Université d'Ottawa • University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Dineshbalu BALAKRISHNAN

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Master of Computer Science

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Design and Implementation of a Personal Assistant for Mobile Device Users Using Agent Technology

A. Karmouch

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

M. Barbeau

L. Logrippo

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Design and Implementation of a Personal Assistant for Mobile Device Users using Agent Technology

By

Dineshbalu Balakrishnan, B.E.

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

Master of Computer Science

Ottawa-Carleton Institute for Computer Science
Department of Computer Science
School of Information Technology and Engineering
Faculty of Engineering

University of Ottawa
Ottawa, ON, Canada

Your file Votre référence
ISBN: 0-494-01406-7
Our file Notre référence
ISBN: 0-494-01406-7

Abstract

Recent developments in the mobile device market and emerging technologies for mobile wireless devices have realized the presence of mobile devices in every day human life activities. New applications, software platforms, and execution environments are being developed for mobile devices. The design and implementation of a personal assistant for mobile device users is an effort taken to develop a new application for mobile devices.

Programs which semi-autonomously act on behalf of software applications or users and execute tasks are called agents. Agent technology is used to design the personal assistant, which falls under the category of software agents. The personal assistant acts on behalf of mobile device users and provides assistance to users to use various Internet applications on mobile devices. Assistance is provided based on the particular user's personal preferences.

The personal assistant is broken down into different agent components and forms a partially ad-hoc multi-agent system. Though the personal assistant is intended for mobile device users and executes in mobile devices, some of its components are executed in a workstation i.e., desktop PC. The personal assistant's components that are executed in a workstation act as a gateway of communication for the personal assistant. This approach resolves problems such as unstable execution environments, overloading, and insufficient resources in mobile devices.

The personal assistant is executed on the PersonalJava runtime environment and the MicroFIPA-OS agent platform. The personal assistant's components in the workstation are executed on the Java Runtime Environment and the FIPA-OS agent platform. The personal assistant could assist a mobile device user while executing tasks such as file and media transfer, e-mail formatting, address book maintenance, and report retrieval.

Acknowledgements

I would like to thank my supervisor Dr. Ahmed Karmouch for his constant guidance, motivation, and support. I would like to thank NCIT, NRC, and MITEL for their suggestions and monetary support that helped in completing my masters' program. I would like to thank my colleagues of Multimedia and Mobile Agent Research Laboratory, University of Ottawa, for their timely suggestions and companionship. I wish to acknowledge my parents and sister for providing constant support and appreciation through all stages of my masters' program. Finally, I would like to extend my thanks and gratitude to all others who helped me to succeed.

Contents

List of Tables

List of Figures

List of Abbreviations

ACC	Agent Communication Channel
ACL	Agent Communication Language
AP	Agent Platform
FIPA-OS	Foundation of Intelligent Physical Agents – Open Source
GUI	Graphical User Interface
HTTP	Hyper Text Transport Protocol
MP3	MPEG audio encoding layer III
MTS & MTP	Message Transport Service and Protocol
PDF	Portable Document Format
UIA	User Interface Agent
USB	Universal Serial Bus
XML	eXtensible Markup Language

CHAPTER 1

Introduction

The mobile device market is growing rapidly and mobile devices have evolved from an expensive luxury to an indispensable requirement. Sophisticated applications are being developed for mobile devices and this is made possible due to recent increase in the processing power and memory capacity of mobile devices. More than 1 out of 10 people (approximately 700 million users) have mobile devices such as *mobile phones* and *Personal Digital Assistants* (PDA), and this ratio continues to increase rapidly [1, 2]. Over the past decade mobile devices' weight, size, and cost have dropped over 20% every year [1, 2]. The demand for innovative yet efficient applications for mobile devices continues to grow at over 40% every year [1, 2], resulting in new compact mobile devices such as *tablet PC's* and *Personal Digital Pens*. The Internet usage has enormously increased spanning from e-mail to controlling household appliances.

Emerging technologies have made Internet usage in mobile devices a reality. New trends and emerging technologies focus on solving user tasks with least user intervention. The personal assistant discussed in this thesis is an effort taken to develop a new application

for mobile devices. Programs which semi-autonomously act on behalf of software applications or users and execute tasks are known as agents. Agent technology is used to design the personal assistant, which falls under the category of software agents. The goal of the personal assistant is to act on behalf of mobile device users and provide assistance to users to use various Internet applications on mobile devices. In other words, the personal assistant makes employing Internet in mobile devices easygoing. Assistance is provided based on the particular user's personal preferences. Therefore, each user could have a customized personal assistant just like their backpack or agenda [3]. The personal assistant is implemented on *FIPA* (Foundation of Intelligent Physical Agents) compliant [4, 5] *FIPA-OS* (Foundation for Intelligent Physical Agents – Open Source) [6] and *MicroFIPA-OS* (Micro edition of Foundation for Intelligent Physical Agents – Open Source) [7] agent platforms, and the *Java programming language* [8]. Though the personal assistant is intended for mobile devices, the agent-based system could also execute in high-power mobile devices such as *laptops* and *tablet PC's*.

This chapter further discusses motives for designing and implementing the personal assistant. The chapter then states the main objectives of the thesis. Next, the principal contributions are stated and important terminologies used in the thesis are briefed. Finally, an outline of the thesis is provided.

1.1 Motivation

The Internet is accessible in mobile wireless devices using new trends and emerging technologies. At the same time, Internet applications for mobile devices are complex to use for users, particularly novice users [9]. Mobile device users have to (i) deal with large

amount of data on a small-sized screen, (ii) use the touch-screen keypad, and (iii) use small-sized buttons. The complexity is also because of capability restrictions, compatibility restrictions, and wireless constraints in mobile wireless devices. Along with mobile device sales, the number of novice users starting to use mobile devices is also increasing [10]. In order to deal with these constraints, new and sophisticated applications, software platforms, and execution environments are being developed for mobile devices [11]. The above discussed constraints motivated to develop an application that focuses on solving user tasks with least user effort and intervention.

1.2 Objective and Requirement

The main objective of the thesis is to develop a personal assistant for mobile device users which provides a framework which users can use to perform tasks in mobile devices with least intervention. The personal assistant should semi-autonomously act on behalf of mobile device users in order to reduce users' intervention while performing tasks such as e-mail retrieval and file/ media transfer.

The requirements of the thesis are summarized as follows:

- Develop a personal assistant model which semi-autonomously acts on behalf of users to perform tasks in mobile devices.
- Develop the personal assistant model using agents to resolve constraints in current mobile devices.
- Design an interactive GUI for the personal assistant model to reduce the use of the touch-screen keypad and small-sized buttons in mobile devices.

- Enable inter-platform agent communication in order to integrate the personal assistant model with other agent-based systems.

This agent-based personal assistant model to assist users to perform tasks in mobile users is a novel idea in the field of agent technology and mobile computing. Other related personal assistants are discussed in the related work section (section 2.7).

1.3 Thesis Contribution

The main contributions of the thesis are listed herein as follows.

- Design and implementation of the personal assistant which provides a framework to assist mobile device users to perform various tasks in mobile devices. This contribution is discussed in chapter 4.
- Design and implementation of the proxy agent which resolves problems in mobile devices such as unstable execution environments, overloading, and insufficient resources. This contribution is discussed in section 4.1.5.
- Design and implementation of the adaptation agent which resolves compatibility problems in mobile devices. This is achieved by decreasing the hardware and software requirements of mobile devices where the personal assistant could execute. This contribution is discussed in section 4.1.6.
- Configuration of the MicroFIPA-OS agent platform to allow inter-platform communication. This contribution is discussed in section 5.1.1.

1.4 Related Terminology

This section lists and defines selected important terms used in the thesis for reader's convenience.

- **Agent:** Agents are programs that act semi-autonomously on behalf of users and applications as a guide or coach. It is an umbrella term representing a wide body of current research and development [14]. Agents can be created, moved, cloned, and destroyed.
- **Personal Assistant:** Personal assistants fall under the category of software agents. A personal assistant is built for a single user to act and perform autonomous actions only on that particular user's behalf.
- **Mobile Agent:** Agents that dynamically move from one location i.e., agent environment to another under their own control to perform tasks are called mobile agents.
- **Agent Platform:** An agent platform is the execution environment wherein agents are created, maintained, and destroyed. An agent platform comprises of platform-specific agents (Agent Management System and Directory Facilitator) and the Message Transport Service (MTS).
- **Agent Communication Language (ACL):** The ACL is the communication medium used by agents to communicate and negotiate with each other.
- **Ad-hoc Communication:** An ad-hoc communication is a type of spontaneous communication wherein software or devices communicate directly with each other without any centralized controlling system.

- **Policies:** Policies are rules and configurations set by users [15]. The particular matching policy acts as a guide for decision-making.
- **Agent Adaptation:** Agent adaptation is the process of adjusting agents to something such as environmental changes in an agent platform or unknown messages and file formats.
- **Proxy:** A proxy is a system or application that is authorized to act for another system or application.
- **External Agent / Inter-Platform Communication:** Agents that reside in remote networks or platforms are known as external agents. Any agent communication involving foreign agents is called inter-platform agent communication.

1.5 Thesis Outline

The thesis presents a design model to assist mobile device users to use various Internet applications. The personal assistant could also be integrated with ad-hoc based applications such as the one described in section 5.2. By means of this integration, the personal assistant could utilize services offered by external applications as well.

The thesis has so far introduced the personal assistant, discussed motives for developing a personal assistant, stated the personal assistant's objectives and principal contributions, and discussed important terminologies related to the thesis. The rest of the thesis is organised as follows.

The *Background and Related Work* chapter presents an overview of concepts that help better understand the thesis. The chapter first introduces agent technology by defining agents, stating agents' general properties, classifying agents, and discussing agents' pros

and cons. Next, the FIPA-OS agent platform and the MicroFIPA-OS agent platform including agent communication concepts are described. The chapter further reviews and surveys agent mobility, ad-hoc communication, policies, and related work on personal assistants. This is followed by the *Overview and Feature Set* chapter. The chapter first introduces the personal assistant and states the personal assistant's main objectives. The chapter then illustrates various user-oriented services and integrated services that could be assisted by the personal assistant. Next, the significant aspects of the personal assistant when compared with other personal assistants are discussed. Finally, various technical challenges faced while designing and implementing the personal assistant are discussed. This is followed by the *Design and Usage Scenarios* chapter which describes the design model of the personal assistant. The chapter illustrates all basic components that make up the personal assistant. This is followed by various usage scenarios of the personal assistant. This is followed by the *Implementation and Results* chapter which discusses implementation details and results of the personal assistant. The chapter elaborates on the *Mobile Agent-based Ad-hoc Communication System* and also discusses integration details of the personal assistant with this ad-hoc system. Finally, the personal assistant's flexibility and performance levels are discussed. This is followed by the final chapter *Conclusion* which concludes the thesis and discusses possible future research directions.

CHAPTER 2

Background and Related Work

The use of mobile devices to work on vital applications like those that require better security and software infrastructure are still at the beginning stage [16]. There is a common interest among researchers towards providing global and efficient technologies, standards, platforms, execution environments, and security for mobile devices [16]. Currently, it is a challenge to develop platforms and execution environments for mobile devices using agent technology [5, 14]. FIPA played a major role in popularising agent technology [4]. FIPA introduces platforms, communication techniques, and runtime environments for agent execution [4].

This chapter first elaborates on agent technology by defining agents, listing agents' common properties, classifying agents, and discussing agents' pros and cons. Next, the FIPA-OS agent platform and the MicroFIPA-OS agent platform are described. The chapter further describes the Agent Communication Language (ACL) and the Agent Communication Channel (ACC). The chapter finally discusses and surveys agent mobility, ad-hoc communication, policies, and the personal assistant's related work.

2.1 Agent Technology

Certain applications related to information technology can be made more efficient by using agent technology [14]. One such example is discussed in section 2.6. Agents usually have a given itinerary which they follow in order to complete tasks and report results. In the case of intelligent agents, the itinerary may change dynamically depending upon agents' status at a particular terminal or node in a network. Agents can be dynamically created, moved, cloned, and destroyed. Agent technology is sometimes known as push technology when information is autonomously fetched and delivered by agents [17]. Agents can also be viewed as a component of a software application in certain cases [18]. This section further defines agents, states agents' common properties, and finally classifies agents.

2.1.1 Agent Definition

An agent is an entity that represents a person, an organization, or a software application, which independently or by interacting with other agents executes a task or set of tasks. Till date there is no unique definition for agents since agent technology can be applied to several fields. In short, agents are defined as programs that act semi-autonomously on behalf of users and applications as a guide or coach.

The following are some common agent definitions.

Pattie Maes defines a software agent as "*A computational system which has goals, sensors, and effectors, and decides autonomously which actions to take, and when*".

Pattie Maes defines autonomous agents as "*Computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed*".

IBM defines intelligent agents as "*Software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in doing so, employ some knowledge or representation of the user's goals or desires*". IBM's agent definition is a practical and general definition of an agent.

2.1.2 Agent Properties, Classification, and Types

Even though agents are heterogeneous, there are some general properties that most agents do possess or recommended to possess. They are [19, 20],

- Autonomous
- Communicative/Cooperative
- Reactive
- Intelligent

Some agents could be proactive and possess goal directed behaviour [19, 20]. Agents possessing all the above mentioned properties will effectively carry out assigned tasks.

Agents can be classified based on their properties. As an example, agents which are autonomous are classified as *expert assistants, autonomous robots, softbots,* and *synthetic agents* [21]. Software agents are classified as *user agents, service agents,* and *intelligent agents* [18]. Information gathering agents such as *news filtering agents* and *weather reporting agents* are in existence [19, 22]. Agents are also used in entertainment industry, like for example, *MP3 agents* and *gaming agents* [11]. Besides these application-specific

usages, agents can also be used in various fields including medical services, personal shopping, tourism, government services, etc [18].

2.2 Pros and Cons of Agents

Agent technology has several advantages and disadvantages when compared with other information technologies. The pros and cons of agent technology are discussed and listed below.

Pros of agent technology:

- Agents could become prominent as the Internet and web technologies continue to grow. In the past decade, several fields in computer science have adopted agent technology [11].
- *Agent-based computing* can be considered as a natural extension to *object-oriented programming* due to similarities in organising and handling complex structures [11]. The distributed nature of agents makes them extensible and simple.
- Agents are well suited for mobile device applications due to their small size and properties such as adaptability, scalability, mobility, etc. *Consumer oriented agent technology* is the future, says Paul Schulz, founder and acting chief of *Talking Point Inc.*
- Applications for mobile wireless devices using agent technology need limited bandwidth as the mobile device's wireless interface should be active only when sending and receiving data.

- Information monitoring and filtering agents reduce mobile device users' problems in viewing large amount of data in small-sized low resolution screen.
- *Voice recognition, mobility, location sensitivity,* and *personalized intelligent filters* can be enabled in software applications using agents [4, 23].

Cons of agent technology:

- Agents are not humans. If agents act on behalf of users, how, and to what level can users trust their agents? [16]. Users are responsible for all autonomous actions made by their agents [19, 16]. This has several security issues.
- Nascent agent platforms and execution environments for mobile devices are prone to errors. This increases security concerns while using agents in mobile device applications.
- Agents running in mobile devices could face wireless connection constraints as discussed in section 3.5.
- Implementing ideas like *agents learning from people*, *people learning from agents*, and *agents improving the creative performance of people* consume time. This is because,
  - agents should have access to all data that may be relevant,
  - accessed data should be scriptable and recordable, and
  - every users' behaviour is different.

2.3 Agent Platforms

An agent platform is the execution environment wherein agents are created, moved, cloned, and destroyed. Agent platforms are generally made up of platform-specific agents and transport services. *FIPA compliant* agent platforms follow necessary *FIPA specifications* [4]. Agents could communicate with other agents within the same platform or with agents in remote platforms. Agents can move from one location to another within the same platform, or can move from one platform to another i.e., mobile agents. Agent platforms should recognize foreign agents and messages received from foreign agents. Until now agent platforms are built as applications on operating systems.

Major agent platforms that are in existence include *FIPA-OS* [5, 6], *LEAP* [http://leap.crm-paris.com], *JADE* [http://sharon.cselt.it/projects/jade], *ZEUS* [http://more.btexact.com/projects/agents/zeus], *and Grasshopper* [http://www.ikv.de/products/grasshopper]. Compact agent platforms such as the *MicroFIPA-OS* agent platform [7] and the *LEAP* (Lightweight Extensible Agent Platform) are specifically designed for mobile devices to deal with the processing power and memory capacity restrictions in mobile devices. All the above mentioned open-source agent platforms comply with FIPA specifications. The FIPA-OS agent platform is one of the most widely used agent platforms [4, 5]. The FIPA-OS agent platform supports a majority of FIPA specifications, frequently releases updated versions, and has a well-managed active mailing list [5, 6].

The personal assistant is executed on the MicroFIPA-OS agent platform in a pocket PC. Some of the personal assistant's components are executed on the FIPA-OS agent platform in a workstation like desktop PC. Both the FIPA-OS and MicroFIPA-OS agent

platforms are described next in sections 2.3.1 and 2.3.2 respectively and a tabulated comparison is made between the same in section 2.3.3.

2.3.1 FIPA-OS Agent Platform

The FIPA-OS agent platform is the world's first open source FIPA compliant agent platform released by Nortel Networks in 1999 [24]. The FIPA-OS agent platform is now managed by Emorphia Ltd. The FIPA-OS agent platform is used in European collaborative research projects including SHUFFLE, CRUMPET, PATTERNS, and AgentCities [25]. The latest release of the standard FIPA-OS agent platform is version 2.2.0. The FIPA-OS agent platform has been primarily tested in Windows 95/NT environments and has also been successfully used in Linux and Solaris environments [7, 26]. The minimum system requirements to run the FIPA-OS agent platform are Pentium 166 MHz processor, 64 MB RAM, and 4 MB free disk space [26]. The basic software requirement to run the FIPA-OS agent platform is the Java Runtime Environment (JRE) [26]. All other third party software used by the FIPA-OS agent platform can be downloaded for free.

The FIPA-OS agent platform is a component-based agent development toolkit developed in Java. The platform-specific agents i.e., the Agent Management System (AMS) and the Directory Facilitator (DF), and the Message Transport Service (MTS) make up the FIPA-OS agent platform. The FIPA reference model is shown in figure 2.1.

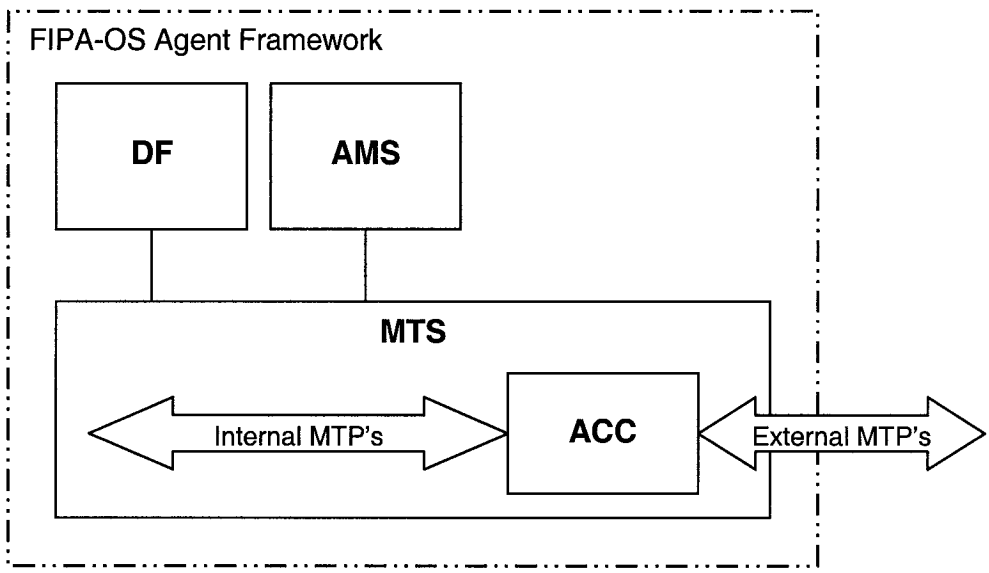

Figure 2.1 The FIPA Reference Model [26]

The DF provides yellow pages service to other agents [26]. *Platform management functions such as monitoring agent lifecycles and ensuring correct behaviour of entities are provided by the AMS* [26]. The Message Transport Service (MTS) is used to transfer ACL messages between agents both within and across agent platforms. Various Message Transfer Protocols (MTP) including the *fipaos-rmi, fipaos-ssl-rmi, corba*, and *http* are available for use in the MTS. The Agent Communication Channel (ACC) (discussed in section 2.3.5) is a part of the MTS and it *supports interoperability both within and across agent platforms* [26]. The FIPA-OS agent platform can be logically viewed as four layers, each made up of pluggable components [25]. The stack-based layered model of the FIPA-OS agent platform is shown in figure 2.2.

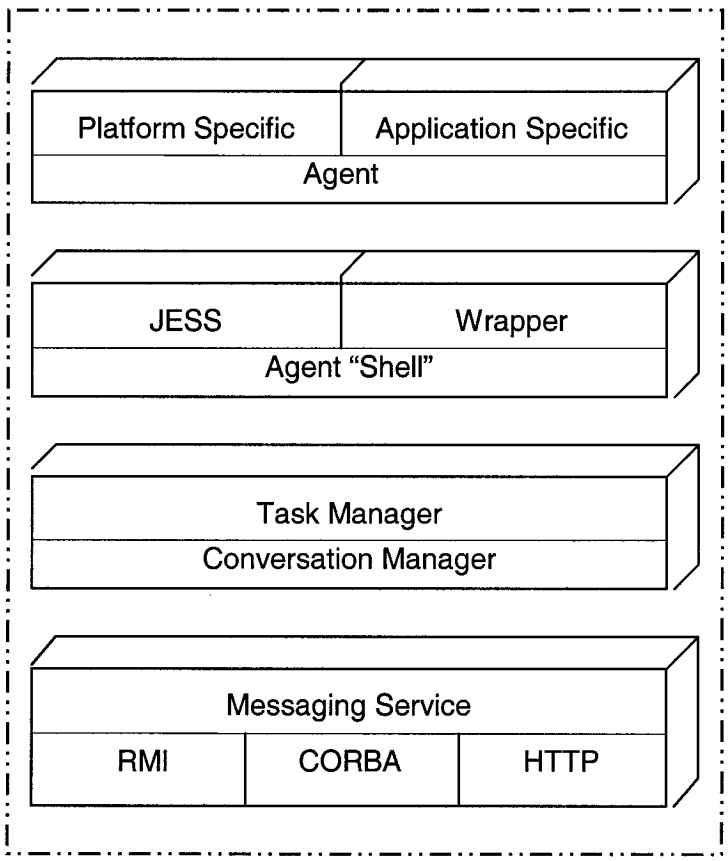

Figure 2.2 The FIPA-OS Layered Model [25]

The first layer is the *agent implementation layer* which is made up of platform-specific and application-specific agents. The next layer is the *agent shell layer. The FIPAOSAgent class provides a shell for agent implementation, which could be used by simply extending this class* [25]. The third layer is made up of the *task* and *conversation managers*. The task manager splits agent functions into smaller units, which independently carry out tasks and return results. *The Conversation Manager provides the ability to track conversation states, as well as mechanisms for grouping messages of the same conversation together* [25]. The final layer i.e., the messaging service layer is discussed in section 2.3.5.

