simply connect to the relay and send messages to selected peer groups or to other peers. Besides messages, JXME clients can also send requests to join a group and search for or create resources such as peers, groups, and pipes. On the other hand, the jxtacldc.jar file is used to program the mobile P2P clients.

More than one way exists to run a JXTA relay. The easiest one, though, is to use the JXTA shell. JXTA shell 2.3.6 can be downloaded from http://download.jxta.org/index.html and unzipped in a particular folder. After navigating to the \jxtashell2.3.6\shell folder, the relay is started by double-clicking the run.bat or run.sh file. Initially, a JXTA configuration window is shown. Figure 8-Figure 11 below show how the shell is configured to act as a relay for JXME applications. If the shell somehow needs to be reconfigured, one should simply delete the *.jxta* folder to erase the previously saved configuration.

The main functions that have to be carried out by a mobile peer to be connected to the JXTA P2P network are shown in Figure 6, where the method invocation is given in square brackets. The given sequence will be the same for any kind of application; however, the parameters for the steps are specific to MediCol. A single JXTA relay can support any number of peer groups and each student can join any number of peer groups by repeating the steps 4-9.

2) Development environment for project starter: which library files are needed?

The EclipseME Eclipse plugin provides a very convenient and affordable IDE for the development of CLDC/MIDP applications. The JXME project home page (http://jxme.jxta.org) offers easy access to the JXME source code, jar file, and demo programs. The version 2.1.1 of JXME, being stable, is downloaded and unzipped in a particular folder. For the creation of JXME applications, we need to use the jxtacldc.jar file from one of the demo folder's lib folder (e.g., "C:\jxme2.1.1\midp\demo\chat\lib"). However, we may also use the JXME source files available under the "C:\jxme2.1.1\midp\src\net\jxta\j2me" folder by adding them to our EclipseME project. The emulator offered by the J2ME Wireless Toolkit is matured enough to be used for

executing the application.

3) Identify the types of peers, groups, and floor control for the new application

The collaborative application should allow peers to discuss privately as well as within a group. Multiple types of peers may exist, in this case, an instructor and a student. Each peer should subscribe or listen to two pipes, namely the *MediColPgrp* pipe and a *JxtaUnicast* pipe. While the latter is used for private interaction, the former is a *JxtaPropagate* pipe that will be used for group communications among the peers. A special group, *MediColGroup*, is created to limit the scope of discovery. The collaborative application should offer some sort of floor control to adapt its GUI according to the role of each peer. For example, only the instructor has the authority (functionality) to prompt a student for an answer to a question.

4) Implement a workflow for smooth handling of interactions and resolve user interface issues to render application-specific data to the peers

By continuously polling the relay for messages, the peers will be able to receive both private and public messages from other peers. A workflow can thus be designed to allow the peers to ask and answer questions. The relay is expected to queue the messages in the order they were issued. Further capabilities may be added to the relay, such as archiving messages and data or implementing workflow logic to control the distribution of messages to the peers.

5) Other issues: conventional database functionalities, extended relay functionalities, appropriate use of web services, and custom-made UI items

J2ME offers very few GUI features; hence the CustomItem class should be used to create custom GUI items for appropriate rendering of the UML diagram forwarded by the relay.

The RMS might be used to save the user's profile and the relay's configurations to minimize input from the peers. We do not require any traditional database functionality in this particular application; hence no need for third-party library component.

1. Create a peer instance [*PeerNetwork.createInstance("Eleni")*]
2. Connect to the relay [*peer.connect("http://129.128.23.90:9700", state)*]

   [The relay's URL is specific to MediCol]
3. Check group existence [*peer.search("GROUP", "Name", "MediColGroup", 1)*]

   [The peer group name "MediColGroup" is specific to MediCol and all peers join this single group, whose name is chosen automatically by the application]
4. If the group does not exist, do the following:
   a. Create it [*peer.create("GROUP", "MediColGroup", null, null)*]
   b. Search for the group [*peer.search("GROUP", "Name", "MediColGroup", 1)*]
5. Join the group [*peer.join(searchedGroupId, null)*]
6. Re-create peer instance to effectively join and communicate within the *MediColGroup* [*PeerNetwork.createInstance("Eleni", searchedGroupId)*]
7. Search for UML_PRO_PIPE [*peer.search("PIPE", "Name", "UmlPgrp", 1)*]
8. If the pipe does not exist, do the following:
   a. Create it [*peer.create("PIPE", "UmlPgrp", pipeId, "JxtaPropagate")*]
   b. Search for the pipe [*peer.search("PIPE", "Name", "UmlPgrp", 1)*]
9. Subscribe/Listen to the pipe [*peer.listen(searchedProPipeId)*]
10. Create and search for the UML_UNI_PIPE as follows:
    a. Create it [*peer.create("PIPE", "Eleni", null, "JxtaUnicast")*]
    b. Search for the pipe [*peer.search("PIPE", "Name", "Eleni", 1)*]
11. Subscribe/Listen to the pipe [*peer.listen(searchedUniPipeId)*]

Figure 6: Order of operations to get connected to the JXTA network

The relay may have to be extended to include message archiving processes, so that the instructor can keep track of participating peers and the extent of participation over time.

Shall there be any need for web service components; they can be easily made available to the JXME clients by running a Tomcat server on the same machine as the relay. Appropriate client stubs will then be generated using the J2ME Toolkit.

The main operations that a mobile peer and the JXTA relay perform to enable the peer to get disconnected from the JXTA P2P network are shown in Figure 7, where the most important method invocations are given in square brackets. In the event a peer crashes or the handheld's battery is down, the peer may neither properly leave the JXTA network nor receive messages from other peers nor get assigned specific questions by the mentor. In this case, the JXTA protocol has to be extended with a mechanism that will force the JXME peers to send an acknowledgement upon receiving any message. Consequently, if it happens that a peer crashes or has its battery drained out, then it will be unable to send back acknowledgements to the relay. The latter will then be able to deduce the fact that particular peer is no more active in the JXTA network.

2. Send a "LEAVE" message to the JXTA relay [*PeerNetwork.send("MediColGroup", searchedProPipeId, "JxtaPropagate", leaveMessage)*]
3. JXTA relay modifies the "LEAVE" message and propagates it to all peers informing them of the change in the list of online peers
4. The peer leaving the JXTA network stops polling the JXTA relay
5. JXTA relay will delete the leaving peer's endpoint address, unicast pipe, and advertisement when his/her advertisement's time-to-live (TTL) expires

Figure 7: Order of operations to get disconnected from the JXTA network

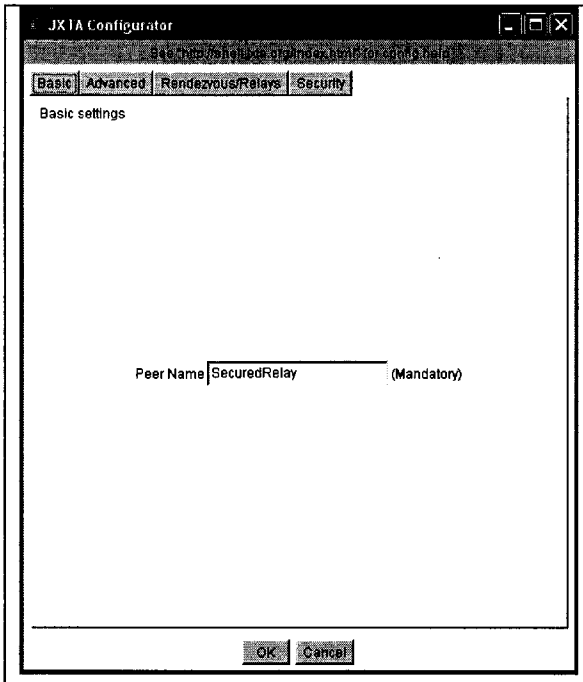


Figure 8: Shell configuration (Tab 1)

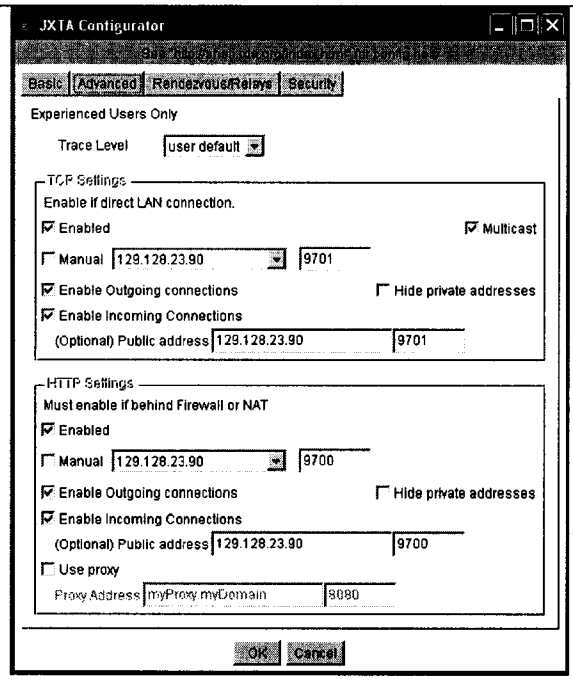


Figure 9: Shell configuration (Tab 2)

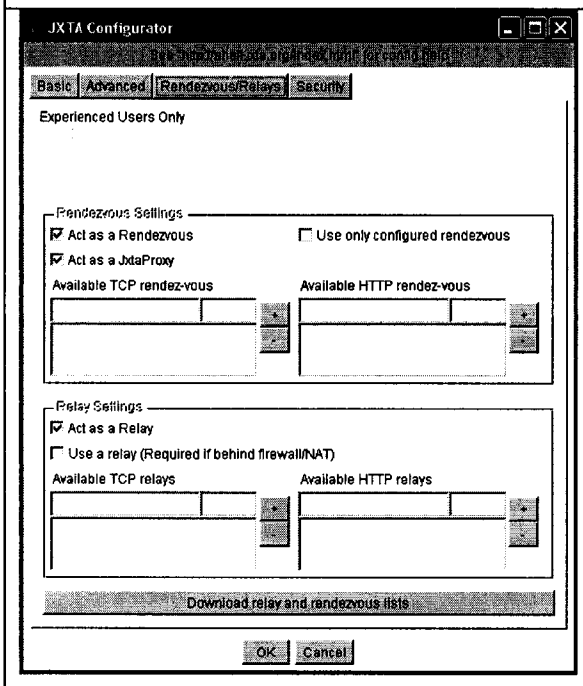


Figure 10: Shell configuration (Tab 3)

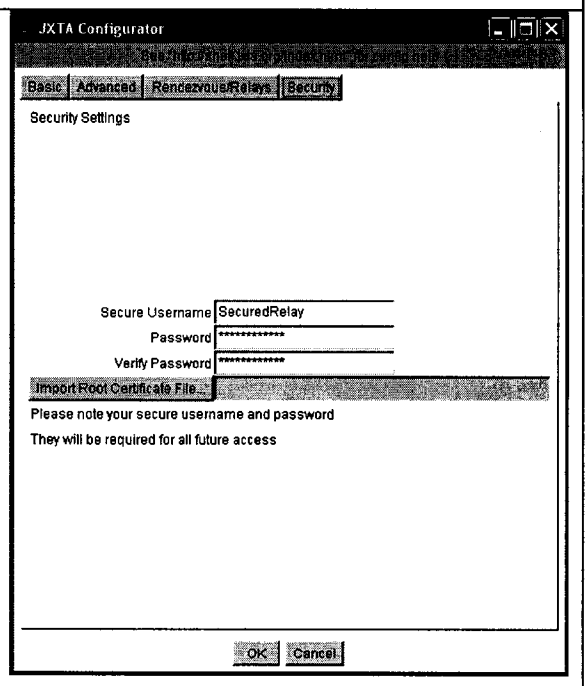


Figure 11: Shell configuration (Tab 4)

Chapter 4 MediCol: Context-based Collaboration and Portfolio Maintenance

This chapter reviews thoroughly the MediCol architecture as well as its various components. The interactions among the different components within the MediCol application are analyzed.

4.1 Architectural Description

The real-life scenario will essentially consist of a mentor and several students collaborating wirelessly via their PDAs. As we have already pointed out earlier, the JXTA relay makes the collaboration process possible. Figure 12 shows the architecture and main components of the MediCol application residing on the PDA, which consists of:

1. The *MediCol MIDlet* or User Interface Manager: initiates, creates, and manages the mobile user interface for the various medical activities that students carry out.
2. The *E-Logbook*: allows students to maintain an integrated representation of their portfolio.
3. The *JXME component*: extends the application with P2P features and allows the students to collaborate during their surgery rotation.
4. The *Forum Builder*: allows medical students to cross-reference questions and answers by creating structured and unambiguous conversational threads.
5. The *Textbook Reader*: enables peers to browse and read PDF e-books, and select a passage as context or attachment.
6. The *JXTA relay*: handles all the JXTA-related resource-intensive processing work including *data archiving* and *peer management* processes. The *MediCol* web service eliminates the need for questions or answers to accommodate large e-book passage attachments. Such attachments are sent as a request consisting of several parameters to extract the selected passage from the appropriate PDF e-book residing on the relay.

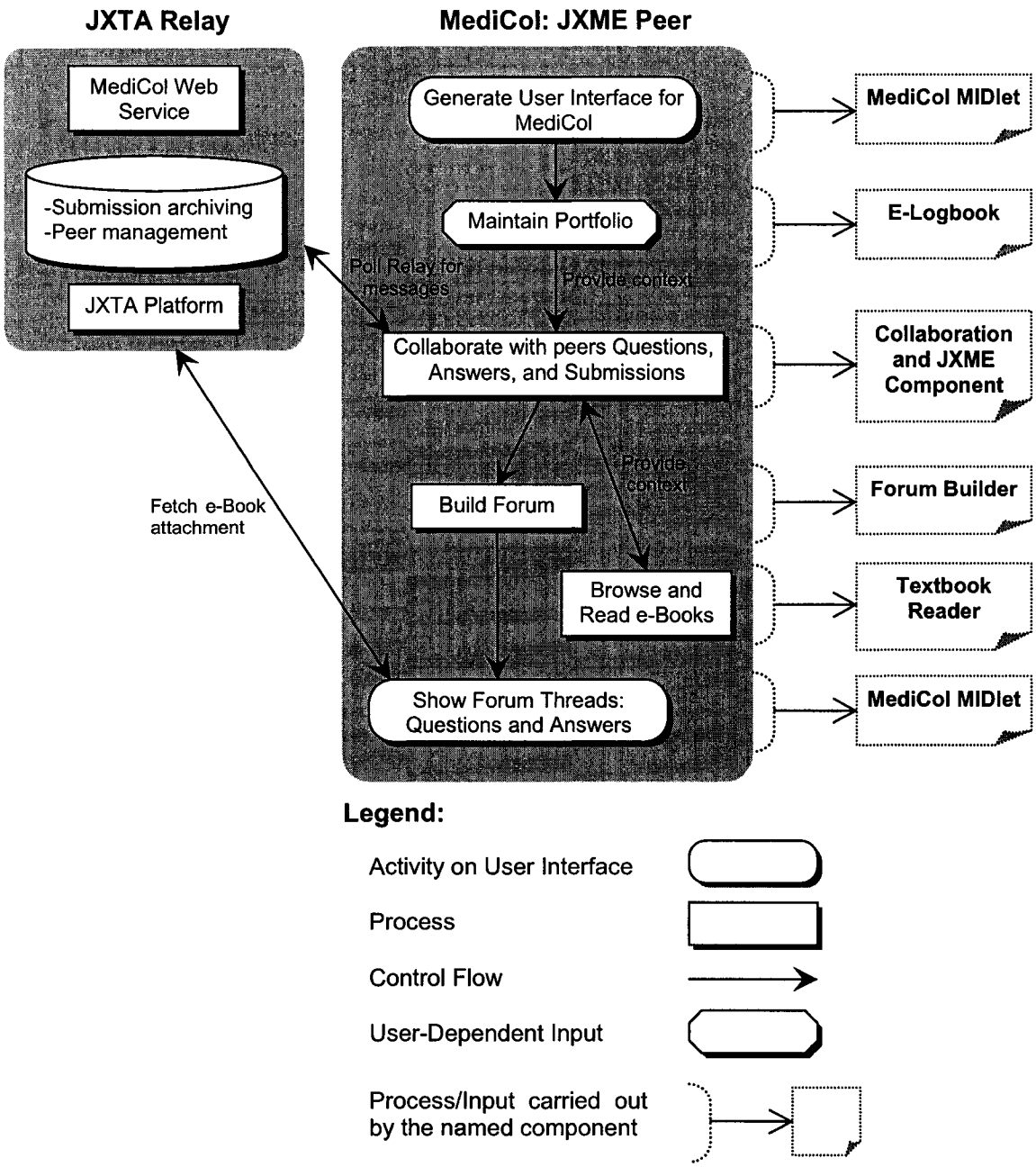


Figure 12: MediCol components and workflow

A simple sequence diagram illustrating the interactions between these main components is shown in Figure 13. The interactions between the MediCol MIDlet and E-Logbook components allow students to record their medical activities locally on their handhelds. Sub-section 4.1.2 will shed more light on this. The interaction between the MediCol MIDlet, Chat, and JXTA relay enables students to collaborate on their daily unique

medical cases, as discussed in sub-section 4.1.3. Sub-sections 4.1.3 and 4.1.4 discuss how the interactions between the Chat, Textbook Reader, and E-Logbook components allow students in providing contexts to their question and answer messages. Moreover, the interaction between the ForumBuilder and MediCol Web Service, as discussed in sub-section 4.1.5, enables students to view their e-book attachment.

