INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®
Design and Implementation of Distributed File Access for Mobile Devices

Jing Li

A Major Report in The Department of Computer Science

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Concordia University Montreal, Quebec, Canada

August 2002

© Jing Li, 2002
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
Abstract

The objective of this project is to explore a proposal to provide seamless file transferring between mobile device and remote servers, while the mobile device is moving among local stations based on IP network. At first, three proposals are presented. One of the best proposals (the third one) is selected for implementation. The implementation is done by taking advantage of several prevailing software techniques: J2ME, RMI and Java Servlet. J2ME is a platform to simulate the mobile devices (PDA). RMI is a Java package to construct a distributed system, which is an interpreter among the mobile device, local stations and remote server. The protocol used in the communication between mobile device and local station is HTTP, which is supported by Java Servlet.

There are two challenges in the implementation of the proposal. The first one is how to choose the techniques to connect mobile devices and the fixed IP stations and server. J2ME and RMI work perfectly with each other, in the simulation of the mobile connectivity among a distributed system. The second challenge is how to design the system in order to guarantee the file transferring seamlessly. Making three separate classes, MobileAgent, Placeholder and Requestholder, successfully solves this problem.
By Using the above three java based techniques, the result of the implementation of the proposal shows a good performance in simulating mobile file transfer between mobile devices and file servers.
Acknowledgements

I would like to express my sincere gratitude to my supervisor, my mentor – Dr. R. Jayakumar, for accepting me as his student, instructing me so much in this field.

I am also thankful to Dr. T. Radhakrishnan. Without his contribution, this report would not have been possibly completed.

Specially, I would like to thank my dearest parents and my beloved husband. Their greatest love is always my invincible backup.
Table of Contents

1. INTRODUCTION .................................................................................................................. 1

2. RESEARCH ON MOBILE DEVICES AND DISTRIBUTED NETWORK SYSTEM ....................... 4
   2.1 TERMINOLOGY ................................................................................................................. 4

3. THE CONCEPT OF MOBILE DISTRIBUTED FILE ACCESS SYSTEM (MDFAS) ....................... 6
   3.1 DEFINITION OF MDFAS .............................................................................................. 6
   3.2 REQUIREMENT OF MDFAS ......................................................................................... 6

4. SOLUTIONS ............................................................................................................................ 8
   4.1 SOLUTION 1 ................................................................................................................... 8
      4.1.1 Functionality of the system ......................................................................................... 9
      4.1.2 Scenario of the system ............................................................................................... 10
      4.1.3 Mobile Devices and Middleware(FA, HA) ................................................................. 11
   4.2 SOLUTION 2 .................................................................................................................... 11
      4.2.1 Problem Statement ................................................................................................... 11
      4.2.2 Functionality of the system ....................................................................................... 12
      4.2.3 Scenario of the system ............................................................................................... 13
   4.3 SOLUTION 3 .................................................................................................................... 14
      4.3.1 Architecture of the system ......................................................................................... 14
      4.3.2 System functional requirement ................................................................................... 16
      4.3.3 System Non-Functional requirements ......................................................................... 17
      4.3.4 Scenario of the system in FA .................................................................................... 17
      4.3.5 Use case design .......................................................................................................... 19
      4.3.6 Class diagram design ................................................................................................ 22
      4.3.7 Sequence diagram of the design ................................................................................ 23

5. IMPLEMENTATION ............................................................................................................... 29
   5.1 REQUIREMENT FOR SYSTEM DEVELOPMENT: ............................................................ 29
      5.2.1 Class MobileAgent .................................................................................................... 29
      5.2.2 Class PlaceHolder .................................................................................................... 30
      5.2.3 Class PlaceHolderImpl ............................................................................................ 30
      5.2.4 Class RequestHolder ............................................................................................... 30
      5.2.5 Class Mobile ............................................................................................................. 30
      5.2.6 Class HomeAgent .................................................................................................... 31
   5.3 HOW TO INSTALL THE SOFTWARE FOR THE SYSTEM .............................................. 31
   5.4 HOW TO RUN THE SYSTEM .......................................................................................... 32

6. TEST ....................................................................................................................................... 39
   6.1 OBJECTIVES ................................................................................................................... 39
   6.2 DETAIL TESTING IN THIS PROJECT ............................................................................. 39

7. CONCLUSION AND FUTURE WORK .................................................................................. 47
List of Figures