2.3.2 MicroFIPA-OS Agent Platform

The MicroFIPA-OS agent platform is a compact lightweight version of the FIPA-OS agent platform. The MicroFIPA-OS agent platform is designed specifically for mobile devices such as *pocket PC's* and *mobile phones* to deal with the processing power and memory capacity restrictions in mobile devices [7]. This lightweight agent platform is developed in University of Helsinki, Finland, as a part of the *CRUMPET project* [23, 27]. The MicroFIPA-OS agent platform is based on the *PersonalJava Specification* [28] and has been successfully tested on *Linux for iPAQ* and *PocketPC/WindowsCE 3.0 & 2.11* [29]. The MicroFIPA-OS agent platform has also been successfully tested on *Microsoft PocketPC 2002* as a part of this thesis. The MicroFIPA-OS agent platform can run on any device that supports the PersonalJava specification and virtual machine, and has sufficient memory [29]. The lightweight agent platform's components can be added or removed depending upon the memory capacity of the device in which they are running.

Most of the functions and Application Programming Interfaces (API) in the MicroFIPA-OS agent platform are similar to those in the FIPA-OS agent platform. Most agents that are implemented on the FIPA-OS agent platform can also run on the MicroFIPA-OS agent platform, but with certain limitations. This is due to component limitations in the MicroFIPA-OS agent platform and restrictions in mobile devices' processing and memory capacity. For example, the MicroFIPA-OS agent platform only supports the *http* transport protocol to perform single and inter-platform communications, whereas the FIPA-OS agent platform supports various MTP's. MicroFIPA-OS agents can be deployed in two different ways [29]. In the first approach, MicroFIPA-OS agents run as a part of the FIPA-OS agent platform and FIPA-OS platform agents i.e., the AMS and DF run

on a fixed network. In the second approach, MicroFIPA-OS agents run independently and host platform-specific agents. The MicroFIPA-OS agent platform also supports a new minimal mode for creating agents, which do not use tasks and conversations [29]. The basic interaction between a standard FIPA-OS agent platform running in a desktop PC and a tailor-made MicroFIPA-OS agent platform running in a pocket PC is via a wireless link. This is shown below in figure 2.3.

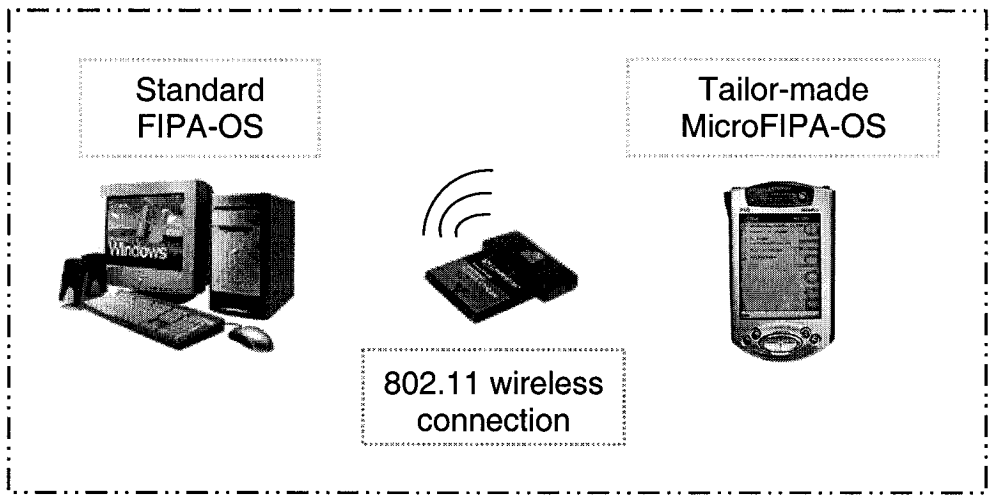

Figure 2.3 Basic Interaction between Agent Platforms

2.3.3 FIPA-OS and MicroFIPA-OS Agent Platforms Comparison Summary

Although both the FIPA-OS and MicroFIPA-OS agent platforms are FIPA compliant and the micro edition is a lightweight version of the FIPA-OS agent platform, there are few divergences between the two agent platforms that are worth remarking. Table 2.1 shown below lists the similarities and differences between the FIPA-OS and MicroFIPA-OS agent platforms.

Table 2.1 FIPA-OS and MicroFIPA-OS Agent Platforms Comparison Chart

FIPA-OS	MicroFIPA-OS
Signifies Foundation for Intelligent Physical Agents - Open Source agent platform	Signifies Micro edition of FIPA-OS agent platform
Designed for workstations such as desktop and laptops PC's	Designed for mobile devices such as pocket PC's and mobile phones
Occupies approximately 20 MB memory space	Needs approximately 5 MB memory space (developer-controlled)
Implemented in 100% pure Java and supports majority of FIPA specifications	Made up of limited functions and API's in the FIPA-OS agent platform
Supports fipaos-rmi, fipaos-ssl-rmi, corba, and http transport protocols to communicate with remote platform agents	Only supports the http transport protocol due to mobile devices' memory and processing power restrictions
Needs a java runtime environment to execute agents	Needs a PersonalJava based runtime environment to execute agents
The ACL is used for communication between agents	Also uses the ACL for communication between agents
Has a fully functioning ACC	Has a lightweight ACC

2.3.4 Agent Communication Language

Agents communicate and negotiate with other agents through single-platform communication or inter-platform communication. The FIPA Agent Communication Language (ACL) is the communication medium used for agent communication. The ACL is designed for both single and inter-platform communications and it is the only communication medium that is in existence for agents [6, 30].

The ACL is a set of primitives that allows an agent to state its intentions. An ACL message is an expression in Knowledge Query and Manipulation Language (KQML) format. The expression's arguments are made up of terms or sentences in Knowledge Interchange Format (KIF), formed from words in the ACL vocabulary. The elements that form the ACL vocabulary and their purposes are listed below in table 2.2.

Table 2.2 List of FIPA ACL Message Elements [30]

Element	Category of Elements
Performative*	Type of communicative act
Sender*	Participant in communication
Receiver*	Participant in communication
Reply-To	Participant in communication
Content*	Content of message
Language	Description of content
Encoding	Description of content
Ontology	Description of content
Protocol	Control of conversation
Conversation-id	Control of conversation
Reply-With	Control of conversation
In-Reply-To	Control of conversation
Reply-By	Control of conversation

All the above listed parameters may not be used in all FIPA ACL messages. The parameters needed for actual agent communication vary during communication. The only parameter that is mandatory is the *performative*, as it phrases the type of agent communication that is in progress. Other most commonly used parameters include the

sender, *receiver*, and *content*. The mandatory and commonly used parameters are highlighted in table 2.2 with a '*' symbol. The sender and receiver parameters state senders' and receivers' name, address, and platform details. The content element contains the actual content that is being transferred. An example ACL message making use of commonly used parameters is shown below in figure 2.4.

ACL message

```
(inform : sender      A_Agent
        : receiver    B_Agent
        : language    prolog
        : ontology    ack_bagent
        : reply-with  id
        : content     "service(receive, success)")
```

- *inform* is the "performative"
- "content" is the actual content of communication
- *A_Agent* is informing *B_Agent*, in *Prolog* (language) as an acknowledge
- Any reply by *B_Agent* will have *id* in "in-reply-to" parameter
- *ack_bagent* is the "ontology" i.e., content additional information

Figure 2.4 An Example ACL Message using *inform* Performative

2.3.5 Agent Communication Channel

The Agent Communication Channel (ACC) is a part of the MTS. The ACC is used for agent interoperability both within and across agent platforms [26]. Currently, the FIPA-OS agent platform has a fully functioning ACC and the MicroFIPA-OS agent platform has a

lightweight ACC that forwards incoming messages based on recipient agent identifiers. The ACC may use different MTP's for agent communication. If one MTP fails to send a message, the multiplexer tries other available MTP's. Also, one MTP may be used for single-platform communication and another MTP for inter-platform communication. As an example, the FIPA-OS agent platform could use the *fipaos-rmi* MTP for single-platform communication and could use the *http* MTP for inter-platform communication as shown below in figure 2.5. The ACC may even obtain information from platform-specific agents to complete its tasks [31].

In figure 2.5 shown below, *agent1* and *agent2* belong to the same FIPA-OS agent platform and use the *fipaos-rmi* transport protocol for single-platform communication. *Agent2* uses the *http* transport protocol for inter-platform communication with *agent3*, which runs on the MicroFIPA-OS agent platform.

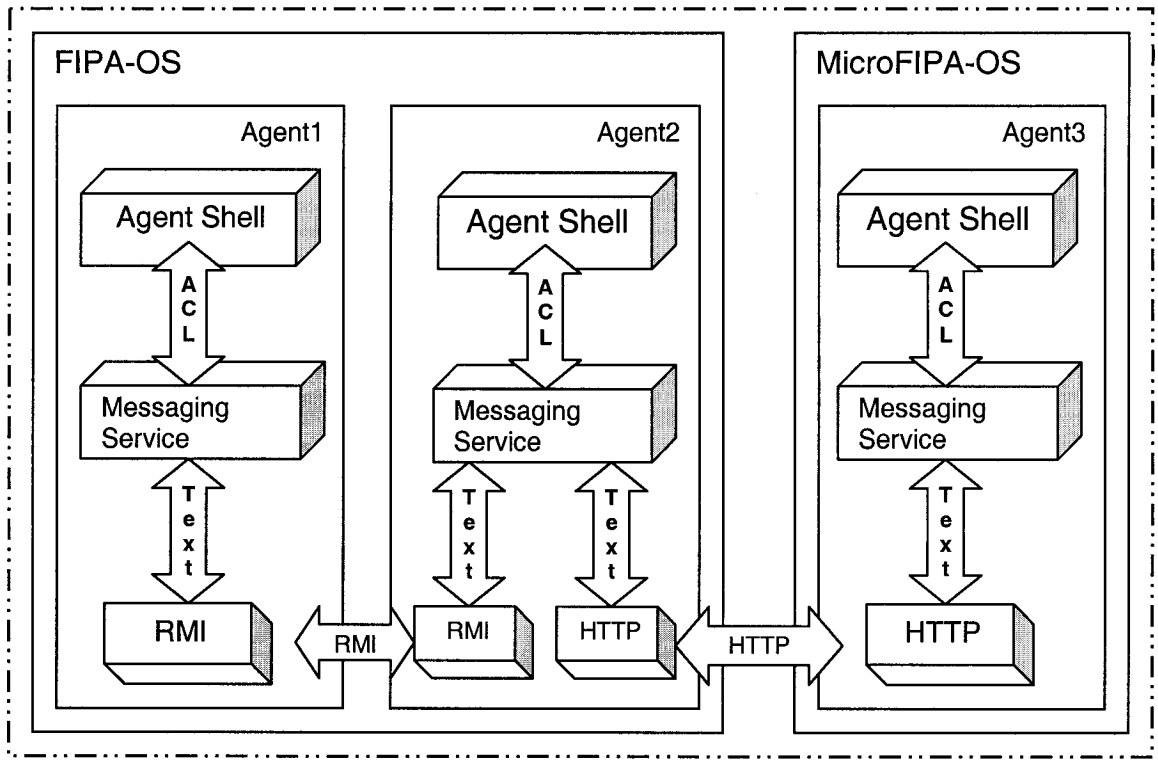

Figure 2.5 Schematic Representation of Single and Inter-Platform Communications

An agent can send a message to another agent residing in a remote agent platform in four distinct ways as shown in figure 2.6 [31].

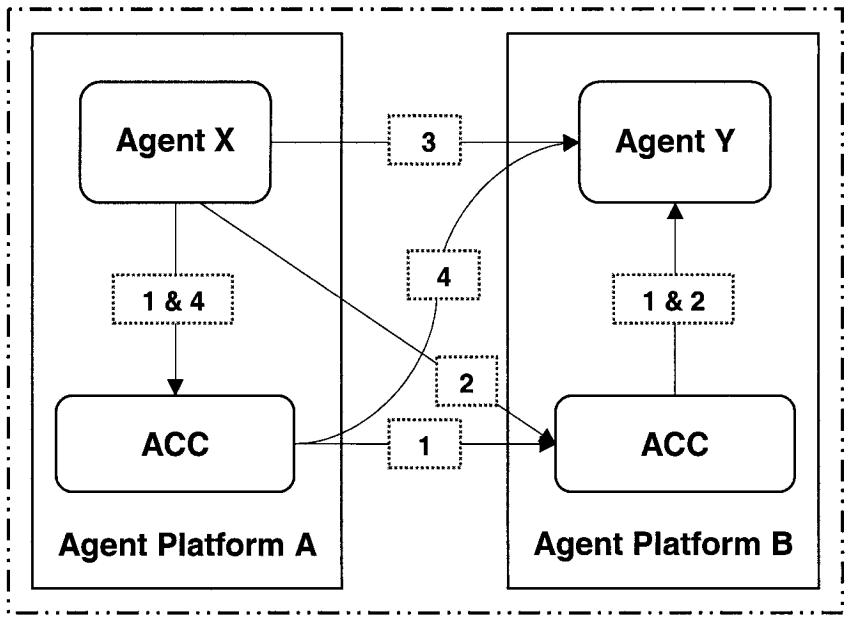

Figure 2.6 Four Methods of Communication between Agents on Remote Agent Platforms [31]

(1) → Agent X sends the message to be transferred to its local ACC in agent platform A. The ACC selects a suitable MTP and makes use of the same to send the message to the intended remote ACC in agent platform B. The remote ACC in turn delivers the received message to Agent Y in its agent platform B.

(2) → When Agent X has access to one of the remote ACC's interfaces, it can directly send the message to the particular remote ACC. The remote ACC in turn delivers the received message to Agent Y in its agent platform B.

(3) → Agent X uses a direct communication mechanism and sends the message directly to Agent Y in the remote platform. This method is not dealt by FIPA.

(4) → Agent X sends the message to be transferred to its local ACC in agent platform A. The ACC directly delivers the message to Agent Y in the remote agent platform B depending upon the destination address [31].

2.4 Agent Mobility

Agents communicate with each other either locally or remotely. Inter-platform communications can also be performed by using the concept of agent mobility [32]. The most innovative feature of an agent is mobility [32]. Using the concept of agent mobility,

- Agents are created at one location i.e., agent environment.
- Created agents are moved to another more resourceful location.
- Moved agents execute tasks in their new foreign location.

Agents possessing the agent mobility feature are called mobile agents [32]. The agent mobility feature significantly improves agents' performance [32]. By using the agent mobility feature, the network traffic could significantly be reduced. Mobile agents approximately need four times lesser bandwidth to complete tasks involving intense remote communication, when compared with the client-server approach [32]. Mobile agents complete their tasks in less time and reduce latency [32] when compared with non-mobile agents. Also, the mobile agent based approach doesn't need full-time active network connection to perform tasks i.e., after the agent code has been migrated, operations can be performed in the disconnected mode. Operating in the disconnected mode is a significant feature considering current wireless network constraints discussed in section 3.5.

Agent systems could result in performance bottlenecks due to lack of resources or overloading [33]. The performance bottleneck problem could arise in the personal assistant discussed in the thesis. The performance bottleneck problem is predicted and handled by using the proxy and adaptation agents in a workstation as discussed in sections 4.1.5 and

4.1.6. Common solutions to the performance bottleneck problem include delegating agents' tasks to other agents, making agents autonomous, and migrating agents to foreign hosts [33]. An alternate approach to the agent mobility is the agent cloning [33]. The agent cloning approach developed by *The Robotics Institute*, Carnegie Mellon University, solves insufficient resources and agent overloading problems. According to the agent cloning approach, *agents may clone, pass tasks to other agents, die, or merge*. Therefore, the agent cloning approach features both the agent mobility and transfer of tasks. For example, an agent clone could be created on a foreign host and tasks could be transferred from an overloaded agent in the local host to the cloned agent. This cloned agent could finally die after performing the required tasks. If additional resources are available at the local host then agents could first be cloned locally and later be migrated to a foreign host [33].

2.5 Ad-hoc Communication

An ad-hoc communication is a type of spontaneous communication wherein software or hardware devices communicate directly with each other without any centralized controlling system. An ad-hoc communication takes place in an ad-hoc network. Mobile and immobile devices dynamically create ad-hoc networks i.e., an ad-hoc network is formed instantly, on-demand, and without preparations. Mobile devices connected by short-ranged wireless links dynamically create ad-hoc wireless networks with no fixed infrastructure [34]. Mobile agents themselves could organize and administer ad-hoc wireless networks i.e., nodes that enter and leave a network manage the network [35]. *Thus, an ad-hoc network may have any network topology and may or may not have a gateway to any particular fixed network* [35].

Ad-hoc wireless networking is influenced by wireless constraints and also because of the fact that ad-hoc wireless networks don't have a fixed infrastructure [35]. The main constraints that influence wireless networks include *dynamic connectivity, unpredictable latency,* and *limited resources* [35]. As a solution, various predictive architectures could be used to reduce the impact of wireless constraints in ad-hoc wireless networks [35]. Another important concern of ad-hoc wireless networks is security. Several frameworks proposed for ad-hoc wireless networks' security are still in development phase [35].

2.6 Policies

Policies are rules or conditions set by the user in order to govern the behaviour of entities with a specific domain. Each policy generally consists of a triggering event, a set of conditions, a set of actions, and a life time. Policies [15] are generally applied in security – for restricting access, management – to assign rules for participating entities, and conversational policies – to structure and carry out conversations between entities. There are several policy languages that aim at formalizing the specification of policies so that they can be represented and interpreted by machines [15]. Future applications could learn from their current system behaviour and create policies at run-time.

Policies can be considered as a tool for managing agents as a particular matching policy acts as a guide for decision-making. As an example, policies could direct agents to either cache the necessary information or forward the information without caching. Thus, the behaviour of an agent system can be modified without influencing system architecture.

2.7 Related Work

Several research groups and organizations are currently focussing on agent technology [11]. Personal virtual assistants are also being developed, and some personal assistants are commercially ready with distinct features [11, 23]. Companies which develop commercial personal assistants include *Webley* [http://www.webley.com], *BotBox* [http://www.botbox.com], *eGain* [http://www.egain.com], and *Active Buddy* [http://www.activebuddy.com]. Commercial personal assistants are also developed by research groups. For example, *Information Society Technologies* (IST) has developed a *personal travel assistant* [23]. Several research groups are also working on agent security [16]. Once agents are proven to be secure, agent technology could set a new paradigm in problem solving [16].

The tasks performed by the personal assistant discussed in this thesis can also be accomplished by using other already well-established non-agent technologies. For example, similar tasks can also be performed by using the client-server approach, peer-to-peer technology, API, etc [36]. This thesis uses an agent-based approach to develop a personal assistant for mobile device users. One of the personal assistant's components i.e., the proxy agent resides in a workstation. The proxy agent acts as a gateway of communication for the personal assistant as discussed in section 4.1.5. This approach resolves agent overloading and resource limitation problems in mobile devices. There is no other personal assistant model which uses the approach employed in this thesis. In the client-server approach, the personal assistant's components in the mobile device would act as the client and the personal assistant's components in the workstation would act as the server. The client-server approach is outlined in figure 2.7 [36].

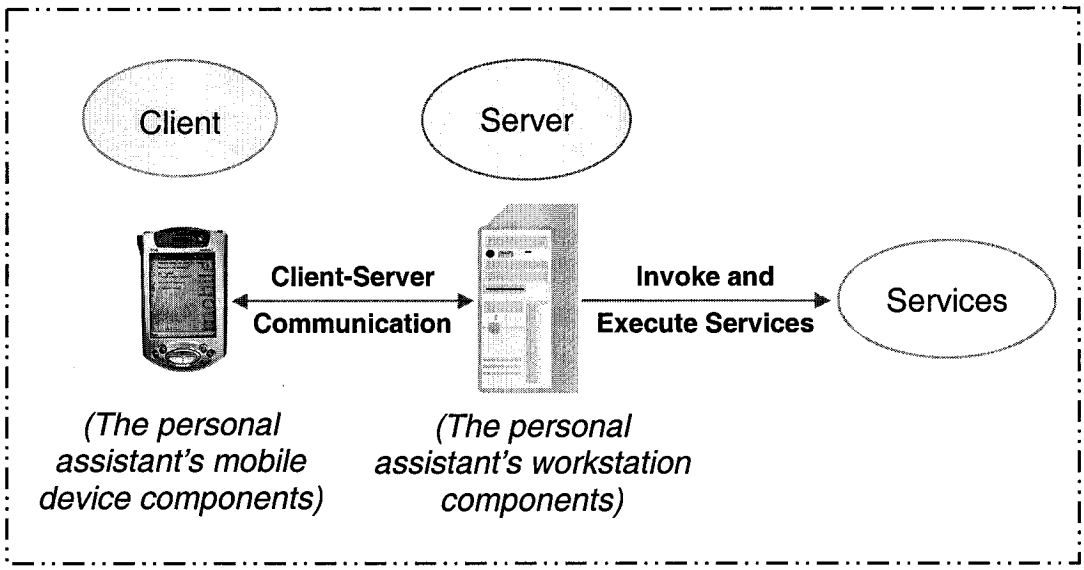

Figure 2.7 The Client-Server Approach

As seen from the above figure, the client i.e., the personal assistant communicates with its server (proxy agent) and vice-versa. The client states its needs and provides necessary information to the server. The server in-turn does what is needed to execute requested tasks.

