

T04907

**Shared Storage in J2ME:
A Multi-Agent System Approach**

T4907



Developed by

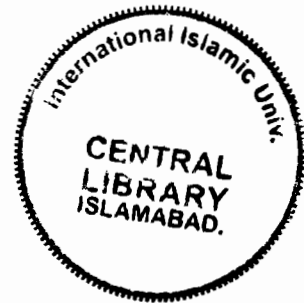
Muhammad Ainan Sadiq

(261-FAS/MSCS/F05)

Supervised by

Dr. Naveed Ikram

Mr. Zohaib Zafar



**Department of Computer Science
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad.**

Dated: _____

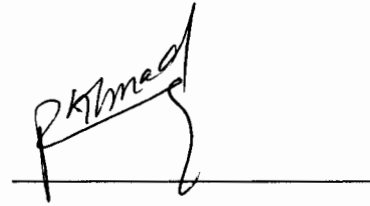
FINAL APPROVAL

It is certified that we have read the thesis, entitled “**Shared Storage in J2ME: A Multi-Agent System Approach**”, submitted by Muhammad Ainan Sadiq Reg. No. 261-FAS/MSCS/F05. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University Islamabad for MS Degree in Computer Science.

PROJECT EVALUATION COMMITTEE

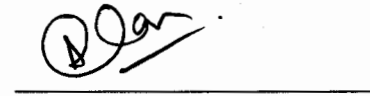
External Examiner:

Dr. Hafiz Farooq Ahmed,
School of Electrical Engineering & Computer Science,
NUST Institute of IT,
Rawalpindi, Pakistan.



Internal Examiner:

Mr. Adnan Ashraf,
Department of Computer Science,
Faculty of Basic and Applied Sciences,
International Islamic University,
Islamabad, Pakistan.



Co-Supervisor:

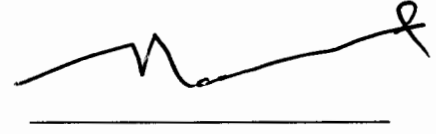
Mr. Zohaib Zafar
Department of Computer Science,
Faculty of Basic and Applied Sciences,
International Islamic University,
Islamabad, Pakistan.



Supervisor:

Dr. Naveed Ikram

Department of Computer Science,
Faculty of Basic and Applied Sciences,
International Islamic University,
Islamabad, Pakistan.



In the Name

of

ALLAH

The Most Beneficent

The Most Merciful

A thesis submitted in partial fulfillment of the requirements for the degree of

MS in Computer Science

in the Faculty of Basic and Applied Sciences,

International Islamic University, Islamabad, Pakistan

Project In Brief

- Project Title:** Share Storage in J2ME: A Multi-Agent Systems approach
- Organization:** International Islamic University, Islamabad, Pakistan.
- Objective:** The objective of the research in the area of Multi-Agent System is to provide a mechanism for sharing record store among different MIDLETs on same as well as MIDLETs on different devices in J2ME based lightweight multi-agent systems.
- Undertaken By:** Muhammad Ainan Sadiq
Reg. No. 261-FAS/MSCS/F05
- Supervised By:** Dr. Naveed Ikram (IIUI)
Mr. Zohaib Zafar (IIUI),
Faculty of Basic and Applied Sciences,
International Islamic University, Islamabad.
- Started On:** _____
- Completed On:** _____
- Research Area:** Shared Storage, RMS, Multi-Agent Systems, J2ME, distributed environment.
- Tools:** J2ME, Borland JBuilder 2005 Enterprise Edition, JDK 1.5,
Sun Java Wireless Toolkit 2.5.

Abstract

Current research trend towards mobile computing emphasizes the need for distribution of data among various clients in mobile environment. Sharing the data in J2ME opens the data to be accessible for all MIDLETS present on the device, hence creating a number of vulnerabilities to the confidential information that is intended to be shared with specific MIDLETS, e.g. a malicious MIDLET can delete, copy or move the Shared record store in J2ME based application. This project aims at overcoming sharing of record store problem through the concept of Multi-Agent System by providing a shared storage with explicit access to authenticated & authorized MIDLETS. With this solution data can be shared among specific MIDLETS. The implementation uses the SAGE-Lite framework as a solution to our proposed work.

Acknowledgements

With a debt of gratitude, which cannot be adequately expressed in words, we thank our supervisors **Dr. Naveed Ikram** and **Mr. Zohaib Zafar**, for their advice, guidance, and endless support during our research. Their practical and sharp vision in research has not only been invaluable for our work on this thesis but also for developing our taste in research and our development as researchers. We are also indebted to **Mr. Zohaib Zafar** for his unforgettable generous support for our research.

We extend sincere gratitude to all of our teachers whom had been guiding me through out our course work and increased our knowledge. Their knowledge, guidance and training enabled me to carry out this research & development work.

Dr. Hafiz Farooq Ahmad being an expert in the field of Multi-Agent System and pioneer of Scalable Agent Grooming Environment (SAGE), and SAGE lite helped a lot during our research work. His experience and wisdom enabled us to refine our idea to a mature concept. We are also thankful to the NUST School of Electrical Engineering and Comouter Scinces and Dr. Hafiz Farooq for allowing us to work on the framework of SAGE-Lite and making this project possible.

We would like to offer our appreciation to **Dr. Naveed Ikram** for his insightful suggestions, encouragement and guidance in the field software engineering. We are also indebted to **Mr. Shakeel** and **Mr. Adnan Ashraf** for their unforgettable generous support for the fulfillment of our degree requirements.

We express our sincere gratitude to our external examiner **Dr. Hafiz Farooq Ahmad**, who gave us good comments on research ethics and his useful suggestions helped us refining our work more efficiently.

We extend our sincere gratitude to our friends **Mr. Syed Muhammad Ali Shah, Mr. Naseer Gul and Mr. Adeel** for their insightful and technical suggestions at various stages of this project.

Last but not least, we are deeply indebted to our families for their generous support and the financial sacrifice for making this possible. We once again would like to admit that we owe all our achievements to our most loving parents, who mean most to me, for their prayers are more precious than any treasure on the earth. We are also thankful to our truly, sincere and most loving brother, sisters, friends and class fellows who mean the most to me, and whose prayers have always been a source of determination for me.

Muhammad Ainan Sadiq

261-FAS/MSCS/F05

Dedication

We would like to dedicate our work to

ALMIGHTY ALLAH,

Who has always showered His endless blessings upon me.

We also dedicate this work to our

FAMILY AND FRIENDS

Whose sincere prayers and love were a source of strength for me,

And made this project successful.

Declaration

We hereby declare and affirm that this thesis neither as a whole nor as part thereof has been copied out from any source. It is further declared that we have completed this thesis and accompanied software application on the basis of our personal efforts, made under the sincere guidance of our supervisor. Where as necessary references and acknowledgment has been made. If any part of this report is proven to be copied out or found to be a reproduction of some other, we will stand by the consequences. No portion of the work presented in this report has been submitted in support of an application for other degree or qualification of this or any other University or Institute of learning.

Muhammad Ainan Sadiq

261-FAS/MSCS/F05

Table of Contents

Chapter No.	Content	Page No.
1.	Introduction	1
1.1	The Research Perspective:	2
1.2	The Research Objectives:.....	2
1.3	Thesis Contribution:.....	3
1.4	Thesis-Related Paper:	4
2.	Literature Review:	6
2.1	Architecture Of J2ME Record Management System (RMS).....	8
2.2	Record Stores	9
2.3	Record Stores Sharing.....	9
2.4	Shared Storage Vulnerabilities.....	9
2.5	Problem Definition:.....	10
3.	Multi-Agent Systems	13
3.1	Evaluation of Multi-agents Platforms:.....	13
3.1.1.	FIPA-OS:	14
3.1.2.	Grasshopper:	15
3.1.3.	JADE-LEAP (Lightweight Extensible Agent Platform):.....	15
4.	Multi-Agents Based Record Sharing Architecture	18
4.1	SAGE-Lite	18
4.1.1.	Architecture of SAGE-Lite	18
4.1.2.	Light-weight Agent Management System (AMS-Lite)	19
4.1.3.	Light-weight Message Transport Service (MTS-Lite).....	20
4.1.4.	Light-weight Agent Communication Language (ACL-Lite)	20
4.1.5.	ACL-Lite Message Structure	20
4.1.6.	Context-Awareness in SAGE-Lite.....	21
4.1.7.	AMS-Lite Design	21
4.1.8.	Visual Management Service	22
4.2	Agent Lifecycle Manager.....	23
4.3	Managing DF-Lite.....	24
4.4	Managing Services	25

4.5	Proposed Solution	25
4.6	Implementation of Proposed Solution.....	29
4.7	Advantages of proposed solution	32
5.	Evaluation	34
5.1	Testing Scenarios:	34
5.1.1.	Scenario 1: MIDLet with Shared Record Store:	34
5.1.2.	Scenario 2: MIDLet with Private Record Store:	36
5.2	Results & Discussion:	38
5.3	Advantages of Research Work:.....	39
5.4	Conclusion	39
5.5	Future Work	40
6.	Bibliography	42
	APPENDIX: Research Paper.....	44

Table of Figures

Figure No.	Content	Page No.
2.1	Internal Architecture of Record Store	9
2.2	Shared record store vulnerabilities	10
4.1	Architecture of SAGE-Lite	18
4.2	View of registered agent	23
4.3	Record Store Sharing based on SAGE-Lite	26
4.4	Agents accessing record stores	27
4.5	Agents accessing record stores from different devices	27
4.6	Application architecture based on proposed solution	29
4.7	Patient's temperature manual entry	30
4.8	Graph drawn by an Agent.....	31
4.9	View Treatment Schedule	31
5.1	MIDLET Launch	35
5.2	Data Entry Form	35
5.3	Record Storage Info	35
5.4	Malicious MIDLET	36
5.5	Patient MIDLET	37
5.6	Data Entry Page	37
5.7	Storage Information.....	37
5.8	Malicious MIDLET	37

Table of Tables

Chapter No.	Content	Page No.
2.1	Evaluation of technologies.....	8
5.1	Evaluation of proposed solution	38

Chapter 1

INTRODUCTION

1. Introduction

Today's computing trend is more towards ubiquitous computing, which is diverging a number of database applications to the mobile environment and facilitating the users to achieve the usefulness of anywhere anytime computing. Different m-commerce applications are being deployed on small handheld devices, to facilitate the transactions and data processing in a mobile environment. Java 2 Micro Edition (J2ME) [1] is one of the leading technology in handheld device applications.

One of the J2ME features is that it provides built in caching mechanism for locally storing data on a mobile device. The Mobile Information Device Profile (MIDP) adds APIs (Application Programming Interface) for user interaction, network connectivity, and persistent storage. Two profiles have been developed for J2ME: MIDP 1.0 and 2.0. J2ME has a Record Management System (RMS), an Application Programming Interface (API) that provides persistent storage on local device. A few MIDP-enabled handheld devices support the traditional file system; RMS [2] is the only feature in J2ME for local data storage and is essential to writing any application that relies on local persistent data. RMS stores all the records in a file with extension ".db" called record store. The application developed in J2ME for mobile devices is called MIDLET and each MIDLET suit (group of related MIDLETs) can own one or more record stores.