FIGURE 3-1 REQUIREMENT OF MDFAS ......................................................... 7
FIGURE 4-1 SYSTEM ARCHITECTURE OF SOLUTION 1 ................................. 9
FIGURE 4-2 DIAGRAM OF MOBILE DEVICES AND MIDDLEWARE ............... 11
FIGURE 4-3 GSM IN MOBILE DEVICES FILE ACCESS SYSTEM .................. 12
FIGURE 4-4 SYSTEM ARCHITECTURE OF SOLUTION 3 ............................... 15
FIGURE 4-5 SYSTEM CONSTRUCTION ..................................................... 18
FIGURE 4-6 BEFORE AGENT MOVING .................................................. 18
FIGURE 4-7 AFTER AGENT MOVING .................................................... 19
FIGURE 4-8 USE CASE DIAGRAM OF FOREIGN AGENT .............................. 20
FIGURE 4-9 USE CASE DIAGRAM OF HOME AGENT ............................... 20
FIGURE 4-10 USE CASE DIAGRAM OF MOBILE ............................... 21
FIGURE 4-11 MOBILEAGENT CLASS DIAGRAM .................................... 22
FIGURE 4-12 MOBILE CLASS DIAGRAM ............................................. 22
FIGURE 4-13 HA & FA CLASS DIAGRAM ........................................... 23
FIGURE 4-14 CONNECT FA1 SEQUENCE DIAGRAM .................................. 24
FIGURE 4-15 CONNECT FA2 SEQUENCE DIAGRAM .................................. 25
FIGURE 4-16 DATA REQUEST IF AGENT EXIST SEQUENCE DIAGRAM ......... 26
FIGURE 4-17 DATA REQUEST IF AGENT NOT EXIST SEQUENCE DIAGRAM .... 27
FIGURE 5-1 HA(orchid.cs.concordia.ca) SIDE SCREEN .................................. 33
FIGURE 5-2 FA1(sunset.cs.concordia.ca) SIDE SCREEN .................................. 34
FIGURE 5-3 FA2 (dahlia.cs.concordia.ca) SIDE SCREEN .................................. 35
FIGURE 5-4 MOBILE SIDE SCREEN 1 ..................................................... 36
FIGURE 5-5 MOBILE SIDE SCREEN 2 ..................................................... 37
FIGURE 5-6 J2ME WIRELESS TOOLKIT 1.0.4 SCREEN ................................ 38
FIGURE A-1 THE FLOW CHART OF METHOD Process .................................. 52
FIGURE B-1 MOBILE DEVICES CLIENT USE CASE DIAGRAM .................. 57
FIGURE B-2 FOREIGN AGENT USE CASE DIAGRAM ................................ 58
FIGURE B-3 HOME AGENT USE CASE DIAGRAM .................................... 58
FIGURE B-4 OBJECT DIRECTORY SERVER USE CASE DIAGRAM ................. 59
FIGURE B-5 OBJECT SERVER USE CASE DIAGRAM .................................. 59
FIGURE B-6 REGISTRYMANAGER CLASS DIAGRAM ............................... 60
FIGURE B-7 SESSIONSTATEREQUEST & DATAREQUEST CLASS DIAGRAM .. 60
FIGURE B-8 LOCATIONHANDLER CLASS DIAGRAM .................................. 60
FIGURE B-9 MOBIL DEVICEREQUESTHANDLER CLASS DIAGRAM ............. 61
FIGURE B-10 MOBILEDEVICEDATAPROCESSOR CLASS DIAGRAM .............. 62
FIGURE B-11 SESSION BEGIN & END SEQUENCE DIAGRAM ..................... 63
FIGURE B-12 LOCATION & DATA SEQUENCE DIAGRAM .......................... 64
FIGURE B-13 HANDLEREGISTRYREQUEST SEQUENCE DIAGRAM ............ 64
1. Introduction

Although the Internet offers access to information sources worldwide, typically we do not expect to benefit from that access until we arrive at some familiar point—whether home, office, or school. However, the increasing variety of wireless devices offering IP connectivity, such as PDAs, handhelds, and digital cellular phones, is beginning to change our perception of the Internet.

To understand the contrast between the current realities of IP connectivity and future possibilities, consider the transition toward mobility that has occurred in telephony over the past 20 years. An analogous transition in the domain of networking, from dependence on fixed points of attachment to the flexibility afforded by mobility, has just begun.

Mobile computing and networking should not be confused with the portable computing and networking we have today. In mobile networking, computing activities are not disrupted when the user changes the computer’s point of attachment to the Internet. Instead, all the needed reconnection occurs automatically without any interaction [15].

Considering the above fact, it is worth to setup a system connecting mobile devices and fixed local stations based on IP network. Three proposals are presented in this paper to show the possibility of the system for this purpose. The first one is a broadcasting based solution. The location of a mobile device is
broadcasted to all the adjacent local stations, such that the mobile device immediately knows where mobile device moves from, when mobile device roams to local station. Although using a centralized station can optimize the solution, it introduces other new problems such as bottleneck and session state movement. The second solution gives a different way of describing the mobile location, which gives the coordinate (X,Y) of a mobile device by means of Global Position System (GPS). The third solution three is a query-based one. When a mobile device is moving around local stations, the location of the mobile device is always stored and updated in one remote server. The local station has to query the remote server to get the location where the mobile device moves from.

After consideration of various aspects, the third proposal is chosen as the test bed for the purpose of simulating and testing the solution. The technique is another important consideration in deciding how the system is implemented. Three Java based techniques, J2ME, RMI and Servlet, are chosen as the tools to implement the proposal, because Java is nicely independent of platform.

J2ME is a new, very small Java application environment. It is a framework for the deployment and use of Java technology in the post-PC world. It is suitable for the simulation of the mobile devices in the proposal stated in this project.

Java Remote Method Invocation (RMI) enables the programmer to create distributed Java technology-based to Java technology-based applications, in
which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. A Java technology-based program can make a call on a remote object once it obtains a reference to the remote object, either by looking up the remote object in the bootstrap naming service provided by RMI or by receiving the reference as an argument or a return value. A client can call a remote object in a server, and that server can also be a client of other remote objects. RMI uses object serialization to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism.

Java Servlet technology provides web developers with a simple, consistent mechanism for extending the functionality of a web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side -- without a face. Java servlets have made many web applications possible. We use HTTP protocol to transfer the file between mobile devices and local IP stations.