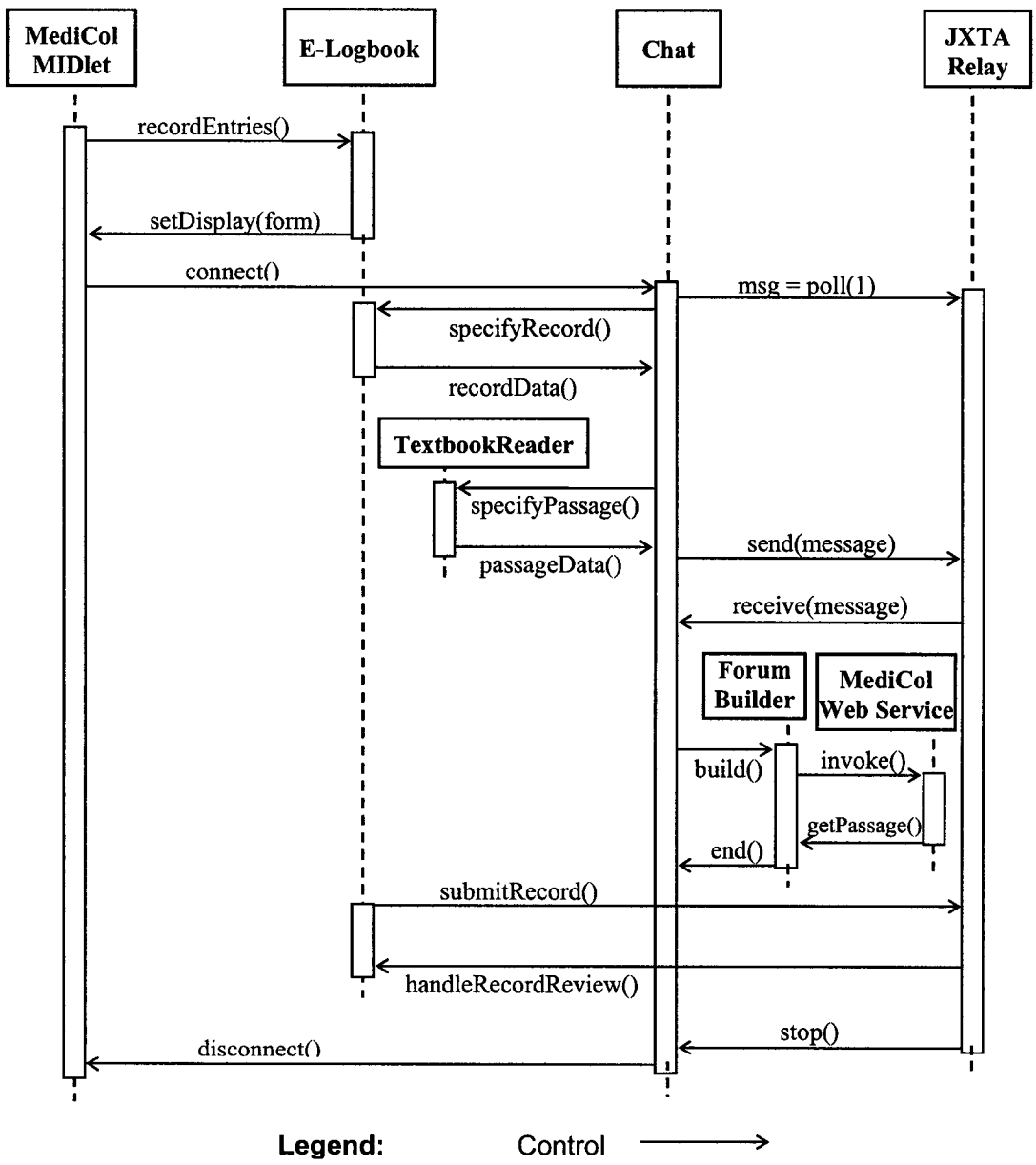


Figure 13: Interaction between main components in the MediCol architecture

4.1.1 The MediCol MIDlet

After launching the MediCol application, the MediCol MIDlet initiates the user interface by creating necessary objects and displaying the authentication form to the users. This form, as illustrated in Figure 14, allows the peers to log in the MediCol application. The "logInPeer" method is exposed from the MediCol web service as do the "logOffPeer" and "readPdfText" methods. These will be discussed in section 4.1.5. The "logInPeer" method returns the role, student or mentor, of the peer. The peer's username serves as a unique identifier for him/her and is also used for the peer to join the JXTA network. Once a medical student has logged in the MediCol application, s/he will be presented with the MediCol form (Figure 15) consisting of a toolbar at the top and the E-Logbook component. While the E-Logbook will be discussed in details in the next section, we consider the toolbar component in this section.

The MediCol toolbar is accessible in two different states, *connected* and *disconnected.* Figure 15 shows the toolbar in the disconnected state, where the student has not yet joined the JXTA network. In this particular state, no student can participate in any collaborative activities. Essentially, students can only log off from their MediCol application, use their E-Logbook component to record the medical activities they indulge in, and finally join the JXTA network to participate in various collaborative activities. On the other hand, the toolbar in the connected state is shown to students who have already joined the JXTA network. Consequently, in this state, the toolbar provides the peers with additional functionalities such as leaving the JXTA network to stop participating in collaborative activities, posting and receiving context-based messages, submitting records for review, and recording medical activities using the E-Logbook component. This is illustrated below in Figure 16. Reviewing the different messages (like questions and answers), accessing the conversational forum to view details like the contexts or attachments, and reading e-books using the Textbook reader are among the additional functionalities provided by the toolbar in the connected state.

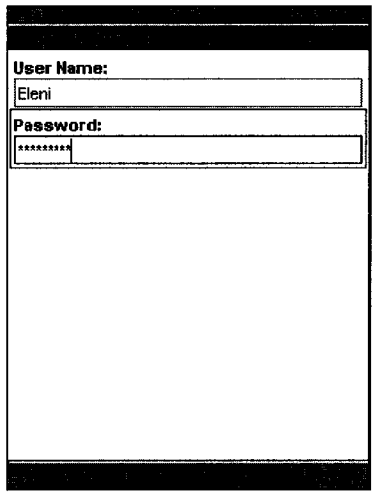


Figure 14: Authentication

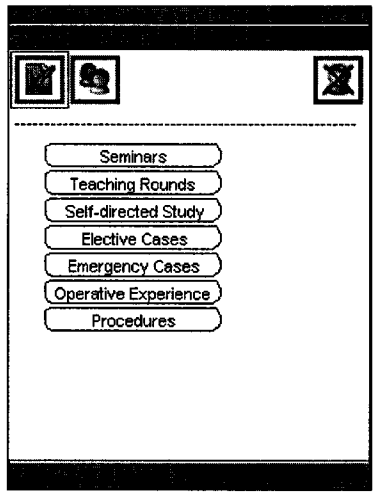


Figure 15: Initial screen after logging in

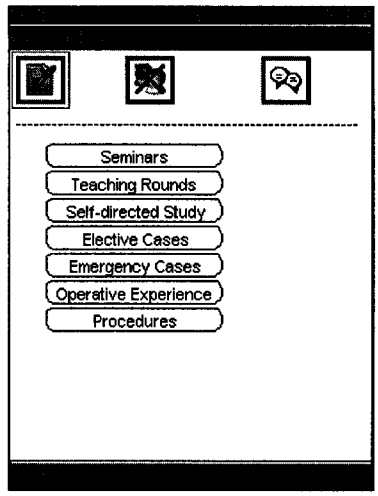


Figure 16: After joining

4.1.2 The E-Logbook

Medical students often need to maintain a paper-based notebook or logbook, which is used to evaluate the students' performance during their surgery rotation apprenticeship program. At the University of Alberta, the Surgery 546 course is dedicated to the medical apprenticeship stage. The latter requires medical students to participate in various activities and record them in their logbook. The recorded activities also need to be signed off by a surgeon or mentor. The logbook is used as the basis of the students' performance evaluation at the end of their apprenticeship program. The E-Logbook component of the MediCol application attempts to represent, as closely as possible, the paper-based notebook electronically. Among the various activities that the student has to participate in, are:

1. seminars and teaching rounds that the student has attended,
2. various related subjects that the student has studied by himself,
3. elective and emergency cases at which the student was present,
4. operative experience that the student gained while assisting with surgery cases, and
5. different procedures that the student has performed.

The E-Logbook component of the MediCol application consists of a form (an instance of the Form class) corresponding to each of the above medical activities. The forms allow medical students to add entries to their E-Logbook repository. Besides the insertion of entries to the E-Logbook database, this component also enables students to edit and delete their previous entries, submit the entries to their mentor, and use the entries as context to a conversational message. The E-Logbook database was implemented using the Mimer SQL Mobile v9.2 for the reasons presented in Chapter 1. The E-Logbook component, which includes a pop-up menu, was implemented by subclassing the CustomItem class as illustrated in Figure 17. The next few paragraphs provide more details on this.

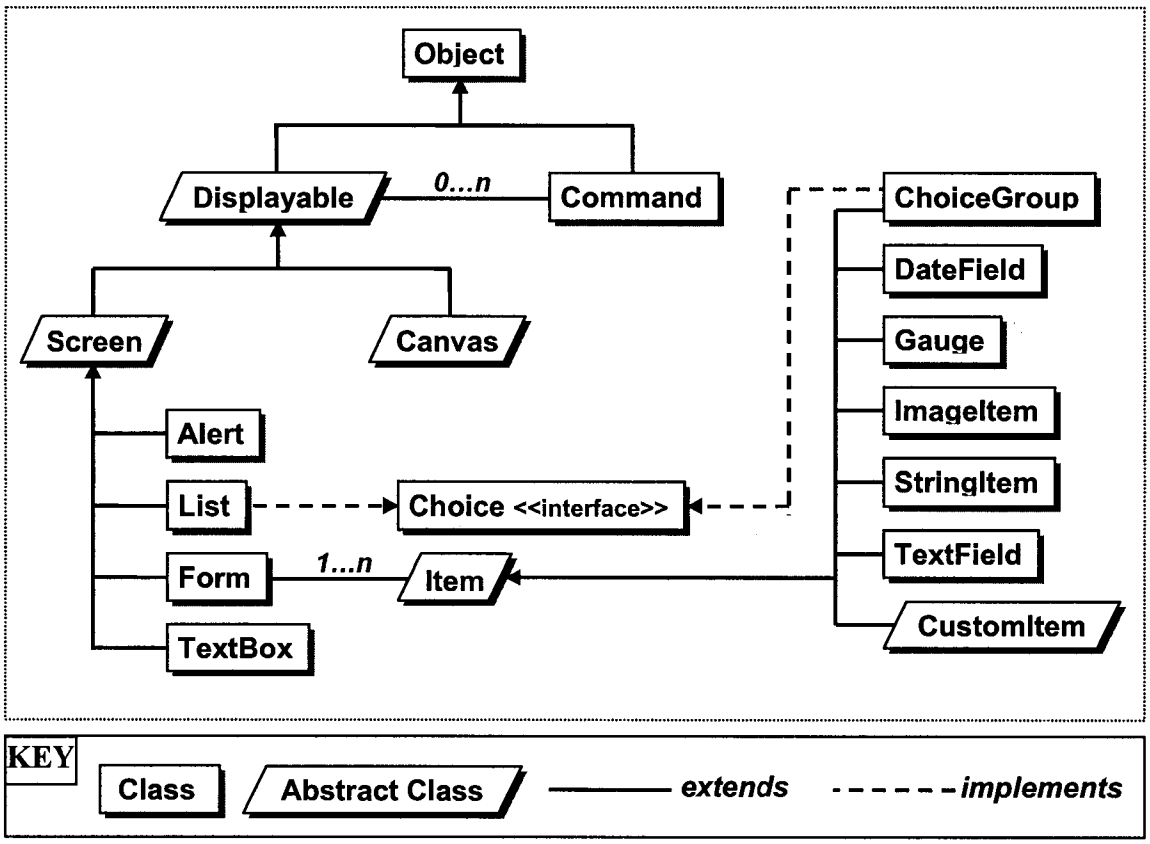


Figure 17: MIDP User Interface Widgets [OWJ]

Any MIDP application requiring custom GUI components has to create a class X that extends the CustomItem class and implements the following five abstract methods:

- *protected int getPrefContentWidth(int height)*

- *protected int getPrefContentHeight(int width)*
- *protected int getMinContentWidth()*
- *protected int getMinContentHeight()*
- *protected void paint(Graphics g, int w, int h)*

The Graphics class is used to draw text, shapes, lines and images in the newly created GUI component's content area; the *paint* method is a callback method used by the device implementation to show the GUI component on the screen. The *repaint()* method of the Graphics class is invoked to refresh and redraw the GUI component on the device's screen. The methods *getColor()* and *getFont()* of the Display class are used to understand the device's look and feel. When the custom-developed GUI item appears, *showNotify()* is called; *hideNotify()* is invoked when it is scrolled out of sight. A re-layout can be forced by invoking *invalidate()*. To interact with a custom GUI item, the following methods must be overridden to handle the generated events appropriately:

- *protected void keyPressed(int keyCode)*
- *protected void keyReleased(int keyCode)*
- *protected void keyRepeated(int keyCode)*
- *protected void pointerPressed(int x, int y)*
- *protected void pointerReleased(int x, int y)*
- *protected void pointerDragged(int x, int y)*

The first three methods are suitable for devices with several keys for input like mobile phones, while the last three ones are for devices with a stylus. To determine which of these events a particular device can generate, the *getInteractionModes()* method is invoked to handle the events appropriately.

4.1.3 JXME and Collaboration

The collaboration component is implemented on top of JXME and allows MediCol to support the natural interactions among the medical students and their supervising

physicians during their rotation. Students discuss among themselves issues related to the cases they have reviewed, the operations they have attended and the materials they have studied. During some of the activities, such as emergency and/or elective operations they attend or help perform, the mentor may also pose specific questions to a student of the group.

4.1.3.1 The JXME component

The JXME component is the lightweight stripped-down version of the JXTA P2P platform for resource-constrained devices. Although JXME performs only the minimal functionalities of the more traditional JXTA, a static relay is configured to carry out the resource-intensive tasks like queuing and routing of messages, caching of advertisements, endpoint-resolving, and message broadcasting. Composed of only five classes as shown in Figure 18, JXME allows mobile devices to connect to the JXTA network to send and receive binary-formatted messages.

A. The PeerNetwork class

This class is an abstraction for the JXTA network and specifies the operations that a JXME-based application can invoke on such networks. The following table summarizes these operations.