Due to the limitations of J2ME Record Management System (RMS), data when made public is accessible to all the MIDLETs present on the device and hence creating a number of vulnerabilities to the confidential information.

Our work is to share the data among specified MIDLETs on same device as well as on different devices and protect it from malicious MIDLETs access. Our solution uses the concept of multi-agent systems for developing a mechanism of Access Control in order to make the information accessible only to the authenticated or authorized MIDLETs. This is achieved by introducing a layer of multi-agents that handle the whole data manipulation from inside or outside the device. Two MIDLETs

can exchange data through their agents. An agent can perform desired action only on its MIDLET data. In order to perform an action on different MIDLET's data the agent of first MIDLET sends a command to the agent of second MIDLET that responds to the agent of first MIDLET with the desired results. In this way shared storage vulnerabilities are minimized. The benefit of using multi-agents system approach is to save redevelopment of legacy systems, only a wrapper agent needs to be developed. Secondly the data remains protected i.e. inaccessible to malicious MIDLETs and can only be shared among authorized MIDLETs. Furthermore, performance is also increased by using multi-agents instead of using other techniques discussed in related work.

1.1 The Research Perspective:

The critical points kept in mind while developing applications providing shared and distributed environment for handheld devices are keeping the data secure and only accessible to valid users and applications. They also provide transparent data sharing among users or applications.

In this perspective, the thesis adopts a view that recommends a framework for shared and distributed data between different MIDLET Suits.

1.2 The Research Objectives:

The goal of this thesis is to develop a mechanism to securely share data among different MIDLETs based on lightweight multi-agent systems that are targeted to run on small handheld devices. Following are some specific objectives to be met by the research work:

- Data can be shared among multiple MIDLETs
- To provide an architecture that is independent of specific applications and is applicable for rang of J2ME applications.
- To define a security policy for authentication and authorization of different MIDLETs through their agents.

The benefit of this research work is to assist the J2ME application developers to develop open source distributed applications. There is no need to use third party tool for data storage and sharing. The objective is to secure the J2ME record store and share it among the specific group of MIDLETS. This framework will enable the developer to develop a lightweight secure, distributed and shared environment.

1.3 Thesis Contribution:

This thesis enhances the understanding of distributed databases on MIDP enabled devices that is to be distributed among different MIDLETS. This approach of sharing of data is based on Multi-agent system, where access to data is given through different agents on the device. Granting access to data specifically for a set of MIDLETS will enable application developer to restrict the existing vulnerabilities associated with making the data shared using traditional techniques.

The thesis makes the following specific contributions:

- Survey of research work on existing techniques for sharing data in mobile applications on same or different devices or platforms.
- Highlights the requirements for developing mobile applications with distributed sharing of data. These requirements are specified for lightweight multi-agent systems applications targeted for small handheld devices.
- Describes an approach to develop a multi-agent based record store sharing mechanism that uses the existing record store and gives its access through its agents.
- Complements an existing multi-agent framework SAGE LITE [3] with the feature of shared storage local and remote devices.
- Applies the approach on SAGE-Lite (Scalable Fault Tolerant Agents Grooming Environment-Lite) framework [1]. This application demonstrates novelty in the record store sharing and its access to other multi-agent system based applications. It also draws some preliminary observations, lessons, and insights that could stimulate future research in the area of distributed data processing in lightweight multi-agent systems based applications.

1.4 Thesis-Related Paper:

A research paper titled: “*Shared storage in J2ME: A Multi-Agent Systems approach*” is based on the work presented in this thesis.

Chapter 2

LITERATURE SURVEY & PROBLEM STATEMENT

2. Literature Review:

Designing and implementing complex software systems e.g. m-commerce and other intelligent applications for handheld devices is rapidly growing. Persistence and distribution of data is crucial for these systems, as the small handheld devices are not as powerful as that of their counterpart – desktop machines.

Keeping in view the constrained environment and limited resources on these devices a number of mechanisms are provided for data persistence on these devices, e.g. J2ME provides Record Management System (RMS) for locally storing the data on the device. RMS is a system for managing records in J2ME. A record is an individual data item.

The records in RMS consist of a variable length binary field. This is contrary to the typical Database Management System (DBMS). The validation and data consistency checks cannot be applied to the records in RMS. The application developers are required to apply validation and data consistency checks to interpret the contents of records in RMS [2].

We find a number of alternatives for data sharing among same and different devices. Tuple-space is one of the well known and elegant ways of sharing information in distributed environment among communicating parties where processes write and read tuples [4].

JavaSpaces [5] implements the concept of tuple space through Java programming language. It is a service of Jini [6], which forms a distributed network of clients and services. JavaSpaces provide an easy way for communications facility in mobile distributed applications. Due to the fact that Jini technology is dependent on Remote Method Invocation (RMI), which is not supported by number of handheld devices, so JavaSpaces is not suitable for distributed mobile applications. Moreover, JavaSpaces requires a resource rich environment.

In order to make the handheld devices Jini compatible a number of different approaches were defined. These include modifying Java virtual machine, modifying Jini, or introducing a non-standard proxy. In JiniME [7], a J2ME virtual machine is changed to make MIDP devices Jini capable. This approach severely restricts the set

of devices, on which applications can be run.

MobileSpaces [8], on the other hand, require no changes to the standard J2ME environment. Like JavaSpaces, applications developed using MobileSpaces capture events through notify() method [8]. Each time when event is triggered regarding notifications about tuples, notify method is called, resulting in resource hungry application.

Although a light weight framework TinyDB [9] was developed but its main limitation is not providing distributed storage.

Another light weight storage system based on serialization framework [9] allows CLDC enabled J2ME devices to store data on local as well as remote storage spaces requiring similar semantics. So this framework does not support heterogeneous environment.

Oracle Database Lite 10g [10] is a unified and a comprehensive environment for development and deployment of high impact solutions intended for mobile and lightweight environments. It is an extension to Oracle Database 10g for enterprises focusing at increased employee throughput, reduced operative cost, and enhanced client satisfaction. It provides an extension to grid environment for mobile and embedded devices, allowing enterprise data accessible to remote employees even being offline. Moreover, it gives data synchronization permitting users to reliable and secure exchange of data with corporate Oracle Database.

The limitation of using Oracle Lite database is that its server and client requirements are too resource demanding for mobile environment. Besides, it requires a license to be purchased for every user, hence not a cost effective solution.

The related work in field of multi-agent system for developing applications for small handheld devices is found in Locker Rental Service [11], in which a persistent storage medium on a static device is preserved for different agents known as locker place. All agents communicate with their respective locker agents to operate on their data. Here the author had recommended multi-agent system as a middle-ware between a MIDLET and Locker place. Locker rental service provides a centralized storage

mechanism, which has a number of issues like maintenance problems of a single database, availability, a separate device for storage etc.

Table 2.1 shows the evaluation of technologies discussed in the above survey.

Parameters	Technologies			
	Java Spaces	Mobile Spaces	Web Services	Oracle Lite
Device / J2ME Compatibility	No	Low	High	Low
Heterogeneity	No	Low	High	Low
Resource Hungry	Yes	Yes	Yes	Medium
Server Required	Yes	Yes	Yes	Yes
Open source	No	No	No	No

Table 2.1 Evaluation of technologies

Table 2.1 presents a comprehensive comparison of different technologies according to five parameters as discussed in literature survey. Following are the major drawbacks in all the above mentioned technologies:

- They require a server
- They are not open source
- They are resource hungry.

These parameters are assigned values on the basis of following definitions:

LOW: Compatible with High End mobile devices e.g. PDAs having sufficient memory and processing resources.

MEDIUM: Compatible with Smart phones that contain a complete operating system.

HIGH: Compatible with J2ME enabled devices, with limited resources and having minimum specification of MIDP 2.0, CLDC 1.1, JSR-82.

2.1 Architecture Of J2ME Record Management System (RMS)

The architecture of J2ME record management system (RMS) contains small footprints for storing and manipulating persistent data on local device.

2.2 Record Stores

A record store is a chronological collection of records – associated to a record store. Every record can be accessed through the record store. In fact, record store ensures that records are read and written individually, with little possibility of data corruption. RMS maintains unique numeric sequence number for each record in a record store known as Record ID. Maximum 32 Unicode characters unique name is defined for record store within MIDLET suite creating it. Record store sharing mechanism is not provided in MIDP 1.0, while this mechanism is provided in MIDP 2.0, which allows record stores to be shared among other MIDLET suites. In each case the record store is recognized by the name of MIDLET suite, its vendor name and the record store name itself.

Time-stamp and version information is also maintained by record stores to enable applications to discover when it was last modified [2].



Fig: 2.1. Internal Architecture of Record Store

2.3 Record Stores Sharing

Record stores are made accessible to all other MIDLETS using its shared property. The default property AUTHMODE_PRIVATE allows record store to be accessible only to the MIDLET suite that created it. Record store can be shared by changing the property to AUTHMODE_ANY. When the record stores are shared they can be writable or read-only. If the shared data is confidential, then it can be read by all MIDLETS on the device making it more vulnerable to malicious MIDLETS on the device if it is shared in read/write mode [12].

2.4 Shared Storage Vulnerabilities

Confidential data or information can be susceptible to an attack outside the RMS that results it to be accessed and manipulated from device utilities (without using a MIDLET) [12]. The shared record store can be deleted by calling the deleteFile() method of the class RecordStoreFile [12] of low level MIDP API shown below in fig 2.2.

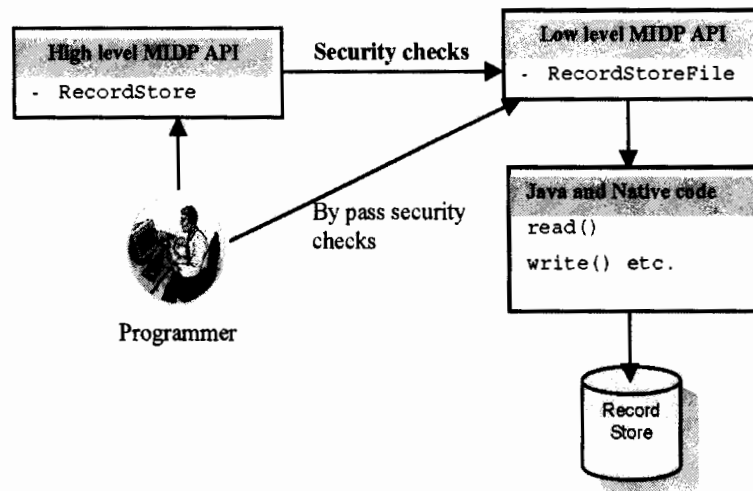


Fig 2.2 Shared record store vulnerabilities [12]

Developing applications for these devices poses some problems when the data of one MIDLET is required to be shared among multiple MIDLETS. Debbabi et al. [12] provide the Security Evaluation of J2ME platform and explains the vulnerabilities that are associated with the shared storage of data. Non shared record stores can be accessed and modified by the MIDLETS creating them, but the shared record stores can be accessed from any MIDLET on the device. So MIDLETS cannot share their record stores with only a specific subset of MIDLETS. Shared Record stores are vulnerable to any attack from outside the Record Management System (RMS) of J2ME.