The implementation system consisted of three remote Unix hosts simulating local IP stations and Home Agent respectively, and one windows station simulating the mobile device.
2. Research on Mobile Devices and distributed network system

2.1 Terminology

*Mobile agent:* An autonomous program performs that some task on its own and moves across a computer network in order to complete its calculation.[10]

*Mobile Object:* It is the object of mobile agent that handles all the information about the Mobile Device. It moves between two or more Foreign Agent (FA), as Mobile Device moves around.[10]

*Object Serialization:* Object Serialization is the process of writing the state of an object to a byte stream. This is useful when you want to store the state of an object in the persistent storage for example to a file. Once the objects state is stored in a persistent storage area, the object deserialization process restores the object state back. The object serialization process is required in Java RMI (Remote Method Invocation). Once the object is serialized you can send the object to a hard disk or across the network. You can also say object serialization is the process of converting the Java object into bit-blot.[11]

*Serialization:* The mechanism used by RMI to pass objects between JVMS, either as arguments in a method invocation from a client to a server or as return values from a method invocation. Main uses of serialization are:[6]

- As a persistence mechanism, if the stream being used is FileOutputStream, then the data will automatically be written to a file.
- As a copy mechanism `ByteArrayOutputStream` to create duplicates of the original objects.
- As a communication mechanism: if the stream is coming from a socket, it can be automatically sent over the wire to a receiving socket.

*Distributed system:* A collection of loosely coupled processors interconnected by a communication network.[8]

*Distributed File System (DFS):* It is a file-service system whose users, servers, and storage devices are dispersed among the various sites of a distributed system.[8]
3. The concept of Mobile Distributed File Access System (MDFAS)

Before the description of three solutions based on the three proposals, it is necessary to introduce the concept of Mobile File Access Distributed System (MDFAS).

3.1 Definition of MDFAS

The MDFAS is a combination of mobile access distributed system and file transfer system. It gives the mobile devices (i.e. PDA, cell phone) the ability of transferring files between the file server and itself, while it is in mobile status. In general, MDFAS should be automated to support accurate services for Mobile Devices. It has the following features:

- A Mobile device is able to send/receive files while it is moving around local stations.
- File must be transferred/stored in a safe way.
- The response time, when mobile device moves from one station to another, should be short enough to keep the file transfer seamless.

3.2 Requirement of MDFAS

The elements involved in MDFAS is shown in the following figure:
The MDFAS basically consists of four elements: mobile device, local station, and Internet connection and file server. The connection between a mobile device and local station is wireless. Local station is actually a node with two interfaces. The mobile side interface is in charge of the communication with the mobile device. It can handle the roaming of the mobile device. The network side interface is in charge of connection between local station and file server. All the data transfer from the mobile device to IP network has to be nicely handled in local station, in order to guarantee the correctness and entirety of the data. File server has to keep a record of the necessary information (i.e., the location of the mobile device).

The key point in MDFAS is how to synchronize the data transfer when mobile device is moving around local stations. Certain technology must be utilized to realize this function.
4. Solutions

According to the ideas presented in the three proposals, three solutions are introduced in this report. All three solutions are the extensions of the basic MDFAS model. Their system structure and functionalities will be stated in detail in the following.

4.1 Solution 1

Solution 1 provides a broadcasting based solution that the location of mobile device is broadcast to all the adjacent local stations so that mobile device immediately knows where a mobile device moves from. When a mobile device roams to a new local station, the station can get all session states from old location. Using a centralized station, called Home Agent, the solution can be optimized although it still introduces other new problems such as the problem that Home Agent may be involved in session state movement. The following figure shows the optimized solution.
4.1.1 Functionality of the system

- Manually initialize a MOBILE DEVICE
- Display the current read/write information
• Update the current read/write information
• Handle data transfer between a MOBILE DEVICE and servers
• Store middleware status information when a MOBILE DEVICE is online
• Relocate new address from local area
• Provide a MOBILE DEVICE interface for read/write

4.1.2 Scenario of the system

The following steps (as indicated in Figure 4.1) illustrate the operation:

1. MOBILE DEVICE sends initial registration request to Home Agent
2. MOBILE DEVICE sends data service request to Foreign Agent (A) and update location information
3. Foreign Agent (A) forwards update location request to Home Agent
4. Home Agent sends data service request to directory server and object server
5. Directory server sends data service request to object server
6. Object server acknowledges the data service request to Home Agent (A) and returns required data.
7. Home Agent then responds to Foreign Agent (A) about data information.
8. Foreign Agent (A) continues responding to MOBILE DEVICE about data information.
9. MOBILE DEVICE sends other data service request commands, for example: read file, write file.
10. When MOBILE DEVICE moves from Foreign Agent(A) to Foreign Agent(B), MOBILE DEVICE is detected by Foreign Agent(B)
11. Foreign Agent (B) informs Home Agent immediately about the new MOBILE DEVICE by sending update location request.

12. At the same time, when Foreign Agent(A) disconnects from MOBILE DEVICE, Foreign Agent (A) have to upload the current information about MOBILE DEVICE to Home Agent.

13. Home Agent receives the latest status information about MOBILE DEVICE, and Home Agent forwards status information to Foreign Agent(B).

14. New connection between MOBILE DEVICE and Foreign Agent (B) is set up.

15. MOBILE DEVICE receives updated data from Foreign Agent(B).

4.1.3 Mobile Devices and Middleware (FA, HA)

The following figure shows the type of messages exchanged among them.

![Diagram of Mobile Devices and Middleware](image)

**FIGURE 4-2 DIAGRAM OF MOBILE DEVICES AND MIDDLEWARE**

The PDA client can send update location and data transferring to foreign agent A. When foreign agent A receives the update location request, it will forward to home agent. When the PDA client moves to another foreign agent B, foreign agent B sends getsessionstate request to home agent, home agent will send updatesessionstate request to foreign agent A. Upon receiving response, Home agent will send updatesession request to foreign agent B.