Another alternative approach is by using mobile agents as shown below in figure 2.8. In this approach there is no need for separate client and server. The mobile device itself runs the mobile agent server. The server executes the mobile personal assistant i.e., the mobile agent, dispatches the mobile agent to the appropriate service agent to execute tasks, and also controls other on-going services. To enable this approach in mobile devices, light-weight agent platforms for mobile devices should fully support mobility [32], which still isn't the case.

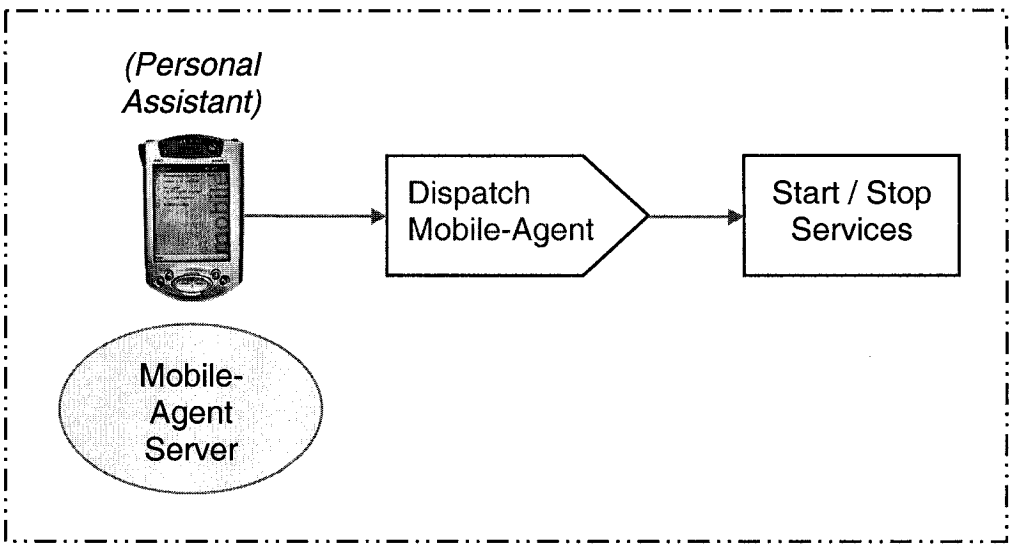

Figure 2.8 The Mobile Agent Server Approach

Performance bottleneck problems could arise in agent systems, particularly in multi-agent systems due to agent overloading or resource lacking in devices such as handhelds [33]. Mobile agents can resolve performance bottleneck problems in agent systems as discussed in section 2.4. The performance bottleneck problem is handled in this thesis by using the proxy agent in a separate workstation as discussed in section 4.1.5.

Most of the currently available personal assistants' design approaches are limited to assist users in specific applications. The main difference between personal assistants stated above and the personal assistant described in this thesis is that the latter tries to be as generic as possible i.e., not restricted to assist specific applications only. For example, consider the *personal travel assistant* designed by *Information Society Technologies* (IST) research group, Queen Mary College, London, UK [23, 27]. The *personal travel assistant* is based on agent technology and provides travel related assistance only. The *Personal travel assistant* is a part of the CRUMPET (Creation of User-friendly Mobile services

Personalised for Tourism) project [27]. Important points to consider in the CRUMPET in regard to the personal assistant discussed in the thesis are as follows [35, 6].

- The main goal of the CRUMPET is to offer value-added nomadic services to mobile device users.
- The CRUMPET applies agent technology for service creation, deployment, distribution, and personalisation for mobile device users.
- The CRUMPET dynamically adapts tourism service as per changing technical environments.
- The CRUMPET can execute in IP networks, Wireless LAN's, and mobile networks supporting WAP technology. The CRUMPET follows the W3C standards for device independent service delivery. The CRUMPET supports next generation mobile devices including PC hybrid terminals.
- The CRUMPET is implemented on a standards-based open source agent framework.

2.8 Summary

This chapter mainly focussed on introducing agent technology and discussed the pros and cons of agent technology. The FIPA-OS and MicroFIPA-OS agent platforms are then discussed in detail. This was followed by the FIPA Agent Communication Language (ACL) and Agent Communication Channel (ACC) description. The chapter further discussed and surveyed agent mobility, ad-hoc communication, policies, and the personal assistant's related work.

The next chapter introduces the personal assistant and states the personal assistant's main objectives. Next, the services that could be assisted by the personal assistant are illustrated. The next chapter also discusses the significant aspects of the personal assistant when compared with other personal assistants. Finally, the next chapter discusses technical challenges faced while designing and implementing the personal assistant.

CHAPTER 3

Overview and Feature Set

This chapter first introduces the personal assistant and states the personal assistant's main objectives. The chapter then illustrates user-oriented services and integrated services assisted by the personal assistant. Next, the significant aspects of the personal assistant when compared with other personal assistants are discussed. Finally, various technical challenges faced while designing and implementing the personal assistant are discussed.

3.1 Introduction

Personal assistants are also termed as Personal Internal Assistants (PIA). Personal assistants designed using agent technology fall under the category of software agents and are intended to run in mobile devices. Personal assistants are defined as *programs that act semi-autonomously on behalf of users or applications as a guide or coach.* The main objective of the personal assistant discussed in this thesis is to act like a virtual secretary for mobile device users and provide assistance to users to perform tasks in mobile devices. A personal

assistant is built for a single mobile device user to act and perform semi-autonomous actions only on that particular user's behalf [19]. Each mobile device user could have a customized personal assistant based on his/her preferences and requirements. The personal assistant's GUI can include visual figures and icons, thereby reducing mobile device users' discomfort in using the touch-screen keypad and small-sized buttons.

The personal assistant is capable of interacting with external agents and applications. An implementation of one such personal assistant integrated with the *mobile agent-based ad-hoc communication system* is described in section 5.2.1. This personal assistant should also be flexible and adaptive than being standards-based, as wireless technologies would continue to evolve in future [12].

3.2 Features and Services

The personal assistant isn't restricted to provide assistance to any specific Internet application for mobile device users but also to various Internet applications. The basic idea of the personal assistant is to work autonomously on behalf of a particular user by keeping track of the actions performed by the user. The personal assistant's components communicate with each other in order to perform tasks. If necessary, the personal assistant's components would communicate with external agents in remote platforms in order to perform tasks. External agents could even be other personal assistants. For example, if a mobile device user wants to fix a suitable time, place, and duration for meeting with his/her colleagues, he/she can instruct his/her personal assistant to do the same by specifying his/her case-specific preferences. The process of meeting schedule negotiation is discussed in section 3.2.1. Few common tasks performed by personal assistants are as follows.

- Access and store most of the frequently used information.
- Organize information hierarchically according to users' personal settings and preferences.
- Quickly search and locate documents of interest and importance.

3.2.1 User-Oriented Services

The personal assistant's level of assistance or involvement in any application is based on the particular user's preferences and requirements. Some common services that could be assisted by personal assistants in mobile devices are described below in figure 3.1.

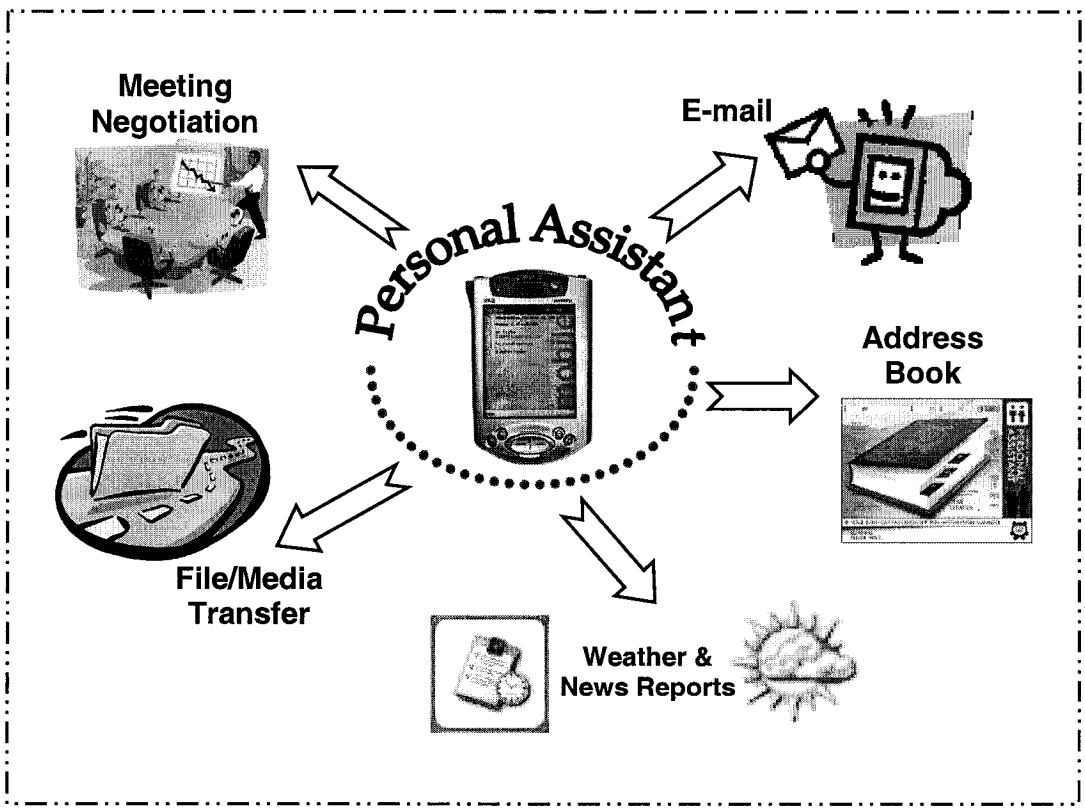

Figure 3.1 Representation of User-Oriented Service Assistance

- **Meeting Schedule Negotiation**: If a mobile device user wants to fix a suitable time, place, and duration for meeting with his/her colleagues, he/she can instruct his/her personal assistant to do the same by specifying his/her case-specific preferences. The case-specific preferences may include the meeting title, location, date, and time. The personal assistant first retrieves the names of the user's colleagues and thereafter retrieves the names and addresses of corresponding personal assistants, if any. The personal assistant communicates and negotiates with external personal assistants until all personal assistants have agreed upon the final meeting schedule. Finally, the personal assistant informs the fixed schedule to the mobile device user for confirmation.

- **File/Media Transfer**: Since agents can communicate only using the ACL, it's not feasible for agents to directly attach and send files to other agents. Instead, the file to be transferred is first converted as a string of bytes. The string is then wrapped in the ontology or content ACL field, and finally sent as an ACL message [37]. This method is feasible by making use of the personal assistant's proxy and adaptation agents (discussed in sections 4.1.5 and 4.1.6). File transfers could also be performed by using "sockets". The source agent transmits a file in a specific port number and the target agent receives the file by listening to the same port number.

- **E-mail**: Each time the wireless connection is activated between the personal assistant's components in the mobile device and workstation, the mobile device side components retrieve the user's e-mails, if any, from the proxy agent. The proxy agent would have already adapted any unknown messages by making use of the adaptation agent.

- **Address Book**: The personal assistant maintains the user's contact information, wherein contacts would be arranged according to the user's preferences such as family contacts, friends, frequent contacts, etc.

- **Weather and Other Reports**: The personal assistant could also act as an information gathering agent by providing the user with up-to-date information regarding weather, headline news, etc. The information is retrieved and filtered based on the user's preferences [22, 38].

3.2.2 Integrated Services

The personal assistant is integrated with the *mobile agent-based ad-hoc communication system* discussed in section 5.2. The personal assistant thereby provides integrated services to the user by means of this integration. Services offered to the personal assistant's user depend upon the user's profile and actual list of services offered by the *mobile agent-based ad-hoc communication system.* After profile matching is done, services such as the printing service, PDF service, MP3 service, conferencing service, and sidebar service could be offered to the user. The integrated services are represented in figure 3.2.

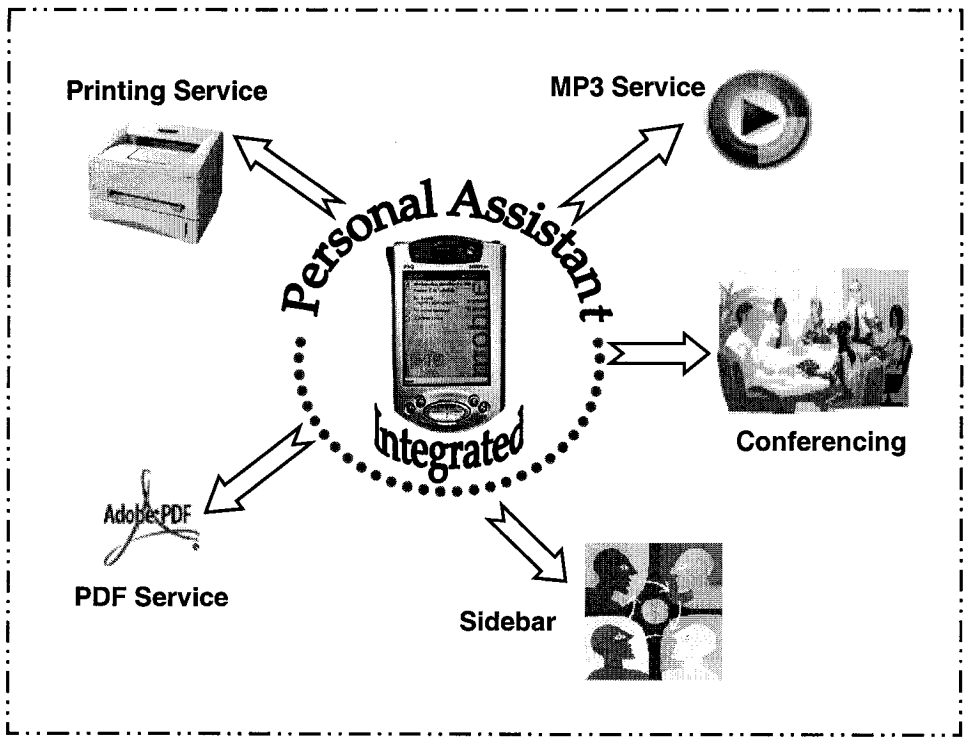

Figure 3.2 Representation of Integrated Service Assistance

- **Printing Service:** Mobile device users using the personal assistant can print documents using the printing service. The printing service would be uniquely represented by using printers' names and locations. Using the printing service's GUI (described in figure 5.21) the user can select a valid file to print and also has the option to change default print properties, such as number of copies. Finally, the status of the print job is sent to the user.

- **PDF Service:** The PDF service is a PDF writer used to convert DOC, PPT, and TXT files to PDF files. The 'PPT to PDF' conversion is a useful feature for pocket PC's without any PowerPoint presentation viewer. The *Adobe Acrobat Reader for Pocket PC* is used to view converted PDF files.

- **MP3 Service:** Mobile device users using the personal assistant can play *mp3* files. A java-based custom-made mp3 player is used to play mp3 files (described in figure 5.23). The user can select mp3 files from the local network or local memory. The selected mp3 file would be buffered directly from the local network to save mobile devices' memory space.

- **Conferencing:** Mobile device users using the personal assistant are invited to join the room conference with users already present in the room. The conference participants could communicate and discuss with each other using text, audio, or video. As this is a public conference, transmitted data will be available to all users currently participating in the conference.

- **Sidebar:** The sidebar service is a part of the conferencing service. Users who wish to communicate privately with other users may initiate the sidebar service. Only the user who initiated the sidebar service and other users who are invited to join the sidebar may participate in the conference. The number of users who could be invited by the initiator is unrestricted. Also, any number of sidebars may be active at a particular time.

3.3 Significant Aspects

Certain significant aspects of the personal assistant model discussed in this thesis when compared with other personal assistants stated in section 2.7 are discussed and listed below.

- One agent can play multiple roles i.e., one personal assistant serves multiple purposes.
- The personal assistant acts as an information filtering agent. By means of this, a large amount of seldom used content in mobile devices' small-sized screen is considerably reduced.
- One of the personal assistant's components i.e., the proxy agent resides in a workstation and acts as a gateway of communication for the personal assistant. By means of this approach,
  - resource limitation problem in mobile devices is resolved,
  - bandwidth requirement is reduced, and
  - network latency is reduced.
- The personal assistant can enable a mobile device to support unrecognized file formats without additional software requirements by utilizing the proxy and adaptation agents.
- The personal assistant needs active wireless interface only when sending and receiving data.
- The personal assistant communicates with external agents and platforms. The personal assistant can be integrated with other agent-based systems. An example integration scenario is discussed in section 5.2.1.2.

3.4 Technical Challenges

Certain technical challenges and constraints faced while implementing the current version of the personal assistant for mobile devices are discussed in this section.

Most of the limitations arise due to the use of a mobile device for agent execution. Currently, mobile devices lack in resources and their processing power is limited. Mobile devices also face frequent environment changes. Due to these drawbacks, the personal assistant often ends up in adaptability problems [12]. Also, both the MicroFIPA-OS agent platform and PersonalJava specification are in nascent stage and prone to minor errors. The personal assistant running in a mobile wireless device also faces wireless network constraints such as limited bandwidth, more latency, and unexpected disconnections when compared with wired networks.

It is believed that the performance of the current version of the personal assistant could be improved by,

- Enabling FIPA's new mobility concepts.
- Meliorating the lightweight agent platform to be as efficient as the FIPA-OS agent platform.
- Enabling voice recognition in the personal assistant.

3.5 Summary

This chapter first introduced the concept of personal virtual assistant for mobile device users and discussed the personal assistant's main objectives. User-oriented services and integrated services which could be assisted by the personal assistant were then illustrated. The chapter also discussed the significant aspects of the personal assistant when compared with other personal assistants. Finally, various technical challenges faced while designing and implementing the personal assistant were discussed.

The next chapter discusses the proposed design model of the personal assistant. The personal assistant's usage scenarios at various instances are also illustrated.

CHAPTER 4

Design and Usage Scenarios

In this chapter, the personal assistant's proposed design model and usage scenarios at various instances are discussed in detail. First, the basic components that make up the personal assistant are elaborated sequentially. Additional importance is given to the proxy and adaptation agents, which provide proxy and adaptation services respectively. The chapter then discusses the proposed personal assistant model. Finally, different scenarios outlining the personal assistant's usage at various instances are illustrated.

4.1 The Basic Components

The top-down problem solving technique is followed to design and implement the personal assistant. The personal assistant is broken down into different components i.e., agents. Each agent is given a name exemplifying its role. The agent-naming scheme specified by FIPA is

followed to ensure that the personal assistant's agents have unique identifiers. One of the main regulations to be followed in the agent-naming scheme is to name an agent with corresponding domain name. The personal assistant comprises of,

- User Interface Agent
- Management Agent
- Personal Agent
- Control Agent
- Adaptation Agent
- Proxy Agent

The above listed agents perform tasks individually and also by interacting with each other [13]. The user, management, and personal agents are designed to reside in java enabled mobile wireless devices. The proxy and adaptation agents are designed to reside in java enabled workstations. The mobile device side agent components communicate with external agents via the proxy and adaptation agents. Control agents are designed to reside in both mobile devices and workstations to establish wireless communication between the mobile device side components and workstation side components.

4.1.1 User Interface Agent

The main function of the *User Interface Agent* (UIA) is to manage interactions between the mobile device user and the personal assistant's GUI. Agents can communicate only using the FIPA compliant *Agent Communication Language* (ACL) [5]. The UIA is

used to convert the information inputted by the mobile device user into ACL messages and vice-versa.

In today's mobile wireless world, all users wish to interact with their machines with ease [9]. This can be accomplished by using visually appealing graphical icons instead of standard texts in graphical user interfaces and by enabling *voice recognition* technology. The UIA represents information in a visually appealing way to the mobile device user. The UIA finally sends converted information i.e., ACL messages to the management agent. All UIA components are represented below in figure 4.1.

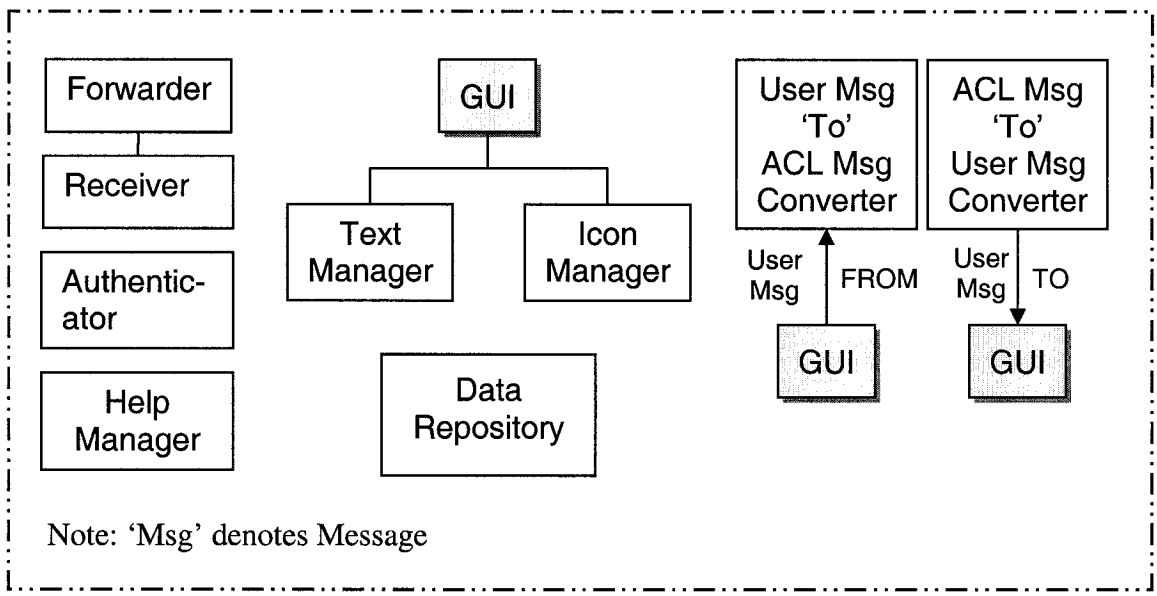

Figure 4.1 User Interface Agent Components

The personal assistant's agents require their own *forwarder* and *receiver* mechanisms for the purpose of communication with other agents (both internal and external). The *authenticator* is used to authorize the mobile device user to start utilizing the personal assistant. The *help manager* provides instant help to the mobile device user. An

instance wherein the *help manager* is used is shown in figure 5.6. A finite list of ACL field values is maintained in a data repository to help the mobile device user to specify values for mandatory ACL fields. The personal assistant's GUI is managed by using the *text manager* and the *icon manager*.