2.5 Problem Definition:

Java 2 Micro Edition (J2ME) [12] provides a large platform for the development of applications intended to be run on small devices. These devices are constrained with small memory, limited computing & processing power. Developing applications for these devices poses some problems when the data of one MIDLET is required to be shared among multiple MIDLETS. In [12] Debbabi et al. provide the Security Evaluation of J2ME platform and explains the vulnerabilities that are associated with the shared storage of data. Non shared record stores can be accessed and modified by the MIDLETS creating them, but the shared record stores can be accessed from any

MIDLET on the device. So MIDLETS cannot share their record stores with *only a specific subset of MIDLETS*. Shared Record stores are vulnerable to any attack from outside the Record Management System (RMS) of J2ME.

- Record stores can be accessed from the device's utilities (without using a MIDLET).
- Record stores can be manipulated as files (copied, renamed, deleted, etc.).
- No encryption method is specified to protect sensitive record stores on the device.

Chapter 3

Multi-Agent System

&

Comparison of FIPA Complaint

Multi-Agent System

3. Multi-Agent Systems

A multi-agent system (MAS) [5] is a system composed of multiple autonomous agents with the following characteristics:

- Each agent cannot solve a problem unaided,
- There is no global system control,
- Data is decentralized, and
- Computation is asynchronous.

Multi agent systems provide execution environment for intelligent software (agents). There are different standard governing bodies. We especially focus on Foundation for Intelligent Physical Agent (FIPA) standards that we follow throughout the thesis.

3.1 Evaluation of Multi-agents Platforms:

Liljedahl [13] compiles a list of different multi-agents platforms, which is reproduced here for evaluation on the basis of different parameters:

- ADK
- Agentsheets
- AgentTool
- Bee-gent
- CABLE
- Comet Way
- CORMAS
- Cougaar
- DECAF
- FIPA-OS
- Grasshopper
- JACK
- JADE/LEAP
- JIVE
- Kaariboga
- Living Markets

- MASSYVE KIT
- RETSINA
- ZEUS

A comprehensive analysis on different multi-agent platform concludes that only three platforms (FIPA-OS, Grasshopper and JADE/LEAP) support mobility and FIPA standards.

3.1.1. FIPA-OS:

MicroFIPA-OS [13] is an extension of the FIPA-OS agent toolkit, designed to execute on medium to high-level PDA where Personal-Java is installed. This platform has been redesigned to be deployable on PDA like the Compaq iPaq running Linux or PocketPC operating system. The drawback of this platform is that only one agent is recommended to run on each PDA.

In this framework two-layer architecture with security layer on top of FIPA-OS has been designed. MicroFIPA-OS security framework [15] does not facilitate the Speech Acts, which allow web entities to update the existing policies (delegation of permissions, revocation of permissions, and cancellation of previous requests) automatically.

- Security: low.
- Heterogeneity: medium.
- Scalability: medium, only one agent per PDA is recommended.

3.1.2. Grasshopper:

Grasshopper [13] is an agent platform which supports mobile agents that can travel across different platforms. The Grasshopper platform supports both FIPA and MASIF through add-on modules. This platform runs under Windows, Windows CE and UNIX operating systems. Grasshopper is designed to implement the complex distributed workflow applications and dynamic software deployment at client/server systems. This platform is also used for development of third generation mobile telecommunication systems and mobile commerce applications. A web-hopper extension for grasshopper can be used to deploy agents on a web-server and to access agents through a web browser.

Grasshopper platform uses trusted classes [14]. These trusted classes override the Security Manager and are not checked for access. In case of Grasshopper this leads to a security leak. In Grasshopper the `checkAwtEventQueueAccess` method has not been implemented. Due to this vulnerability it is possible to access the event queue associated with the graphic interface and the graphic events can be traced easily. In this platform there is no security check on calls to the `checkPropertyAccess` method. This vulnerability causes the access and modification to any property that is available in the system.

3.1.3. JADE-LEAP (Lightweight Extensible Agent Platform):

JADE-LEAP [13] platform was designed to be a lightweight FIPA compliant platform deployable on PDAs and mobile phones as well as on workstations. The LEAP project is the extension of JADE platform and it can execute on various operating systems and java VM by use of different profiles. This platform supports TCP/IP over both wireless and fixed networks. The LEAP platform consists of several modules. The main drawback of this platform is that it relies on a main-container (containing DF and AMS) which is running on a computer elsewhere in the network.

- Security: Medium, A plug-in called Jade-S is used to add security functions.
- Heterogeneity: High.
- Scalability: High, because of the small footprints. According [13] thousands of agents can run simultaneously on this platform.

Chapter 4

Proposed Solution

MULTI-AGENTS BASED RECORD

SHARING ARCHITECTURE

4. Multi-Agents Based Record Sharing Architecture

In order to solve the problem mentioned in previous section, We use the concept of Multi-Agents System. In multi-agent system single task is divided among multiple agents on the same or different platforms. The proposed architecture is based on the existing FIPA compliant multi-agent system framework, namely SAGE-Lite [3].

4.1 SAGE-Lite

SAGE-Lite is a lightweight context aware multi-agent system, which senses the capabilities of the light-weight devices and reacts accordingly. SAGE Lite can also function as standalone system.

4.1.1. Architecture of SAGE-Lite

The architecture of SAGE-Lite contains three main modules [3].

- Light-weight Agent Management System (AMS-Lite)
 - Light-weight Directory Facilitator (DF-Lite)
 - Light-weight Visual Management Service (VMS)
- Light-weight Message Transport service (MTS-Lite)
- Light-weight Agent Communication Language (ACL-Lite)

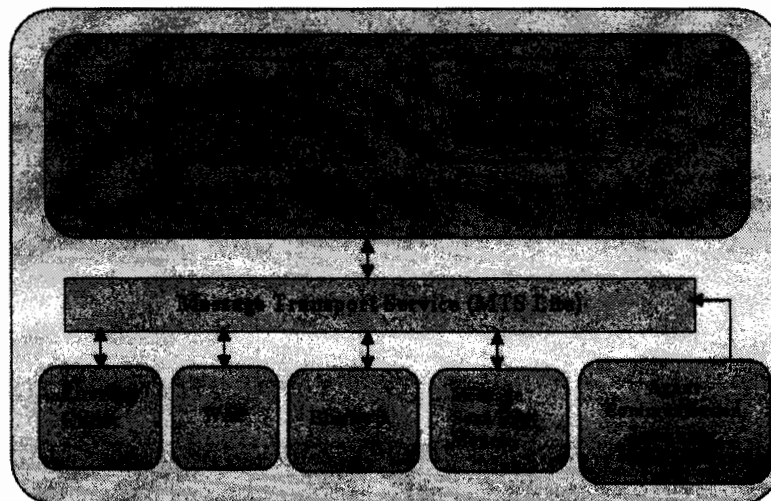


Fig 4.1: Architecture SAGE-Lite [3]

4.1.2. Light-weight Agent Management System (AMS-Lite)

According to FIPA specifications Agent Management System (AMS) is a mandatory component of the Agent Platform. The AMS covers the supervisory control over access to and use of the Agent Platform. Only one AMS exists in a single Agent Platform. The AMS maintains a directory of AIDs which contains transport addresses for agents that have been registered with the Agent Platform. The Agent Management System offers white page services to other agents.. Each agent must register with an AMS to get a valid AID [3].

Directory Facilitator (DF) is an optional component of Agent Platform, but it must be implemented as a Directory Facilitator service. The DF provides yellow pages services to other agents. Agents may register their services with the DF or query the DF to find out what services are available or have been offered by other agents. Multiple DFs may exist within an Agent Platform [3].

In SAGE framework, AMS and DF act as two different system agents. Operations of platform can be handled through a service called Visual Management Service (VMS) which is also a part of AMS-Lite. Portable devices may locate at different locations in an unpredictable way, and they have no prior knowledge about the available services and resources by other agents; they need to discover the available services on run time. So when our mobile device enters into new zone, record of all the agents residing on the device will also be shifted into a new zone through this local DF. Otherwise our device needs to remain connected with home agent platform [3].

Another reason was that when two devices will search for specific services using Bluetooth and there is not any local DF, then each device needs to connect with its server every time for this purpose.

4.1.3. Light-weight Message Transport Service (MTS-Lite)

In SAGE-LITE, the MTS-LITE module communicates through WAP and Bluetooth. The Message Send/Receive Module is responsible for the message buffering and envelope codec is responsible for the encoding of the envelope in Bit Efficient Representation.

4.1.4. Light-weight Agent Communication Language (ACL-Lite)

ACL-Lite is responsible for creation of a message that is understandable by all agents involved in the multi-agent system. Through this package all agents will create a message through some predefined rules defined by FIPA. And the message will be sent to the required destination. At the reception end, the agent will take its own decision based on the ACL Message [3].

4.1.5. ACL-Lite Message Structure

ACL-Lite is FIPA compliant so its message structure is same as defined by FIPA specifications. ACL-Lite message consists of a set message parameters. Precisely which parameters are needed for effective agent communication will vary according to the situation; the only mandatory parameter in all ACL-Lite messages is the Performative, although it is expected that most ACL-Lite messages will also contain sender, receiver and content parameters [4].

If an agent does not recognize or is unable to process one or more of the parameters or parameter values, it can reply with the appropriate not-understood message.

Some parameters of the message might be omitted when their value can be deduced by the context of the conversation. However, FIPA does not specify any mechanism to handle such conditions, therefore those implementations that omit some message parameters are not guaranteed to interoperate with each other.

4.1.6. Context-Awareness in SAGE-Lite

Another important aspect of the architecture is that it supports the context awareness feature. Context-Awareness is something to do with the location of the user or the environment in which the user is roaming. Such context aware systems give services to the user depending upon the surrounding environment. If the user is in a business environment, business related services would be provided to the user [3].

But the concept of context awareness emphasizes on the device based context awareness i.e. the services should be provided to a device, depending upon the capabilities of the devices. If the device does not support GUI, then the service that it should be provided should not have GUI too, or if it has lower processing power then the services with minimal features should be sent to the devices. For the purpose the device will first send its full specifications to the main container then SAGE-Lite will send the service to the device according to the specifications.

4.1.7. AMS-Lite Design

AMS-Lite is a mandatory component of SAGE-Lite. It controls the whole agent platform. When agent is created it needs to be registered with the Agent Platform [3]. It maintains the directory of Agent ID's. Every Agent will have a unique ID.

Functions of AMS-LITE

Following are the functions of AMS-Lite:

- register
- deregister
- search
- modify
- get description

According to the FIPA specifications every agent has a lifecycle associated with it.

Architecture of AMS-LITE

AMS-Lite manages the whole platform. Following are the components of AMS-Lite

- Manage DF-Lite
- Manage Agent Lifecycle
- Generate unique AIDs
- Manage Services
- Manage the Registered Agent Repository

4.1.8. Visual Management Service

VMS is a service that offers a graphical interface to the platform administration. The agent offers many services that show the state of the Agent Platform and it also offers various tools to request administrative action from the AMS-Lite and the DF-Lite [3].