Table 1: PeerNetwork APIs and operations

JXME APIs	Operation
connect(relayURL, byte[0]) : byte[]	Connecting to a relay
search("PIPE", "Eleni") : int	Searching for resources
listen("MediCol", pipeID, "JxtaPropagate") : int	Subscribing to a pipe
poll(timeout) : Message	Polling a pipe for messages
send("Eleni", pipeID, "JxtaPropagate", Message) : int	Sending a message

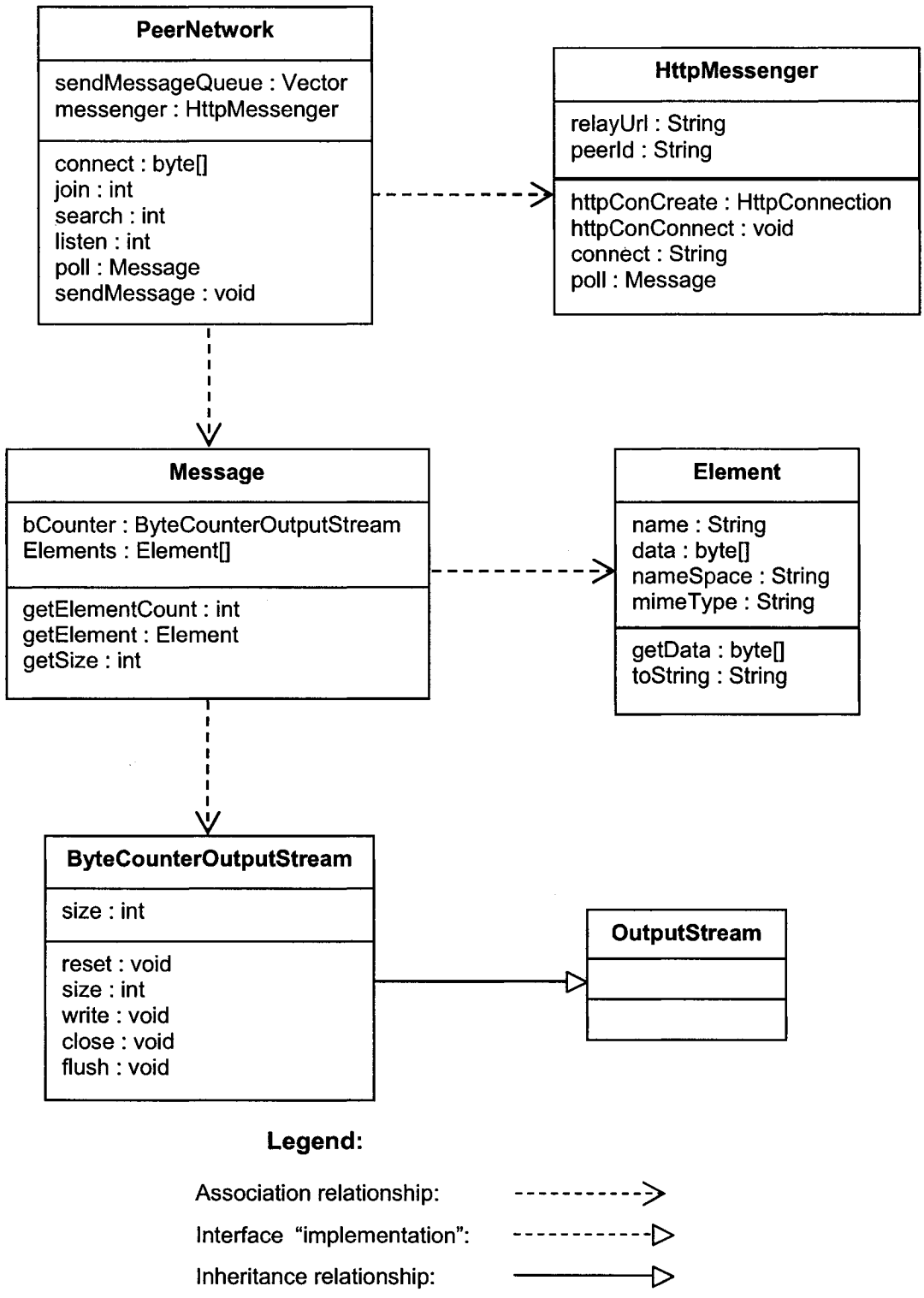


Figure 18: Class diagram for the JXME component

B. The Element and Message classes

JXME applications send messages through the JXTA network using the Message class. Comprised of at least one Element, a Message disseminates important information (chat dialogs, E-Logbook record, e-book passage, and others) between the peers in a particular peer group. The code snippet in Figure 19 shows an example of a Message consisting of two Element.

```
Element elms[] = new Element[2];
elems[0] = new Element("Query", "Should we submit assignment?", null, null);
elems[1] = new Element("Reply", "OK, go ahead and submit", null, null);
Message m = new Message(elms);
```

Figure 19: Constructing a JXME Message

C. The HttpMessenger class

It provides a messaging service for the JXME peers and facilitates the sending and receiving of messages. To receive messages, a peer first subscribes ("listen") to at least one pipe and then establishes a polling relationship with the JXTA relay server.

D. The ByteCounterOutputStream class

Extending the OutputStream class, ByteCounterOutputStream counts bytes without actually buffering the stream in memory. It enables the pre-computation of the size of a Message, for example, to set the Content-Length HTTP header in an outgoing HTTP request.

4.1.3.2 The Collaboration component

As part of the natural interactions, MediCol supports *chatting*, i.e., it enables a peer A to send a private message to another peer B by selecting B from A's list of online group members. The implication is that the application needs a way of keeping track of who is in the group and available for collaboration. JXTA's peer-discovery protocol enables peers to search for other peers but it relies on advance knowledge of the peers' names. Since it is not always possible to know the names of all the peers in advance, a more

general process is needed for maintaining awareness of the peer group. In MediCol, group awareness is maintained by the relay, whose basic JXTA-specified capabilities have been extended with special-purpose data-archiving and peer-group management functionalities. Each time a peer joins or leaves the group, the relay updates its list of online peers and broadcasts it. The relay sends a message containing the list of online peers along the output end of the peer group's propagate pipe to all the peers who, in turn, update their lists. When a peer joins the group, i.e., when it connects to the relay, it prompts the relay about whether the MediCol peer group exists. In the case of the first peer connecting, the group does not exist yet; it is created in response to the request and the first peer joins it.

Besides private chatting between two peers, MediCol also supports a threaded public discussion forum among the peers. Each peer listens to two pipes simultaneously by regularly polling them for messages. First, it listens to the input end of its own point-to-point pipe for private incoming messages from other peers. Second, it listens to the input end of MediCol's (peer group's) propagate pipe for public messages like questions and answers. Chat is implemented using point-to-point pipes. The originator of the chat selects the partner from his list of online peers, composes his message and sends it through the output end of the point-to-point pipe of the selected recipient. The message first goes through the relay before being forwarded on the output end of the selected recipient's point-to-point pipe. The recipient receives the message on the input end of his point-to-point pipe since he was listening to it.

Public discussion, implemented using the MediCol propagate pipe, connect every peer to every other peers via the relay. The public discussion forum is organized around threads that encapsulate all related messages, i.e., questions, answers and follow-up questions. Each message, whether a question or an answer, may have one or more annotations or attachments providing context for the message. Currently, the contextual attachment can be either an e-book passage or a portfolio entry. This way, apprentices can cross-reference their practical tasks and their textbook learning. All questions and answers have a globally unique ID, which is used to cross-reference related messages in a thread, i.e., grouping answers to questions and follow-up questions. Questions and answers are public and are posted by choosing the MediCol peer group from the list of online group

members. These messages are sent along the output end of the peer group's propagate pipe. To answer a question, a peer first chooses the particular question from the pending question list. If the mentor has not already assigned the selected question to another apprentice, the peer's answer is posted to the group.

The mentor may assign questions to the apprentices. To do so, the mentor first selects a question from the question list and then chooses the "Assign to" action to select an apprentice from the list of online apprentices. After the question is assigned, the selected apprentice receives a notification concerning the question assignment. The assignment message is actually sent along the selected apprentice's point-to-point pipe.

Apprentices submit their completed portfolio entries to the mentor for review. The mentor can then either approve or reject the submission. The mentor's review is notified to the appropriate apprentice, who can later edit the rejected submission(s) and submit them again. The history of all the submissions and their corresponding reviews are logged at the relay for references and evaluations.

To sum up, MediCol consists of five different types of messages (a) questions, (b) answers, (c) submissions, (d) submission acknowledgements, and (e) assignments. While the first three are subclasses of the original "Message" class in JXME; the latter two are simply instantiated as "Message". Figure 20 shows the main classes of the collaboration component, which we simply call Chat, although it supports both private and public interactions.

4.1.4 The Textbook Reader

Medical students often need to maintain electronic textbooks for their self-directed studies during their rotation. The Textbook Reader component of the MediCol application enables students to read their textbooks on their mobile devices and highlight passages as contexts to their question and answer messages. Here, we discuss a few issues related to the development of this component.

Initially, we tried to hook up the MediCol system with existing e-book readers. However, the fact that such readers were developed using C/C++ meant that JNI was required to invoke the reader from the MIDP-based (Java-based) application. Since JNI is not

supported by J2ME/MIDP, we could not use existing e-book readers and had to develop our own. Due to time limitation and the project focus, a sophisticated e-book reader was not on our agenda.

Composed of three classes as shown in Figure 21, our reader allows a student to read text-based e-books (PDF e-books are compressed using the LZW algorithm and will thus consume too much memory as well as CPU resources for decompression and rendering on handhelds). The text-based e-books may need to be partitioned if they are too large in size.

4.1.4.1 The StateManager class

StateManager, as the name implies, manages the state of the reader. It stores the position of the marker for the current textbook when the reader exits. When the students open the reader again, the StateManager retrieves the marker's previously-stored position for the appropriate textbook and sets the marker to this position. The Record Management System is used to store and retrieve the marker's position for each textbook that the student maintains on the PDA.

4.1.4.2 The TextManager class

TextManager reads the text from the appropriate textbook residing on the SDCard of a student's device and holds it in memory. It then feeds the in-memory text to the TextbookReader class for proper rendering to the students.

4.1.4.3 The TextbookReader class

TextbookReader follows an algorithm, shown in Figure 23, to render the textbook data and offer navigational facilities as part of the reader. The algorithm works as follows. Fairly standard values for the width (charWidth) and height (charHeight) are used to calculate the number of characters per line (CPL), which indicates when the reader should change line and start rendering on the next line [Line 1]. The blue marker is drawn on the top right corner of the buffered image and the variables i, k, and l are properly initialized [Line 2]. Variable k ensures that only one complete page is rendered each time

during navigation and reading; variable i keeps track of the character position in the textbook, while variable l keeps track of the position just after a word. The condition for rendering a page is that neither the height of the page (pageHeight) nor the size of the textbook is exceeded [Line 3]. If the character at i is End-Of-Line (EOL), the text position i is incremented until the character at i is not EOL [Line 4]. The algorithm then gets the character at position i [Line 5]. A check is made to determine if c is part of a word [Line 6], whereby a word consists of at least one character excluding EOL, horizontal tab, or normal space. Because c is part of a word, the position of the character just after the complete word is determined and assigned to l [Line 7]. Since we now know the length of the word, we determine the total number of characters including this word and compare this total with the CPL [Line 8]. If this total is greater than the CPL, the line is changed to the next one [Line 9]. We then store the display position for character c in a temporary variable [Line 10]. A check is made to determine if c is a horizontal tab or normal space [Line 11]. If the character at i is a space, it is rendered on the buffered image at the determined display position [Line 12]. The text position i is incremented until the character at i is neither a space nor EOL [Line 13]. The rest of the statements are skipped and execution resumes at the beginning of the while loop [Line 14]. If c is a valid character (neither space nor EOL), it is rendered at the display position, which was previously stored in a variable [Line 16]. The text position i is then incremented [Line 17]. A check is made to determine if one more valid character on the current display line will exceed the CPL value [Line 18]. The display line is then changed to the next line because the current one is full [Line 19].