4.1.2 Management Agent

The purpose of having the *Management Agent* as a part of the personal assistant is to manage and store system and context information including other agent details i.e., the agent name server. The necessary information is cached in the local database and subsequently parsed in order to respond to other agents' requests [37]. The received data is managed by applying a set of pre-built policies. Here, policies [15] are hard-coded with the management agent specifying a set of conditions set by the user. The particular matching policy acts as a guide for decision-making. As an example, policies could direct the management agent to either store the received message from the UIA or forward the received message without storing. Decisions based on policies are made by analyzing the received message's size, format, etc. All management agent components are shown in figure 4.2.

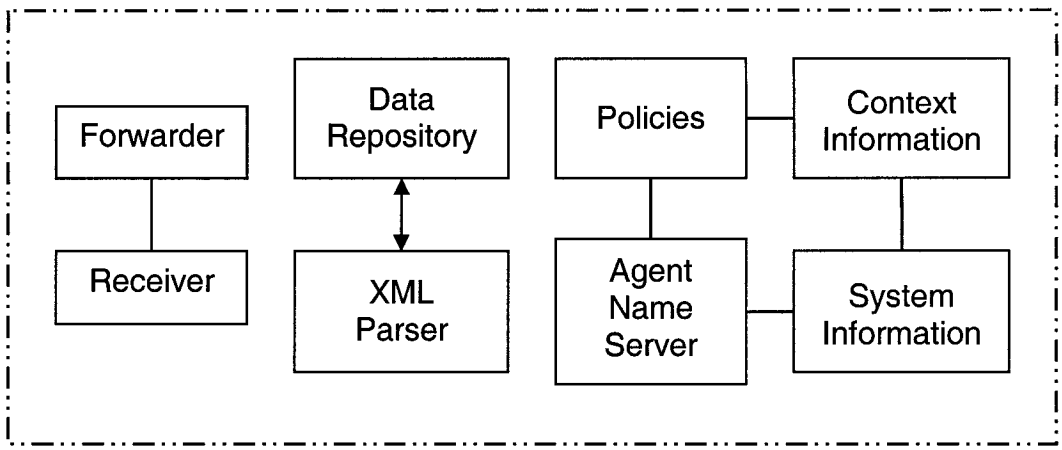

Figure 4.2 Management Agent Components

4.1.3 Personal Agent

The *Personal Agent* first communicates with the control agent running in the mobile device to get the current status of wireless connection between the mobile device and the desktop PC. This desktop PC executes the proxy and adaptation agents discussed in sections 4.1.5 and 4.1.6 respectively. When the wireless connection is active, the personal agent communicates with the proxy agent. The personal agent delegates tasks and messages to the proxy agent and also receives certain tasks and messages from the proxy agent. The personal agent's main function is to understand and process requests made by external agents and form appropriate replies. In order to perform this function, the personal agent may communicate and retrieve data from the management agent and may also communicate with the UIA to get responses from the mobile device user. The personal agent also performs selected adaptation functions and acts as a partial adaptation agent, which is discussed in section 4.1.6. All personal agent components are represented in figure 4.3.

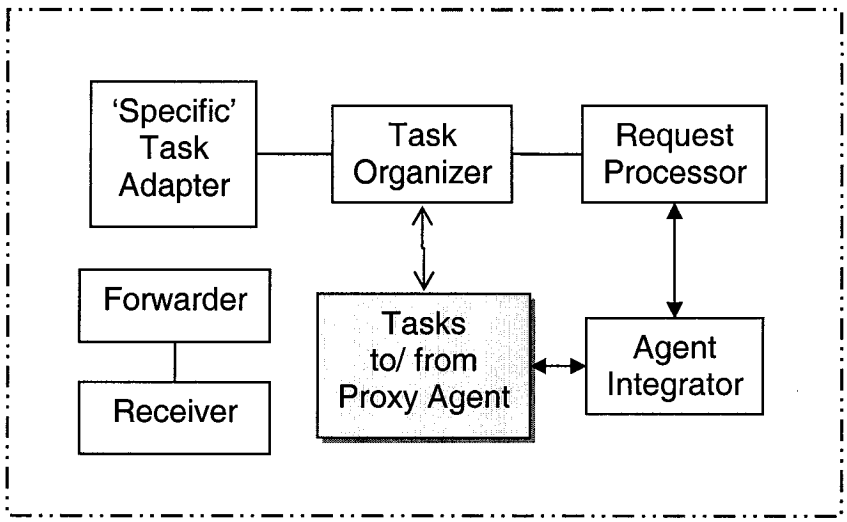

Figure 4.3 Personal Agent Components

4.1.4 Control Agents

Control Agents are designed to establish wireless communication between the personal assistant's components in the mobile device and workstation [40]. Since mobile device agents can communicate with external agents only using wireless link, it is necessary to consider bandwidth constraints and disconnection problems in wireless communication. Control agents reside in both mobile device and workstation and establish a wireless interface between the two devices whenever it is necessary and feasible using a common transport protocol (*http*). The functions performed by control agents are illustrated in figure 4.4.

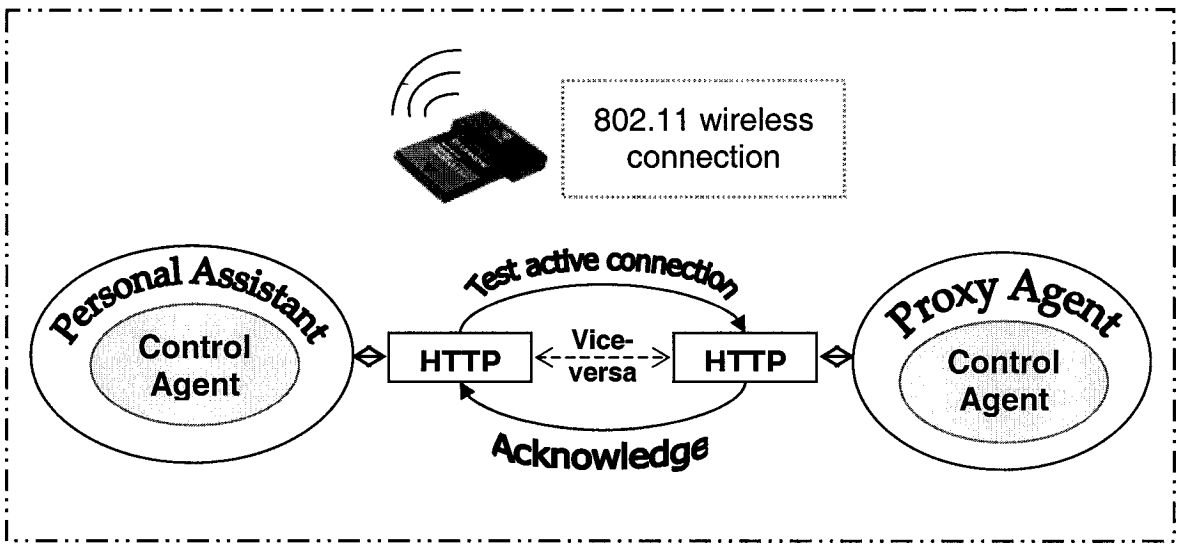

Figure 4.4 Control Agents' Functions

The life time of service agents and temporary agents (discussed in section 5.2) are decided by the control agent by analyzing various factors including current network traffic and resources currently used by agents.

4.1.5 Proxy Agent

The ***Proxy Agent*** [41] is designed to reside in a java enabled workstation. The proxy agent is executed on the FIPA-OS agent platform. By running on the FIPA-OS agent platform, the proxy agent can use several functions and transport protocols that are not available in the lightweight version. Therefore, the proxy agent has higher possibility to successfully communicate with external agents. The main reason to separately design a proxy agent is to overcome constraints faced by the personal assistant's components executing in a mobile device on the MicroFIPA-OS agent platform. Agents running in mobile devices face overloading, resource lacking, and other problems discussed in section 3.5. The proxy agent acts as a proxy to the personal assistant's components in the mobile

device. By doing so, the proxy agent reduces the workload of the personal assistant's components in the mobile device and acts as a gateway of communication for the personal assistant. The design representing the proxy agent as a gateway of communication is shown below in figure 4.5.

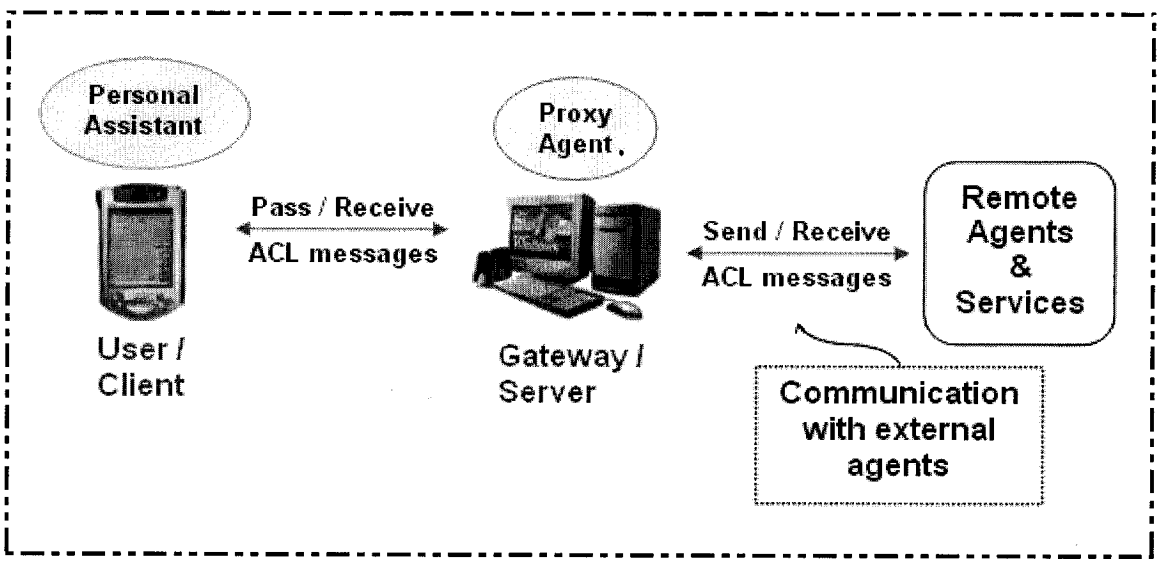

Figure 4.5 Proxy Agent Functions

From figure 4.5, it can be seen that any communication between the personal assistant and remote agents takes place via the proxy agent. The proxy agent approach can be compared with the client-server approach wherein the personal assistant's components in the mobile device act as the client and the proxy agent acts as the server [36]. The client-server approach and other related approaches are discussed in section 2.6.

The proxy agent's usage scenario is discussed below.

- The proxy agent first intercepts any incoming data to the personal assistant.
- Then, the proxy agent suitably adapts the received data using the adaptation agent (discussed in section 4.1.6).
- Finally, the proxy agent forwards the adapted data to the personal assistant's components in the mobile device.

As any regular external agent cannot be expected to tackle communication and other issues in MicroFIPA-OS platform agents, the use of the proxy agent is essential for prompt communication. Also, the proxy agent can be used to perform additional proxy services described in section 4.1.5.1. All proxy agent components are shown in figure 4.6.

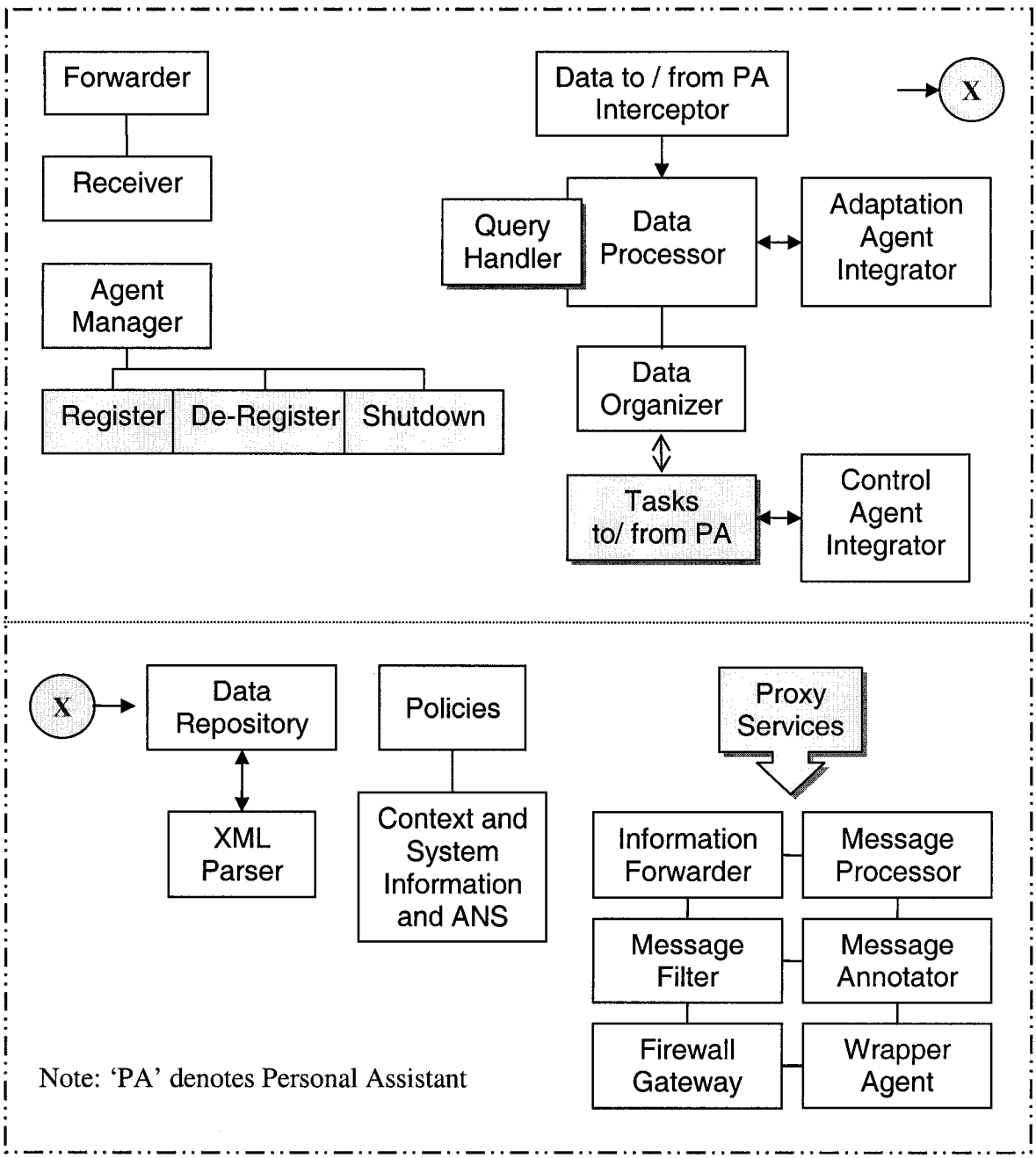

Figure 4.6 Proxy Agent Components

The proxy agent follows all basic procedures specified by the agent platform, such as *register*, *de-register*, and *shutdown*. The proxy agent also has its own *forwarder* and *receiver* mechanisms for the purpose of communication with other agents. The proxy agent is mainly made up of the *message interceptor, processor,* and *organizer*. The *message*

interceptor intercepts any message or data from/ to the personal assistant. The *adaptation agent integrator* is used to interoperate with the adaptation agent. The *message processor* along with the *adaptation agent integrator* analyzes, processes, and adapts the received data. The *query handler* handles external agent requests. The *message organizer* processes, filters, and stores the adapted data before forwarding the data to the personal assistant's components in the mobile device. All management agent components discussed in section 4.1.2 are also included in the proxy agent to perform management agent tasks. The *control agent integrator* is used to interoperate with the control agent in the workstation, which assists to perform inter-platform communication. The *adaptation agent integrator* and the *control agent integrator* can be viewed as a part of the proxy agent's *forwarder* and *receiver* mechanisms. It can also be noticed from the above figure 4.6 that additional components which perform proxy services discussed in section 4.1.5.1 are also included.

4.1.5.1 Proxy Services

Some common proxy services include *forwarding data, message processing / filtering / annotating*, and *firewall / protocol gateways* [41]. All proxy services are in one way or other helpful for the personal assistant to perform its tasks. The proxy agent not only acts as a gateway of communication for the personal assistant, but could also feature the above mentioned proxy services to act as a multi-purpose agent. The proxy agent could act as a *wrapper agent* as well; thereby it could simultaneously connect to one or more software applications identified by software descriptions [42]. The proxy agent could then transfer software applications to its workstation.

4.1.6 Adaptation Agent

The *Adaptation Agent* is used as a part of the proxy agent. It is used to adapt different types of messages and file formats, so that the personal assistant's components in the mobile device could understand and interpret those messages and file formats [43, 44, 45]. The adaptation agent restricts the time taken by external agents to respond, ignores delayed replies, and performs many other functions.

Mobile devices have limited capability and support limited number of messages and file formats [29]. Data formats that are recognised in the workstation where the proxy agent resides may not be recognized in the mobile device where the personal agent resides. For example, if the user receives an e-mail with a PDF attachment, the adaptation agent adapts the received PDF file to a string of bytes and forwards the same as an ACL message to the personal agent. Therefore, the adaptation agent enables mobile device users to view unrecognized file formats without additional requirements in mobile devices. Few other general functions and uses of the adaptation agent are discussed next in figures 4.7, 4.8 and 4.9.

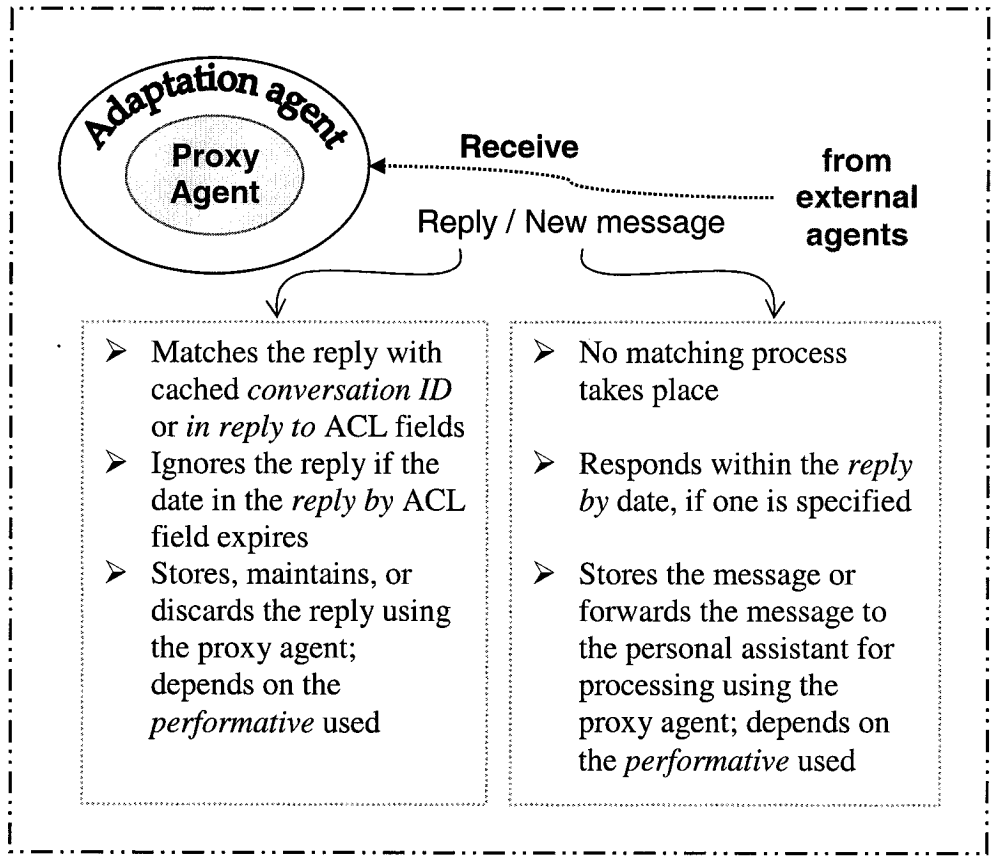

Figure 4.7 Message Handling by the Adaptation Agent

The above shown figure 4.7 tabulates the adaptation agent's processes when a reply or a new message sent to the personal assistant is received by the proxy agent. In addition to storing, processing, forwarding, and maintaining messages, the adaptation agent filters ACL messages before forwarding them to the personal assistant [46]. This is shown next in figure 4.8.

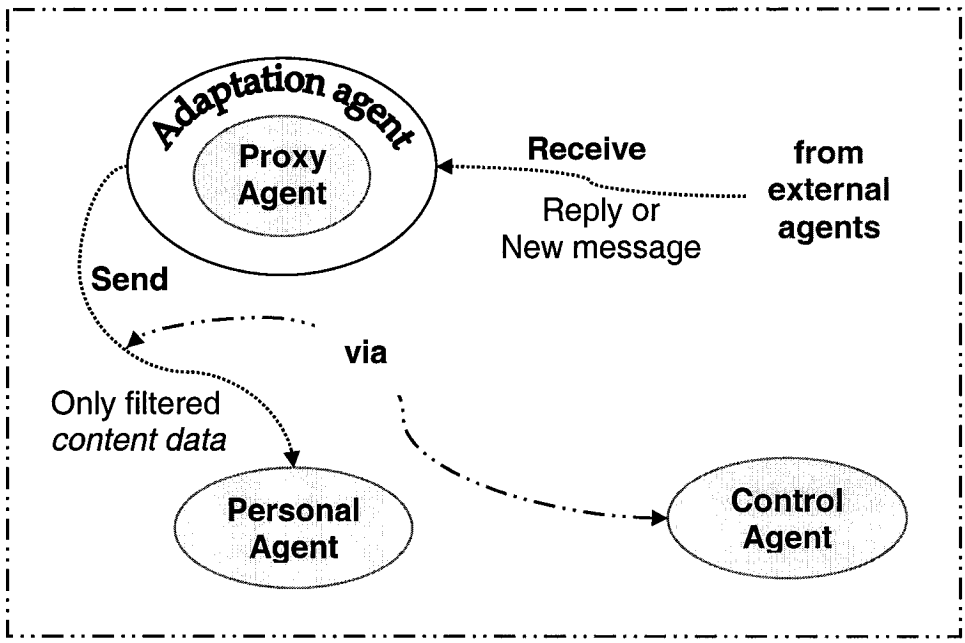

Figure 4.8 ACL Fields Filtering by the Adaptation Agent

The *content* field and other mandatory fields, if any, are only sent to the personal agent depending upon the *performative* used [30]. ACL messages are sent to the personal assistant only after confirming active wireless connection using the control agent. The adaptation agent handles the *not understood* message and performative. This is shown next in figure 4.9.