Key Contributions of VMS:

The key contributions of VMS are

- Easy Navigation of platform.
- Easy Management of the platform.
- Easy Monitoring of the platform.
- Easy Inspection of the platform.

Functionalities of VMS

The graphical user interface provides the user with a lot of flexibilities and operations that can be performed on the agents residing in the platform and the platform itself [3]. These functionalities include:

- View registered agents.
- View Agent information.
- View Machine information.
- View Platform information.
- Create new agent.
- Suspend agent.

- Resume agent.
- Kill agent.
- Send message(s) to other agents.
- Shutdown Platform.

Fig 4.2 shows the registered agent which can be viewed through Visual Management Service, and the operations that can be performed on that agent e.g. Suspend, Deregister and Resume etc. Select a particular agent from the agent list. A menu box will appear and ask for the operations that need to be performed for that agent

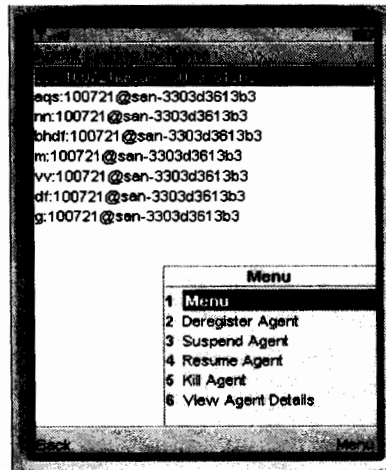


Fig 4.2: View of registered agents and operation that can be performed [3]

AMS-Lite Controller

It is the main component of AMS-Lite which controls all the sub-components of AMS-Lite [3]. VMS interacts with this module. It provides the object persistence. Whenever system is booted the entire registered agent becomes active. It manages the following modules of AMS-Lite:

4.2 Agent Lifecycle Manager

AMS-Lite can instruct the underlying platform to perform the following operations in order to manage the agent Lifecycle

- Suspend Agent
- Create Agent

- Resume Agent
- Execute Agent

Other operations of FIPA agent lifecycle will be performed when the light-weight agent platform (SAGE-Lite) will communicate with SAGE.

Following are the state of agent residing on SAGE-Lite [3]:

- **Active**

The MTS delivers messages to the agent as normal.

- **Initiated/Waiting/Suspended**

The MTS either buffers messages until the agent returns to the active state or forwards messages to a new location (if a forward is set for the agent).

- **Unknown**

The MTS either buffers messages or rejects them, depending upon the policy of the MTS and the transport requirements of the message.

The state transitions of agents can be described as:

- **Create**

The creation or invocation of a new agent.

- **Invoke**

The invocation of a new agent.

- **Suspend**

Puts an agent in a suspended state. This can be initiated by the agent or the AMS.

- **Resume**

Brings the agent from a suspended state. This can only be initiated by the AMS.

4.3 Managing DF-Lite

Lightweight directory facilitator basically provides the registration of agents and their services into the database. Whenever system boots up, DF-Lite is queried to get the record of previously registered agents (also called object persistence). DF-Lite is also responsible for parsing of record [3]. Following attributes of agents resides in DF-Lite

- name
- transport addresses
- description
- state
- owner
- locators

And following attributes of services reside in the DF-Lite

- serviceId
- name
- owner
- description
- state

4.4 Managing Services

Agents provide yellow page services to other agents. When one agent needs any service it gives the description of that service to Agent Controller. It in return searches the DF-Lite to check the availability of the service. If that service is not found on machine then it invokes the Bluetooth module and searches the devices that are in the range. If that service is found on any other device it respond to the Agent controller and then agent can directly interact with the agent providing that service through ACL messaging [3].

4.5 Proposed Solution

The multi agent layer is added to MIDET for data sharing. This multi agent layer contains DF– Lite, AMS – Lite, MTS – Lite, VMS - Lite and ACL - Lite [3]. Agents communicate through ACL - Lite. RMS manages the record store of a MIDLET and DF – Lite performs data manipulation on that record store through RMS. Agents and there services will be created through VMS-Lite. AMS manages all the agents on the platform.

Following figure proposes the architecture for record store sharing without making it public to all other MIDLETS.

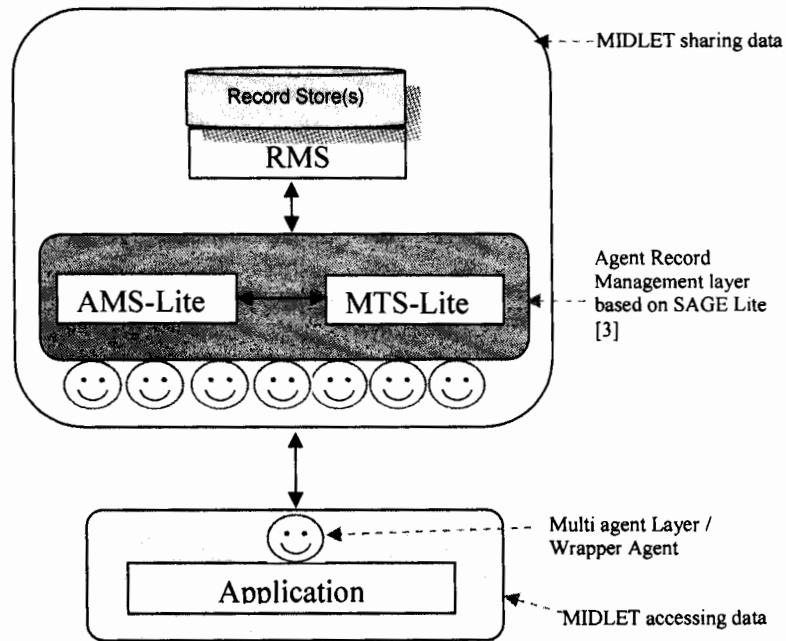


Fig 4.3: Record Store Sharing based on SAGE-Lite

In figure 4.3 “Agent Record Management layer” acts as a mediator between two MIDLETS sharing the data. The use of multi-agents system on this layer enhances the performance of the system, as all the actions on data (read, write, search etc) are divided among agents.

In the proposed technique actions (save, update, delete, search etc) on non shared or private data are published as services of the agents and any other agent can access data through these services. The requesting agent can belong to same or different MIDLET.

Two types of applications can access the data through agents:

Legacy systems applications can access data through the wrapper agent and multi-agent based applications can access data through their agents by communicating with the Agent Record Management Layer.

Following figure shows, how the agents on the Agent record Management Layer work:

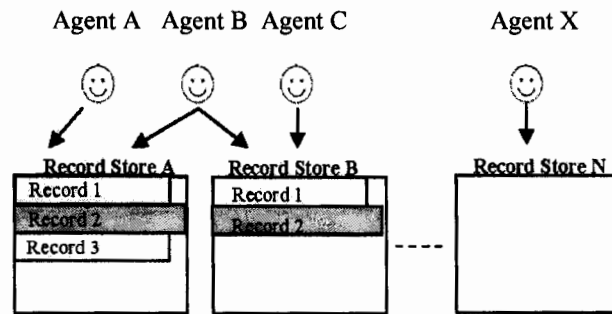


Fig 4.4: Agents accessing record stores.

“Agent A” can perform desired actions record store A. “Agent B” can perform desired actions on Record stores A & B. “Agent C” can perform actions on Record store B &so on.

Following figure shows, the record store sharing between different MIDLETS:

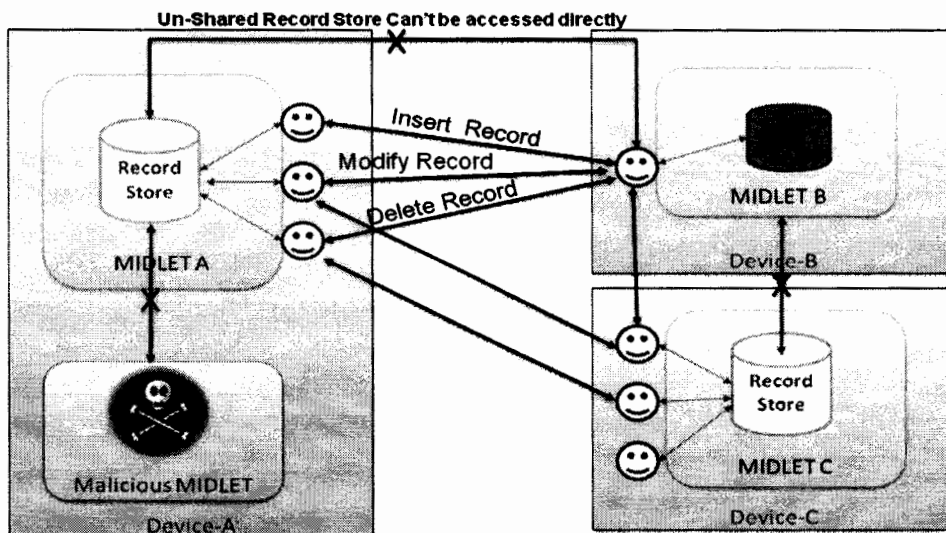


Fig 4.5: Agents accessing record stores from different devices.

The figure 4.5 shows data manipulation between four MIDLETS on three mobile phone devices. There is one Malicious MIDLET on device A. In this Malicious MIDLET Low level API 'RecordStoreFile' is used to access the shared record store of MIDLETS on the same device.

To prevent Malicious MIDLETS from accessing the Record Store of MIDLET A, we have changed the Mode property of MIDLET A's Record Store to AUTHMODE_PRIVATE. Now the Record Store is only accessible to MIDLET A. Here in "MIDLET A" we have introduced a new Layer named "Agent record Management Layer", this layer is based on SAGE LITE [3] Multi agent framework. We have created three agents for Save, Modify and Delete actions on record store. Only the Save, Modify and Delete actions can be performed through the agents of "MIDLET A". So MIDLET without agents cannot access the record store of "MIDLET A". "MIDLET A" is deployed on device A as shown in Fig 4.5.

On device B a legacy MIDLET named "MIDLET B" is deployed. A wrapper agent is created on "MIDLET B" so that MIDLET B can perform actions on record store of "MIDLET A". The wrapper agent of "MIDLET B" communicates with agents of "MIDLET A" through Agent Communication Language (ACL) [3] as shown in Fig 4.5.

On device C a "MIDLET C" copy of "MIDLET A" is deployed. Agents of "MIDLET C" communicate with agents of "MIDLET A" through Agent Communication Language (ACL) [3] as shown in Fig 4.5.

So the record store is shared between multiple MIDLETS on multiple devices without exposing to Malicious MIDLETS, which is our goal.

4.6 Implementation of Proposed Solution

To demonstrate the record store sharing using a multi-agent system approach we have implemented the proposed architecture by developing a Health Care application for mobile devices, based on SAGE-Lite framework [3]. This application while keeping the record store private provides access to specific MIDLETS on same or remote devices through its agents. This application has three parts Patient, Doctor and Nurse.