4.2 Solution 2

4.2.1 Problem Statement
Global Server Mobile (GSM) Functionality:

In this section we will describe the functions of GSM from the perspective of users. The GSM system consists of three major interconnected subsystems that interact with each other and with the users through certain network interfaces. The subsystems are called the Base Station Subsystem (BSS), Network and Switching Subsystem (NSS), and the Operation Support Subsystem (OSS). The Mobile Station (MS) is also a subsystem, but is usually considered to be a part of the OSS for architectural purpose.

4.2.2 Functionality of the system

This solution gives a different way of describing the mobile location, which gives the coordinate \((X, Y)\) of the mobile device by means of Global Position System (GPS). Therefore, at the edge area where mobile may move to a new location, the Old Foreign Agent can find more suitable other Foreign Agent and move its object to the other Foreign Agent. The following figure shows the principle.

![Figure 4-3 GSM in Mobile Devices File Access System](image)

**Figure 4-3 GSM in Mobile Devices File Access System**

MSC: Mobile Server Center

GSM: Global Server Mobile

12
NT: neighbour table

- FA (foreign agent) can move to different object host (station)
- FA can decide when, how and where to move at the new address
- Mobile Station Center can calculate the coordinate (x, y) of the Mobile device

4.2.3 Scenario of the system

1. Mobile device sends signal every 10 seconds
2. When FA gets this information, FA find out whether or not the Mobile device is on the edge of the area.
3. When Mobile device is at the edge of the area, FA checks neighbor table on the current object host (server). Every object host has different neighbor table, it means it is configured manually in static manner.
4. When the Mobile device is in the area, the connection between the Mobile device and the Station continues.
5. FA can move to different object host after FA gets the new address of the station.
6. In GSM, mobile devices send signal in several seconds, two adjacent mobile station accept these signals.
7. FA can move one mobile object to a new suitable FA.
4.3 Solution 3

This solution is an inquiry-based system from the perspective of Mobile Device(MD) address searching. The IP addresses are stored in home agent. When MD moves from FA1 to FA2, FA2 does not know where MD moves from. FA2 must ask home agent (HA) where MD comes from. After getting the IP address of FA1 from HA, FA2 is now able to get mobile agent from FA1.

4.3.1 Architecture of the system

In Foreign Agent station, there are three objects, Mobile Agent object, which does data processing, PlaceHolder object, which moves mobile agent by using RMI and query location information with HomeAgent, RequestHolder object, which communicates with the Mobile device.
Figure 4-4 System Architecture of Solution 3
4.3.2 System functional requirement

This system consists of three elements: mobile device (MD), Foreign Agent (FA) and Home Agent (HA). MD is the mobile equipment (i.e., PDA or cell phone). Foreign Agent is actually the local station in the IP network, which is the connection point of the mobile device and the IP network. Home Agent (HA) is a node in the IP network playing the role of the file server, and address management for mobile devices. The above three elements work together to set up a system of MDFAS. The functionalities of each element are specified as follows:

*Mobile Device*

Mobile agent is simulated by the tool, J2ME, as an independent simulation program. It communicates with FAs by means http protocol. It is the mobile end of the file transfer.

*Foreign Agent*

Foreign agent is the core of the system. It is the intermediary between mobile devices and file server. As a result, it brings up three basic requirements:

- Communicate with mobile devices: All the data, either from the transferred file, or from the communication messages, must be properly handled by FA, so that when the mobile device moves around FAs, the connectivity is seamless. This is done by RequestHolder.
- Handle the mobile agent: Each mobile agent containing the dynamic status information for the session of file transfer corresponds to one mobile device. When a mobile device moves around FAs, the mobile agent corresponding to that mobile device moves around in consequence. The action of the mobile agent movement among FAs must be handled by FA appropriately. This is done by PlaceHolder.

- Communicate with Home Agent: The data for file transfer and mobile location tracking must be updated by the Home Agent.

_Home Agent_

Home agent is actually the place registering and storing the current address of the mobile agent.

4.3.3 System Non-Functional requirements

- The users must manually activate the Mobile device at first time from the original FA

- Three conditions must be satisfied to run the system
  1. Three remote hosts, one for HA, two for FA.
  2. Windows machine running simulation of Mobile device.
  3. Valid Internet connection.

4.3.4 Scenario of the system in FA
The FA consists of three parts: RequestHolder, PlaceHolder and Mobile Agent, corresponding to the three functional requirements described above. How they cooperate to complete the migration of mobile agent is the key point in the design of this system.

**Figure 4-5 System construction**

**Figure 4-6 Before agent moving**
FIGURE 4-7 AFTER AGENT MOVING

1. RequestHolder gets the request from MD.

2. PlaceHolder(PH) in FA2 gets the object of Mobile Agent (MA) through RMI

3. PH2 calls moveto() of PH1 to inform PH1 of the new address where MA has moved to.

4. PH2 gets the object reference of PH2 through using RMI

5. PH1 calls method pass() of PH2 to transfer the object of MA of PH1 to PH2.

6. Agent manager 1 kills it’s mobile agent

7. Agent manager 2 loads the agent which was transferred from FA1

8. Agent Manager 2 creates new object from real agent

9. New object agent starts to work

4.3.5 Use case design

As shown in Figure 4-8, the Foreign Agent has four function modules, one is for processing requests from the Mobile PDA, one is for Handling Read/Update Data requests. The other two are for Handling Move procedure and Get/Update agent location information module.
FIGURE 4-8 USE CASE DIAGRAM OF FOREIGN AGENT

FIGURE 4-9 USE CASE DIAGRAM OF HOME AGENT
The Home Agent has two function modules as shown in Figure 4-9, one is to handle `updateAgentLocation` request from FA, the other is to handle `GetAgentLocation` request from FA.