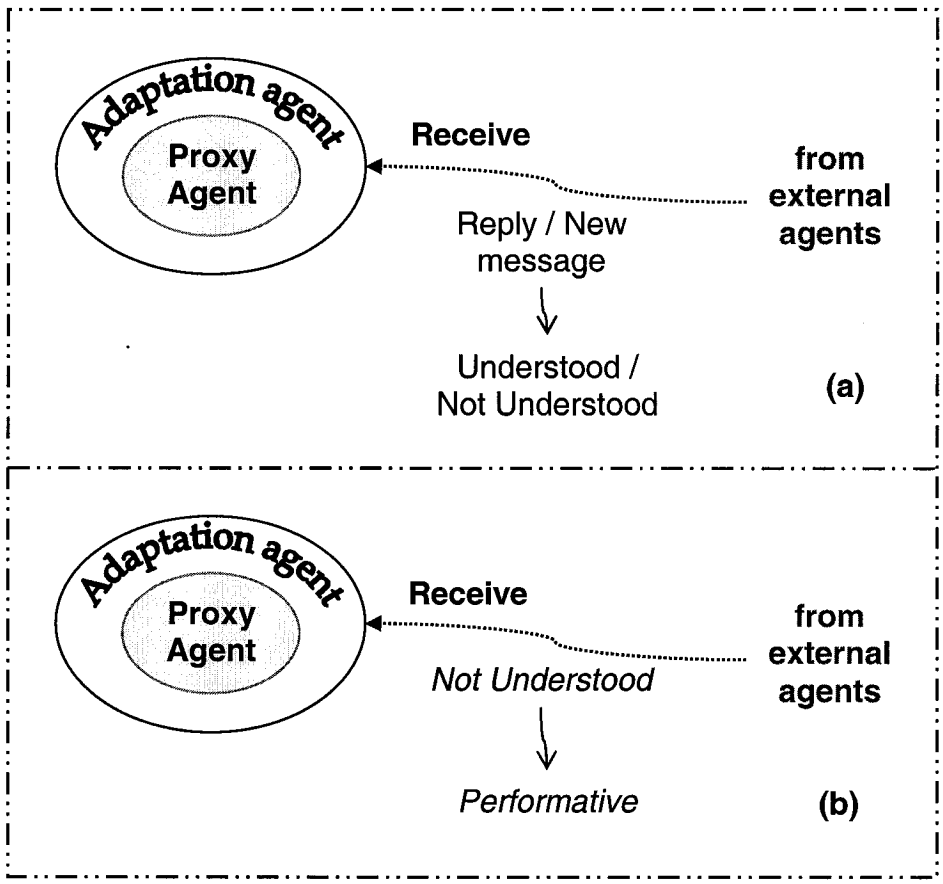

Figure 4.9 Handling *Not Understood* Message and Performative by the Adaptation Agent

There are two cases wherein the concept of 'not understood' comes into effect. In the first case, a new message or reply sent by an external agent is not understood by the proxy agent. In the second case, an external agent sends a *not understood* ACL performative as a reply, if the external agent doesn't understand the message or reply sent by the proxy agent. These two cases are shown above in figures 4.9(a) and 4.9(b) respectively. When the message received is not understood, the adaptation agent detects and imports new ontology and languages which match the received message structure [4, 30]. When the adaptation agent receives a *not understood* ACL performative as a reply, it re-sends the actual message to the corresponding receiver with all possible modifications to ACL fields. The adaptation

feature is expected to be the most necessary requirement for any agent in the near future considering the heterogeneous nature of agent environments. All adaptation agent components are represented below in figure 4.10.

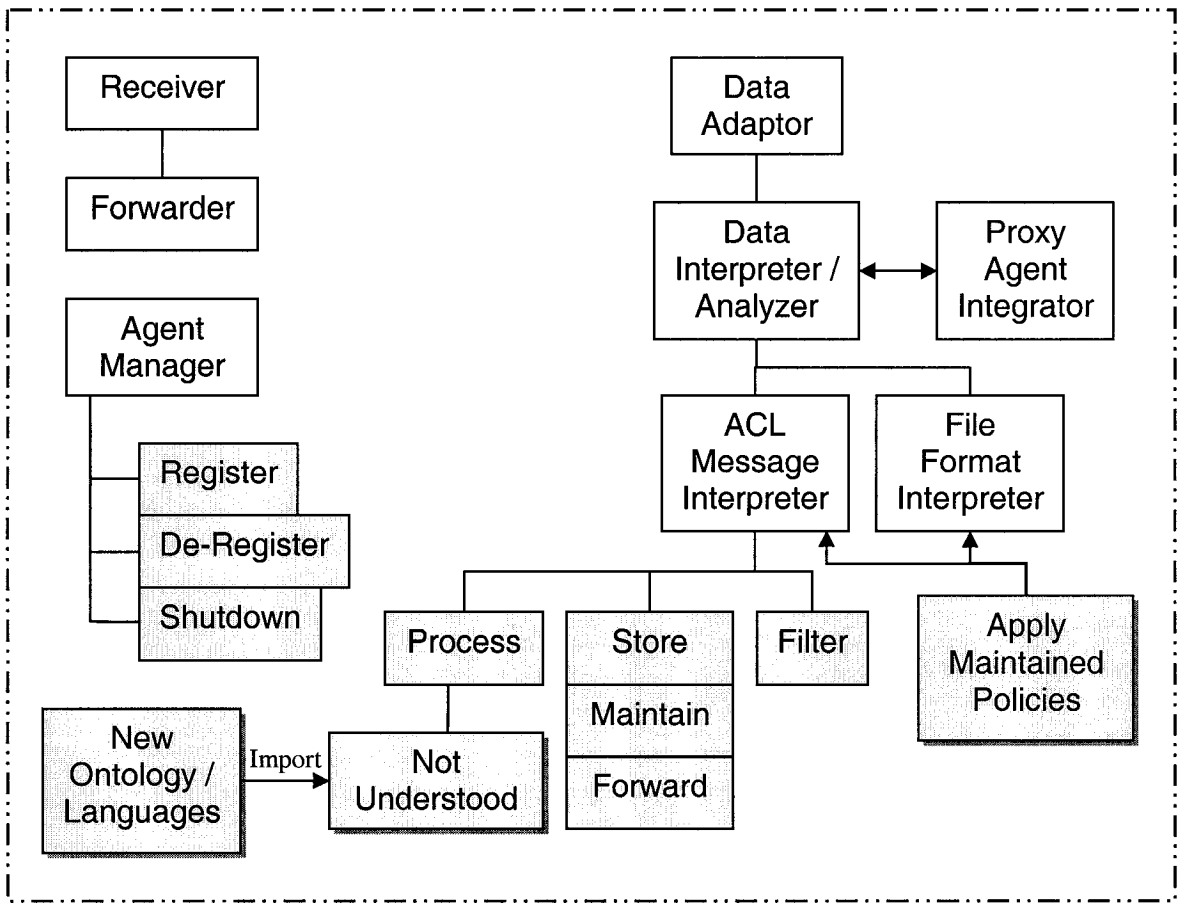

Figure 4.10 Adaptation Agent Components

Similar to the proxy agent, the adaptation agent also follows all basic procedures specified by the agent platform. The adaptation agent is mainly made up of the *data interpreter*, *analyzer*, and *adapter*, which perform adaptation agent operations with the help of policies. The *proxy agent integrator* is used to interoperate with the proxy agent. The *data interpreter and analyzer* along with the *proxy agent integrator* analyze the received

data from the proxy agent and perform required adaptation using the *data adapter*. The *ACL Message Interpreter* acts as a management agent and has several sub-components to process and manage ACL messages. An example is shown above in figure 4.9 (b).

4.2 Architectural Overview

The architectural overview of the personal assistant model is discussed in this section. The overall block diagram of the personal assistant model is shown below in figure 4.11.

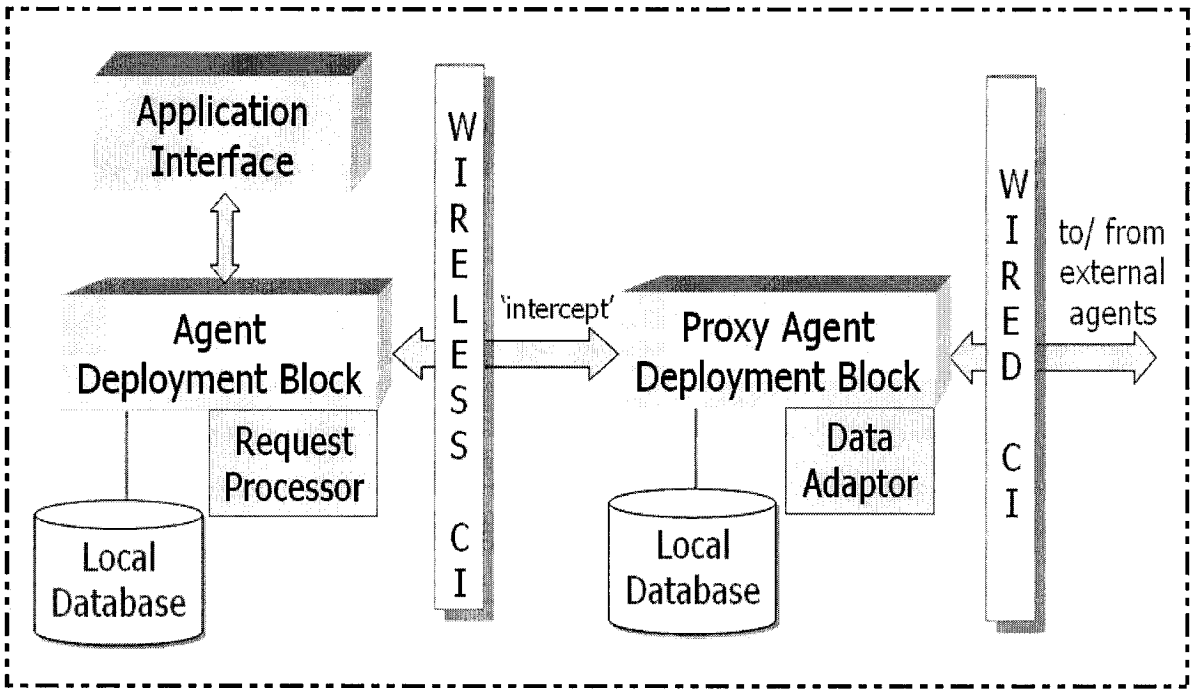

Figure 4.11 Personal Assistant Model – Block Diagram

The personal assistant model has an *Application Interface* which manages the user interface, interacts with the mobile device user to collect the input information, converts the inputted data into ACL messages, and forwards the same to the *Agent Deployment Block*

(ADB). The ADB is the most important block in the personal assistant model as it is the core of all operating agents. The ADB analyzes and processes the inputted information and if necessary, caches the processed data in the local database. The ADB is also responsible for processing external agent requests and sending appropriate replies. Then, the processed data is sent to the appropriate agent through a wireless interface.

The *Proxy Agent Deployment Block* (PADB) intercepts the data sent by the ADB and if necessary, it further processes, adapts, and caches the intercepted data. The required adaptation is performed using the *Data Adaptor* block and the necessary information is cached in the local database. Finally, the PADB forwards the processed data to the intended receiver through a wired interface. The next section discusses usage scenarios outlining the use of the personal assistant in various contexts.

4.3 Usage Scenarios

This section discusses usage scenarios of the personal assistant. First, a general scenario employed by the personal assistant is outlined. This is followed by case-specific scenarios such as,

- Sending a message or request to an external agent.
- Sending a request reply to the corresponding external agent.
- Receiving a message or request from an external agent.
- Receiving a request reply from a known external agent.

A general scenario employed by the personal assistant to perform its tasks is shown below in figure 4.12.

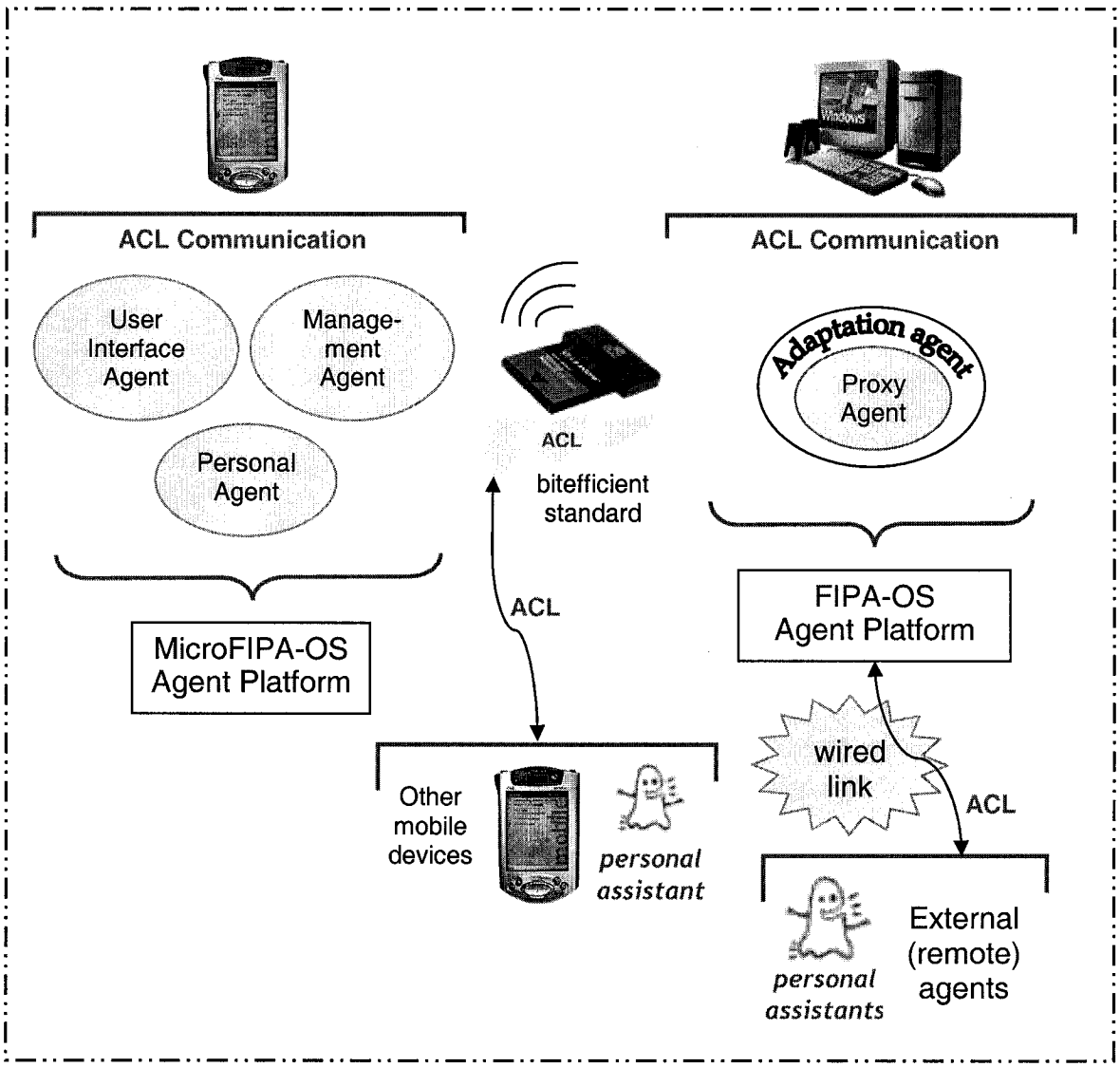

Figure 4.12 A General Scenario of the Personal Assistant Model

The user interface, management, personal, proxy, adaptation, and control agents are the software components that makeup the personal assistant. As seen from the above figure 4.11, the personal assistant's components that execute in a mobile device are implemented

on the MicroFIPA-OS agent platform and the proxy and adaptation agents execute separately in a workstation on the FIPA-OS agent platform.

The inputted information is processed by the personal assistant's components in the mobile device and if necessary, sent to the proxy agent through an 802.11 wireless interface. The personal assistant's components communicate with each other using the ACL. The ACL messages used for communication are optimized for efficient communication by following FIPA specific bit-efficient standards [40]. The proxy agent along with the adaptation agent further processes and adapts the received data, searches and locates the address of the intended receiver, and finally forwards the result as ACL messages to the intended external agent. The inter-platform communication between the personal assistant's components in the workstation and external agents takes place through a wired interface. The external agent may be any normal agent, application, software, or other personal assistant as well. Also, the proxy agent can communicate directly through a wireless interface with other personal assistants residing in mobile devices i.e., the proxy agent can act as a gateway of communication for more than one personal assistant [13].

4.3.1 Scenario 1. The Personal Assistant in Sending Mode

The *personal assistant in sending mode* scenario discusses the steps taken by the personal assistant to send a message or request to an external agent. The scenario also briefs the process of sending a reply to the corresponding external agent. The personal assistant's sending mode scenario is shown below in figure 4.13.

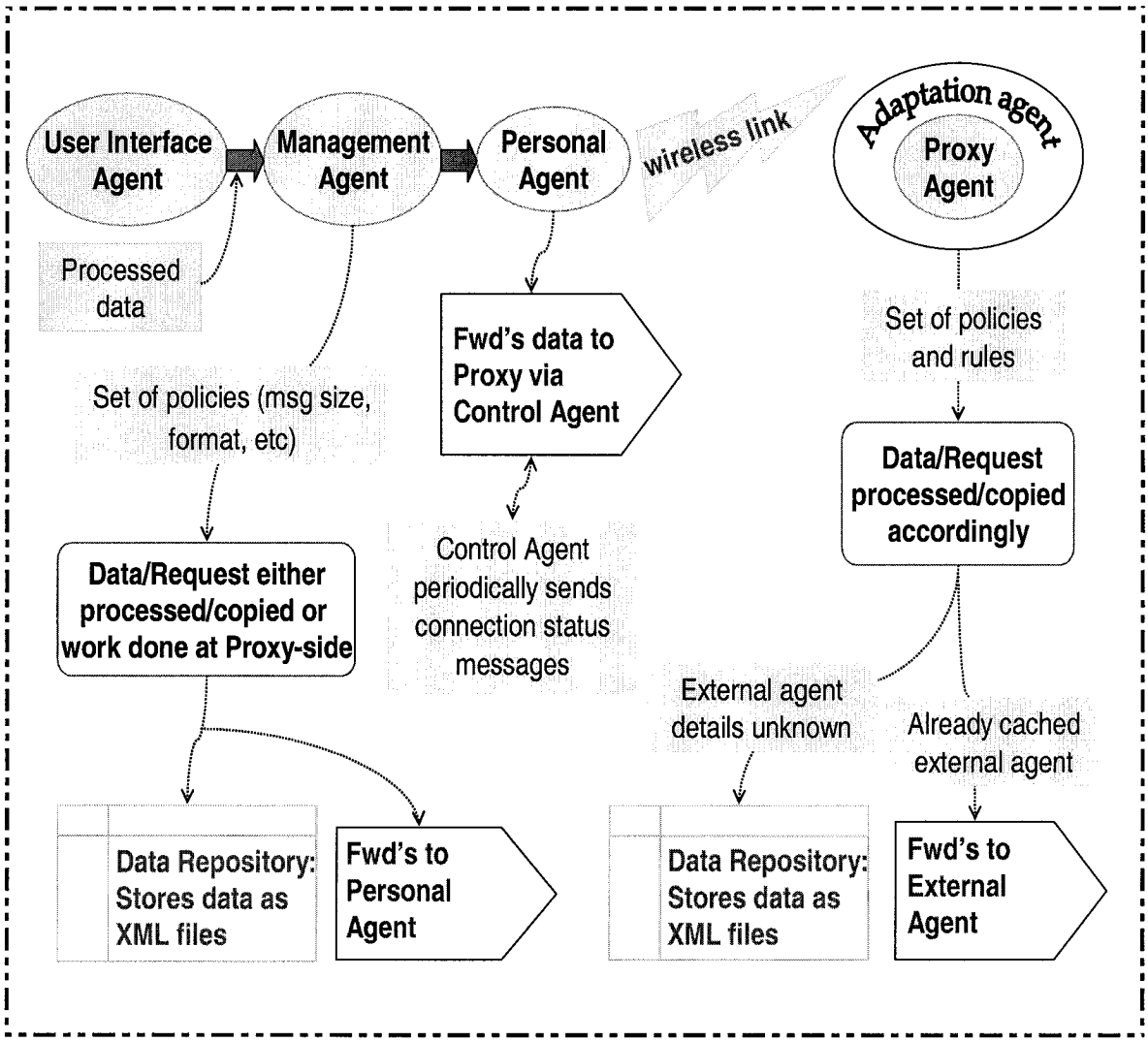

Figure 4.13 Scenario1: The Personal Assistant in Sending Mode

The User Interface Agent is the first agent that comes into consideration while executing the personal assistant. The UIA collects information from the user as discussed in section 4.1.1. The UIA first receives the data inputted by the user in his/her preferred input format, then processes and converts the received data into ACL messages, and finally forwards the result to the management agent. The management agent manages the received data by applying pre-defined user-specific policies. The management agent may also communicate with the personal agent to obtain case-specific cached results. If necessary, the

processed data is cached in an XML file for further processing. The personal agent searches and retrieves cached results in order to process requests made by external agents. The results of processed requests are then managed by the management agent. The processed data is sent to the proxy agent residing in a workstation through a wireless interface using control agents in both devices.

The proxy agent could act as a management agent and further process the received data. This depends on the level of management on the mobile device. If necessary, processed data is then adapted by the adaptation agent so as to be compatible with external agents. Finally, the proxy agent forwards the result to appropriate external agents. The proxy agent also stores the intended receiver's name, address, and other related fields for future use, if they are not already cached. The scenario elaborating the personal assistant's usage in receiving mode is discussed next.

4.3.2 Scenario 2. The Personal Assistant in Receiving Mode

The *personal assistant in receiving mode* scenario is shown below in figure 4.14. The scenario discusses the steps taken by the personal assistant when a message is received from an external agent. The scenario also briefs the procedure followed when a reply is received for a previously sent message from the corresponding external agent.