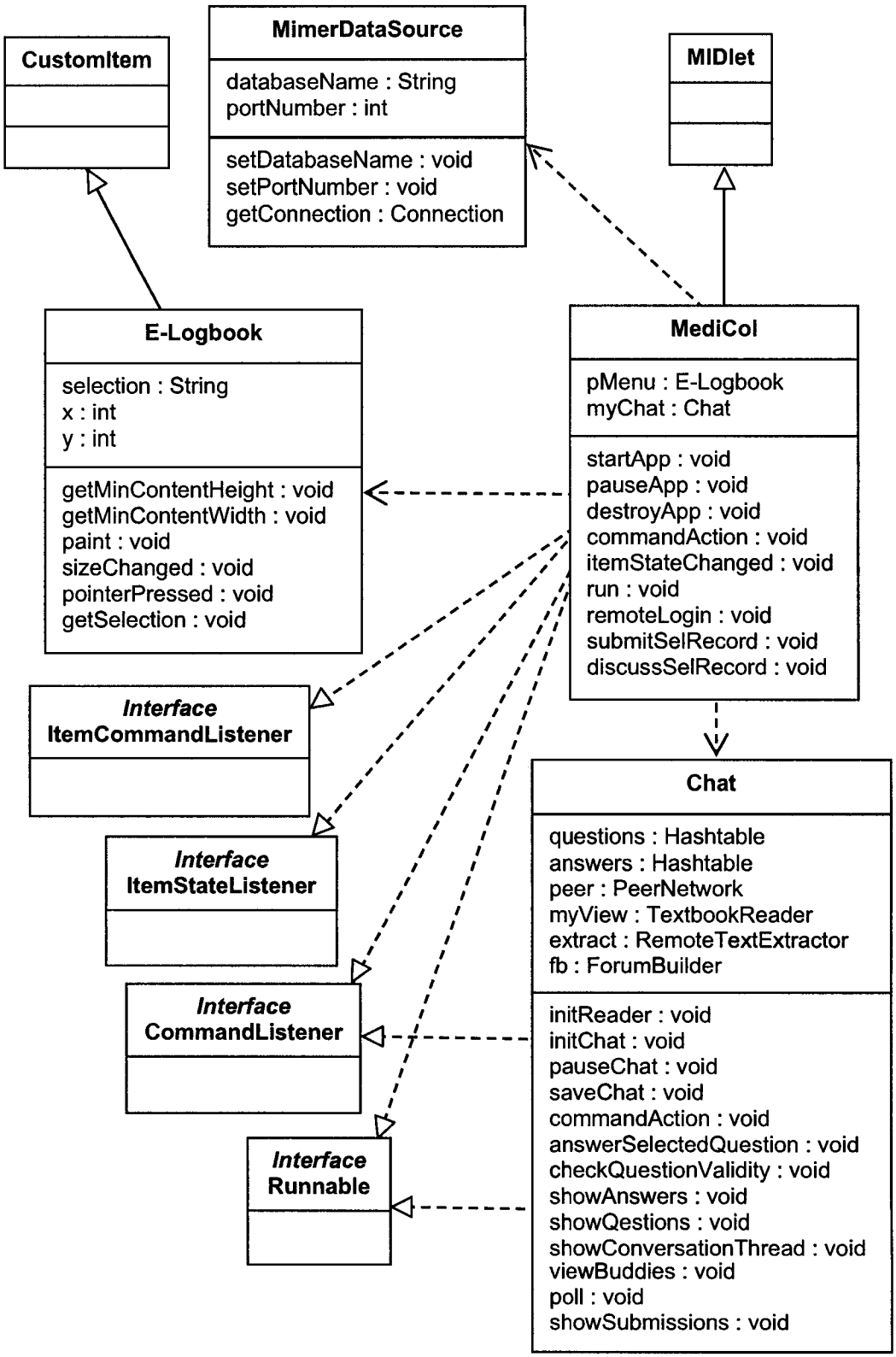


Figure 20: Class diagram for the Collaboration component

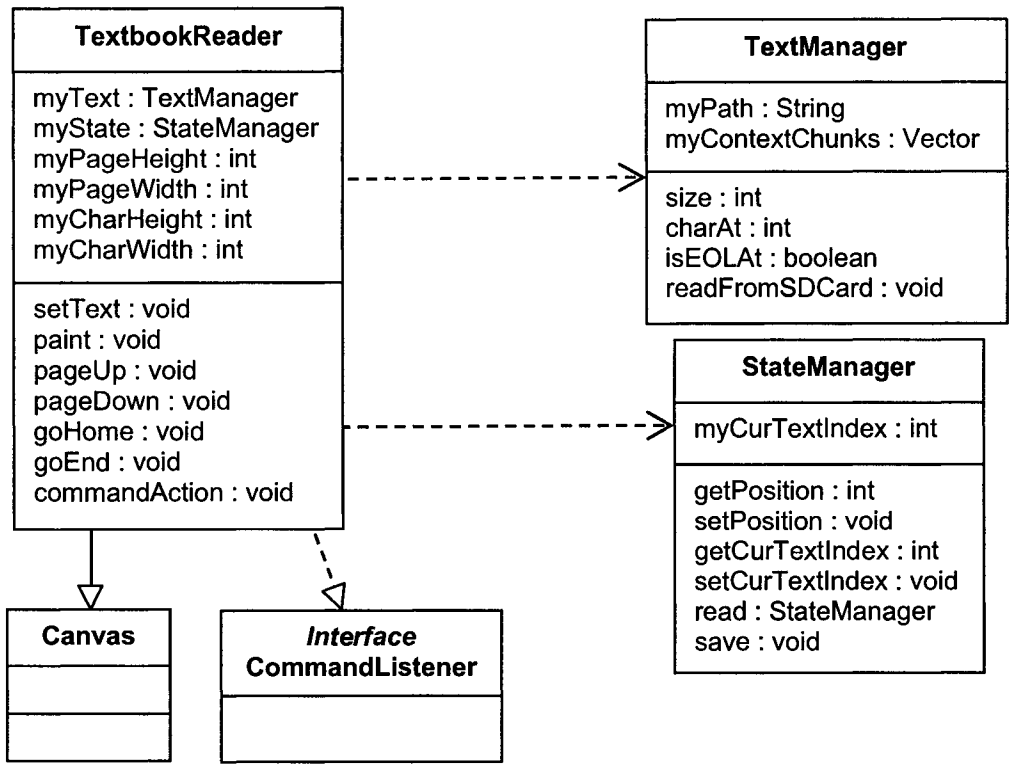


Figure 21: Textbook Reader class diagram

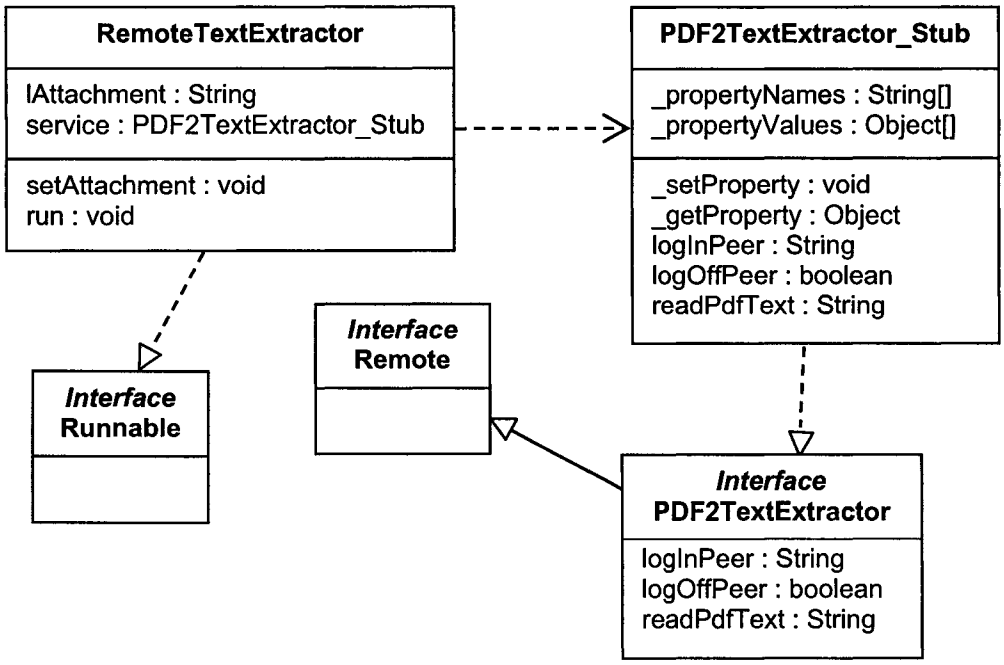


Figure 22: MediCol web service client stubs

```
1. Use charWidth and charHeight to calculate CPL (no. of characters per line)
2. Draw the blue marker on the right and Set k=i=0, l=i
3. while (k < pageHeight && i < total_textbook_size) {
4.     Determine text position until character at i is not EOL
5.     c = get character at position i
6.     if (c is part of a word) {
7.         Set l to the position of the character just after the word
8.         Check if the word fits on current line by using l and CPL
9.         Otherwise, set line to next line
10.        Determine position to display c on buffered image
11.    } else if (c is a space) {
12.        Render c at the calculated display position
13.        Determine text position until character at i is not EOL or space
14.        Continue with while loop
15.    }
16.    Render c at previously-determined display position if c is valid
17.    i++;
18.    if (current no. of characters + 1 > CPL) {
19.        k = k + charHeight;
20.    }
21.}
```

Figure 23: TextbookReader algorithm

4.1.5 The MediCol Web Service

The main motivation behind using the MediCol web service is to reduce the size of a JXME message, be it a question or an answer. Having subjected MediCol to a few experiments, it was found that the size of JXME messages is limited to approximately 1.25 KB. On the other hand, we do not want to limit the size of a passage that a student can select from a textbook. Consequently, we came up with the idea of sending messages with request parameters instead of the complete selected passage. This way, both the message size constraint and the unrestricted selected passage size vision are maintained. The request parameters consist of the first and last names of the author, the e-book title,

the starting and ending positions provided by the marker, and the e-book partition (if one exists). The MediCol web service is therefore invoked using the parameters to extract annotated passages and ship them to the peer students.

The MediCol web service also exposes methods to log students in and off the MediCol application. This way, the students' usernames are used as peer-generated IDs to connect to the JXTA network.

The J2ME Wireless Toolkit was used to automatically generate the client stubs from the WSDL file. Figure 22 shows the relationship between the generated stub classes.

4.2 Implementation Status

The MediCol architecture was tested using the J2ME Wireless Toolkit [SJWT]. MediCol uses the optional profiles JSR75 [JSR75] for accessing files stored on the SD (Secure Digital) Card and JSR172 [JSR172] for accessing web services. While the toolkit provides the emulated environment, it is not a really suitable IDE for application development. Hence, the EclipseME [EJ2MEP] Eclipse plug-in was used for application development and testing. Four different emulators are provided by the toolkit; their characteristics are summarized in Table 2 including those for the device (HP iPAQ) used for the application deployment. The heap size available to the emulators and iPAQ for application execution ranges from 512 KB to 2 MB.

Table 2: Characteristics of toolkit emulators and physical device [Mor01]

Device name	Screen size (pixels)	Canvas size	Colors	Input
DefaultColorPhone	240 x 320	240 x 289	4096	ITU-T
DefaultGrayPhone	180 x 208	180 x 177	256 Grayscale	ITU-T
MediaControlSkin	180 x 208	180 x 177	4096	ITU-T
QwertyDevice	636 x 235	540 x 204	4096	Qwerty
HP iPAQ 2750	240 x 320	240 x 320	65536	Foldable Qwerty

4.2.1 Library and API Usage Review

MediCol uses Sun Microsystems' JXTA Relay, which was extended with data archiving and peer management processes using J2SE. In contrast, the JXME Client component classes use the J2ME MIDP APIs, PDAP File Connection APIs, Web Services APIs, and the MIDP-based third-party library from Mimer SQL Mobile 9.2 [MSM]. Table 3 summarizes the use of the various packages by these sub-components.

Table 3: J2ME MID Profile, Optional Profiles and Third-party library packages

Package and API	Description
User Interface *javax.microedition.lcdui*	Features for the implementation of user interfaces for MIDlets.
Persistence *javax.microedition.rms*	Provides a mechanism for MIDlets to persistently store and retrieve data.
Networking *javax.microedition.io*	Networking support based on the *GenericConnection* framework from the *CLDC.*
Application Lifecycle *javax.microedition.midlet*	Defines interactions between MIDlets and the environment in which they run.
Core *java.io, java.lang, java.util*	System input and output classes; Language and Utility Classes included from J2SE.
Web Services [Optional] *javax.microedition.xml.rpc, javax.xml.rpc, java.rmi*	Supports the representation of client stubs and invocation and web service.
PDAP File Connection [Optional] *javax.microedition.io.file*	Support for reading and writing data persistently to files on external storage like SD Cards.
Mimer SQL Mobile [3rd party] *com.mimer.jdbc*	Provides JDBC support for MIDP applications.

4.2.2 Rationale behind using an emulator

The advantages of using an emulator for the initial testing of an MIDP application resemble those for using simulation prior to the construction of an aircraft. The UI,

memory/heap usage and leaks, input minimization, and code optimization must all be verified and properly tested for before an application can actually be deployed on a physical device. Among such fierce conditions, an emulator is absolutely necessary to identify and handle any anomalies. Through the use of an emulator, an application's behaviour can be monitored as closely as possible to that on the physical device.

4.2.3 Shortcomings of an emulated environment

Although it is recommended to use an emulator prior to deployment, it should be borne in mind that there are a few limitations associated with the use of an emulator [Mor01]:

- The emulator tries to behave the closest possible to a physical device. The behaviour of an application may differ drastically in practice.
- The emulator runs on a desktop; thus it may not properly reflect the memory constraints of a physical device. There may be a difference in the performance between the emulator and physical device, because emulators are considered to be faster than handhelds.

Despite these drawbacks, experts still consider that emulators play a significant role in the successful development and deployment of MIDP-based applications. We were no exception to this rule.

4.2.4 Code Review and Optimization

The J2ME Wireless Toolkit comes with such tools as a memory monitor, network monitor, and a profiler. In order to better understand the runtime behaviour of MediCol, we used these three tools extensively in a few preliminary experiments. The latter consist of the following sequence of activities, which were monitored incrementally.

A. Using the E-Logbook component: logging a few medical entries,

B. Using the Chat component: using the logged entries for collaboration,

C. Using the Textbook Reader: reading an e-book and adding a passage as context,

D. Using the ForumBuilder: viewing the forum threads,

E. Invoking the MediCol web service: invoking the MediCol web service to retrieve

an attached e-book passage, and

F. Regular polling for new messages: polling the relay in a normal fashion.

We present graphically the results of the aforementioned preliminary tests in the forthcoming section accompanied by useful insights and interpretations.

4.2.4.1 Experimental Analysis

The snapshots of the memory monitor, network monitor, and profiler are shown below after having performed the predefined sequence of activities. Among them, the memory monitor's snapshot is the most important one since it depicts the memory/heap usage during the execution of those set of activities.

1. *Memory monitor*: It shows an application's usage of the memory resource. Figure 24 illustrates the variation of MediCol's memory usage against the sequence of activities ranging from A to F. It is very interesting to note that the Textbook Reader and ForumBuilder consume much more memory than the other components. Besides, on average, the execution of MediCol requires around 160KB of memory. This seems satisfactory or even good since 500KB of memory were allocated for the experiment runs.

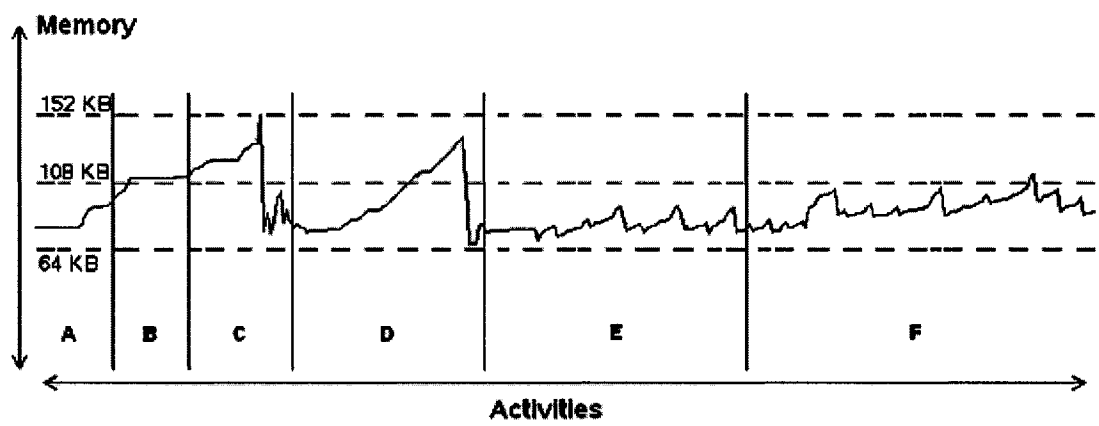


Figure 24: Memory monitor for the MediCol application

2. *Network monitor*: It logs all the requests and replies for such protocols as HTTP/HTTPS, SMS, MMS, SSL, and Datagram. This really helps programmers in visualizing the actual contents of messages, both requests and replies, noting any

anomaly, and taking appropriate course of action. For instance, initially when we attached an e-book passage of size around 2KB, the message (question or answer) was not getting delivered to the relay. The reason was that the size of a JXME message is limited to approximately 1.25KB. By using the network monitor, we were easily able to note this error and handle it properly. The network monitor for the execution of activities A through F is shown in Figure 25.

3. *Profiler*: It allows the programmer to monitor accurately the application's execution thanks to the use of a call graph, as shown in Figure 26. The profiler informs the developer of the frequency with which a method is invoked or a class is instantiated among other such information. Based on these, a programmer may decide to perform code optimization to minimize the use of a highly resource-intensive component. From Figure 26, it is quite obvious that the Private and Public chat (polling) features are executed quite frequently in a thread.

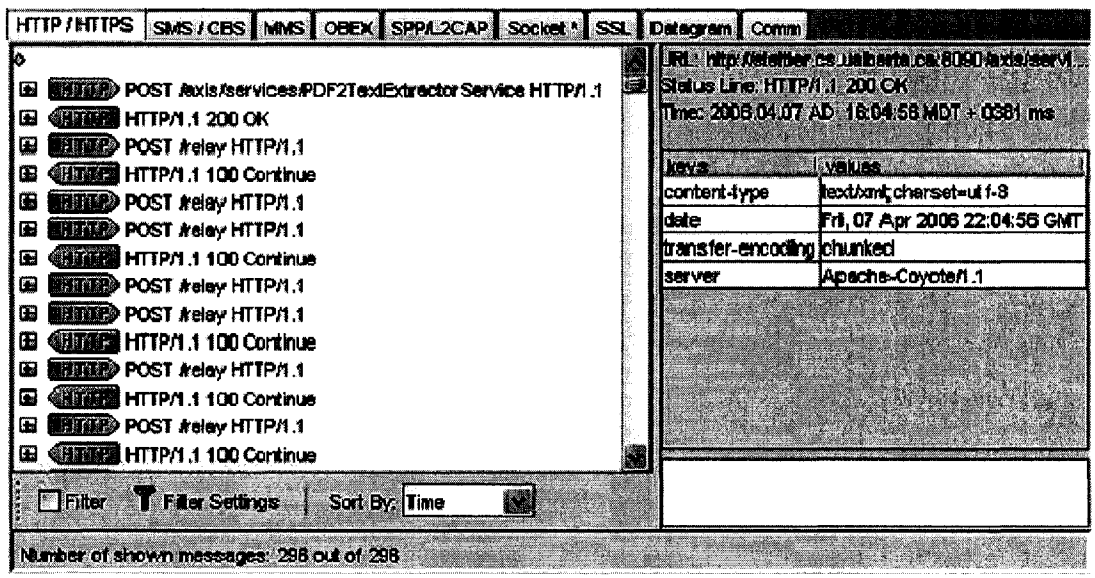


Figure 25: Network monitor for the MediCol application

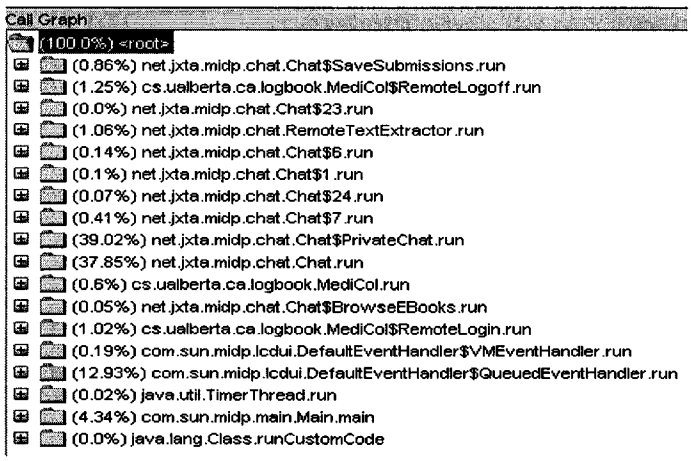


Figure 26: Profiler for the MediCol application

Chapter 5 Evaluation

As previously described, the traditional apprenticeship program of medical students can be enriched by MediCol's collaborative and interactive learning activities. However, MediCol's run-time assessment is of utmost importance for its adoption in the medical field. This chapter describes the various experiments that were carried out along with the results.

5.1 Illustrative scenario

The capabilities of MediCol are better illustrated in this section through a representative usage scenario, as shown diagrammatically in Figure 27, and reference should be made from here to the motivational scenario presented in sub-section 1.2. The next few paragraphs elaborate on how the MediCol application meets the proposed motivational scenario.

Figure 27 shows the static JXTA relay at the center and an ad-hoc group consisting of four medical students and their mentor. Three of the students, Alice, Bob and Caleb, attend a seminar on 'Acute abdomen and hernia'. They all add an entry in their respective MediCol portfolios, as illustrated in Figure 28 and Figure 29, to keep notes on the seminar and join the MediCol group to collaborate.

Alice asks a question to the group (message number 1) and adds the 'Acute abdomen and hernia' seminar entry of her MediCol portfolio as context to the question by selecting the 'Discuss' option from Figure 28. The message is sent over the output end of the group's pipe. This message goes first to the JXTA relay, which then broadcasts it to all online peers listening to the input end of the group's pipe.