![Diagram](image)

**Figure 4-10 Use Case Diagram of Mobile**

The Mobile has three function modules (Figure 4-10), one is to send `Open/Close` request to FA, one is to send `Connect/Disconnect Request` to FA, the third one is to send `Read/Update Request` to FA.
4.3.6 Class diagram design

**Figure 4-11 MobileAgent class diagram**

**Figure 4-12 Mobile class diagram**
4.3.7 Sequence diagram of the design

4.3.7.1 Connect FA1

As shown in Figure 4-14, Mobile handles to connect FA1, then invokes init method of FA1 RequestHolder to init FA1 RequestHolder and to init FA1 PlaceHolder object with its local Naming Service.

FA1 RequestHolder receives Get Http request from mobile and forwards it to its local PlaceHolder object. FA1 PlaceHolderImpl will responsible for processing the request. It will first check if local mobile agent exists. If the mobile agent exists, it will process the request from mobile. Otherwise, it will send request to ask the Home Agent the location of the mobile agent. If the mobile agent doesn’t
exist in any Foreign Agent, a new mobile agent will be created in local agent.

Then, send request to inform Home Agent that agent location will be updated.

---

**FIGURE 4-14 CONNECT FA1 SEQUENCE DIAGRAM**
4.3.7.2 Connect FA2

Mobile handles to connect FA2, then invokes init method of FA2 RequestHolder to init FA2 RequestHolder and to init FA2 PlaceHolder object with its local Naming Service. FA2 RequestHolder receives Get Http request from mobile and forwards it to its local PlaceHolder object. FA2 PlaceHolderImpl is responsible for processing this request. It will first check if local mobile agent exists. If the mobile agent doesn’t exist, it will send request to ask the Home Agent the location of the mobile agent. If the mobile agent exists in a remote Foreign Agent, FA2 will get the remote Agent location from Home Agent while move mobile agent from FA1.

Figure 4-15 Connect FA2 Sequence Diagram
to FA2. Then, send request to inform Home Agent that agent location will be updated.

4.3.7.3  Data Request If Agent Exist

![Sequence Diagram]

**Figure 4-16 Data Request If Agent Exist Sequence Diagram**

Mobile executes open request, PlaceHolderImpl processes the request. First, PlaceHolderImpl checks if Agent exists in local station. If it exists, Mobile Agent opens one of the remote files on distributed file system and loads the whole contents of file into its local buffer when mobile sends open request.
4.3.7.4 Data Request If Agent not exist

Mobile executes open request, PlaceHolderImpl processes the request. First, PlaceHolderImpl checks if Agent exists in local station. If it doesn’t exist, it will send request to ask the Home Agent the location of the mobile agent. If the mobile agent exists in a remote Foreign Agent, FA2 will get the remote Agent location from Home Agent while moving the mobile agent from FA1 to FA2. Then, it sends request to inform Home Agent that agent location will be updated. Mobile
Agent opens one of remote files on the distributed file system and loads the whole contents of file into its local buffer when mobile sends open request.

4.3.7.5 Move Procedure

![Sequence Diagram](image)

Figure 4-18 Move Procedure sequence diagram

FA1 PlaceHolderImpl calls MoveTo method of FA2 with location address of FA1, FA2 PlaceHolderImpl calls Pass method of FA1 with Mobile Agent of FA2.

4.3.8 Summary

In summary, the tracking of the location of mobile agent has to be done through the inquiring of HA. This makes FA less loaded in terms of determining the trace of mobile agent. But it results in longer time locating the location of mobile agent. When the system carries time critical data transfer, real time application for example, certain mechanism must be found to solve the problem of slow response. Apparently, the system structure in this design is rather simple, which implies great advantage in the employment of this technique in real world.
5. Implementation

5.1 Requirement for system development:

- Hardware:
  
  IBM compatible PC with CPU equivalent to Intel Pentium 200 or higher

  Minimum hard disk space: 50 MB

  Minimum RAM: 64MB

  Operating system: Microsoft Windows 2000/RedHat Linux 6.0 higher

- Software:

  JDK 2 standard edition [11] version 1.2.1 or later

  J2EE Enterprise Edition [16]

  SSH secure shell client [12]

  JSDK2.1 small server [13]

  J2ME wireless Toolkits for Windows 1.0.4 [14]

5.2 Class description

We select solution 3 as the system configuration for implementation. The coding
is all based on Java technology. Following is the description of the classes. The
description of methods is given in the Appendix.

5.2.1 Class MobileAgent

MobileAgent class is responsible for processing the file access request, such as
Open, Close, Read, and Update from mobile devices. For the consideration of
object mobility, Class MobileAgent inherits from Class Serializable because any
objects inheriting Serializable can be used as a parameter in RMI method to
provide mobility functionality from one location to the other location without loss of state the object.

5.2.2 Class PlaceHolder

PlaceHolder class is an interface, which inherits from Class Remote, to define two public methods (MoveTo, Pass) that can be invoked remotely by RMI.

5.2.3 Class PlaceHolderImpl

PlaceHolderImpl class, which inherits from unicastRemoteObject class and PlaceHolder interface, is used to analyze requests from the mobile device and manage mobility of the Mobile Agent by implementing MoveTo and Pass method of PlaceHolder interface.

5.2.4 Class RequestHolder

RequestHolder class, which inherits HttpServlet class, is responsible for the reception of HTTP requests (GET and POST) from the mobile devices and forwards its request to its local PlaceHolder Object.