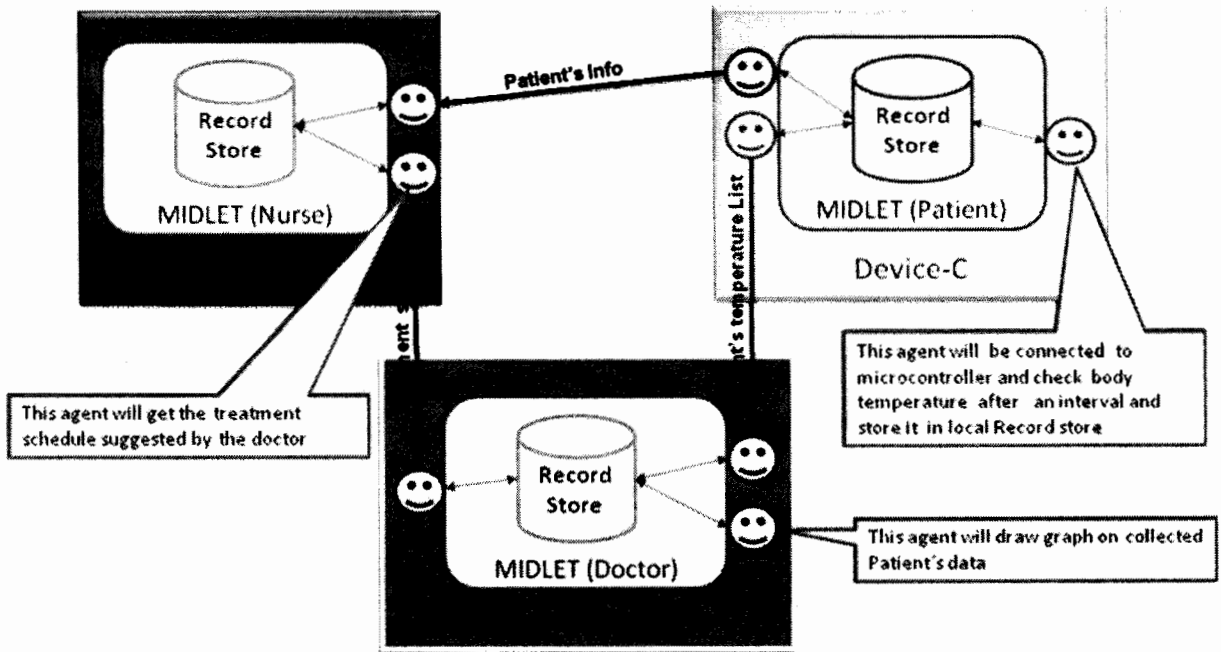


Fig 4.6: Application architecture based on proposed solution.

The application is deployed and tested on Sun Java Wireless Toolkit 2.5. The reason for choosing this Toolkit was its support for J2ME MIDP 2.0, CLDC 1.1 and Bluetooth API.

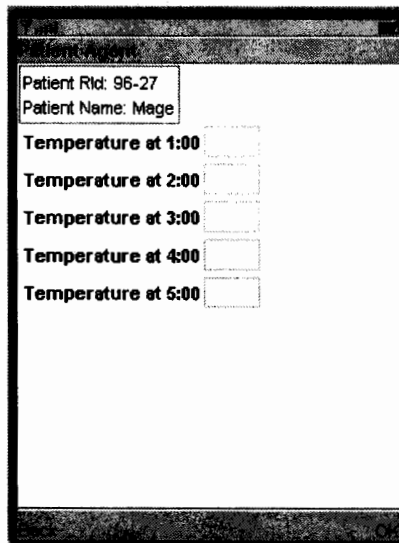
The application is divided into three main modules name as Patient, Doctor and Nurse. In Patient Module there are three agents; one that gets temperature of Patient from device (Microcontroller) or through manual form, Second agent serves the doctor's Agents requesting for Patient's temperature. Third one serves the Nurse's Agents requesting for Patient's information.

7/11-4907

In Doctor Module there are also three agents; one that gets list of daily temperature of Patient from Patient's Agent, Second agent draws graph on the collected data for doctor's analysis. Third one serves the Nurse's Agents requesting for treatment schedule suggested by the doctor.

Nurse Module contains two agents; one that gets Patient information from Patient's Agent, Second agent gets the treatment schedule from doctor's agent.

Following are some screen shots of application



The screenshot shows a mobile application interface with a dark header and footer. The main content area is white and contains the following text and input fields:

- Patient RId: 96-27
- Patient Name: Mage
- Temperature at 1:00
- Temperature at 2:00
- Temperature at 3:00
- Temperature at 4:00
- Temperature at 5:00

Fig 4.7: Screen for manual entry of Patient's temperature.

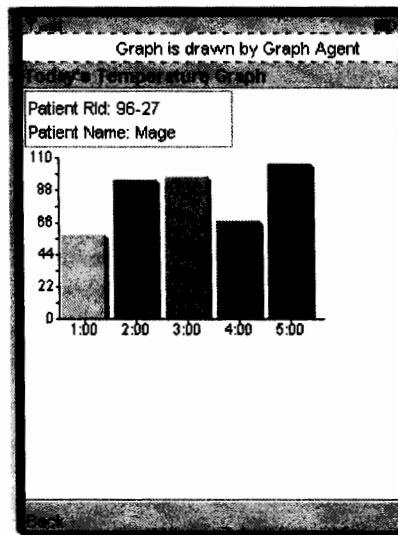


Fig 4.8: Screen to show the graph drawn by an Agent.

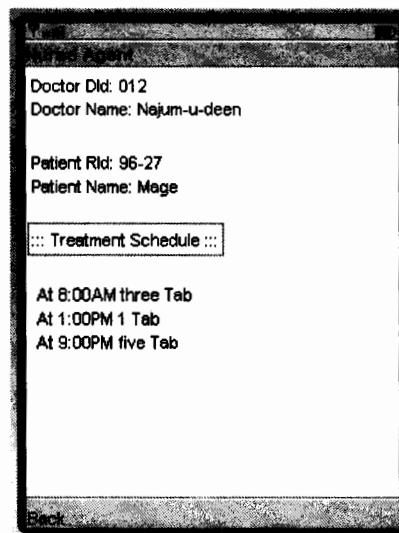


Fig 4.9: View Treatment Schedule Screen for Nurse.

On the basis of treatment schedule the Nurse will start the treatment of Patient.

Note: The Patient may be a person or an animal.

From the above example we have successfully implemented the modification and sharing of data among specified MIDLETS. As the record store was defined with PRIVATE authentication mode, so no other MIDLET suites were able to access that record store.

4.7 Advantages of proposed solution

- Data can be shared between multiple MIDLETS on same and different handheld devices.
- Light weight and fast data access, as task are divided between multiple agents.
- Open Source as SAGE Lite is going to be open source.
- Record is inaccessible to malicious MIDLETS.
- To provide an architecture that is independent of specific applications and is applicable for rang of J2ME applications

Chapter 5

EVALUATION & CONCLUSION

5. Evaluation

This section describes the criteria through which we intend to judge the success of our proposed work. The results were tested on simulator having specification of Nokia N73 device, which supports J2ME applications with Bluetooth API. However, all the devices having the following specifications will support proposed work:

- ❖ J2ME enabled device
- ❖ MIDP 2.0 JSR -118
- ❖ CLDC 1.1 JSR -139
- ❖ Bluetooth Functionality
- ❖ JSR-82 API support for Bluetooth

We have considered the following scenarios for evaluation purpose:

- i) Using the existing techniques for record store sharing and vulnerabilities caused by the existing technique.
- ii) Implementing the proposed technique (as detailed in Chapter 4) and who to overcome the vulnerabilities caused with the existing technique.

5.1 Testing Scenarios:

We used a Malicious MIDLET, which can read the information from the record store files of other MIDLETS having their AUTHMODE_ANY (i.e. visible to all other MIDLETS on that device). We used this MIDLET to test our system, as we are sharing the data with only specific set of MIDLETS. This malicious MIDLET uses the low level APIs to access the record store files having their shared mode enabled.

5.1.1 Scenario 1: MIDLet with Shared Record Store:

PatientWithSharedRS MIDLET:

This MIDLET is designed to demonstrate the vulnerabilities caused by making the record store mode to shared in which the record store of the MIDLET becomes accessible to all the MIDLETS on the device. In this scenario when the patient

information is entered and stored in the record store, then the malicious MIDLET should be able to access the information.

STEPS:

- 1) Enter the patient information in the aforementioned MIDLET, as shown in fig 5.2
- 2) Press the OK button on the screen of Fig. 5.2, the data will be saved in the record store of the MIDLET, as shown in Fig. 5.3.
- 3) Run the Malicious MIDLET, which reads the information from the Shared Record Store.

Expected Output:

Malicious MIDLET should be able to access the confidential information of the patient as stored in the record store. Refer to Fig. 5.5

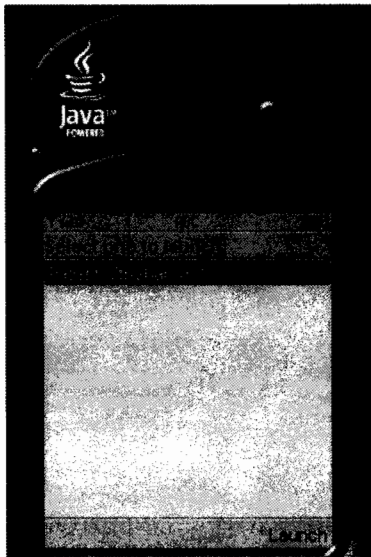


Fig5.1: MIDLET Launch

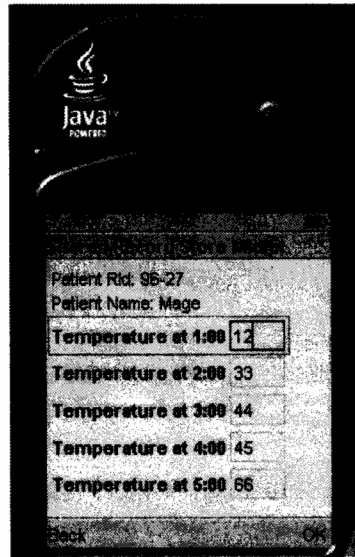


Fig 5.2: Data Entry Form

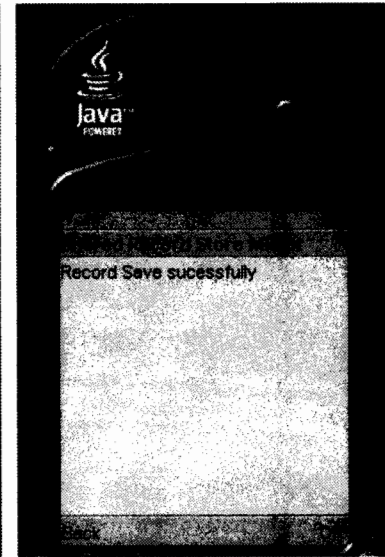


Fig5.3: Record Storage Info

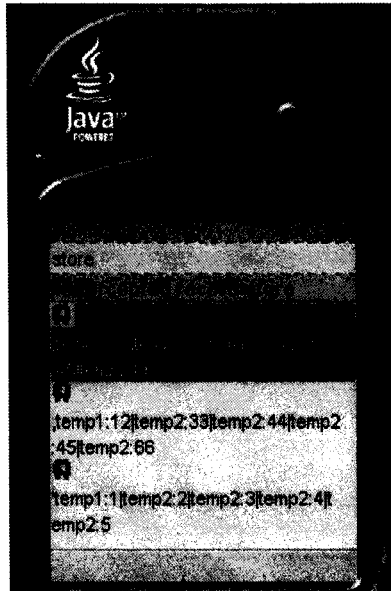
Malicious MIDLET:

Fig.5.4: Malicious MIDLET

5.1.2. Scenario 2: MIDLet with Private Record Store:**PatientAgent MIDLET:**

This MIDLET is developed to evaluate the success of our proposed work. In this MIDLET we have made the record store `AUTHENTICATION_MODE=PRIVATE` and thus made record store invisible to all the other MIDLETS present on the device. We have provided the access to record store of the MIDLET through a layer of multi-agents, so the malicious agents cannot access the record store directly.