Alerted by a short beep from their PDAs announcing the reception of a new message, the students are informed by the application that they have just received a new question. At this point, they may choose to inspect their current list of pending questions for a brief look at the new question as shown in Figure 30. They may also check their discussion forum for more details. The forum contains threads consisting of questions and their corresponding answers and follow-up questions. Bob views only one thread, as illustrated in Figure 31. Upon selecting the thread, Bob can see the new question as a hyperlink,

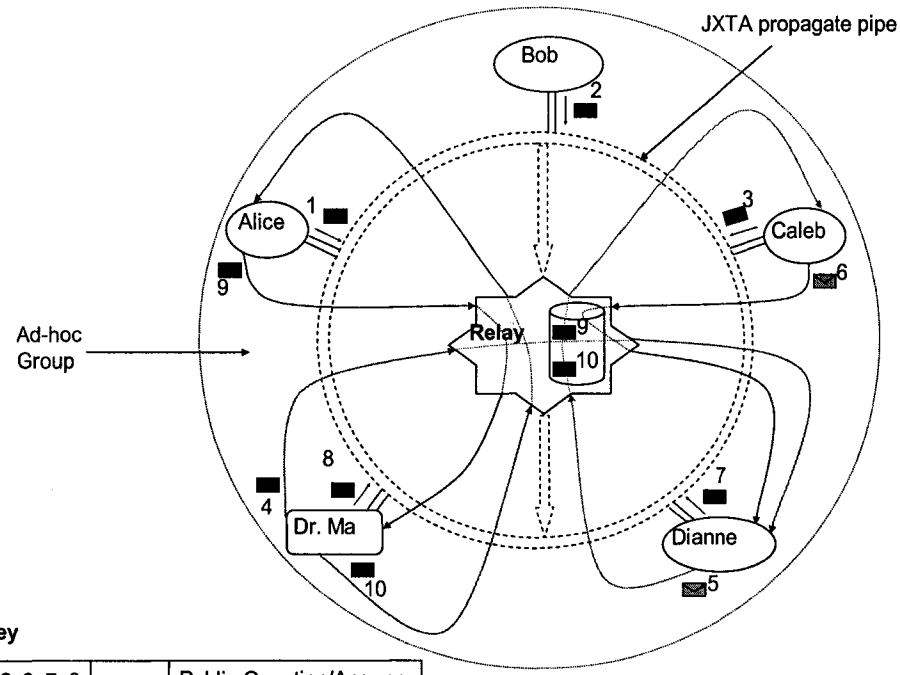
shown in Figure 32, since it has an attachment, without any answer. Bob views the attachment, Figure 33, by selecting the question with the PDA stylus. Bob thinks he has a good answer to Alice's question; therefore, he selects it and proceeds to answer it. He is confident that his answer is sufficient and does not need to be supported by any further context. Consequently, he sends his answer without attachment (message number 2).

Bob's answer message is transmitted similarly to Alice's question. Again, the application informs all connected peers of the reception of a new answer. The students open their discussion forum and select the only thread available. It contains a hyperlinked question and a plain answer, as shown in Figure 34. A few minutes later, the mentor Dr. Ma joins the peer group and another student, Dianne, follows. The relay broadcasts a message to signal the arrival of new group members. Alice, Bob, and Caleb are informed about the new group members by a short beep on their respective PDAs.

Caleb asks a question referring to Bob's answer (message number 3). This message is also sent over the output end of the group's pipe to the relay before being broadcast to all peers listening on the input end of the group's pipe. Dr. Ma decides to assign this question to Dianne (message number 4) because nobody has attempted it for quite some time now. He does so by first selecting the particular question from his list of questions and then choosing Dianne from his list of online group members. This is illustrated in Figure 35.

The assignment message is sent over the output end of Dianne's pipe to the relay. Dianne's PDA receives it on the input end of her pipe from the relay and the application informs her that she was assigned a question by the mentor. Dianne asks a quick question to Caleb privately to learn more about the earlier messages (message number 5). This message is private: the relay transmits the message from output end of Caleb's point-to-point pipe to the input end of the same pipe. Caleb receives the private message and sends an answer to Dianne (message number 6). The relay again transmits this private answer to the input end of Dianne's pipe. Only Dianne receives the answer sent by Caleb and, based on this answer, Dianne gets the gist of the discussion. Dianne then answers the question assigned to her with a reference to a passage from an e-book as context (message number 7). Figure 36 illustrates this clearly.

As described earlier, the answer message is broadcast by the relay over the output end of the group's pipe resulting in all online peers to receive the message from the input end of the group's pipe. The group members open the thread in their forum and can now view the initial hyperlinked question, the plain answer, the follow-up question, and a new hyperlinked answer, Figure 37. When they select the latter with their stylus, they can view the attached passage from Dianne's e-book, Figure 38. To enable sharing text from PDF e-books, a web service is invoked to extract the passage using the attached e-book title, author's first and last names, and the beginning and terminating byte positions of the excerpt. Dr. Ma approves the answer and replies with a positive comment (message number 8). All connected peers receive the message in the same way as described earlier.



Key

1, 2, 3, 7, 8	■	Public Question/Answer
4	■	Question Assignment
5, 6	■	Private Question/Answer
9, 10	■	Submission/Response

Message Numbers	Peers in Group
1, 2	Alice, Bob and Caleb
3, 4, 5, 6, 7, 8, 9, 10	Alice, Bob, Caleb, Dianne and Dr. Ma

Figure 27: An illustrative scenario

By selecting 'Submit' from Figure 28, Alice submits her 'Acute abdomen and hernia' seminar entry, which she made earlier in her MediCol portfolio, to Dr. Ma (message number 9). This is a private message and, therefore, the relay sends it over the output end

of the mentor's pipe since the latter is listening to the input end of the same pipe. Only Dr. Ma receives the submission. He opens his list of submissions to view all submissions. A few minutes later, he approves Alice's submission as shown in Figure 39 (message number 10). The history of the submission, message numbers 9 and 10, is saved at the relay before it is sent over the output end of Alice's pipe as the latter is listening to the input end of the same pipe. If Dr. Ma had rejected Alice's submission, he would have then been able to send a comment on why he did so along with the rejection response.