5.2.5 Class Mobile

Mobile class, which inherits Midget class and CommandListener class, is responsible for simulating any mobile device, such as palm device, that can use J2ME to communicate with foreign agent server.
5.2.6 Class \textit{HomeAgent}

Home Agent class, which inherits HttpServlet class, is responsible for the HTTP request from FA to get or update location request.

5.3 How to install the software for the system

- \textit{Server side installation}
     Download J2SE from java.sun.com, then install it.
     Download J2EE, and install it.
  3. Install Java Servlet Development Kit 2.1 (J2SDK), a small web server.
  4. Set \texttt{CLASSPATH}
     \begin{verbatim}
     setenv CLASSPATH /usr/java1.2/bin:~/.jsdk2.1/servlet.jar:..
     \end{verbatim}
  5. Set permission
     \begin{verbatim}
     chmod 700 startserver
     chmod 700 stopserver
     \end{verbatim}
  6. Copy servlet
     \begin{verbatim}
     copy servlet class and other compiled class to
     /jsdk2.1/webpages/WEB-INF/servlets
     \end{verbatim}
• Mobile side installation

      Download J2SE from java.sun.com, then install it.

  2. Install Java 2 Platform, Micro Edition(J2ME)
      Download J2ME from java.sun.com, then install it.

  3. copy the directory of mobile to /j2me/apps directory

5.4 How to Run the system

We have described the system deployment steps above. It is time to put all the things together to see how the entire system works on both the server side and the client side.

Server Side:

1. For Home Agent (HA):

   • Connect to orchid.cs.concordia.ca
   • cd /home/grad/jingli/orchid-jsdk2.1
   • run startserver
Using classpath: /server.jar:/server.jar:/usr/java1.2/bin: .

orchid.jingli % JSDK WebServer Version 2.1

Loaded configuration from file:/home/grad/jingli/orchid_jsdk2.1/default.cfg
endpoint created: :8190

Note: Serverlet is running at port 8190, which is specified in the code. It can be modified to any free port for user in the host. But the code needs to be recompiled to effect the change.

2. Foreign Agent1 (FA1):
- Connect to sunset.cs.concordia.ca
- cd ~/sunset_jsdk2.1/webpages/WEB-INF/servlets
- rmiregistry 2050 &
- cd ~/sunset_jsdk2.1/
- run startserver

![Image of a screen with code]

**Figure 5-2 FA1(sunset.cs.concordia.ca) SDL Screen**

Note: Serverlet is running at port 2050, which is specified in the code. It can be modified to any free port for user in the host. But the code needs to be recompiled to effect the change. To recompile the source code of PlaceHolderImpl.class

```
Rmic -v1.2 PlaceHolderImpl
```
3. Foreign Agent 2 (FA2)

- Connect to dahlia.cs.concordia.ca by SSH
- `Cd ~/dahlia_jsdk2.1/webpages WEB-INF/servlets`
- `rmiregistry 2050 &`
- `Cd ~/dahlia_jsdk2.1`
- Run startserver

**Figure 5-3 FA2 (DAHLIA.CS.CONCORDIA.CA) SIDE SCREEN**

Using classpath: `./server.jar./servlet.jar/usr/java1.2/bin:`

dahlia.jngli % JSDK WebServer Version 2.1

Loaded configuration from file:/home/grad/jngli/dahlia_jsdk2.1/default.cfg

endpoint created: :8190

35
Note: Serverlet is running at port 8190, which is specified in the code. It can be modified to any free port for user in the host. But the code needs to be recompiled to effect the change.

Client side:

Figure 5-4 Mobile side screen 1
The Mobile side needs to have the J2ME Wireless Toolkit to run Mobile project. From the mobile interface, select launch, then screen display menu from which one can select any command to connect FA1 or FA2.
open connection OK
length is:1
Read OK
Response is:Close OK
of true private choice," today upheld the use of public mony for religious school 00074555

readfirst request
open connection OK
length is:1
Read OK
Response is:Close OK
of true private choice," today upheld the use of public mony for religious school 00074555

readfirst request
open connection OK
length is:1
Read OK
Response is:Close OK

In a decisive 445-4 ruling that the majority called a logical outgrowth of recent cases

Execution completed successfully
1344276 bytescopy executed
4444 thread switches
212 classes in the system (including system classes)
2715 dynamic objects allocated (308184 bytes)
4081 garbage collections (294120 bytes collected)
Total heap size 500000 bytes (currently 417028 bytes free)
Execution completed successfully
32144 bytecode executed
0 thread switches
319 classes in the system (including system classes)
319 dynamic objects allocated (21852 bytes)
0 garbage collections (1930 bytes collected)
Total heap size 500000 bytes (currently 480580 bytes free)

FIGURE 5.6 J2ME WIRELESS TOOLKIT 1.0.4 SCREEN
6. Test

6.1 Objectives

Software testing is one of the major processes of system development. The objective of testing is to ensure that the system conforms to its requirement specifications (i.e. Software Verification) and the system implementation has met the expectation of the customer (i.e. Software Validation). A successful testing process should systematically uncover different classes of errors in a minimum amount of time and with a minimum amount of effort. It also demonstrates that the software is working as stated in the specifications. The data collected through testing can provide an indication of the software’s reliability and quality.