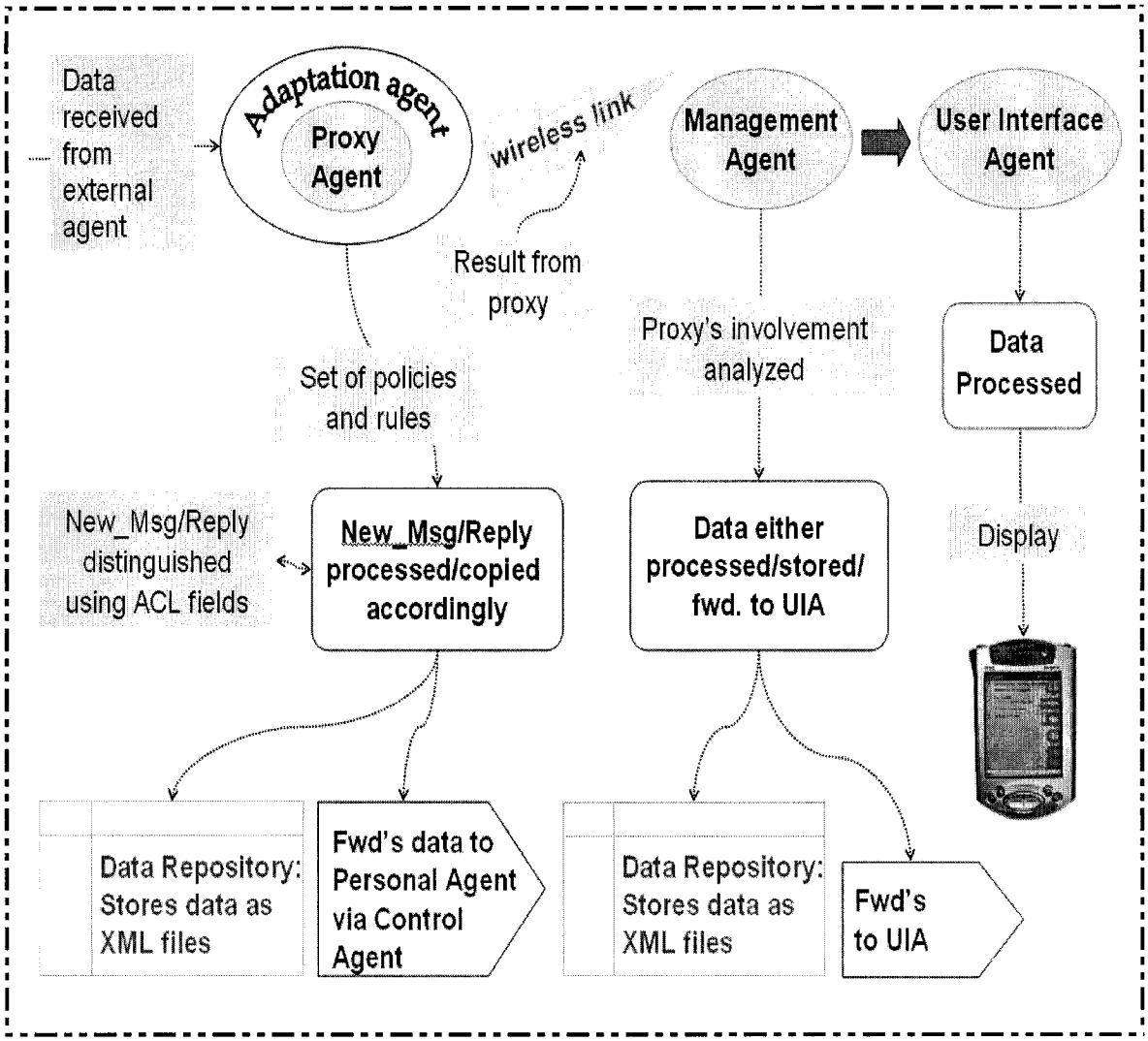

Figure 4.14 Scenario2: The Personal Assistant in Receiving Mode

When the proxy agent receives a message from an external agent which is not in the blocked list of agents, the proxy agent first checks the type of the received message. The received message may be a new message, request, or reply to a previously sent message. The proxy agent distinguishes received message by making use of ACL fields. For example, it uses the *in reply to* and *conversation id* ACL fields [30] to distinguish between a new message and a reply. Also, the proxy agent checks for validity of the received reply by examining the *reply by* ACL field of the original message. Valid replies, requests, and new

messages are further processed and stored. The decision to process, store, reply, or forward the received message is made by analyzing the received message's ACL fields, particularly the *performative* ACL field and applying pre-defined user-specific policies. The result is then sent to the personal agent residing in the mobile device via the control agent for further processing.

The received data is analysed by the personal agent in the mobile device. If necessary, the personal agent directs the management agent to process and store the received data. In certain cases, the processed data is sent to the UIA which converts the processed data into user readable messages and forwards the result to the user's mobile device.

4.4 Summary

This chapter mainly discussed the proposed personal assistant model and the personal assistant's usage scenarios at various instances. Each basic component of the personal assistant was also elaborated.

The next chapter discusses the implementation details and results of the personal assistant. The *mobile agent-based ad-hoc communication system* and integration details of the personal assistant with the ad-hoc system are also discussed. The next chapter finally evaluates the flexibility and performance levels of the personal assistant.

CHAPTER 5

Implementation and Results

This chapter discusses the implementation details of the personal assistant including class diagrams, user interface snapshots, command prompt snapshots, and code snippets. The chapter then describes the *mobile agent-based ad-hoc communication system* in detail including the ad-hoc system's integration details with the personal assistant. Various snapshots representing integrated services are also described in this chapter. Finally, the personal assistant's flexibility and performance levels are discussed.

5.1 Implementation Details

The personal assistant is implemented on a Compaq iPAQ H3850 pocket PC and a Windows 2000 desktop PC. Both devices need to have an agent execution environment to execute agents. The pocket PC uses the MicroFIPA-OS agent platform and the desktop PC

uses the FIPA-OS agent platform for agent execution. Both devices need to be java enabled in order to run the above mentioned agent platforms, as both agent platforms are coded in Java. The personal assistant is coded in Java in order to execute in both the agent platforms. The pocket PC uses the *Jeode* runtime environment [47] and the desktop PC uses the J2SE v1.3.1 [8]. Necessary information is cached by the management and proxy agents as XML files. The created XML files are named after the *conversation id* ACL field to ease information retrieval process. Cached XML files are retrieved using the *xerces* java parser [48] and are viewed on the Pocket IE (Internet Explorer) in the pocket PC and Internet Explorer 4 and above in the desktop PC. This section further elaborates on the communication infrastructure, XML file parsing details, and the personal assistant's class diagrams, user interface snapshots, command prompt snapshots, and code snippets.

5.1.1 Agents and Agent Platforms - Communication Infrastructure

The Agent Communication Language (ACL) is the only communication medium that is in existence for agents to communicate and negotiate with each other. The ACL was elaborated in section 2.3.4.

The communication infrastructure used should be portable, reusable, and efficient. In multi-agent systems agents need to communicate with each other using the ACL in order to exchange information. The communication may take place in a wireless environment which has bandwidth and other wireless constraints as discussed in section 3.5. Therefore, the agent platform used for agent execution should satisfy communication requirements and should use a transport protocol that is widely supported. In order to perform stable communication between agents, the FIPA-OS and MicroFIPA-OS agent platforms are used

to execute the personal assistant. The *http* is used as the default transport protocol for both single-platform and inter-platform communications, as the *http* is the only transport protocol supported by the MicroFIPA-OS agent platform [29]. The MicroFIPA-OS agent platform running in a pocket PC had problems to perform inter-platform communications. These problems were identified and resolved as a part of this thesis. For example, the messaging service layer's components were modified in order to perform inter-platform communications. The TCP/IP is the default transport mechanism as it is both efficient and stable [49]. The different communication layers under consideration are shown below in figure 5.1 [49].

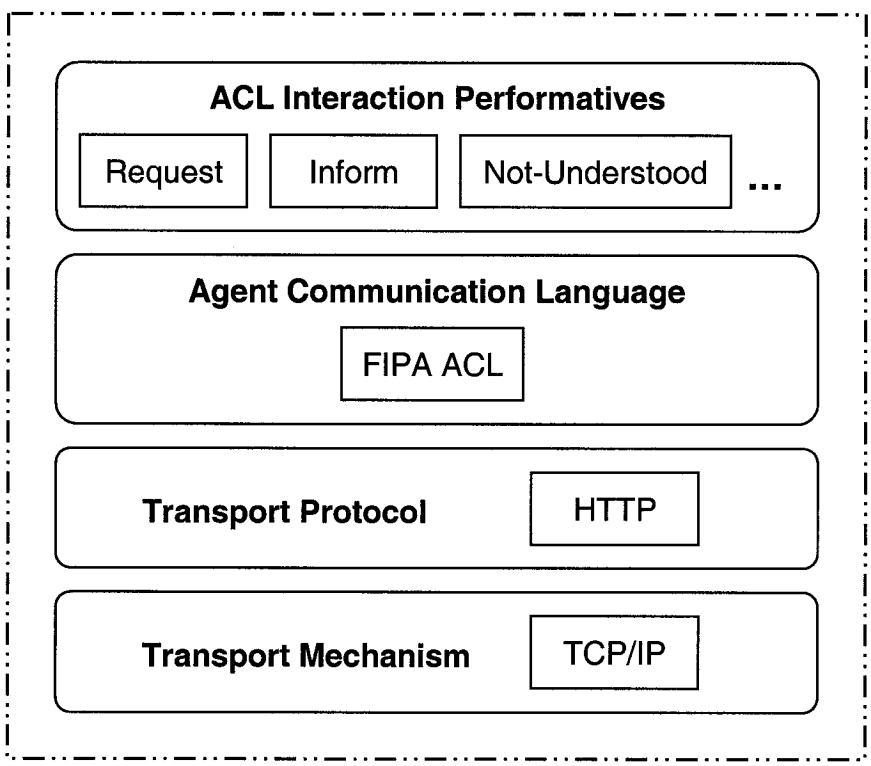

Figure 5.1 Communication Layers Infrastructure

5.1.2 XML File Parsing Details

The contents of any ACL message including the fields *performative*, *sender address*, *receiver address*, *content*, *language*, *encoding*, *ontology*, *protocol*, *conversation ID*, *reply with*, and *in reply to* are stored in an XML file. ACL messages are stored and maintained by the management and proxy agents, if they are required for future use. The management agent generates and maintains XML files in the mobile device and the proxy agent generates and maintains XML files in the workstation. An XML file containing a particular ACL message is either stored in the mobile device or in the workstation or even both depending upon the contents and significance of the particular ACL message. Figure 5.2 shown below represents a cached ACL message in XML format.

```
<?xml version="1.0" ?>
<Document>
  <Performative value="request" />
  <Sender value="UIAgent@pda01.genie.uottawa.ca" />
  <Receiver value="RM@main.genie.uottawa.ca" />
  <Content value="PrintGUI (Receive,Ready)" />
  <Language value="Prolog" />
  <Encoding />
  <Ontology value="RequestService_RM" />
  <Protocol value="FIPA-SL" />
  <ConversationID value="CID_20030624_RM" />
  <ReplyWith value="Service002" />
  <InReplyTo />
  <ReplyTo />
  <ReplyBy value="03-06-25" />
</Document>
```

Figure 5.2 A Cached XML File comprising ACL Elements

Cached XML files can be retrieved for further processing using the *xerces* java parser [48]. The xerces java parser is a compact and fully functioning parser that supports XML 1.0 recommendations and contains advanced parsing functionalities. XML files are displayed to the mobile device user on the Pocket IE (Internet Explorer) in the pocket PC and Internet Explorer 4 and above in the workstation.

5.1.3 Class Diagrams, Snapshots, and Code Snippets

This section discusses significant class diagrams, user interface snapshots, command prompt snapshots, and code snippets involved while implementing and running the personal assistant. First, the class diagram of the personal assistant's components in the mobile device is shown in figure 5.3.

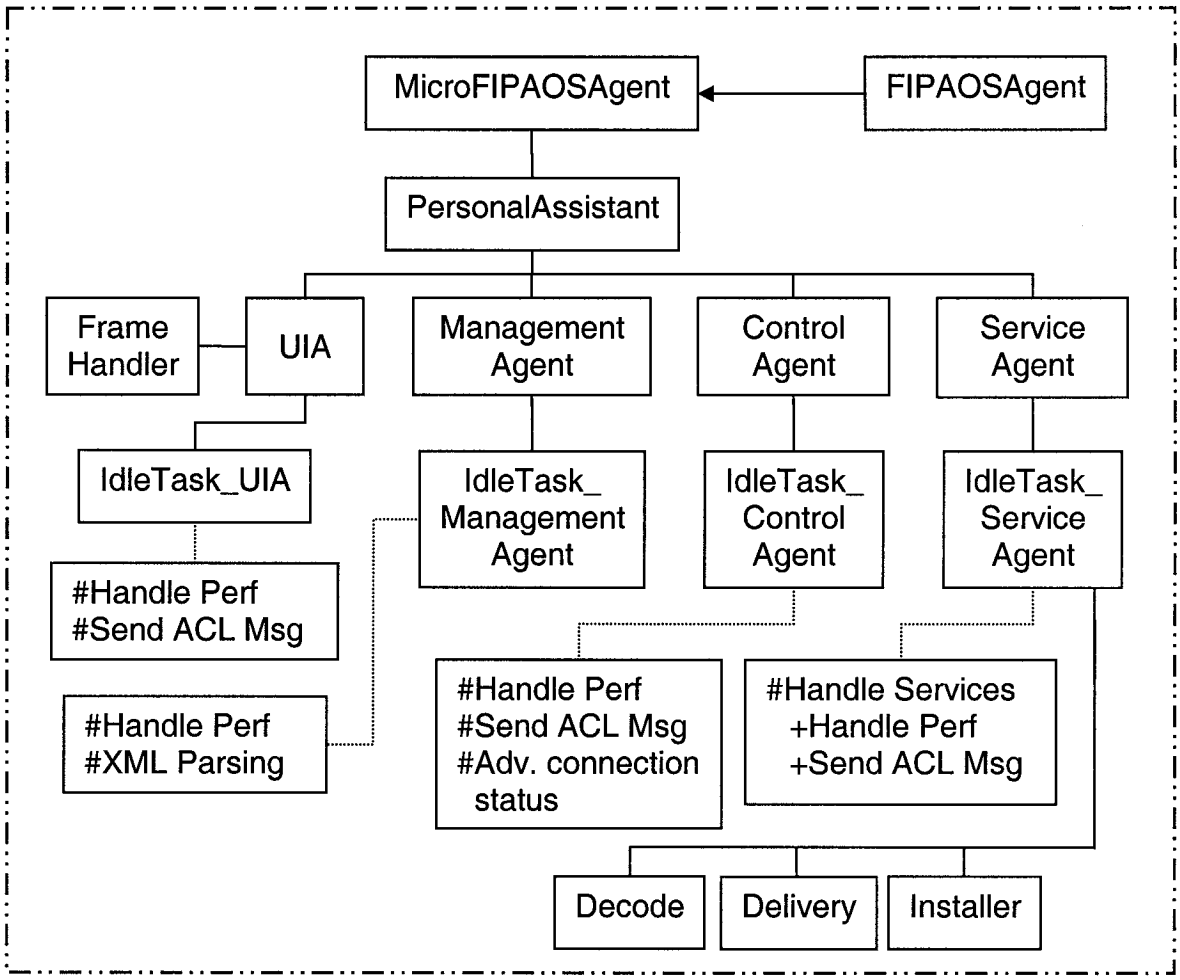

Figure 5.3 Class Diagram of the Personal Assistant's Components in the Mobile Device

As seen from the above figure, the *PersonalAssistant* class has a 'super' named as *MicroFIPAOSAgent* class which helps in creating and initializing agents and also handles the *Task* and *Conversation Managers*. Extending the *FIPAOSAgent* class enables support for the MicroFIPA-OS agent platform [29]. All the basic components of the personal assistant have an inner class named as *IdleTask* to handle agent tasks such as the following.

- Handle various ACL message performatives such as *handleInform*, *handleRequest*, etc.
- Handle all ACL message sending.
- Handle all XML file parsing.

The UIA has a *FrameHandler* to manage the personal assistant's GUI. The *Decode*, *Delivery*, and *Installer* classes are inner classes of the service agent's *IdleTask* (*IdleTask_ServiceAgent*) i.e., the personal agent's *IdleTask*. The *Decode* class is used to decode received messages and files, while the *Delivery* and *Installer* classes are used to transfer files to the proxy agent. The class diagram of the personal assistant's components in the workstation is shown next in figure 5.4.

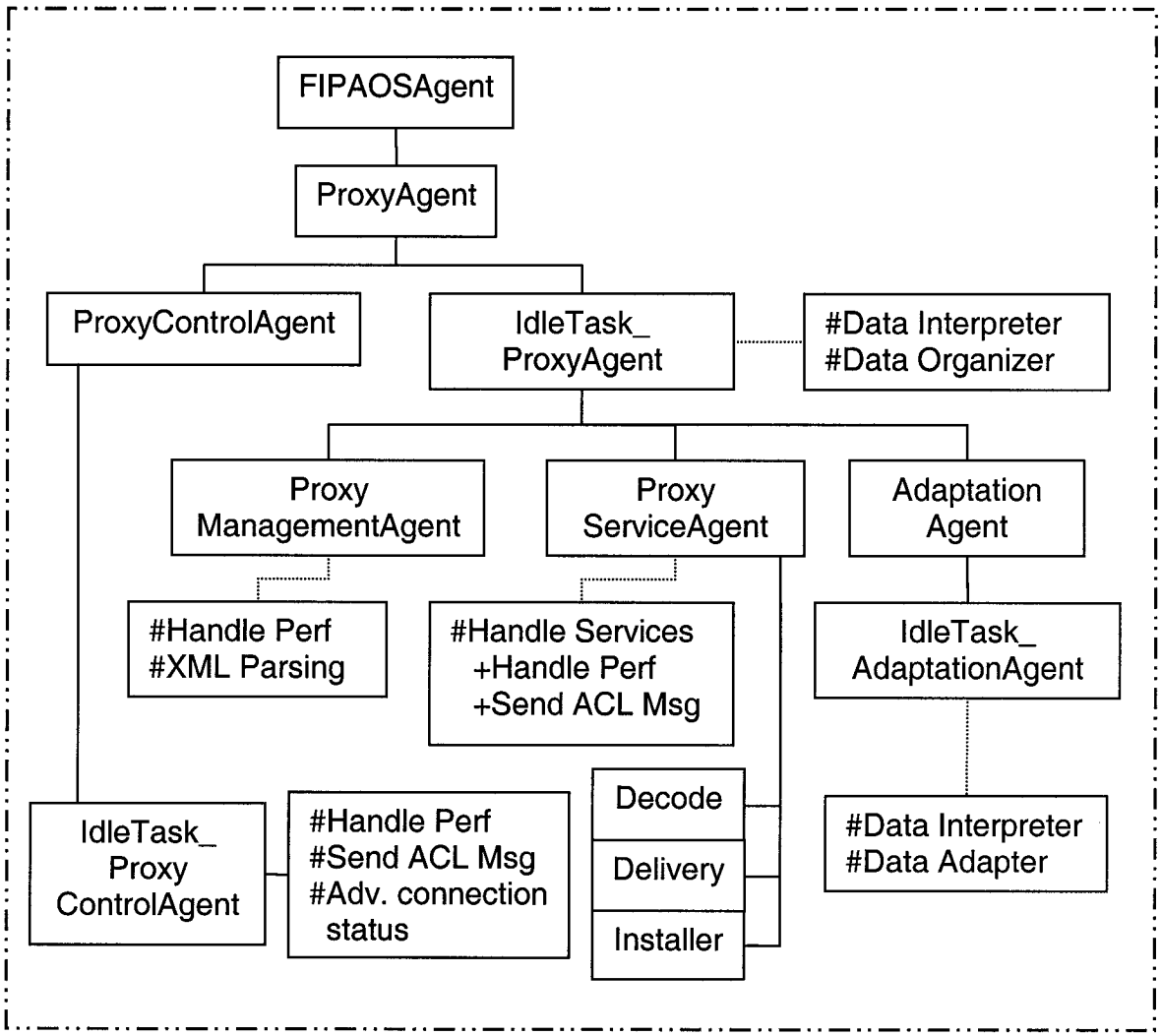

Figure 5.4 Class Diagram of the Personal Assistant's Components in the workstation

The above shown figure represents the class diagram of the personal assistant's components in the workstation i.e., the class diagram of the proxy and adaptation agents. The *ProxyAgent* class has a 'super' named as *FIPAOSAgent* class which helps in creating and initializing agents, and also handles the *Task* and *Conversation Managers*. Similar to the personal assistant's components in the mobile device, the *ProxyAgent* class also has an inner class named as *IdleTask* to handle agent tasks. The *AdaptationAgent* class is

implemented as a part of the proxy agent's inner class to help the proxy agent to provide proxy services. The *ProxyControlAgent* class is one of the inner classes of the proxy agent which helps the proxy agent to perform stable wireless communication with the personal agent in the mobile device. As explained in the class diagram of the personal assistant's components in the mobile device, each *IdleTask* handles tasks such as receiving and sending ACL messages and parsing XML files. The *ProxyServiceAgent* class has the *Decode*, *Delivery*, and *Installer* classes. The *Decode* class is used to decode received messages and files, while the *Delivery* and *Installer* classes are used to transfer files to the personal agent and external agents. The adaptation agent's IdleTask i.e., *IdleTask_AdaptationAgent* handles additional tasks such as data interpreting and adapting.

User Interface Snapshots

A prototype of the personal assistant model mainly contains four GUI frames, namely, the authentication screen, the application area selection screen, the ACL message creation and sending screen, and the cached message retrieval screen. The number of frames available in the personal assistant's GUI depends upon the personal assistant's implementation details.

The mobile device user has to be authorized in order to use the personal assistant as shown below in figure 5.5. The user can specify his/her preferred method to input information.