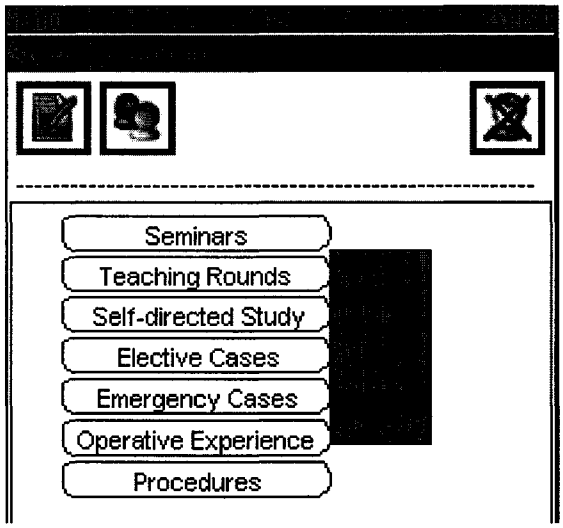


Figure 28: The MediCol User Interface

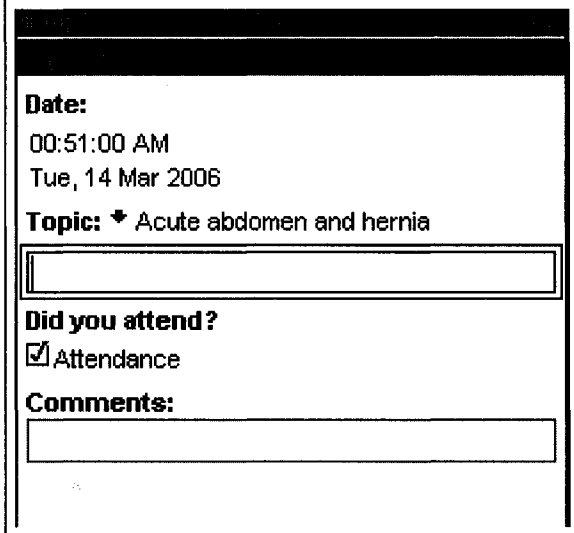


Figure 29: Alice adds a seminar entry to her portfolio

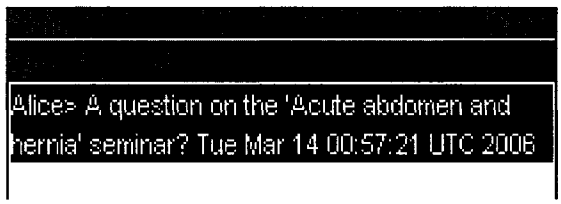


Figure 30: Bob inspects the list of questions

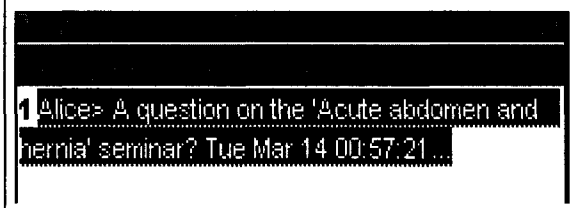


Figure 31: Bob examines the MediCol forum

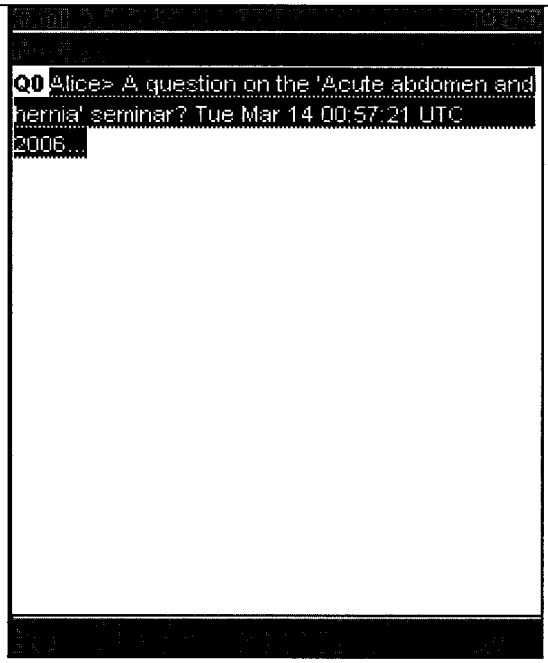


Figure 32: Bob views Alice's question in the MediCol group forum

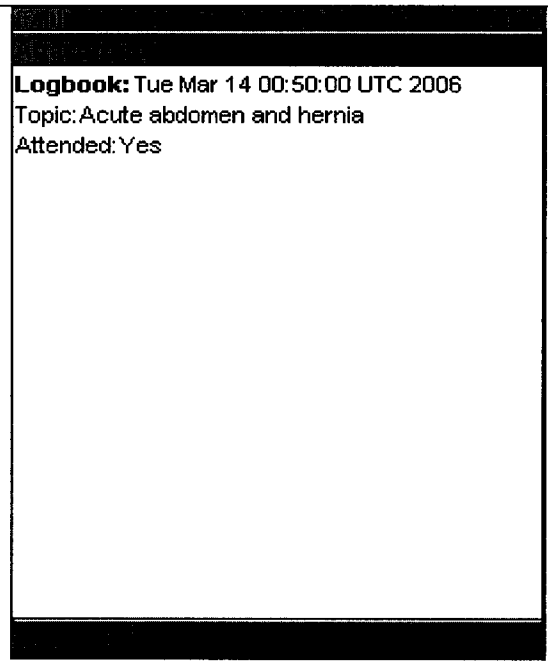


Figure 33: Bob views the E-Logbook attachment of Alice's question

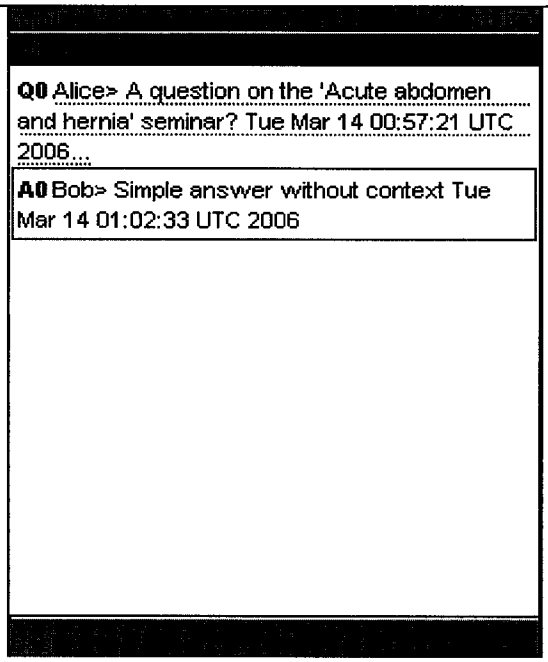


Figure 34: Bob answers Alice's question

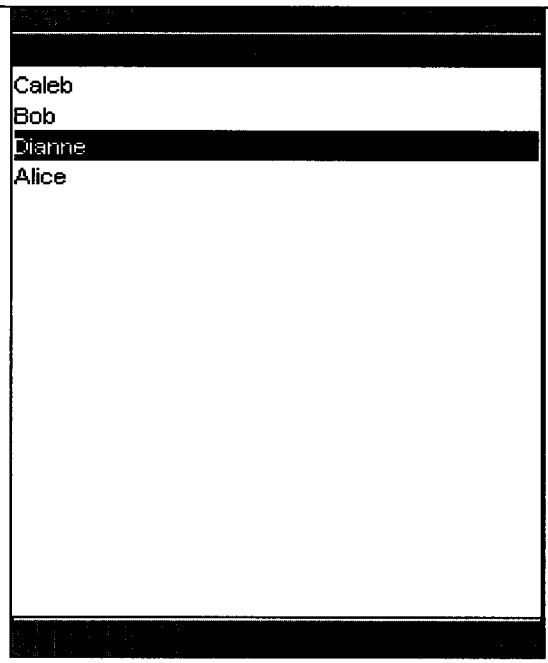


Figure 35: Dianne is assigned Caleb's question

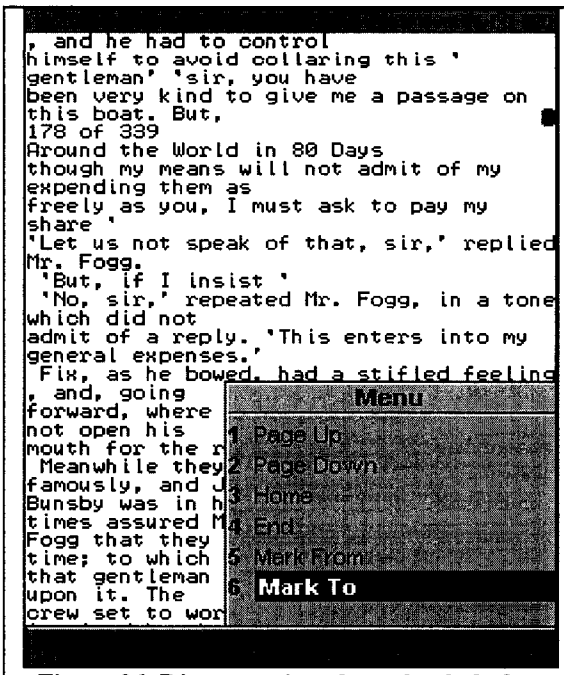


Figure 36: Dianne reviews her e-book, before answering the question

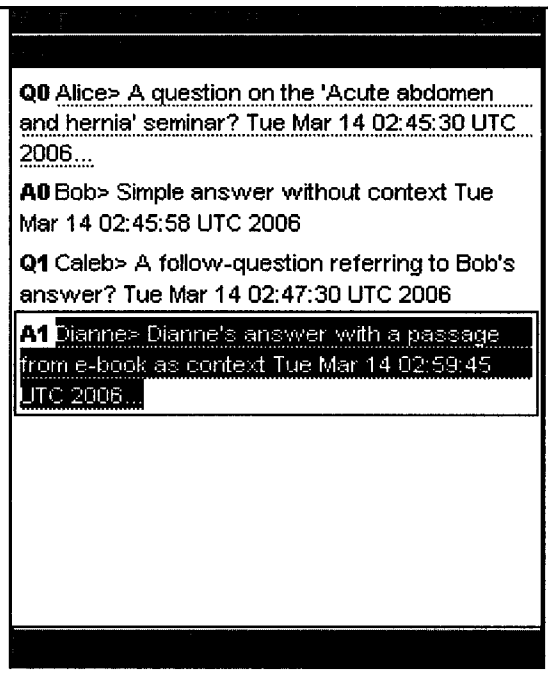


Figure 37: Dianne answers the question to the MediCol group

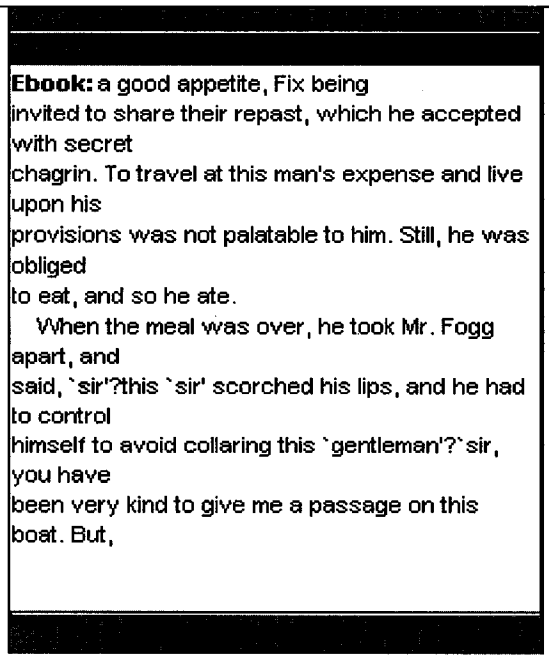


Figure 38: Dianne's selected passage as attachment

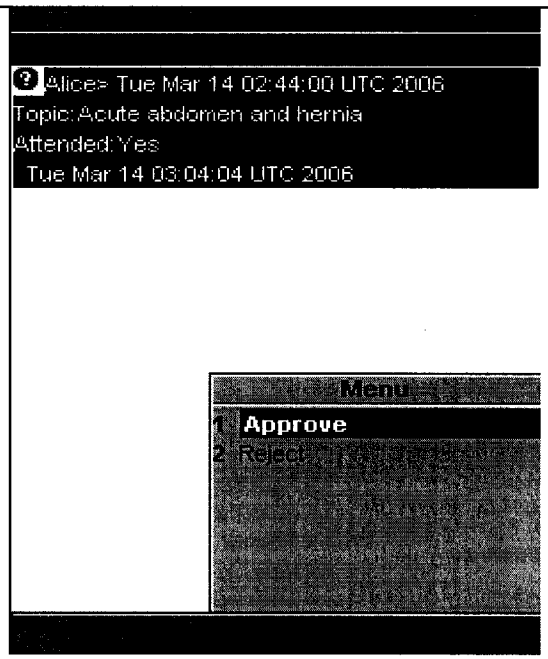


Figure 39: Dr. Ma approves Alice's entry submission

5.2 Experiments

The relevance of the work in this thesis clearly needs an in-depth evaluation with field

experimentation in order to assess the tool's usability and its straightforwardness in its integration in the context of surgery rotation. Before actually asking medical students and instructors to use the tool however, we need to make sure that its performance is acceptable, or otherwise its usability will be fundamentally threatened. More specifically, we have to make sure that messages of realistic size, sent with realistic frequency, will be received fast enough so that the on-line collaboration can be as seamless as possible.

To that end, we have carried out three quantitative experiments to examine how well MediCol scales up with the number of peers, the data throughput of the pipes, and the frequency of message exchanges. Because all messages of mobile peers – private through sender-to-recipient pipes and public through broadcast via the propagate pipe of the group - are forwarded to their destination(s) through the relay, increasing the number of peers and the number of messages exchanged results in a load for the relay's message processing and managing task. In the three experiments reported below, we examined three, increasingly complex, message exchange scenarios and we measured the average time it takes for messages to get delivered to their destinations and the number of messages that actually get delivered successfully until the queue at the relay overflows.

Essentially, we describe two experimental scenarios in the following sections. The first scenario involves peers running on emulators and is detailed in sub-sections 5.2.1, 5.2.2, and 5.2.3. The second scenario, as described in section 5.3, involves one handheld (HP iPAQ Hx2750) with the remaining peers running on emulators.

For both experimental scenarios, the following set-up has been used.

- The JXTA relay and MediCol Web Service reside on a machine with the Intel(R) Pentium(R) 4, 2.80 GHz CPU power, 1 GB RAM, and running Windows operating system.
- All the emulators were executed on machines with the Intel(R) Pentium(R) 4, 2.40 GHz CPU power, 512 MB RAM, and running Linux operating system. For instance, for experiment 1, five Linux machines were used to execute five emulators each.
- In the case of the Windows machine running the JXTA relay and MediCol Web Service components, the additional load was around 0.25% besides these two

processes. In contrast, the Linux machines running the emulators did not have any noticeable additional load.

- The polling interval is the only JXTA parameter that was used throughout the experiments. A value of one second was used as the polling interval, such that JXME peers will poll the JXTA relay every second to download messages.

The following platform-specific issues were considered for the experiments' setup:

- The startup test measures the time it takes to initialize the JXTA platform on a single peer [HD-Aug03]. It takes on average 10 seconds for a JXTA peer startup as illustrated in the script used for our experiments, in Figure 40.
- An additional warm-up period for 1,000 message acknowledgments is included to ensure that the Just-In-Time (JIT) compiler is not influencing the measurements [ATR+01][MM02]. This is performed to achieve a steady state because a slow start may be encountered due to the initial creation of objects required by the relay for processing the requests. We used around 120 message acknowledgements.

5.2.1 Experiment 1: Response time and group size

This experiment attempts to investigate the relay's performance with respect to the number of participants in the collaborative group.

5.2.1.1 Objective and experiment description

The goal of this experiment is to assess the effect of an increasing number of participants on the response time of the relay. Given the fact that the relay has to resolve the endpoint addresses of all participants before broadcasting a message, this experiment tries to estimate the transmission time of a broadcast message with an increasing number of students in the group. Although it is expected that this transmission time will increase, the actual variation is not known yet.

Essentially, the time it takes for a message of a realistic size to be broadcast to all the participants in the group is noted and averaged. This is repeated with an increasing number of peers in the group.

5.2.1.2 *Experiment design*

All the peers joined at the same time and the mentor broadcast a single 80 bytes message to the whole group. The average time (over 5 experiment runs, over all peers) taken by each peer to receive the mentor's message with an increasing number of peers (5, 10, ..., 25) was measured. The following script was used to run the simulation on Linux machines.

```
for every peer where peer ranges from 1..n
do
        start the emulator on desktop machine and wait for 10 seconds
        start the experiment (send messages, time, and average)
done
```

Figure 40: Script for running Experiment 1, 2, and 3

5.2.1.3 *Results and evaluation*

The table below shows the overall results averaged over 5 experimental runs.

Table 4: Data averaged over 5 runs for Experiment 1

Number of peers	Average transmission time (ms)				
	80 bytes	0.25 KB	0.50 KB	0.75 KB	1.00 KB
1	685	824	968	1008	1119
2	694	829	971	1010	1123
3	698	836	975	1011	1125
4	725	840	975	1014	1125
5	738	841	977	1015	1127
6	746	844	981	1017	1128
7	755	847	983	1020	1130
8	758	852	984	1021	1133
9	758	855	987	1024	1136
10	759	856	989	1027	1138
11	763	859	990	1027	1140
12	770	863	991	1028	1143

13	772	866	991	1031	1146
14	777	869	993	1033	1149
15	781	871	994	1036	1154
16	779	870	996	1037	1158
17	783	874	997	1039	1161
18	789	878	997	1042	1164
19	789	880	999	1043	1165
20	790	885	1002	1046	1167
21	792	887	1002	1047	1167
22	795	892	1004	1047	1168
23	799	893	1005	1058	1170
24	803	895	1006	1052	1174
25	807	899	1009	1055	1177

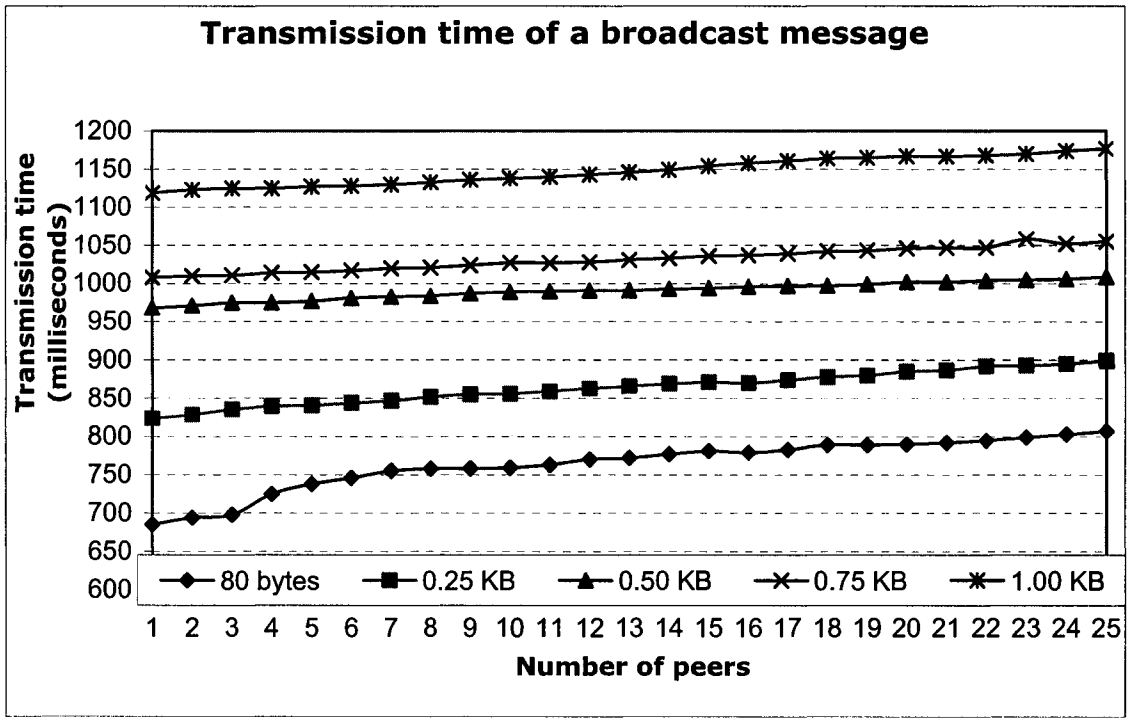


Figure 41: Average transmission time of an 80-bytes broadcast message with an increasing number of peers (variance is around 7.2 ms)

The results are illustrated graphically in Figure 41. The graph shows that the average transmission time increases as the number of peers increases. Due to the increasing number of peers, the processing load on the relay increases causing a corresponding increase in the average transmission time. However, it is interesting to note that the

average transmission time for 25 peers – which is the number we realistically expect in a cohort of surgery-rotation students - is approximately 800 ms. Although the average transmission time increases with increasing number of peers, such increase is not very significant and it seems to taper off. The same observation can be made for an increasing message size and a maximum JXME payload of 1.00 KB produces a really satisfactory communication performance (less than 1.2 seconds) with a peer group size of 25. From this experiment, it can be deduced that a peer group size of between 20 and 25 will enable very satisfactory to good communication performance within the collaboration process. According to the preliminary desktop-based JXTA performance studies done by [TEC05], firewalls and switches would have increased the average transmission time of a broadcast message regardless of the message size.

5.2.2 Experiment 2: Average data throughput of JXTA pipes

This experiment attempts to determine the average data throughputs of the JXTA unicast and propagate pipes.

5.2.2.1 Objective and experiment description

The goal of this experiment is to evaluate the maximum amount of data, in bits per second, that can be transmitted from one peer to another one (unicast) or to a group (propagate). The fact that MediCol uses both JXTA pipes quite extensively makes it even more important to estimate the data throughput of these two pipes. The experiment measures the average time needed to transmit private and group messages of particular sizes. This process is repeated for at least 10 times and the average transmission time is determined; then the average data throughput for that particular message size is calculated.

5.2.2.2 Experiment design

This experiment is set up so that two students and the mentor join at the same time. The mentor first broadcasts a 0.25 KB message to the group and then sends a private message of the same size to one of the students. The goal is to measure the average data throughput in bits per second (bps) of the JXTA unicast and propagate pipes.

We measured the average time t it takes for the reception of k-sized messages along the unicast and propagate pipes (over the execution of 10 experiment runs), where k takes the values 0.25, 0.50, 0.75, and 1.00 KB. We used 1.00 KB as the maximum message size because it seems that JXME cannot accept more than 1.25 KB as the message size, which includes the payload and several control information like the peer group endpoint address. The average data throughput is calculated as (k*8*1024)/t.

5.2.2.3 Results and evaluation

The table below shows the overall results over 10 experimental runs.

Table 5: Data averaged over 10 runs for Experiment 2

		0.25 KB	0.50 KB	0.75 KB	1.00 KB
Throughput (bits per second)	Unicast Pipe	2639	5088	7220	9503
	Propagate Pipe	2450	4201	6077	7282

The average data throughput for the JXTA pipes (unicast and propagate) with an increasing message size is shown graphically in Figure 42.

Two observations can be made here. Firstly, as the message size increases, the average transmission time also increases, however the rate of increase of the message size is much higher than that of the average transmission time. Therefore, we observe a general increase in the average data throughput for both the unicast and propagate pipes. Secondly, the average data throughput of the propagate pipe is lower than its unicast counterpart. The fact that message propagation adds an extra load on the relay, in terms of peer resolving and membership verification, explains this observation. Firewalls and switches would have lowered the average data throughput of both the unicast and propagate pipes regardless of the message size according to the preliminary desktop-based JXTA performance studies carried out by [TEC05].

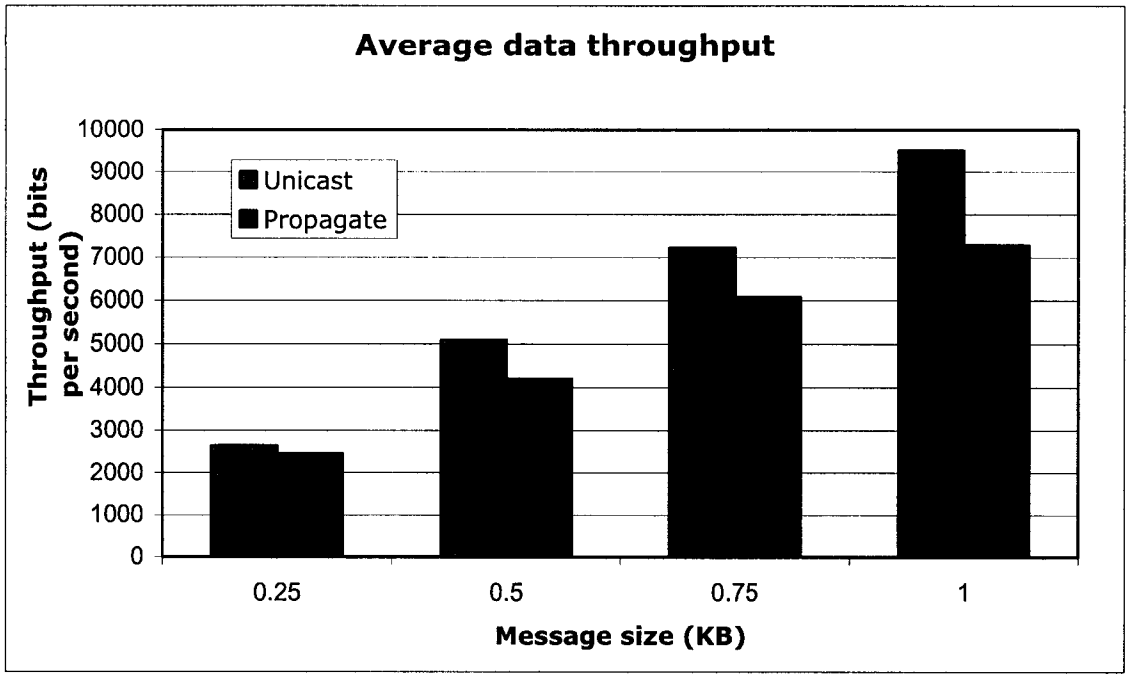


Figure 42: Average data throughput among 3 peers with an increasing message size

5.2.3 Experiment 3: Message sending rate during a real-life scenario

This experiment attempts to estimate the rate at which varying-sized messages can be transmitted to the collaborative group in real-life.

5.2.3.1 Objective and experiment description

The goal of this experiment is to approximate the rate at which a peer can broadcast or send messages infinitely to the group for every peer to receive the message successfully. Although this is not a realistic scenario, it is an added-value to have an idea of the critical limit at which the MediCol propagate pipe functions properly. The experiment measures the number of messages of varying sizes that all the peers in the group receive successfully. This process is repeated for at least 5 times and the number of messages is averaged.