STEPS:

- 1) Enter the patient information in the Patient MIDLET, as shown in fig 5.6
- 2) Press the OK button on the screen of Fig. 5.6, the data will be saved in the record store of the MIDLET (which is made private i.e. invisible to all the other MIDLETS), as shown in Fig. 5.7.
- 3) Run the Malicious MIDLET, which reads the information from the Shared Record Store.

Expected Output:

Malicious MIDLET should NOT be able to access the confidential information of the patient as stored in the record store. Refer to Fig. 5.8

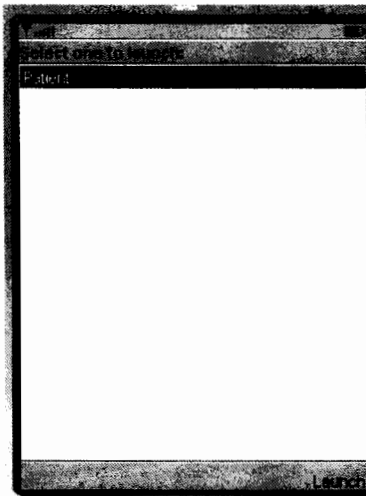


Fig 5.5 Patient MIDLET

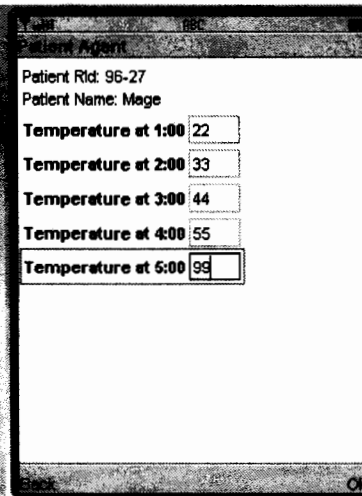


Fig 5.6 Data Entry Page

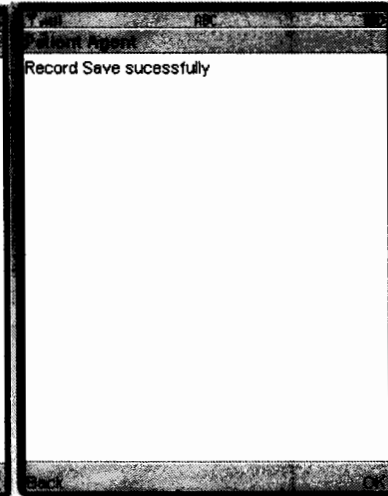


Fig 5.7 Storage Information

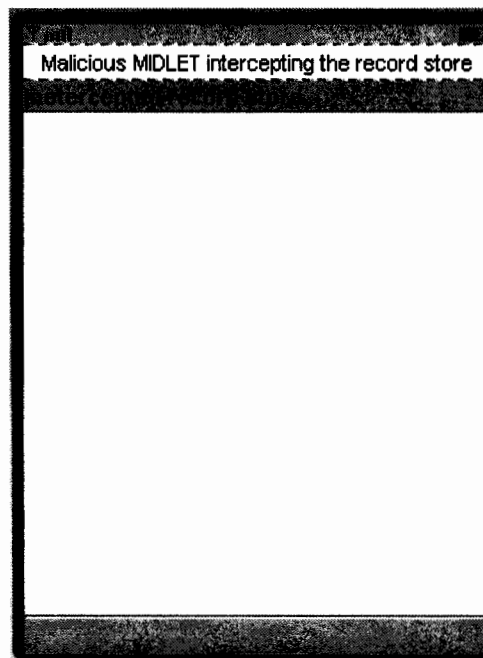
Malicious MIDLET:

Fig 5.8 Malicious MIDLET

5.2 Results & Discussion:

As discussed in the chapter 2, where we surveyed and analyzed different technologies for sharing the database in lightweight mobile devices. The results of analyses are given in the table-2.1 (chapter 2).

In continuation to the previous results of table 2.1 and the analysis of different parameters, we are presenting our proposed solution for the aforementioned problem, by doing a comparison with the same parameters.

Following table presents the side by side comparison of the proposed solution with other existing technologies.

Parameters	Technologies				
	Java Spaces	Mobile Spaces	Web Services	Oracle Lite	Proposed Solution
Device / J2ME Compatibility	No	Low	High	Low	High
Heterogeneity	No	Low	High	Low	Medium
Resource Hungry	Yes	Yes	Yes	Medium	No
Server Required	Yes	Yes	Yes	Yes	No
Open source	No	No	No	No	Yes

Table 5.1 Evaluation of technologies

- i) **Device / J2ME Compatibility:** The compatibility of our proposed work is high due to the availability of J2ME platform on different mobile devices. As SAGE-Lite supports J2ME platform so the proposed work can be implemented through a range of devices having J2ME platform enabled along with MIDP 2.0, CLDC 1.1 and API JSR-82 (for Bluetooth).
- ii) **Heterogeneity:** SAGE-Lite being a FIPA compliant Multi-Agent System supports heterogeneity,
- iii) **Resource Requirement:** Limited memory is required for the work proposed in the previous chapter. This is an advantage over the other technologies available as they require a considerable amount of memory and other resources.
- iv) **Server Requirement:** A specialized server is not required – as SAGE-Lite is a standalone framework, which is not dependent upon a server and can be implemented on different devices separately.

- v) **Open Source:** SAGE-Lite is an about to be an open source framework, which will be available to the multi-agent based community, facilitating the developers to use this framework for developing their applications.

5.3 Advantages of Research Work:

- Using a layer of multi-agents, we have shared the local persistent data on a device, to a specific set of MIDLETS.
- Along with the limited access of data to specific MIDLETS, We have restricted the actions (save, delete, search etc.) that can be performed on the data by invoking the services of agents.
- Our proposed architecture can easily be accommodated in the existing architecture of the application, where the record sharing is required.
- With our proposed approach we have enabled the record stores of a MIDLET to be shared not only on local device but also across remote devices.
- With the implementation of SAGE-Lite, we have improved the system's performance, by performing the desired tasks through multiple agents.

5.4 Conclusion

The growing trend of mobile computing opens door to different issues regarding the data manipulation on small handheld devices. The problem of record store sharing in J2ME is handled through a layer of multi-agent system. The proposed approach provides an environment that is independent of specific applications and can be applicable for different types of applications. J2ME data with this approach can be shared among specific j2me applications through their agents. The use of multi-agents improves the performance of data manipulation on the local or remote device, enabling the applications to give restricted access to their confidential information.

We have reviewed the research work for discussing the shortcomings in the J2ME MIDP record store for developing applications, where the information needs to be shared among specific set of MIDLETS. We then investigated different mechanisms

available for record sharing. The solution we provided makes the shared storage to the desired MIDLETS and improves the system performance. This technique can be used to share the information on the same as well on remote devices.

In evaluating the thesis in the large, we have provided a record store sharing architecture based on lightweight multi-agent system, which if used for development of a lightweight multi-agent system would provide developers to make the data sharing more easily and efficiently.

In evaluating the thesis in the small, we have proposed a two folded solution for lightweight multi-agent systems that are targeted to run on small handheld devices. Firstly, it makes the confidential information inaccessible to other MIDLETS on the local device, preventing the record store from unauthorized access. Secondly, it makes use of the multi-agent system, hence resulting in the improved system performance.

The proposed architecture is implemented on a J2ME based application of SAGE – Lite, which demonstrated the achievement of our proposed solution.

5.5 Future Work

Future work can be related to the concurrency control when the data is being accessed simultaneously by multiple users. The other area that needs to be explored is regarding the authorization and authentication of agents.

REFERENCES & BIBLIOGRAPHY

6. Bibliography

- [1] Sun's Java 2 Micro Edition Platform <http://java.sun.com/javame/index.jsp>
- [2] Databases and MIDP, Part 1: Understanding the Record Management System
<http://developers.sun.com/mobility/midp/articles/databaserms/>
- [3] Hafiz Farooq et. al: Persistent Architecture for Context Aware Lightweight Multi Agent System. The Fifth International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2006), Japan
- [4] D. Gelernter, "Generative Communication in Linda", ACM Transactions Programming Languages and Systems, Vol. 7, No. 1, January 1985, Pages 80-112.
- [5] JavaSpaces™ Service Specification, 1.2.1, Sun Microsystems 2002
- [6] Jini™ Architecture Specification, 1.2, Sun Microsystems, 2001
- [7] A. Kaminsky, "JiniME: Jini™ Connection Technology for Mobile Devices", Information Technology Laboratory Rochester Institute of Technology, August 2000.
- [8] T. Rybicki; J. Domaszewicz; "MobileSpaces -JavaSpaces for Mobile Devices"; Computer as a Tool, 2005. EUROCON 2005.The International Conference on Volume 2, Issue, 2005 Page(s):1076 – 1079
- [9] Philipp Bolliger; Marc Langheinrich; "Distributed Persistence for Limited Devices"; Inst. for Pervasive Computing ETH Zurich, Switzerland
- [10] Karun Bakshi, Oracle Database Lite 10gR2 Feature Overview, June 2006, Oracle Corporation, World Headquarters, 500 Oracle Parkway, Redwood Shores, CA 94065, U.S.A.
- [11] Yolanda Villate, Arantza Illarramendi, Evaggelia Pitoura, "Agent-Based and Mobile, External Storage for Users of Mobile Devices"
- [12] M. Debbabi, M. Saleh, C. Talhi and S. Zhioua: Security Evaluation of J2ME CLDC Embedded Java Platform, in Journal of Object Technology, vol. 5, no. 2, March–April 2006, pages 125–15
- [13] Anders Liljedahl: Evaluation of Multi-Agent Platforms for Ubiquitous Computing, JUNE 2004. SWEDEN

- [14] Giovanni Vigna, Richard A. Kemmerer : Evaluating the Security Of Three Java-Based Mobile Agent Systems Sebastian Fischmeister.
- [15] A Comparison of the Security Frameworks in Agent-Based Semantic Web
Xinyuan Deng
- [16] Sana Farooq, Sana Khalique, Aqsa Bajwa, Obaid Malik, Hafiz Farooq Ahmad, Hiroki Suguri, Arshad Ali, "SAGE LITE: An Architecture and Implementation of Light Weight Multi-agent System", IEEE Proc. of the Sixth International Symposium on ADS (ISADS07) 2007, pp 239-244, 21-23 March (2007).
- [17] Aqsa Bajwa, Sana Farooq, Obaid Malik, Sana Khalique, Hiroki Suguri, Hafiz Farooq Ahmad, Arshad Ali, "Persistent Architecture for Context Aware Lightweight Multi Agent System", Lecture Notes in Artificial Intelligence, Vol. 4411, pp. 57-69 (2007).
- [18] Abdul Ghafoor, Mujahid ur Rehman, Zaheer Abbas Khan, H. Farooq Ahmad, Arshad Ali, "SAGE: Next Generation Multi-Agent System", Proc. Of IEEE International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04), pp.139-145, Vol. 1, (2004).
- [19] Syed M. Ali Shah, Naseer Gul, Hafiz Farooq Ahmad, Rami Bahsoon "Secure Storage and Communication in J2ME based lightweight Multi-Agent System", Proc. Of International Conference on Agents and Multi-Agents Systems: Technology and Applications: South Korea 2008.