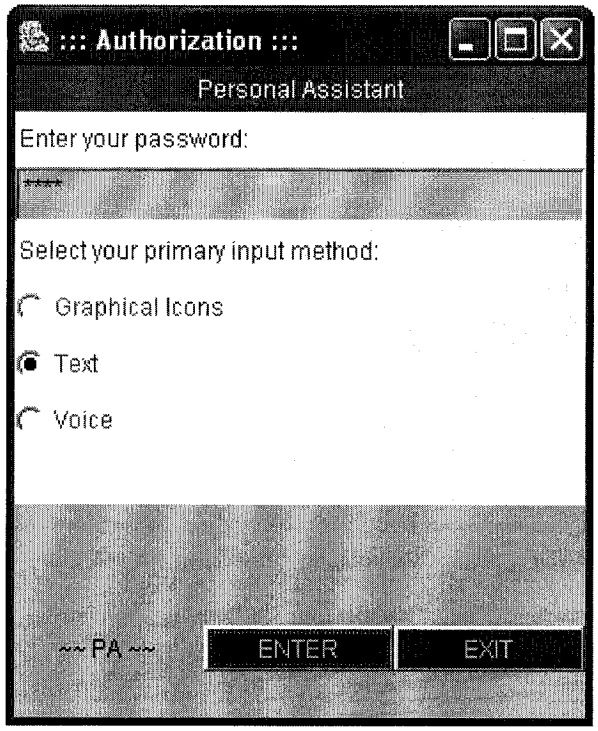

Figure 5.5 User Interface Snapshot – Authorization Screen

The application area selection screen shown next in figure 5.6 is used to specify the type of process that is currently in progress. The current process may be a temporary one-time process such as sending specific information to a known agent, or the process may include sending important files or data to external agents. In the latter case, the personal assistant needs to keep track of data, cache necessary data, and also store names and addresses of remote agents for further communication or information retrieval. The application area selection screen is also used to select between *sending ACL messages* and

retrieving cached ACL messages. The user can get information regarding each option in the screen by clicking on the *Help* button after selecting the desired option.

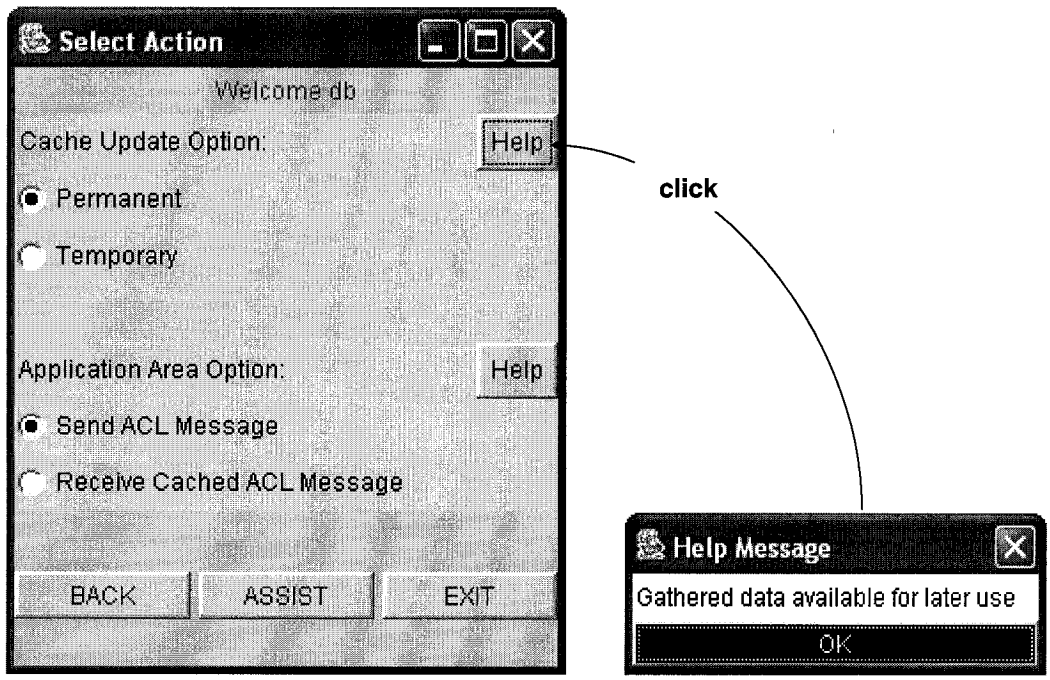

Figure 5.6 User Interface Snapshot – Application Area Selection Screen

The ACL message creation and sending screen is shown next in figure 5.7. This frame allows the user to create an ACL message by filling in mandatory and optional ACL fields. The user can select values from combo boxes for mandatory ACL fields. The mandatory ACL elements are distinguished from optional ACL elements by placing *button* components between them. The user can then send the created ACL message to the intended receiver by clicking the appropriate button.

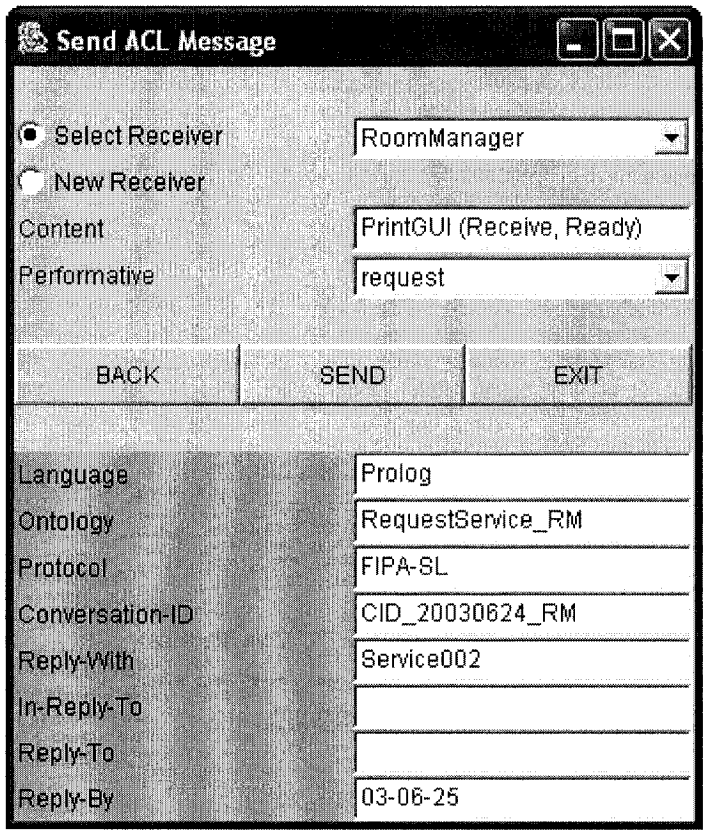

Figure 5.7 User Interface Snapshot – Create and Send ACL Message Screen

Certain received ACL messages are cached by the personal assistant for future use. The frame used to retrieve cached ACL messages is shown next in figure 5.8.

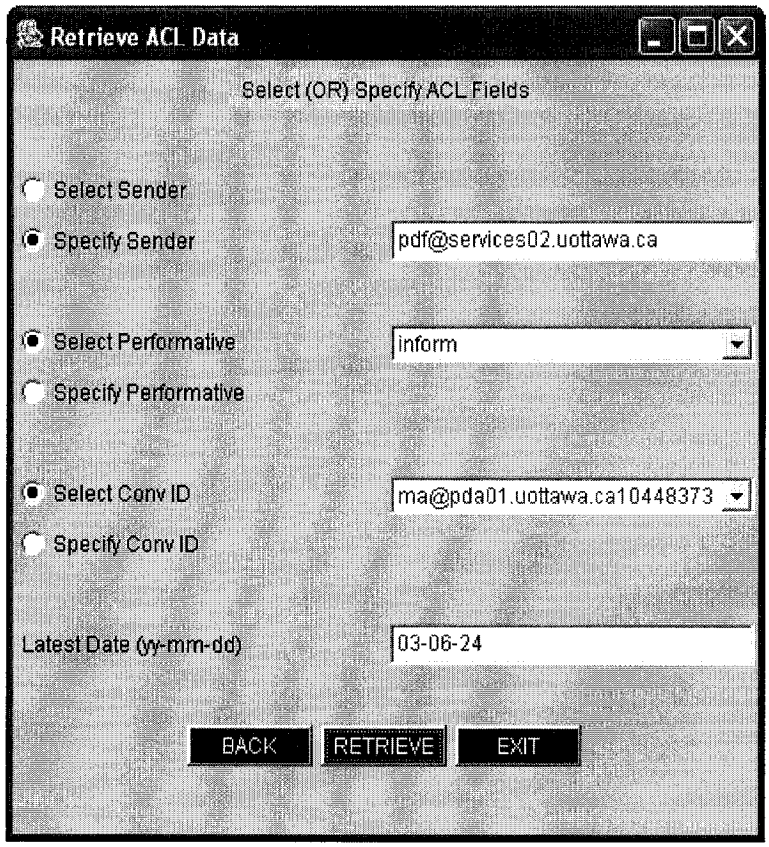

Figure 5.8 User Interface Snapshot – Retrieve ACL Message Screen

Code Snippets and Command Prompt Snapshots:

Significant command prompt snapshots and code snippets involved while implementing and running the personal assistant are as follows. All agents that run on the FIPA-OS and MicroFIPA-OS agent platforms are handled by employing similar procedures i.e., the approach to create, register, deregister, shutdown, and search an agent is similar for all agents. The agent platform diagnostics are similar for all agents. The approach followed to create an agent, register an agent, shutdown an agent, and set diagnostics are shown in figure 5.9.

```
Creation of a new agent
  – new AgentX( args[0], args[1], args[2] );
      @param1 platform_profile: The location of the platform profile
      @param2 name: The agent name
      @param3 owner: The owner of this agent

Registration with AMS and DF (platform agents)
  – registerWithAMS();
  – registerWithDF( AGENT_TYPE );

super.shutdown();
  – invoke shutdown() on the FIPAOSAgent shell preceding deregistration

DIAGNOSTICS.println(ams,this,DIAGNOSTICS.LEVEL_MAX );
```

Figure 5.9 Sample Code representing Agent Creation, Registration, and Shutdown, and Setting Diagnostics

As seen from the above figure, three arguments are mandatory for agent creation. The arguments represent the location of the platform profile, the agent name, and the agent's owner name. All agents should register with their platform-specific agents i.e., with the AMS and DF. Agent shutdown is performed after deregistration by making use of the *FIPAOSAgent* class. The debug level can also be set depending upon the user's preferences.

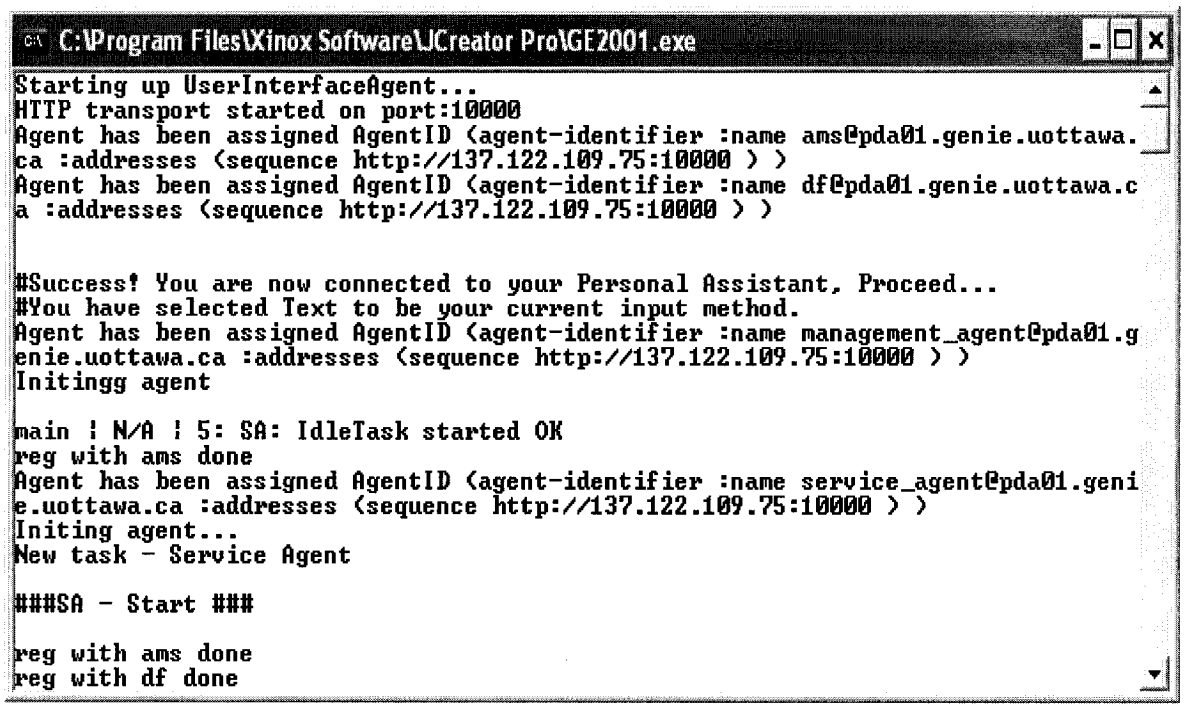

Figure 5.10 Snapshot Displaying Basic Agent Activities

The figure 5.10 shows a command prompt snapshot which represents the transport protocol initiation, transport protocol port assignment, agent creation, agent registration, and *Idle Task* creation. As soon as an agent is created, it will be assigned an Agent ID which includes the agent's name and address. The figure 5.10 first represents the transport protocol initiation and transport protocol port assignment. The figure then represents Agent ID assignments to the AMS and DF followed by Agent ID assignments to the personal assistant's components. The figure also represents the management agent and service agent registration with the AMS and DF. Finally, the *Idle Task* creation is also represented.

A MTS (Message Transport Service) monitor starts along with the MicroFIPA-OS agent platform. A MTS monitor displays all transport activities that take place in the agent platform until the agent platform stops executing. A sample MTS monitor is shown in figure 5.11.

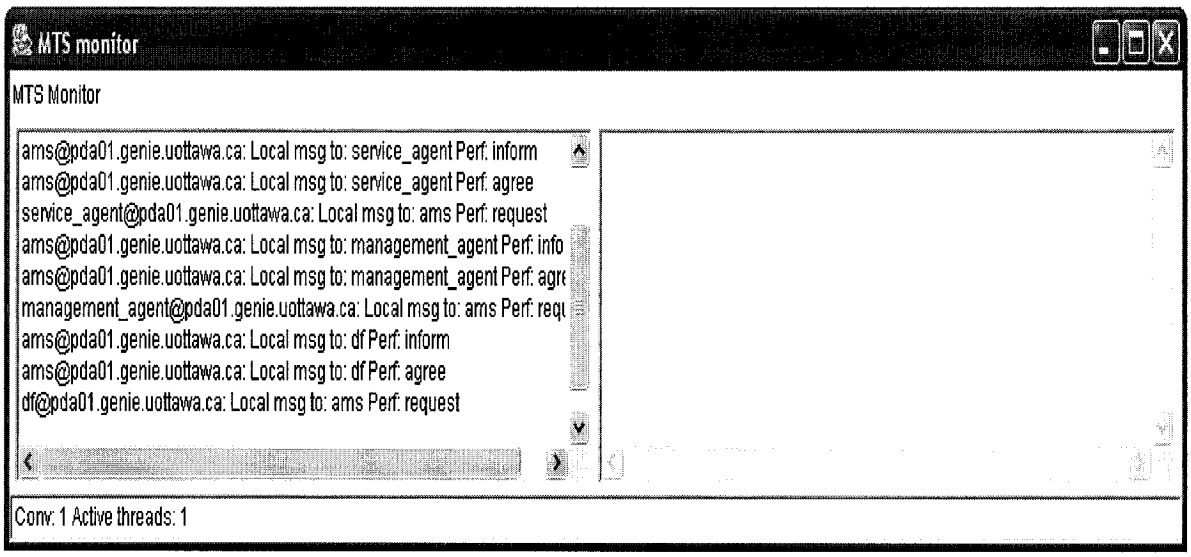

Figure 5.11 Snapshot Displaying MTS Monitor

Agents communicate with each other by using the ACL (discussed in section 2.3.4). The code snippet shown below in figure 5.12 represents a method to send an ACL message. An instance *msg* of the *ACL* class is used to set values for all ACL fields as shown in figure 5.12.

```
1640 //Method to send a new ACL message to a specific target
1641 public void sendMsg()  {
1642     try
1643     {
1644         // Create a new ACL message
1645         ACL msg = new ACL();
1646
1647         //Set values for ACL fields
1648         msg.setPerformative(perfC.getSelectedItem());
1649         msg.setSenderAID(_parent.getAID());
1650         msg.setReceiverAID( _target );
1651         msg.setContentObject("("+fillcontent.getText()+(cnt)+")");
1652         msg.setLanguage(langT.getText());
1653         msg.setOntology(ontoT.getText());
1654         msg.setProtocol(protT.getText());
1655         msg.setConversationID(convT.getText());
1656         msg.setReplyWith(rwT.getText());
1657         msg.setInReplyTo(irtT.getText());
1658         msg.setReplyToAIDs(101.get(2));
1659
1660         System.out.println( "Sending message to " + _target );
1661         this.forward( msg );
1662     }
1663     catch( Throwable e )
1664     {
1665         // Error - display the exception
1666         e.printStackTrace();
1667     }
1668 }//end sendMsg()
```

Figure 5.12 Code Snippet representing an ACL Message Sending

The code snippet shown next in figure 5.13 represents XML file parsing details. The code snippet represents a method to insert information into a new XML file and retrieve information from a cached XML file.

```
1143 XMLImpl xpi = new XMLImpl();
1144
1145 //Vectors to hold Elements and values from XMLParser
1146 Vector tempElements = new Vector();
1147 Vector tempValues = new Vector();
1148
1149 // Path for XML repository
1150 final String XML_PATH = "/\\Nadir\\users\\Public\\xml\\";
1151
1152 //... agent code comes here ... (not related to XML Parsing)
1153
1154     //Create a filename using convID to get XML profile
1155     String fileString = new String(conversationID+".xml");
1156
1157     //tempElements vector has the index of ACL fields
1158     //tempValues vector has values for ACL fields
1159     putBackXML(tempElements,tempValues);
1160
1161     //Parse the XML
1162     xpi.initParser(XML_PATH+fileString);
1163
1164     //Store the parsed values in Vectors
1165     tempElements.removeAllElements();
1166     tempValues.removeAllElements();
1167     tempElements = xpi.getElementsList();
1168     tempValues = xpi.getValuesList();
1169
1170 //... agent code comes here ... (not related to XML Parsing)
1171
1172     /*Method to create XML file for the received ACL message
1173       with it's conversation ID as the filename*/
1174     private void putBackXML(Vector elems,Vector vals){
1175     try{
1176         FileWriter pout = new FileWriter(XML_PATH+fileString);
1177         String st = xpi.makeXML(elems,vals);
1178         pout.write(st);
1179         pout.close();
1180     }catch(Exception exc){
1181         System.out.println(" Problem "+exc);
1182         exc.printStackTrace();
1183      }
1184     }
```

Figure 5.13 Code Snippet representing XML File Parsing Details

Cached XML files can be retrieved using the *xerces* java parser [48]. The above shown figure represents *xpi* as an instance of the *XMLImpl* class. The *XMLImpl* class handles methods to create XML files from vectors. The vectors which hold *elements* and *values* of an XML file are represented as *tempElements* and *tempValues*. XML files are saved in an appropriate location and are named using the *conversation ID* field in ACL messages. The method *putBackXML* is used to create XML files using the vectors *tempElements* and *tempValues*. The method *initParser* in the *XMLImpl* class is used to parse

the necessary XML file, and methods *getElementsList* and *getValuesList* in the *XMLImpl* class are used to retrieve elements and corresponding values. The retrieved elements and values are again stored in *tempElements* and *tempValues* vectors to simplify data processing.

5.2 The Mobile Agent-based Ad-hoc Communication System

This section first describes the *mobile agent-based ad-hoc communication system* in detail. Next, the integration details of the *mobile agent-based ad-hoc communication system* with the personal assistant are elaborated by discussing the integration design outline and integration scenarios. Finally, snapshots representing service GUI's are described.

The *mobile agent-based ad-hoc communication system* is currently in progress at MMAR Laboratory, University of Ottawa, Canada. The goals of the *mobile agent-based ad-hoc communication system* is to first spontaneously detect users and/or services entering a particular physical area, for e.g., a meeting room, and then spontaneously assign workstations, and finally offer services to users depending upon the user's profile. A sensor detects users entering the meeting room with appropriate tags or badges in order to load the appropriate user profile. User profiles vary depending upon the type of the user. For example, the user could be a manager, permanent employee, or guest. Then, authorized users are invited to use selected workstations and services depending upon their profile. Each user may also bring their workstations and/or services. These workstations and services are identified by sensing tags assigned to them. An IP (Internet Protocol) address is dynamically assigned to workstations when they are connected to the network. A service could be offered to a user currently present in the room depending upon the service profile

and the user profile. Available services can be utilized by users using the desktop, laptop, or pocket PC assigned to them or brought by them. An example printing service profile is shown below in figure 5.14.

```
<xml version="1.0" >
 <Document>
<Type value="Service" />
<Device>
      <Device_group value="PRINTERS" />
      <Device_name value="HP LaserJet 4050 N PS" />
 </Device>
<Properties>
      <Color value="no" />
      <sided value="yes" />
      <Memory>
            <minMemory value="8" />
            <maxMemory value="11" />
      </Memory>
      <Resolution value="1200" />
      <PS value="true" />
 </Properties>
<Owner value="CBY-B502" />
<EnteringTime value="" />
<ExitTime value="" />
</Document>
```

Figure 5.14 Printing Service Profile

User profiles, service profiles, and service GUI's are dynamically managed by the *room manager* agent. The *Session Initiation Protocol* (SIP), *service discovery*, and *room policies* help the room manager agent to perform its duties. The room manager agent

maintains a GUI to list the currently available users and services. The room manager agent dynamically moves this GUI to authorized users' machines as soon as they connect their machine to the network. This network connection could be either wired or wireless. Any new service or user available in the meeting room will be spontaneously updated in the GUI maintained by the room manager agent. The user can then select required services from the updated GUI received from the room manager agent. This GUI also lists currently available users in order to join the public conference or start a private conference. Service requests are sent to the room manager agent, which in-turn moves the requested service's GUI to the appropriate user's device. Users can then interact with the received service GUI to perform service execution. The snapshots of various service GUI's are described in section 5.2.1.3. The resource connectivity module provides abstraction of the network layer from other modules as the underlying network may be wired or wireless.

The detailed design of the *mobile agent-based ad-hoc communication system* discussed in this section is shown in figure 5.15.