5.2.3.2 Experiment design

In this experiment, all peers join at the same time. The mentor, again, broadcasts a 0.25 KB message continuously to the whole group in an infinite loop. We repeated the experiment with an increasing number of peers (5, 10, ..., 25) and message size (0.25, .50,

0.75, and 1.00 KB). Each combination of the experiment (like 5 peers for the four different message sizes) was repeated 5 times.

5.2.3.3 Results and evaluation

The table below shows the combined results over 5 experimental runs.

Table 6: Data averaged over 5 runs for Experiment 3

Number of peers	Sending rate (messages per second)			
	0.25 KB	0.50 KB	0.75 KB	1.00 KB
2	6.618	6.137	6.549	6.894
3	5.870	5.325	5.224	4.603
4	4.712	4.520	4.520	4.745
5	4.246	4.407	4.108	4.038
6	2.293	2.306	2.489	2.431
7	2.410	2.516	2.483	2.468
8	2.584	2.406	2.342	2.319
9	2.326	2.482	2.488	2.373
10	2.555	2.560	2.630	2.551
15	2.598	2.555	2.677	2.641
20	2.668	2.545	2.536	2.642
25	2.665	2.516	2.670	2.598

Figure 43 illustrates the results graphically. Note that the sending rate has to be an integer; however we have used floating-point numbers to show the variation for the different message sizes.

The results from this experiment have to be interpreted carefully. Figure 43 shows that the size of the message does not affect the sending rate too much. Furthermore, it shows that the sending rate decreases with an increasing number of peers due to the increasing cost suffered at the relay. The sending rate eventually converges to 2. This does not mean that if there are 20 peers in the group and three peers post three different questions simultaneously to the group, then only two of the three question messages will make it. Because these peers are sending only a one-off message, in the form of a question, as

compared to the above scenario where a peer is sending a message infinitely to the group. Based on the preliminary JXTA performance studies for desktops carried out by [TEC05], firewalls and switches would have lowered the sending rate for all the various message sizes used as well as for the various size of the peer group.

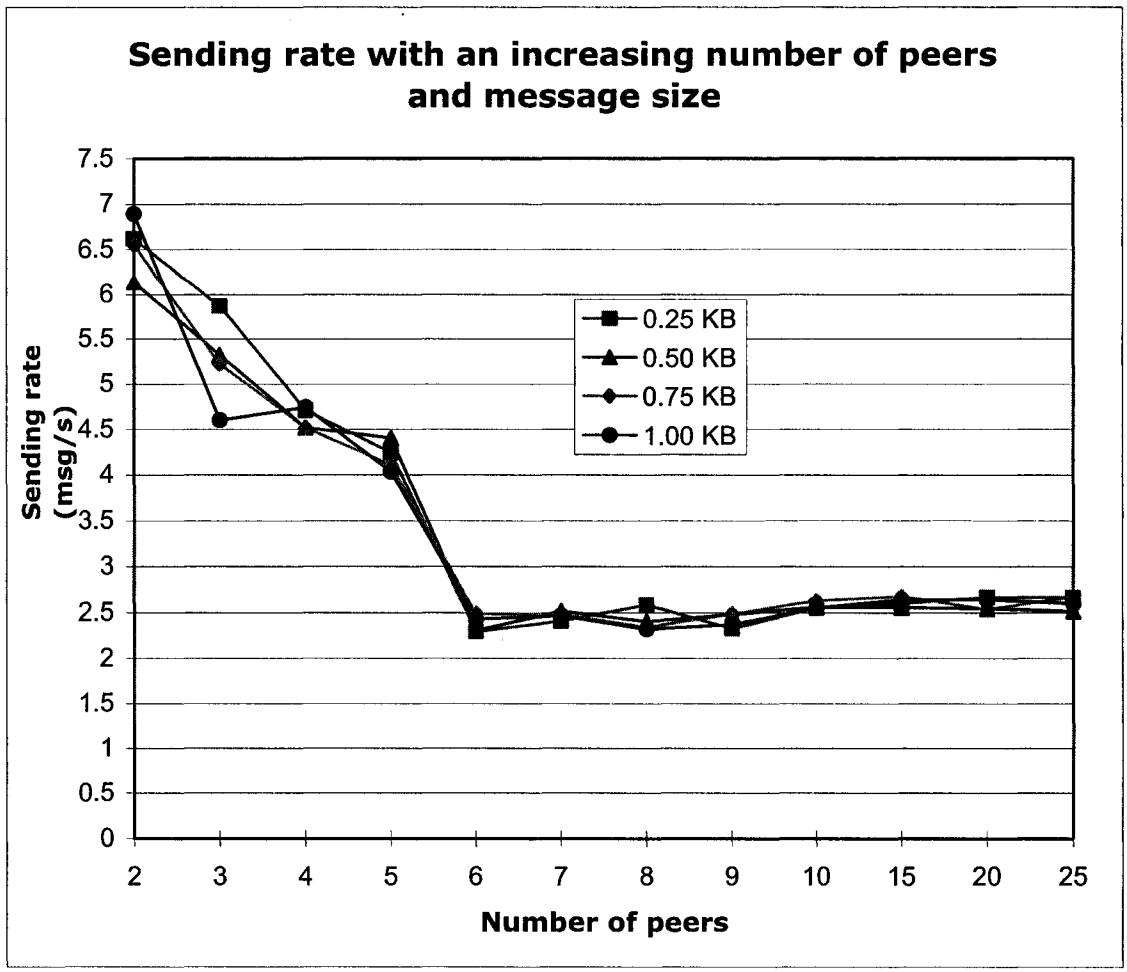


Figure 43: Sending rate with an increasing number of peers and message size

5.3 Pilot trial

It is worth mentioning that, after performing these experiments and obtaining positive initial results, we actually tried the MediCol application on a mobile device. For instance, we used the Pocket PC HP iPAQ Hx2750 model. We proceeded as follows:

- Installing the Mimer Mobile SQL program on the handheld,
- Installing the J9/MIDP runtime environment to run J2ME/MIDP applications,

- Using IBM WebSphere Studio Device Developer (WSDD) development environment to package the MediCol application into a Java Archive (JAR) file that is compatible with the J9/MIDP on the mobile device,
- Deploying the JAR file, with a Java Application Descriptor (JAD) file, on the handheld using the ActiveSync software. Selecting this JAR file on the device will automatically launch the IBM J9 MIDP installer, which will install the MediCol application on the Pocket PC,
- Launching the JXTA relay on a desktop, starting the Mimer SQL database on the device, and then launching the MediCol application from the J9 Manager (manages all the J2ME/MIDP applications stored on a particular device).

The results of this experiment are shown in Figure 44, Figure 45, and Figure 46 below.

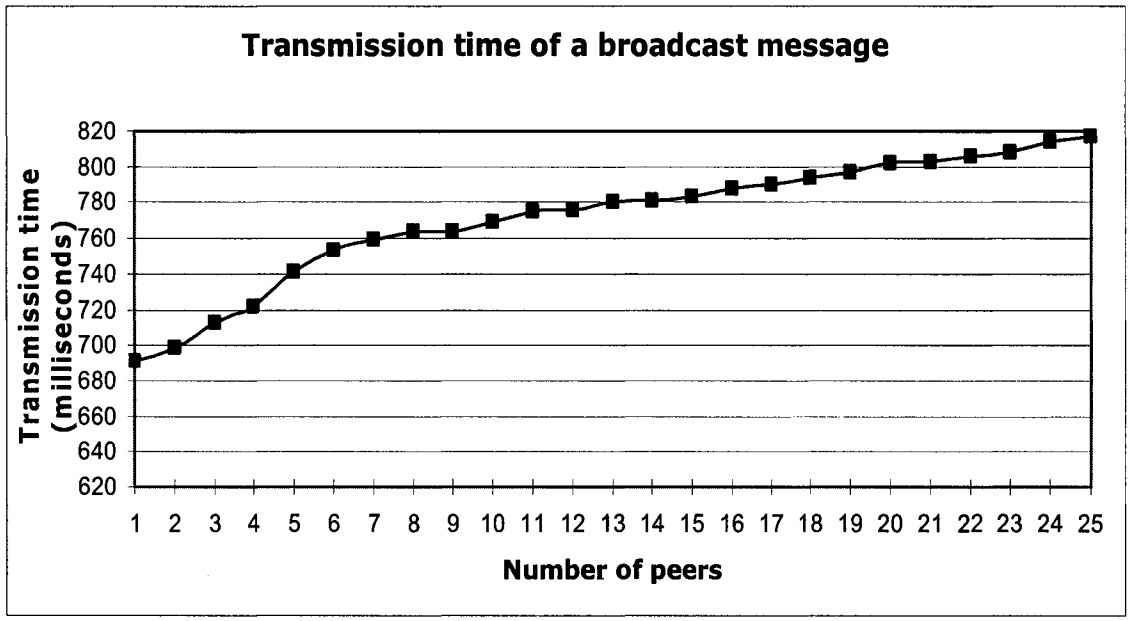


Figure 44: Average transmission time of an 80-bytes broadcast message with an increasing number of peers – this Figure corresponds to Figure 41 (variance is around 12.5 ms)

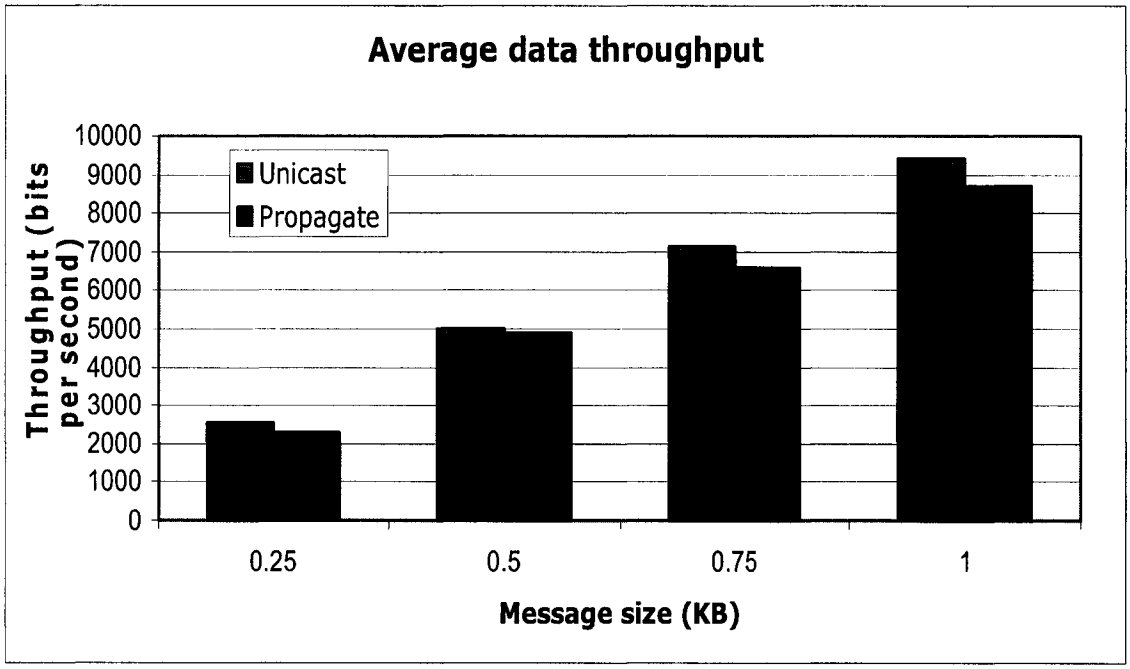


Figure 45: Average data throughput among 3 peers with an increasing message size – this Figure corresponds to Figure 42

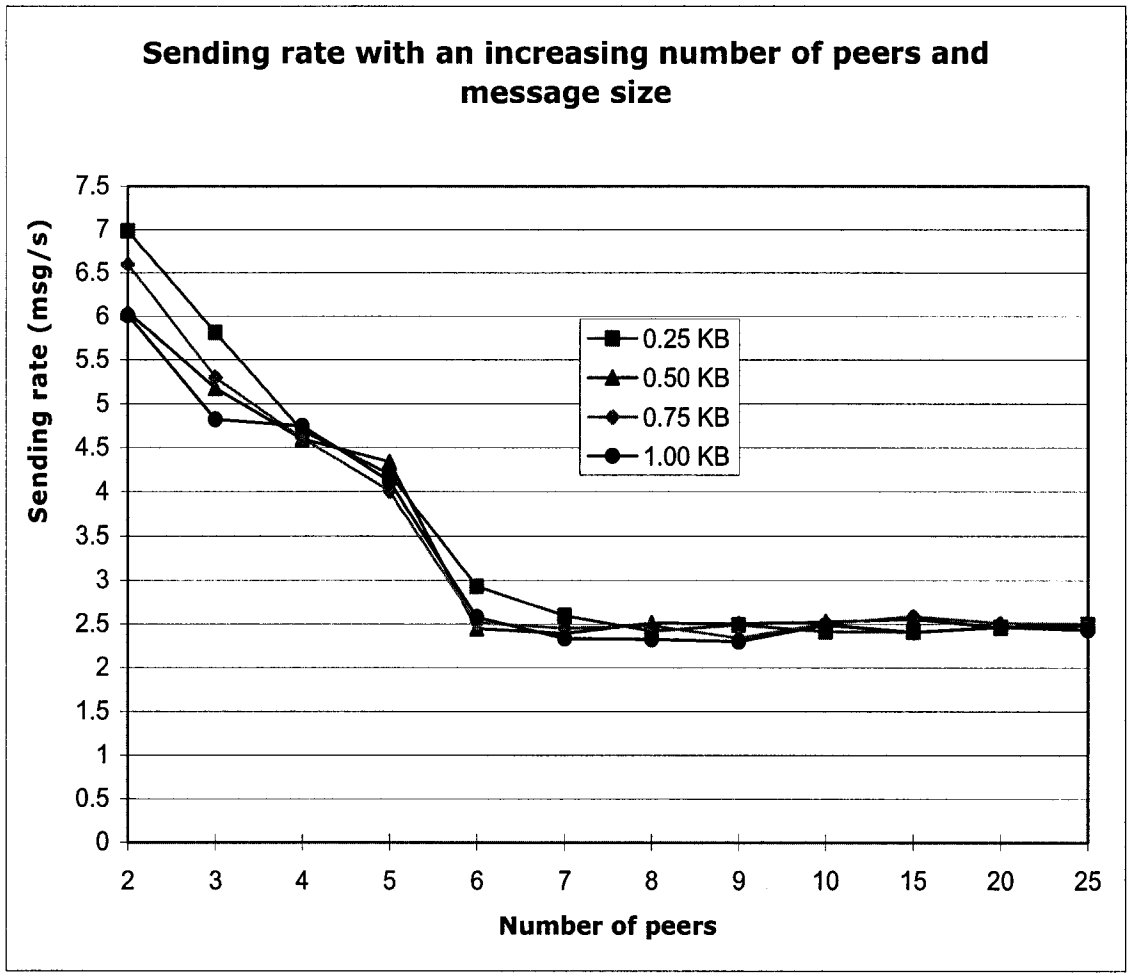


Figure 46: Sending rate with an increasing number of peers and message size - size – this Figure corresponds to Figure 43

Chapter 6 Conclusions and Future work

6.1 Research contributions

The main objective of the MediCol project is to provide a system that can assist medical students during their surgery rotation. Being a JXTA-based application, MediCol allows students to collaborate as well as maintain a portfolio during their rotation. The main contributions of this thesis are outlined below:

- Past and current research in the m-learning field is reviewed with particular focus on the various forms that m-learning materials can take, the m-learning application development cycle, the interaction-design models, and several design issues and constraints.
- J2ME is a very popular and widely-used platform. Since the focus of this project has been on platform-independence, J2ME fits the bill. MediCol has been able to achieve this objective only partially because it uses Mimer SQL for its database functionalities. However, it should be noted that this dependency can be eliminated by using PointBase instead of Mimer SQL.
- As mentioned earlier, the main objective of this project is to provide an architecture that can be adapted to various academic disciplines. MediCol achieves this objective by enriching the apprenticeship experience of medical students. Essentially, the system makes it easier for students to record their medical activities and submit them electronically to their mentor for review without physically meeting the latter. Besides, the students can engage into a question-answer dialogue with their peers and/or mentor to discuss issues or matters related to their rotation. The question and answer messages can have a logbook record and/or an e-book passage as attachment. This project has achieved this particular objective, while properly handling permanent (portfolio) data and ephemeral (chat log) data.
- All the experiments conducted using the MediCol system (as discussed in Chapter 5) show that it is very promising to use the JXME technology to develop collaborative applications in academic or non-academic disciplines. In particular, we have shown

that the JXME-based MediCol collaborative application in the medical apprenticeship context does not introduce inappropriate delays and can really enhance the student-mentor interactions during their surgery rotation.