Appendix A
Research Paper

A Research Paper titled as “Shared Storage in J2ME: A Multi-Agent System Approach” has been accepted in the 3rd IEEE International Workshop on Engineering Semantic Agent Systems (ESAS 2008) in conjunction with COMPSAC 2008.

THEME OF THE WORKSHOP

Applying Semantic Web Technologies in Research and Development of Software Agents, Mobile Agents and Multi-Agent Systems towards INTEGRATING THE DISTRIBUTED WORLDS.

Semantic web technologies render dynamic, heterogeneous, distributed, shared content equally accessible to human reader and software agents. Distributed agents functioning autonomously can utilize semantic Web content to gather and aggregate knowledge, reason and infer new results towards achieving their goals and generating new knowledge. Such knowledge in turn may be disseminated and used to achieve the shared goal of the agents system. Here the vision is to achieve a synergy with multi-agent systems (MAS) technologies whereby both semantics and agents will be equally in the center stage.

ESAS workshop series aims at garnering the synergy of both technologies by taking up both the semantic web and the agent aspects of the common research issue. Topics of interest span a wide spectrum in both theory and practice of autonomous semantic agents, context-aware intelligent agents, agents as semantic web services, software agents, mobile agents, agent architectures, multi-agent systems, agent communities, cooperation and goal seeking through sharing policy and ontology, safety & security in systems, and so on.

Mobile agent and MAS technologies are crucial in realizing multi-party dynamic application systems. Semantic Web technologies augment MAS by enabling agents with functioning based on the semantics of their mission and of the world around them. Agents, implemented as Web services in developing distributed control and processing applications, entail interesting consequences such as situation awareness, semantic composition of services, context sensitive long-lasting transactions, effecting service policies and quality levels, etc. Complex applications could become realizable with novel features such as factory floor automation for flexible production, collaborative discovery of uncharted geography (for example, cooperative labyrinth

discovery), traffic management and info dissemination with facilitation of emergency services,
financial markets forecasting with optimization of portfolio gain.

Shared Storage in J2ME: A Multi-Agent System Approach

Muhammad Ainan Sadiq
International Islamic University Islamabad
ainansadiq@gmail.com

Syed Muhammad Ali Shah
International Islamic University Islamabad
alishah_ph@yahoo.com

Abstract

Current research trend towards mobile computing emphasizes the need for distribution of data among various clients in wireless environment. Sharing the data in J2ME opens the data to be accessible for all MIDLETS present on the device, hence creating a number of vulnerabilities to the confidential information that is intended to be shared with specific MIDLETS, e.g. a malicious MIDLET can delete, copy or move the Shared Record Store, in J2ME based application. This project aims at overcoming record store sharing problem through the concept of Multi-Agent System by providing a shared storage with explicit access to authenticated & authorized MIDLETS. With this mechanism data can be shared among specific MIDLETS. The implementation uses the SAGE-Lite framework as a solution to our proposed work.

1. Introduction

Today's computing trend is more towards ubiquitous computing, which is diverging number of database applications to mobility and facilitating the users to achieve the usefulness of anywhere-anytime computing. Different m-commerce applications are being deployed on small handheld devices, to facilitate the transactions and data processing in wireless environment.

Java 2 Micro Edition (J2ME) [1] is one of the leading technology in handheld device applications. One of the J2ME features is that it provides built in caching mechanisms for locally storing data on a mobile device. The Mobile Information Device Profile (MIDP) adds APIs (Application Programming Interface) for user interaction, network connectivity, and persistent storage. Two profiles have been developed for J2ME: MIDP 1.0 and 2.0.

J2ME has a Record Management System (RMS), an Application Programming Interface (API) that provides persistent storage on local device. A few MIDP-enabled handheld devices support the traditional file system; RMS is the only feature in J2ME for local data

storage and is essential to writing any application that relies on local persistent data [2]. RMS stores all the records in a file with extension ".db" called record store. The application developed in J2ME for mobile devices is called a MIDLET and each MIDLET suit (group of related MIDLETS) can own one or more record stores.

Due to the limitations of J2ME Record Management System (RMS), when the data of record store is set to shared mode then it can be accessed by all the MIDLETS present on the device and hence creating a number of vulnerabilities to the confidential information. The proposed work targets to share the data among specified MIDLETS on local device as well as on different devices and to protect the data from malicious MIDLETS' access. The solution provided uses the concept of multi-agent systems for developing a mechanism for sharing data, which will allow access to private data of a MIDLET. This is achieved by introducing a layer of multi-agents that handles the whole data manipulation from inside or outside the device.

Two MIDLETS can exchange data through their agents. An agent can perform desired action only on its MIDLET data. In order to make an action on different MIDLET data the agent of first MIDLET sends a command to the agent of second MIDLET, which then responds to the agent of first MIDLET with the desired results. In this way the shared storage vulnerabilities are minimized. The benefit of using multi-agents system approach is to save redevelopment of legacy systems, only a wrapper agent needs to be developed. Secondly the data remains protected i.e. inaccessible to malicious MIDLETS and is only shared among authorized MIDLETS. Furthermore, performance is also increased by using multi-agents instead of using other techniques discussed in related work.

Rest of the paper is organized as follows: section 2 describes the related work. Section 3 describes the architecture of J2ME Record Management System (RMS) & its vulnerabilities. Section 4 gives the proposed architecture for J2ME record store sharing. Section 5 explains the evaluation criteria. Section 6 includes the conclusion and hints on the future work.

2. Related Work

Designing and implementing complex software systems e.g. m-commerce and other intelligent applications for handheld devices is rapidly growing. Persistence and distribution of data is crucial for these systems, as the small handheld devices are not as powerful as that of their counterpart – desktop machines.

Keeping in view the constrained environment and limited resources on these devices a number of mechanisms are provided for data persistence on these devices, e.g. J2ME provides Record Management System (RMS) for locally storing the data on the device. RMS is a system for managing records in J2ME. A record is an individual data item.

The records in RMS consist of a variable length binary field. This is contrary to the typical Database Management System (DBMS). The validation and data consistency checks cannot be applied to the records in RMS. The application developers are required to apply validation and data consistency checks to interpret the contents of records in RMS [2].

We find a number of alternatives for data sharing among local and remote handheld devices. Tuple-space is one of the well known and elegant ways of sharing information in distributed environment among communicating parties where processes write and read tuples [4].

JavaSpaces [5] implements the concept of tuple space through Java programming language. It is a service of Jini [6], which forms a distributed network of clients and services. JavaSpaces provide an easy way for communications facility in mobile distributed applications. Due to the fact that Jini technology is dependent on Remote Method Invocation (RMI), which is not supported by number of handheld devices, so JavaSpaces is not suitable for distributed mobile applications. Moreover, JavaSpaces requires a resource rich environment.

In order to make the devices Jini compatible a number of different approaches were defined. These include modifying the Java virtual machine, modifying Jini, or introducing a non-standard proxy. In JiniME [7], a J2ME virtual machine is changed to make MIDP devices Jini capable. The approach severely restricts the set of devices, on which applications can run. MobileSpaces, on the other hand, require no changes to the standard J2ME environment [8].

Like JavaSpaces, applications developed using MobileSpaces capture events through notify() method. Each time when event is triggered regarding

notifications about tuples, notify method is called, resulting in resource hungry application [8]. Although a lightweight framework TinyDB was developed but its main limitation is that it does not provide distributed storage [9].

Another lightweight storage system based on serialization framework allows MIDP enabled J2ME devices to store data on local as well as remote storage spaces requiring similar semantics. [9], so this framework does not support heterogeneous environment.

Different Lightweight DBMSs like Oracle Database Lite 10g, which is a unified and a comprehensive environment for development and deployment of high impact solutions intended for mobile and lightweight environments. It is an extension to Oracle Database 10g for enterprises focusing at increased employee throughput, reduced operative cost, and enhanced client satisfaction. It provides an extension to grid environment for mobile and embedded devices, allowing enterprise data accessible to remote employees even being offline. Moreover, it gives data synchronization permitting users to reliable and secure exchange of data with corporate Oracle Database [10]. The limitation of using third party DBMS includes compatibility issues. Its server and client requirements are too resource demanding for mobile environment. Besides, it requires a license to be purchased for every user, hence not a cost effective solution.

Different multi-agent system platforms exist that support FIPA (Foundation for Intelligent Physical Agents – A standard setting body for MAS) and mobility e.g. MicroFIPA OS [13], JADE-LEAP [14], Grasshopper [13]. We have chosen the SAGE-Lite platform because it is stand alone framework for the lightweight devices and provides the features of robustness, fault tolerance, object persistence and context awareness [3]. The related work in field of multi-agent system for developing applications for small handheld devices is found in Locker Rental Service [11], in which a persistent storage medium on a static device is preserved for different agents known as locker place. All agents communicate with their respective locker agents to operate on their data. Here the author had recommended multi-agent system as a middle-ware between a MIDLET and Locker place. Locker rental service provides a centralized storage mechanism, which has a number of issues such as maintenance problems of a single database, availability, a separate device for storage etc.

3. Architecture of J2ME Record Store Management System (RMS)

The architecture of J2ME record management system (RMS) contains small footprints for storing and manipulating persistent data on local device.

3.1. Record Stores

A record store is chronological collection of records associated to a record store. Every record can be accessed through the record store. In fact, record store ensures that records are read and written individually, with little possibility of data corruption.

RMS maintains unique numeric sequence number for each record in a record store known as Record ID. Maximum 32 Unicode characters unique name is defined for record store within MIDLET suite creating it. Record Store sharing mechanism is not provided in MIDP 1.0, while this mechanism is provided in MIDP 2.0, which allows record stores to be shared among other MIDLET suites. In each case the record store is recognized by the name of MIDLET suite, its vendor name and the record store name itself.

Time-stamp and version information is also maintained by record stores to enable applications to discover when it was last modified [2].