6.2 Detail testing in this project

Since this project is using several java based technologies, such as JDK, J2ME, Servlet and RMI, software debug giving and testing is not as easy as in the IDE like JBuilder and VC++. Due to this reason, a number of print commands were added outputting messages indicating the processing of the program. It is a kind of debug/test information, which gives the developer real run time information. Those test information are listed as follows:

- When MA is connected to FA1

  **HA displays:**
  
  com.sun.web.core.InvokerServlet: init
  
  HA: init
receive request from FA: command=getagentlocation;location=
handle registration

currentAgentLocation is: null

Get Agent Location is done

receive request from FA: command=updateagentlocation
;location=localhost:2050/PlaceHolder

handle UpdateLocation

Update Location OK: new location is:
//sunset.cs.concordia.ca:2050/PlaceHolder

Update Agent Location is done

FA1 displays:

RequestHolder: init
initialize requestholder

begin create placeholder object
initialize placeholder

Initializing placeholder: please wait.

The PlaceHolder is up and running.

get request from mobile

Check if Agent exist

Agent not exist

Get Agent Location Info
New


Read


Receive response: null

oldAgentLocation is: null

can not find location of agent, create a new one

Read


Update Location OK

Agent exists

handle connect request

the buffer is: Connection accepted

Process request <connect> from Mobile is done

• When MA move from FA1 to FA2

FA1 displays:

begin move

Agent exists in my location, begin call remote pass
**FA2 displays:**

RequestHolder: init
initialize requestholder
begin create placeholder object
initialize placeholder
Initializing placeholder: please wait.
The PlaceHolder is up and running.
get request from mobile
Check if Agent exist
Agent not exist
Get Agent Location Info

New

Read
Receive response://sunset.cs.concordia.ca:2050/PlaceHolder
oldAgentLocation is: //sunset.cs.concordia.ca:2050/PlaceHolder
find old location is ://sunset.cs.concordia.ca:2050/PlaceHolder
Get placeholder from: //sunset.cs.concordia.ca:2050/PlaceHolder
call remote move
Mobile Agent is passed here
Update new Agent Location

Read


Update Location OK

Agent exists

handle connect request

the buffer is: Connection accepted

Process request <connect> from Mobile is done

HA displays:

receive request from FA: command=getagentlocation ;location=

handle registration

currentAgentLocation is:

//sunset.cs.concordia.ca:2050/PlaceHolder

Get Agent Location is done

receive request from FA: command=updateagentlocation ;location=//dahlia.cs.concordia.ca:2050/PlaceHolder

handle UpdateLocation

Update Location OK: new location is:

//dahlia.cs.concordia.ca:2050/PlaceHolder

Update Agent Location is done
• When MA open a file from FA1:

  **FA1 displays:**

  get request from mobile

  Check if Agent exist

  Agent exists

  handle open request

  the buffer is: Open OK

  Process request <open> from Mobile is done

• When MA send ReadFirst command to FA1:

  **FA1 displays:**

  get request from mobile

  Check if Agent exist

  Agent exists

  handle readfirst request

  Agent handle read first

  the buffer is: Close OK

  of true private choice," today upheld the use of public money for religious os2345tito

  n

  Process request <readfirst> from Mobile is done

• When MA send ReadNext command to FA1
**FA1 displays:**

get request from mobile

Check if Agent exist

Agent exists

handle readnext request

Agent handle read next

the buffer is:

\[
i
\]

n a decisive 445-to-4 ruling that the majority called a logical outgrowth of recent decisions and

Process request <readnext> from Mobile is done

- When MA send Update command to FA2:

  **FA2 displays:**

  get post request from mobile

  Check if Agent exist

  Agent exists

  handle update request

  begin get text str:

  length is: 18

  the buffer is:

  Open connection OK. updated text string is: Open connection OK

  Agent handle update
• When MA send Close command to FA2

  **FA2 displays:**

  get request from mobile

  Check if Agent exist

  Agent exists

  handle close request

  the buffer is: Close OK

  Process request <close> from Mobile is done
7. Conclusion and Future work

7.1 Conclusion
According to the description of the proposal and the result of the implementation, it looks pretty promising that mobile file transferring could be implemented in IP network, by taking advantage of the new software techniques. Compared with the traditional telephone network, it effectively utilizes the existing tremendous IP network, which has the advantage of low cost.

7.2 Future work
The mechanism proposed in this paper is far less mature.

- There should be more FA involved the simulations.
- The protocol used to transfer the object among FAs should be TCP/UDP.
- Mobile devices should be able to recognize the wireless connection.
- The speed responding the mobile devices should be taken into account as one essential aspect of the system design/implementation.
References


[11] Enterprise Java Specifications v1.1 Sun Microsystems:
[12] SSH secure shell client
http://ftp.ssh.com/cgi-bin/download.cgi?download=sshwkswin

[13] JSDK2.1 small server
http://java.sun.com/docs/books/tutorial/servlets/servletrunner/server-start.html

[14] J2ME Wireless Toolkits 1.0.4


Appendix

A. The description of Class methods of Solution 3

A.1 MobileAgent

Method HandleOpenRequest()

This method simulates the functionality that the Mobile Agent opens one of remote files on distributed file system and loads the whole contents of the file into its local buffer when the mobile device sends open request.

Method HandleCloseRequest()

This method simulates the functionality that the Mobile Agent writes all updated buffer to the remote file on the distributed file system and closes it when the mobile device sends close request.

Method HandleReadFirstRequest()

This method reads the first 100 characters (default value) of the remote file from its local buffer which is already initiated by HandleOpenRequest() method and keeps the current read position value into variable currentPos.

Method HandleReadprevRequest()

This method reads the previous 100 characters of remote file from its local buffer and keeps the current read position value into variable currentPos.

Method HandleReadNextRequest()
This method reads the next 100 characters of remote file from its local buffer and keeps the current read position value into variable currentPos.

Method HandleUpdateRequest()
This method updates part of its local buffer with the contents from mobile.