The results from the experiments conducted using the MediCol application and a few tests carried on the HP iPAQ HX 2750 are very positive to justify further research in the development of apprenticeship-based m-learning applications.

6.2 Future work

Possible future work for the MediCol application includes the following:

- MediCol is an ambitious project and is still in its early stages; however there is room for improving some of its components:
    - The GUI of the forum builder and e-book reader may be significantly improved by creating custom-made GUI items using the CustomItem class.
    - The functionalities of the e-book reader can still be enhanced to provide better browsing, navigation, and reading. The notion of pages, chapters, and so on can be implemented and integrated within the current MediCol application.
- This thesis has provided an initial evaluation of the JXME platform on mobile devices. Due to time limitations, a usability and pilot studies could not be performed. Hence, a usability study should first be done to improve the presentation aspect of the MediCol system and make it more user-friendly. Second, a pilot study should be carried out to evaluate the MediCol system itself in terms of how students feel about the application, whether they are able to perform and log their various activities easily, and to uncover any specific issues (e.g., are handheld devices feasible for entering questions or answers text that are of a fair length?).
- The experimental simulations are not accurate since network delays cannot be predicted precisely. Since the MediCol system has already been executed successfully on a real device, it should be subjected to intense experiments and tests to validate the experimental simulation results of this thesis.
- The vision is to transform the MediCol architecture into a plug-and-play one that can

easily accommodate the activities of any academic discipline.

6.3 Conclusion

Wireless devices like cell phones, two-way pagers, PDAs, and so on are popular these days because they provide instant gratification and convenient services to users without restricting them to a particular place and time. Nowadays, such small devices support additional features like email access, messaging, address book and calendar services, as well as Web browsing. There is a rising need and importance for wireless Web access from portable devices and cell phones.

A lot of research has been reported on e-learning, which has been expanded to the mobile platform. The main objective of this research was to design a new application under the apprenticeship-based class in the m-learning domain, while focusing completely on platform-independence. This therefore justifies the use of Java™ 2 Micro Edition platform (J2ME™) in this project.

This research project has achieved its objectives through the implementation of MediCol and experimental simulations, which show a lot of positive feedback and response in the long run. It is hoped that this thesis will inspire researchers to carry out further research in the apprenticeship-based m-learning domain as well as the collaborative mobile computing area .

Bibliography

[AGT+04] M. Arrigo, M. Gentile, D. Taibi, G. Chiappone, and D. Tegolo; "mCLT: an application for collaborative learning on a mobile telephone"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 11-13

[AHJ05] G. Antoniu, P. Hatcher, M. Jan, and D. Noblet; "Performance Evaluation of JXTA Communication Layers"; *In Proceedings of the 5th International Workshop on Global and Peer-to-Peer Computing, Cardiff, UK, May 2005*

[ALM05] M. Ally, F. Lin, and R. McGreal; "An Intelligent Agent for Adapting and Delivering Electronic Course Materials to Mobile Learners"; *In Proceedings of the MLEARN Conference*, Cape Town, 25-28 October 2005, pp. 5-12

[ATR+01] J. Allin, C. Turfus, A. Robinson, L. Sweet, and J. Bown; "Wireless Java for Symbian Devices"; *Introduction to Java on the Symbian OS; John Wiley & Sons; 2001*

[BB03] L. Bartram and M. Blackstock; "Designing Portable Collaborative Networks; *Colligo Networks, ACM Queue, vol. 1, no. 3, May 2003*

[BBC+04] S. Bull, L. Bridgefoot, D. Corlett, P. Kiddie, T. Marianczak, C. Mistry, N. Sandle, M. Sharples, and D. Williams; "Interactive Logbook: the development of an application to enhance and facilitate collaborative working within groups in higher education"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 39-42

[BBP+05] J. E. Bardram, J. Bunde-Pedersen, and M. Mogensen; "Differentiating between Accountable and Ephemeral Events in the ABC Hybrid Architecture for Activity-Based Collaboration"; *In Proceedings of the IEEE International Conference on Collaborative Computing, pages 168-176, San Jose, USA, December 2005*

[BCM+05] J. Black, P. Castro, A. Misra, and J. White; "Live data views: programming pervasive applications that use 'timely' and 'dynamic' data"; *In Proceedings of the 6th international conference on Mobile data management, pages 294-298, Ayia Napa, Cyprus, May 2005*

[BH03] E. Brandt and P. Hillgren; "Self-produced video to augment peer-to-peer learning"; *In Proceedings of the MLEARN Conference*, London, 19-20 May 2003, pp. 27-34

[BLS+04] W. Byrne, P. Lonsdale, M. Sharples, C. Baber, T.N. Arvanitis, P. Brundell, and R. Beale; "Determining location in context-aware mobile learning"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 43-45

[BMN+03] S. Berger, R. Mohr, H. Nösekabel, and K. J. Schäfer; "Mobile collaboration tool for university education"; *12th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2003)*, Linz, Austria, 9-11 June 2003, pp. 77-80

[CCC+04] F. Cacace, M. Cinque, M. Crudele, G. Iannello, and M. Venditti; "The impact of innovation in medical and nursing training: a Hospital Information System for Students (HISS) accessible through mobile devices"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 47-52

[CJ05] E. P. de Crom and A. de Jager; "The 'ME'-Learning Experience: PDA Technology and E-Learning in Ecotourism at the Tshwane University of Technology (TUT)"; *In Proceedings of the MLEARN Conference*, Cape Town, 25-28 October 2005, pp. 72-107

[CS04] J. Colley and G. Stead; "Mobile learning = collaboration"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 57-58

[CSG05] M. A. M. Carreras, A. F. G. Skarmeta, and E. M. Gracia; "Designing Collaborative Environments and their Application in Learning"; *In Proceedings of the 1st International Conference on Collaborative Computing, San Jose, USA, December 2005*

[DWN03] P. Dawabi, M. Wessner, and E. Neuhold; "Using mobile devices for the classroom of the future"; *In Proceedings of the MLEARN Conference*, London, 19-20 May 2003, pp. 55-59

[FHO04] A. Ferscha, C. Holzmann, and S. Oppl; "Team awareness in personalized learning environments"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 67-72

[Fro04] D. Frohberg; "Mobile learning in tomorrow's education for MBA students"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 81-84

[GHS04] C. Göth, U. Häss, and G. Schwabe; "Requirements for mobile learning games shown on a mobile game prototype"; *In Proceedings of the MLEARN Conference*, Rome, 5–6 July 2004, pp. 95-100

[GSB+04] W. G. Griswold, P. Shanahan, S. W. Brown, R. Boyer, M. Ratto, R. B. Shapiro, and T. M. Truong; "ActiveCampus: Experiments in Community-Oriented Ubiquitous Computing"; *Computer, vol. 37, no. 10, 2004, pp. 73-81*

[HD-Aug03] E. Halepovic and R. Deters; "JXTA Performance Study"; *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, BC, Canada, August 2003*

[HD-Sep03] E. Halepovic and R. Deters; "The Costs of Using JXTA"; *In Proceedings of the 3rd IEEE CS International Conference on Peer-to-Peer Computing, Linköping, Sweden, September 2003*

[HD02] E. Halepovic and R. Deters; "Building a P2P Forum System with JXTA"; *In Proceedings of the 2nd IEEE CS International Conference on Peer-to-Peer Computing; Linköping, Sweden, September 2002*

[HRS03] N. Hine, R. Rentoul, and M. Specht; "Collaboration and roles in remote field trips"; *In Proceedings of the MLEARN Conference*, London, 19-20 May 2003, pp. 69-72

[HSH93] D. Hindus, C. Schmandt, and C. Horner; "Capturing, structuring, and representing ubiquitous audio"; *ACM Transactions on Information Systems Journal, vol. 11, no. 4, pages 376-400, ACM Press, New York, USA, 1993*

[JHS+99] J. Jing, K. Huff, H. Sinha, B. Hurwitz, and B. Robinson; "Workflow and Application Adaptations in Mobile Environments"; *In Proceedings of the 2nd IEEE Workshop on Mobile Computer Systems and Applications, New Orleans, USA, February 1999*

[LBS+03] P. Lonsdale, C. Baber, M. Sharples, and T. N. Arvanitis; "A context-awareness architecture for facilitating mobile learning"; *In Proceedings of the MLEARN Conference*, London, 19-20 May 2003, pp. 79-85

[LLK04] M. Lin, W. G. Lutters, and T. S. Kim; "Understanding the micronote lifecycle: improving mobile support for informal note taking"; *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 687-694, Vienna, Austria, April 2004*

[LVH+02] T. Leinonen, O. Virtanen, K. Hakkarainen, and G. Kligyte; "Collaborative discovering of key ideas in knowledge building"; *In Proceedings of the Computer Support for Collaborative Learning Conference*, Boulder, Colorado, 7-11 January, 2002, pp. 529-530

[McA02] S. McAteer; "Java Will be the Dominant Handset platform"; *MicroDevNet – Micro Java Network; 2002*

[MDC+04] F. Mercier, B. David, R. Chalon, and J. Berthet; "Interactivity in a large class using wireless devices"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 129-131

[MF05] P. Mattila and T. Fordell; "MOOP - Using m-learning environment in primary schools"; *In Proceedings of the MLEARN Conference*, Cape Town, 25-28 October 2005, pp. 222-230

[MFC04] J. Mitić, M. Feisst, and A. Christ; "mLab: handheld assisted laboratory"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 133-134

[MG04] A. Moreno and S. Grande; "A new model for the m-learning emergency scenario in risk contexts: the emergency operator"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 139-142

[Mif03] L. Mifsud; "Learning 2go: making reality of the scenarios?"; *In Proceedings of the MLEARN Conference*, London, 19-20 May 2003, pp. 99-104

[Mit03] A. Mitchell; "Exploring the potential of a games-oriented implementation for m-portal"; *In Proceedings of the MLEARN Conference*, London, 19-20 May 2003, pp. 105-115

[MM02] N. Maibaum and T. Mundt; "JXTA: A Technology Facilitating Mobile Peer-to-Peer Networks"; *In Proceedings of International Mobility and Wireless Access Workshop, pages 7-13, Fort Worth, USA, October 2002*

[MOM96] S. E. McDaniel, G. M. Olson, and J. C. Magee; "Identifying and analyzing multiple threads in computer-mediated and face-to-face conversations"; *In Proceedings of the 1996 ACM conference on Computer supported cooperative work, pages 39-47, Boston, US, November 1996*

[Mor01] Michael Morrison; "Getting to know the J2ME Emulator"; *Article courtesy of sampublishing.*com - *excerpted from Sams Teach Yourself Wireless Java w/J2ME in 21 Days; August 2001*

[MTFM03] M. C. Mayorga-Toledano and A. Fernández-Morales; "Learning tools for Java-enabled phones: an application for actuarial studies"; *In Proceedings of the MLEARN Conference*, London, 19-20 May 2003, pp. 95-98

[Pin04] A. Pincas; "Approaches to just-in-time learning with mobile phones: a case study of support for tourists' language needs"; *In Proceedings of the MLEARN Conference*, Rome, 5-6 July 2004, pp. 157-162

[RST+03] M. Ratto, R. B. Shapiro, T. M. Truong, and W. G. Griswold; "The ActiveClass Project: Experiments in Encouraging Classroom Participation"; *Computer Support for Collaborative Learning, Kluwer, pages 477-486, Bergen, Norway, June 2003*

[RSY+05] M. Rappa, S. E. Smith, A. Yacoub, and L. Williams; "OpenSeminar: A Web-Based Collaboration Tool for Open Educational Resources"; *In Proceedings of the 1st International Conference on Collaborative Computing, San Jose, USA, December 2005*

[SCB00] M. Smith, J. J. Cadiz, and B. Burkhalter; "Conversation trees and threaded chats"; *In Proceedings of the 2000 ACM conference on Computer supported cooperative work, pages 97-105, Philadelphia, US, December 2000*

[SCC03] G. L. Sandoval, E. E. Chávez, and J. C. P. Caballero; "A Development Platform and Execution Environment for Mobile Applications"; *JIISIC, vol. 7, no. 1, paper 4, 2003*

[TEC05] A. Theophilo, M. Endler, and R. Cerqueira; "Evaluation of three approaches for CORBA firewall/NAT traversal"; *In Proceedings of DOA'05, Springer-Verlag Heidelberg*, Agia Napa, Cyprus, 2005, pp. 923-940

[WKG+03] C. Wuthrich, G. Kalbfleisch, T. Griffin, and N. Passos; "On-line instructional testing in a mobile environment"; *Journal of Computing Sciences in Colleges, vol. 18, no. 4, pages 23-29, Consortium for Computing Sciences in Colleges, USA, 2003*

[Zua05] I. A. Zualkernan; "HYDRA: A Light-Weight, SCORM-Based P2P e-Learning Architecture"; *In Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies, pages 484-486, Kaohsiung, Taiwan, July 2005*

Internet Resources

[CB-HSDP] CodeBase: High Speed Database for Programmers. *http://www.codebase. com/products/J2ME/; Last access to site: June 2006.*

[DB2E] DB2 Everyplace: Information Management Software*; http://www-128.ibm.com/developerworks/db2/products/db2e/; Last access to site: June 2006.*

[EJ2MEP] Eclipse's J2ME Plugin. *http://eclipseme.org/; Last access to site: June 2006.*

[Fal02] Mary A. C. Fallon; Handheld Devices: Toward a more Mobile Campus*; Campus Technology Magazine; http://www.campus-technology.com/article.asp?id=6896, 2002; Last access to site: June 2006.*

[Gig] E. Giguère; The Importance of MIDP; *http://www.developer.com/java/j2me/article.php/1453731; Last access to site: June 2006.*

[Gig02] E. Giguère; J2ME Core Concepts*; http://www.ericgiguere.com/articles/j2me-core-concepts.html?noprint=true; 2002; Last access to site: June 2006.*

[JJPG] JXTA2.3 Java Programmer's Guide. *www.jxta.org/docs/JxtaProgGuide_v2.3.pdf; Last access to site: June 2006.*

[JSR172] Web Services Profile: Web Services Access APIs. *http://www.jcp.org/en/jsr/detail?id=172; Last access to site: June 2006.*

[JSR75] PDA Profile: File Systems and PIM APIs. *http://www.jcp.org/en/jsr/detail?id=75; Last access to site: June 2006.*

[LG-INTRO] The Lurker's Guide to J2ME; Introduction to J2ME; *http://www.blueboard.com/j2me/intro.htm; Last access to site: June 2006.*

[LG-WHY] The Lurker's Guide to J2ME; Why J2ME?; *http://www.blueboard.com/j2me/why.htm; Last access to site: June 2006.*

[Mah02] Q. Mahmoud; Wireless Java Security; *http://developers.sun.com/techtopics/mobility/midp/articles/security/; 2002; Last access to site: June 2006.*

[MPI-CJS] Making P2P Interoperable: Creating JXTA Systems. *http://www-128.ibm.com/developerworks/java/library/j-p2pint3/; Last access to site: June 2006.*

[MPI-TJS] Making P2P Interoperable: The JXTA Story. *http://www-128.ibm.com/developerworks/java/library/j-p2pint1.html; Last access to site: June 2006.*

[MSM] Mimer SQL Mobile; *http://www.mimer.com/leftright.asp?secId=172; Last access to site: June 2006.*

[ODL] Oracle Database Lite: Technology Network; *http://www.oracle.com/technology/products/lite/index.html; Last access to site: June 2006.*

[PBM-D] PointBase Micro: DataMirror; *http://www.pointbase.com/products/micro.aspx; Last access to site: June 2006.*

[Sei02] J. M. Seigneur; Jxta Pipes Performance; *http://bench.jxta.org/papers/jmjxtapipesperformance.pdf, 2002; Last access to site: June 2006.*

[SiAMD] Sybase iAnywhere: Mobile Database; *http://www.ianywhere.com/products/mobile.html; Last access to site: June 2006.*

[SJWT] Sun Java Wireless Toolkit. *http://java.sun.com/products/sjwtoolkit/; Last access to site: June 2006.*

[Sul01] S. O. Sullivan; Bluetooth and Java – A Perfect Match?; *http://www.rococosoft.com/docs/Bluetooth_and_Java.ppt; Slides 4 and 28; 2001; Last access to site: June 2006.*

[Tau01] D. A. Tauber; What's J2ME?; *http://www.onjava.com/pub/a/onjava/2001/03/08/J2ME.html; 2001; Last access to site: June 2006.*

[OWJ] Oreilly Wireless Java; Chapter 5; *http://www.oreilly.com/catalog/wirelessjava/chapter/ch05.html; 2001; Last access to site: June 2006.*