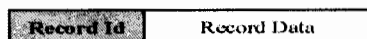


Fig 1: Internal Architecture of Record Store

3.2. Record Stores Sharing

Record stores are made accessible to all other MIDLETs using their shared property. The default property AUTHMODE_PRIVATE allows record store to be accessible only to the MIDLET suite that created it. Record store can be shared by changing the property to AUTHMODE_ANY. When the record stores are shared they can be writable or read-only. If the shared data is confidential, then it is not secure and can be read by all MIDLETs on the device. Also if the data is shared in read/write mode then it is more vulnerable to malicious MIDLETs on the device [12]. In the shared mode the MIDLETs from remote devices cannot access the shared data directly.

3.3. Shared Storage Vulnerabilities

Confidential information can be susceptible to an attack outside the RMS, e.g. it can be accessed and manipulated from device utilities (without using a

MIDLET), which is a serious problem, e.g. The whole Record Store can be deleted by calling the deleteFile() method of the class RecordStoreFile [12]. This is shown below in the figure:

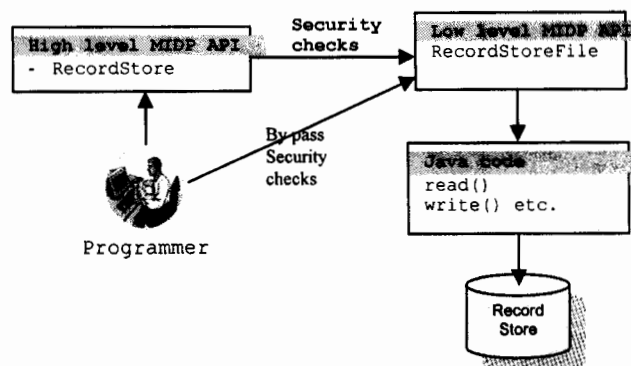


Fig 2 Shared record store vulnerabilities [12]

Development of applications for handheld devices poses some problems when the data of one MIDLET is required to be shared among multiple MIDLETs. Debbabi et al. [12] provide the security evaluation of J2ME platform and explain the vulnerabilities that are associated with the shared storage of data. Non-shared record stores can be accessed and modified by the MIDLETs creating them, but the shared record stores can be accessed from any MIDLET on the device. So MIDLETs cannot share their record stores with only a specific subset of MIDLETs. Shared record stores are vulnerable to any attack from outside the Record Management System (RMS) of J2ME.

The growing trend to introduce the personal agents on mobile devices gives rise to the need of a framework that should facilitate developers to implement agents on mobile devices. Existing multi agent system frameworks (JADE-LEAP, Grasshopper, Micro FIPA-OS etc.) do not provide robustness, context awareness and persistence [3].

4. Multi-Agent Based Record Sharing Architecture

In order to solve the problem mentioned in previous section, we use the concept of Multi-Agents System. In multi-agents system single task is divided among multiple agents on the same or different platforms.

We propose architecture on the basis of an existing FIPA compliant multi-agents system framework, namely SAGE-Lite [3]. SAGE-Lite is a lightweight context aware multi-agents system, which senses the capabilities of the lightweight devices and reacts accordingly. Existing work on secure agents' communication in SAGE-Lite framework [16] gives an advantage of agent security, by restricting unauthorized

agents to misuse the information. Keeping the record store property as AUTHMODE_PRIVATE makes it inaccessible to all other MIDLETS.

In our proposed technique actions (save, update, delete, search etc) on non-shared or private data are published as services of the agents and any other agent can access data through these services. The requesting agent can belong to same or different MIDLET on local or remote device. This is due to the fact that SAGE-Lite enables agents' communication on local or remote devices [3].

Following figure proposes the architecture for record store sharing without making it public to all other MIDLETS.

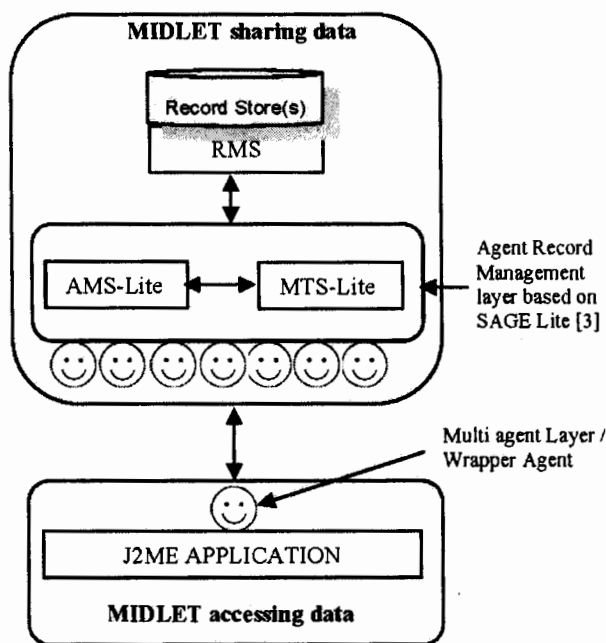


Fig 3 Record Store Sharing based on Multi-Agents

In the above figure Agent Record Management layer acts as a mediator between two MIDLETS sharing data with each other. The use of multi-agents system on this layer enhances the performance of system, as all the actions on data (read, write, search etc) are divided among agents.

Two types of applications can access the data through agents: Legacy systems applications can access data through the wrapper agent and multi-agent based applications can access data through their agents by communicating with the Agent Record Management Layer.

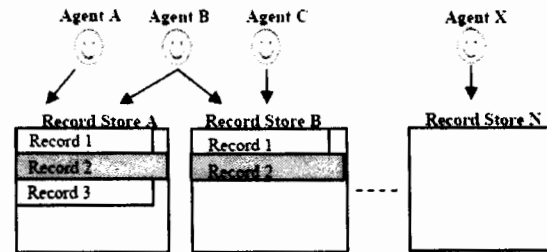


Fig 4 Agents accessing record stores

The above figure shows, how the agents on the Agent Management Layer work.

“Agent A” can perform desired actions on record store A.

“Agent B” can perform desired actions on Record stores A & B.

“Agent C” can perform actions on Record store B & so on.

5. Evaluation

This section describes the criteria through which we intend to judge the success of our proposed work. We have considered the following points for evaluation purpose:

- i. Using a layer of multi-agents, we have shared the local persistent data on a device, to a specific set of MIDLETS.
- ii. Along with limited access of data to specific MIDLETS, we have restricted the actions (save, delete, search etc) that can be performed on the data by invoking the services of agents.
- iii. Our proposed architecture can easily be accommodated in the existing architecture of the application, where the record sharing is required.
- iv. With our proposed approach we have enabled the record stores of a MIDLET to be shared not only on local device but also across remote devices.

To demonstrate record store sharing using the multi-agents system approach we have implemented the proposed architecture by developing an auction application for mobile devices, based on SAGE-Lite framework [3]. This application while keeping the record store private provides access to specific

MIDLETs on same or remote devices through its agents. This application has three parts Advertisement, Search Item and Item Status.

The application is deployed and tested on four NOKIA N70 mobile phones. The reason for choosing this device was its support for J2ME MIDP 2.0, CLDC 1.1 and Bluetooth API. We named those phones as Dev-A, Dev-B, Dev-C and Dev-D. On devices Dev-A, Dev-B, Dev-C we advertised three different Cars with their specification (i.e. make, model, color etc.), expected price and condition. On device Dev-D we applied different queries e.g. display records of car price less or equal to the value of 10,00,000/-. Car condition should be good and specification is corolla etc. The devices Dev-A, Dev-B, Dev-C were sharing their data with Dev-D. Upon receiving the desired results we booked the Car advertised on device Dev-B by changing its status from FOR-SALE to BOOKED. The agents on two devices Dev-B and Dev-D interchanged the data for booking.

From the above example we have successfully implemented the modification and sharing of data among specified MIDLETs. As the record store was defined with PRIVATE authentication mode, so no other MIDLET suites were able to access that record store.

Fig 5 Advertise an Item

Name	Expected Price (Rs)
1. Samsung X700	5500
2. Nokia N70	9950
3. Corolla 2.0D	1100000

Fig 6 View Advertised Items

Fig 7 View Status of Item

Fig 8 Search Item

Fig 9: Search Results

Fig 10: Book Items

6. Conclusion & Summary

The growing trend of mobile computing opens door to different issues regarding the data manipulation on small handheld devices. The problem of record store sharing in J2ME is handled through a layer of multi-agents system. The proposed approach provides an environment that is independent of specific applications and can be applicable for different types of applications. J2ME data with this approach can be shared among specific J2ME applications through their agents. The use of multi-agents improves the performance of data manipulation on the local or remote device, enabling the applications to give restricted access to their confidential information.

Future work can be related to the concurrency control where multiple users are accessing the data simultaneously. The other area that needs to be explored is regarding the authorization and authentication of agents.

6. References

- [1] Sun's Java 2 Micro Edition (J2ME) Platform: <http://java.sun.com/javame/index.jsp>.
- [2] Databases and MIDP, Part 1: Understanding the Record Management System <http://developers.sun.com/mobility/midp/articles/databasesrms/>
- [3] Hafiz Farooq et. al: "Persistent Architecture for Context Aware Lightweight Multi Agent System". The Fifth International Joint Conference on Autonomous Agents & Multi-Agent Systems, Japan, 2006.
- [4] D. Gelernter, "Generative Communication in Linda", ACM Transactions Programming Languages and Systems, Vol. 7, No. 1, January 1985, Pages 80-112.
- [5] JavaSpaces™ Service Specification, 1.2.1, Sun Microsystems 2002.
- [6] JiniT Architecture Specification, 1.2, Sun Microsystems, 2001.
- [7] A. Kaminsky, "JiniME: Jini™ Connection Technology for Mobile Devices", Information Technology Laboratory Rochester Institute of Technology, August 2000.
- [8] T. Rybicki; J. Domaszewicz; "MobileSpaces -JavaSpaces for Mobile Devices"; Computer as a Tool, 2005. EUROCON 2005.The International Conference on Volume 2, Issue, 2005 Page(s):1076 – 1079
- [9] Philipp Bolliger; Marc Langheinrich; "Distributed Persistence for Limited Devices"; Inst. for Pervasive Computing ETH Zurich, Switzerland.

[10] Karun Bakshi, "Oracle Database Lite 10gR2 Feature Overview", June 2006, Oracle Corporation, World Headquarters, 500 Oracle Parkway, Redwood Shores, CA 94065, U.S.A.

[11] Yolanda Villate, Arantza Illarramendi, Evaggelia Pitoura, "Agent-Based and Mobile, External Storage for Users of Mobile Devices".

[12] M. Debbabi, M. Saleh, C. Talhi and S. Zhioua: Security Evaluation of J2ME CLDC Embedded Java Platform, in

Journal of Object Technology, vol. 5, no. 2, March–April 2006, pages 125–15.

[13] Sebastian Fischmeister, Giovanni Vigna, Richard A. Kemmerer, "Evaluating the Security Of Three Java-Based Mobile Agent Systems", Volume 2240/2001, 2001

[14] Anders Liljedahl, "Evaluation of Multi-Agent Platforms for Ubiquitous Computing", JUNE 2004. SWEDEN