Method Read()
This method gets 100 characters from its local buffer.

Method ReadWhole()
This method reads the contents of remote file into wholebuffer.

Method WriteWhole()
This method writes the wholebuffer into remote file.

A.2 Class PlaceHolderImpl

Method CheckIfAgentExist()
This method checks if the mobile agent object exists in its local machine.

Method Process()
The flow chart of method Process
This method is mainly responsible for analyzing request from the mobile device, and moving the mobile agent from remote location to its local location. It will first check if local mobile agent exists. If the mobile agent exists, it will process the request from mobile and call corresponding method of mobile agent according to the kind of request from the mobile device. Otherwise, it will send request to ask the Home Agent server the location of the mobile agent. If no mobile agent exists
in any Foreign Agent server, a new mobile agent will be created in the local server. If the mobile agent exists in a remote Foreign Agent Server, it will call remote MoveTo method to inform the remote Foreign Agent Server to pass mobile agent in its local machine.

Method getTextStr()

This method is used to get text string that is going to be updated.

Method sendBack()

This method is used to send updated text string back.

Method MoveTo()

This method is invoked by the remote foreign agent server that needs the mobile agent. The method will then invoke Pass method of the remote foreign agent server to pass its local mobile agent to remote foreign agent server.

Method Pass()

This method is invoked by the remote foreign agent server for passing the remote mobile agent to its local machine.

Method GetAgentLocationInfo()

This method is used to ask the home agent server for the current location of mobile agent.
Method `UpdateAgentLocation()`

This method is used to send requests to inform the home agent server that the mobile agent has been moved to its local machine.

Method `init()`

This method is invoked by the init method of RequestHolder object to register the new PlaceHolder object with its local naming service.

A.3 Class RequestHolder

Method `init()`

This method is used to create a new placeholder object and bind it to its local naming service.

Method `doGet()`

This method is used to receive GET HTTP request from mobile and forward it to its local PlaceHolder object.

Method `doPost()`

This method is used to receive POST HTTP request, which is mainly for update requests, from mobile and forward it to its local PlaceHolder object.

A.4 Class Mobile
Method **startApp()**

This method is to start Mobile midlet application.

Method **startPage()**

This method is to show the first Welcome page.

Method **pauseApp()**

This method is inherited from midlet and do nothing.

Method **destroyApp()**

This method is inherited from midlet and do nothing.

Method **showText()**

This method is used to show text that receives from foreign agent server.

Method **commandAction()**

This method is used to execute the command that user activate by invoking different method.

Method **handleConnectFAhost()**

This method sends connect command to the foreign agent server with HTTP request.
Method openRequest()
This method sends open command to the foreign agent server with HTTP request.

Method closeRequest()
This method sends close command to the foreign agent server with HTTP request.

Method readfirstRequest()
This method sends readfirst command to the foreign agent server with HTTP request.

Method readnextRequest()
This method sends readnext command to the foreign agent server with HTTP request.

Method readprevRequest()
This method sends readfirst command to the foreign agent server with HTTP request.

Method readPage()
This method sends HTTP GET request to the foreign agent server and receives response from the foreign agent server.
Method updateRequest()

This method sends HTTP POST request to the foreign agent server for updating its local buffer, and received updated contents from the foreign agent server.

B. Design Diagrams of Solution 1

1. Use Case Design

1.1 MOBILE DEVICES client Use Case Diagram

![Use Case Diagram]

**Figure B-1 MOBILE DEVICES CLIENT USE CASE DIAGRAM**
1.2 Foreign Agent Use Case Diagram

Figure B-2 Foreign Agent Use Case Diagram

1.3 Home Agent Use Case Diagram

Figure B-3 Home Agent Use Case Diagram
1.4. Object Directory Server Use Case Diagram

![Diagram of Object Directory Server Use Case](image)

**Figure B-4 Object Directory Server Use Case Diagram**

1.5 Object Server Use Case Diagram

![Diagram of Object Server Use Case](image)

**Figure B-5 Object Server Use Case Diagram**

2. Class Diagram Design

2.1 RegistryManager class diagram

```
Registry Manager
- CurrentPos
- RegistrationRequestHandler()

SessionInterface
- getSessionState()
- putSessionState()

SessionManager
- openSession()
- closeSession()
- getSessionState()
- putSessionState()

SessionAgent
- openSession()
- closeSession()
- getSessionState()
- putSessionState()
```
**Figure B-6 RegistryManager Class Diagram**

- `SessionStateRequest`
  - `getStat()`
  - `updateState()`

- `DataRequest`
  - `getobject()`
  - `readData()`
  - `writeData()`
  - `opname2()`

**Figure B-7 SessionStateRequest & DataRequest Class Diagram**

- `LocationHandler`
  - `updateLocation()`

**Figure B-8 LocationHandler Class Diagram**

- `HALocationRequest`
  - `updateLocation()`

- `FALocationRequest`
  - `updateLocation()`
Mobile Device
RequestHandler

- handleOpenRequest()
- handleReadRequest()
- handleWriteRequest()
- handleCloseRequest()

DataService

- getData()
- putData()

DataServiceImpl

- getData()
- putData()

**Figure B-9** **MobileDeviceRequestHandler** class diagram
Figure B-10 MobileDeviceDataProcessor class diagram
3. Sequence diagram Design:

3.1 Session begin & end sequence diagram

![Sequence diagram]

**Figure B-11 Session begin & end sequence diagram**
3.2 Location & data sequence diagram

Figure B-12 Location & data sequence diagram

3.3 HandleRegistryRequest sequence diagram

Figure B-13 HandleRegistryRequest sequence diagram