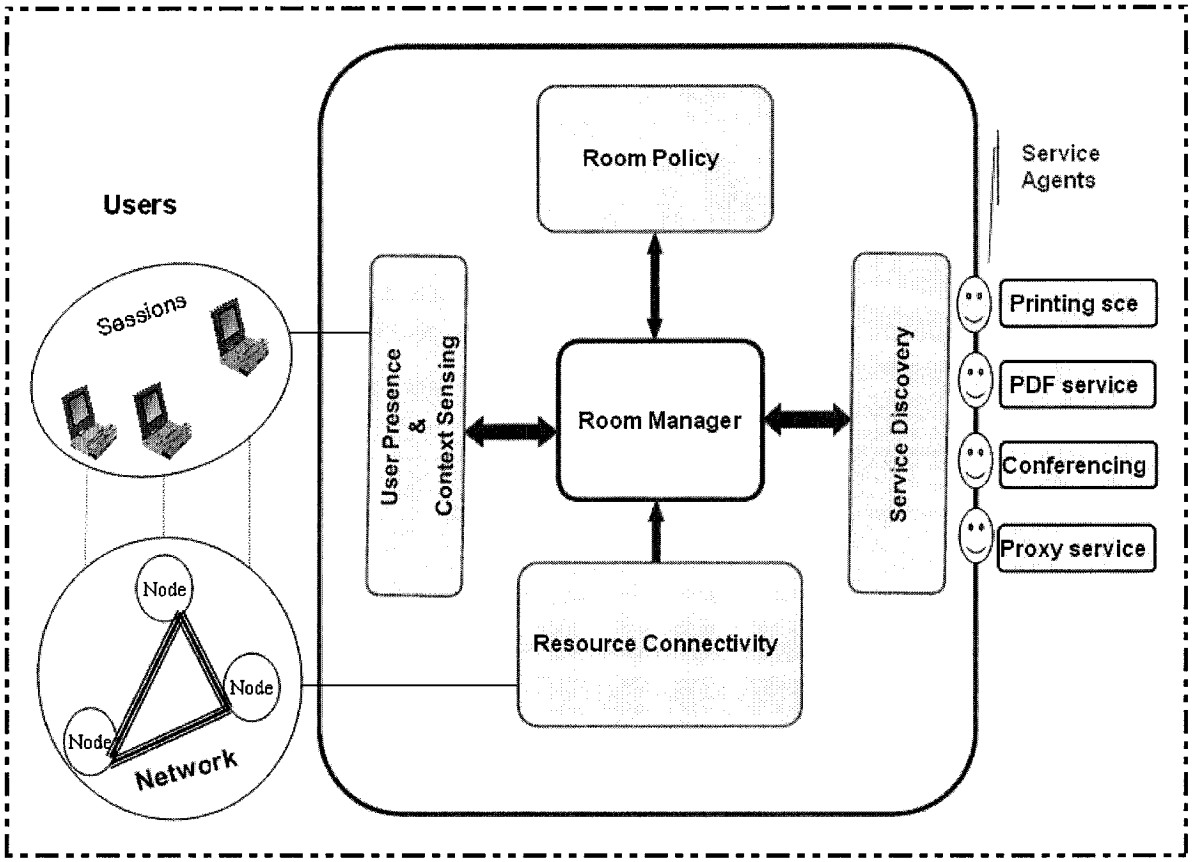

Figure 5.15 Design of the Mobile Agent-based Ad-hoc Communication System

5.2.1 Integration with the Ad-hoc Communication System

The personal assistant discussed in this thesis is integrated with the *mobile agent-based ad-hoc communication system*. Therefore, authorized users using the personal assistant and can also utilize various services offered by the *mobile agent-based ad-hoc communication system*.

The GUI listing the available services and users and the requested service's GUI are dynamically moved to the personal assistant via the proxy agent. The mobile device user can then execute the requested service using the service GUI and forward necessary information to the proxy agent. The role of the proxy agent in the integration of the personal assistant with the *mobile agent-based ad-hoc communication system* is discussed in section

5.2.1.1. The overall design outline of the personal assistant's integration with the *mobile agent-based ad-hoc communication system* is shown below in figure 5.16.

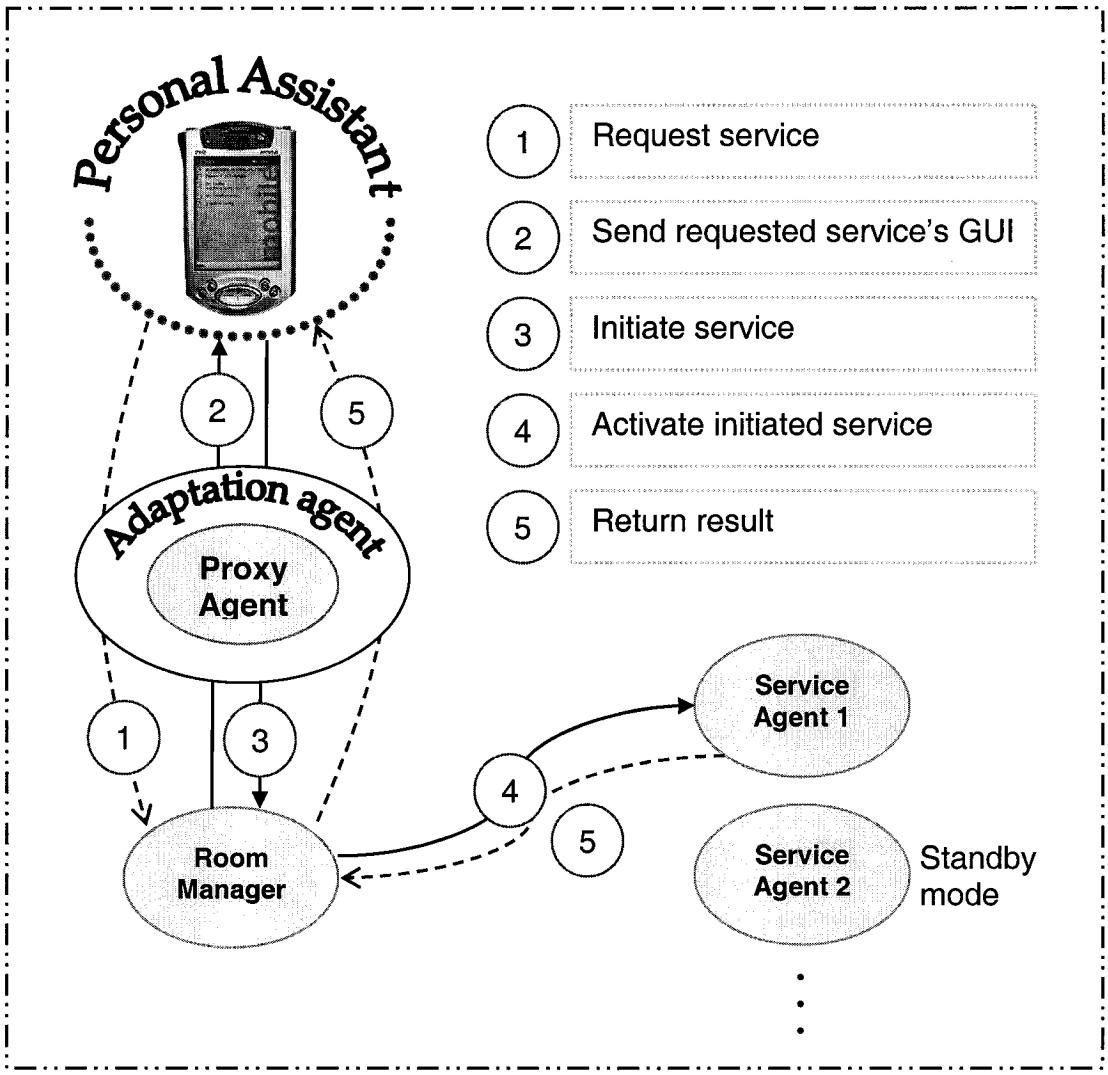

Figure 5.16 Design Outline of the Personal Assistant's Integration with the Ad-hoc Communication System

As seen from the above shown figure 5.16, the design outline doesn't include full integration details. The design outline starts from the point where the mobile device user selects the required service from the available list of services. The design outline assumes

that the mobile device user carrying pocket PC has already been sensed by sensors and has already received the GUI which lists the available services and users from the room manager agent.

The user's request for a particular service is sent to the room manager agent. This is represented by '1' in the above shown figure 5.16. The room manager agent then moves the requested service's GUI to the user i.e., to the user's personal assistant. The personal assistant's component in the workstation i.e., the proxy agent intercepts the data sent to the personal assistant. If necessary, the adaptation agent adapts the intercepted GUI to a format suitable for the pocket PC's java runtime environment. This is represented by '2' in the figure 5.16. The user could then interact with the received service's GUI and initiate the service by providing all required information for service execution. For example, in order to execute the printing service the user should select the file to be printed and should also indicate the number of copies to print. Once the user has initiated the service, the request is passed to the room manager agent via the proxy agent. This is represented by '3' in the figure 5.16. The request is sent to the room manager agent via the proxy agent due to hardware restrictions in the mobile device i.e., the pocket PC doesn't have the capability to directly move files over the network. Therefore, the required files are first moved to the proxy agent using sockets (discussed in section 3.2.1) and then forwarded to a common folder in the network.

As soon as the room manager agent receives the service request with optional file transfer confirmation, it instructs the appropriate service agent to perform the service. The room manager agent forwards all necessary information obtained from the service GUI to the service agent. This is represented by '4' in the figure 5.16. The service agent also

informs the status of the performed service to the appropriate user via the room manager agent and proxy agent. This is represented by '5' in the figure 5.16.

5.2.1.1 Role of the Proxy and Adaptation Agents

The proxy agent is designed as a part of the personal assistant as described in section 4.1.5. The proxy agent's role in the integration of the personal assistant with the *mobile agent-based ad-hoc communication system* is described above in figure 5.16. It could be seen from the figure that any communication between the personal assistant and the room manager agent takes place via the proxy agent. Any data passed between the room manager agent and the personal assistant is intercepted by the proxy agent. The intercepted data is then cached, forwarded, and adapted accordingly. The adaptation is performed by using the adaptation agent which is integrated with the proxy agent. The adaptation agent's processes were elaborated in section 4.1.6.

5.2.1.2 Integration Scenario

A scenario elaborating the integration process of the personal assistant with the *mobile agent-based ad-hoc communication system* is discussed in this section. The steps followed by the mobile device user to execute integrated services are discussed. The overall scenario is described below in figures 5.17 and 5.18. The figure 5.17 represents the mobile device user's service request and the room manager agent's reply for the received request. The figure 5.18 is the continuation of the figure 5.17 and it represents the mobile device user's interaction with the received service GUI, followed by initiation and activation of the required service.

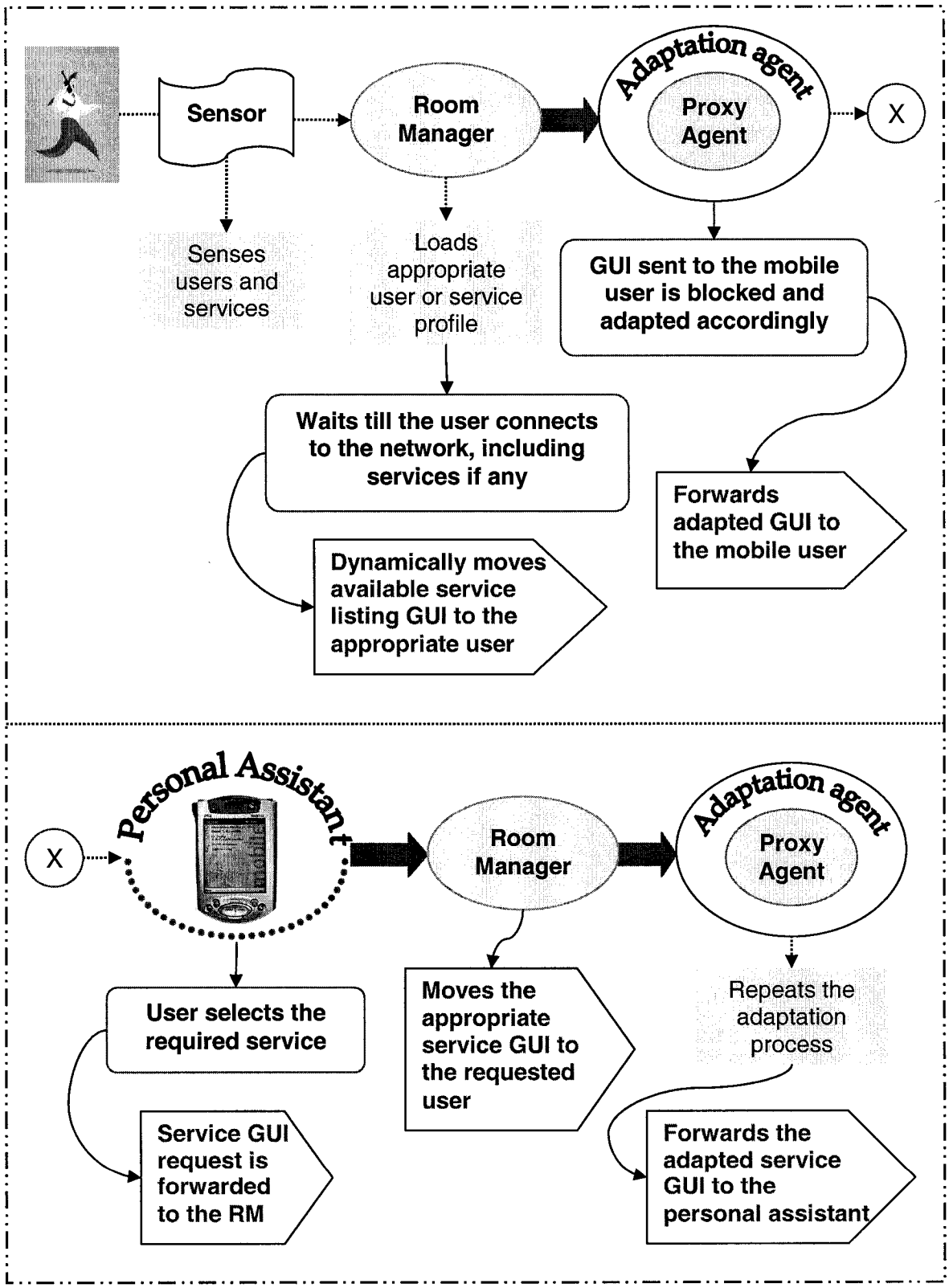

Figure 5.17 Integration Scenario1: Requesting and Receiving a Service GUI

The first integration scenario is represented as a *pseudo-code* for better understanding. The ***pseudo-code*** is as follows.

1. Perform user and service detection process using sensors
2. Load detected user's profile or service's profile using room manager agent
3. If (incoming user = guest) then

   Load guest profile

   Set guest machine

   Endif
4. Go to idle state until user's network connection is made active and appropriate agent platform is started
5. Move profile-based 'services and users' listing GUI from room manager agent to user's device (personal assistant running in pocket PC)
6. Intercept 'services and users' listing GUI using proxy agent
7. Perform GUI adaptation using proxy and adaptation agents; 'forward' adapted GUI to personal assistant in pocket PC
8. Select service required by user
9. Send user's service request to room manager agent via proxy agent
10. Move requested service's GUI from room manager agent to appropriate user
11. Intercept service GUI and 'perform' GUI adaptation using proxy and adaptation agents; 'forward' adapted GUI to personal assistant in pocket PC

The mobile device user then interacts with the received service GUI as shown below in figure 5.18.

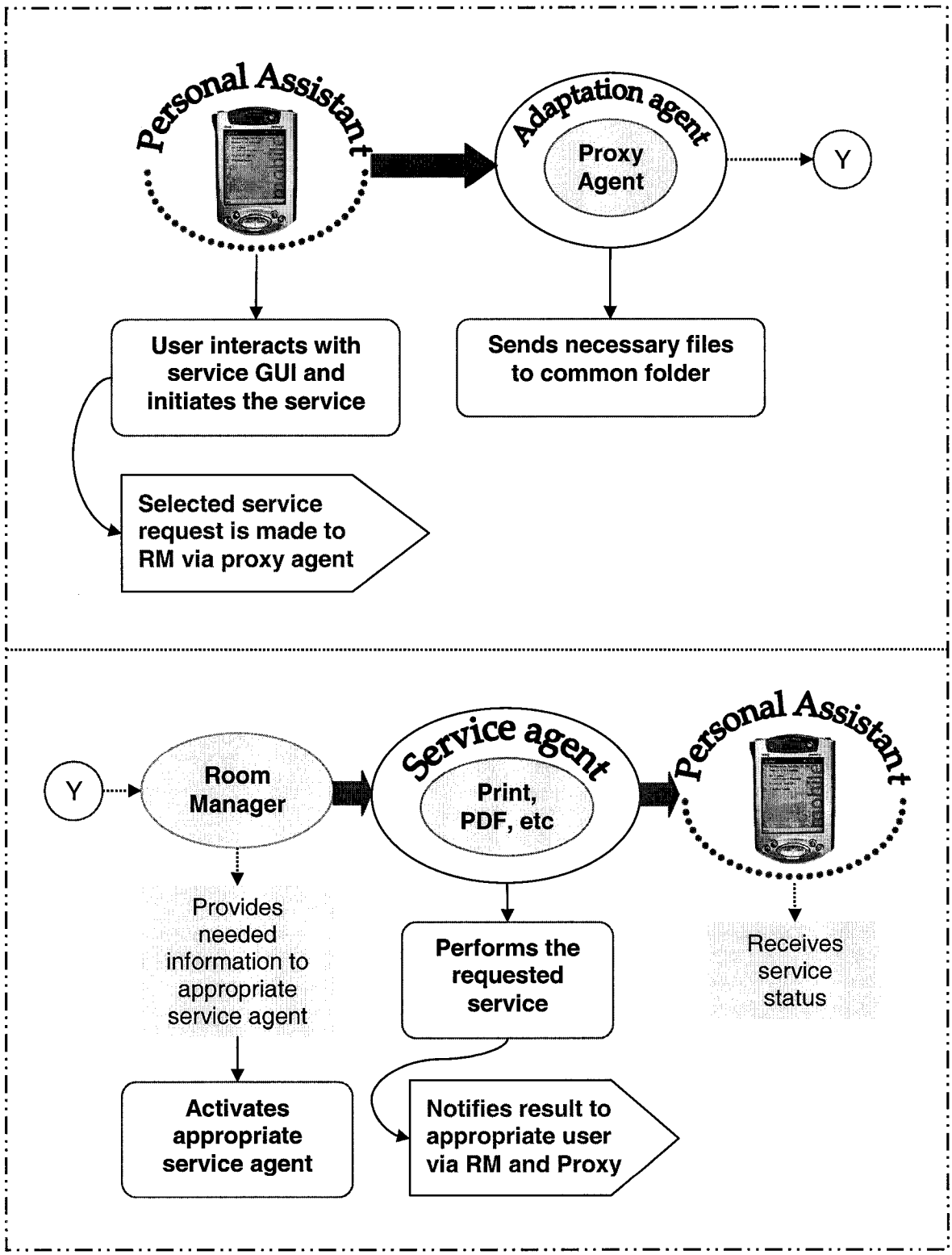

Figure 5.18 Integration Scenario2: Initiating and Activating the Required Service

The second integration scenario is represented as a *pseudo-code* for better understanding. The ***pseudo-code*** is as follows.

1. Initiate required service through received service GUI
2. Append and 'send' necessary information along with service request to room manager agent
3. Intercept service request and transferred information using proxy agent
4. Perform required adaptation and forwarding using proxy and adaptation agents
5. Send authorization and required information to appropriate service agent; 'Activate' service agent through room manager agent
6. Perform appropriate service through service agent
7. Inform service status to appropriate user

5.2.1.3 Integration Result Snapshots

Figures 5.19 and 5.20 shown below represent a snapshot of the GUI which lists currently available services and users. In the snapshot displayed in figure 5.19, the user selects the *HP LaserJet Printer* service from available list of services in order to use the printing service.

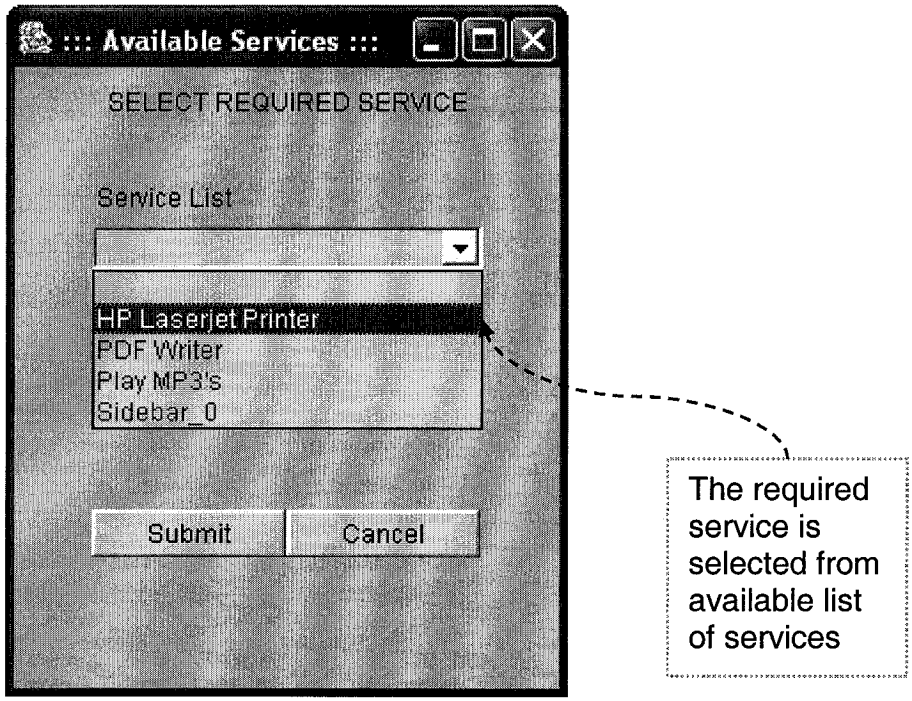

Figure 5.19 Snapshot Displaying Available Services

In the snapshot displayed in figure 5.20, the user selects another user *Tom* for conferencing. The service *Sidebar_0* shown in figure 5.19 can be selected to perform private conferencing with *Tom*.

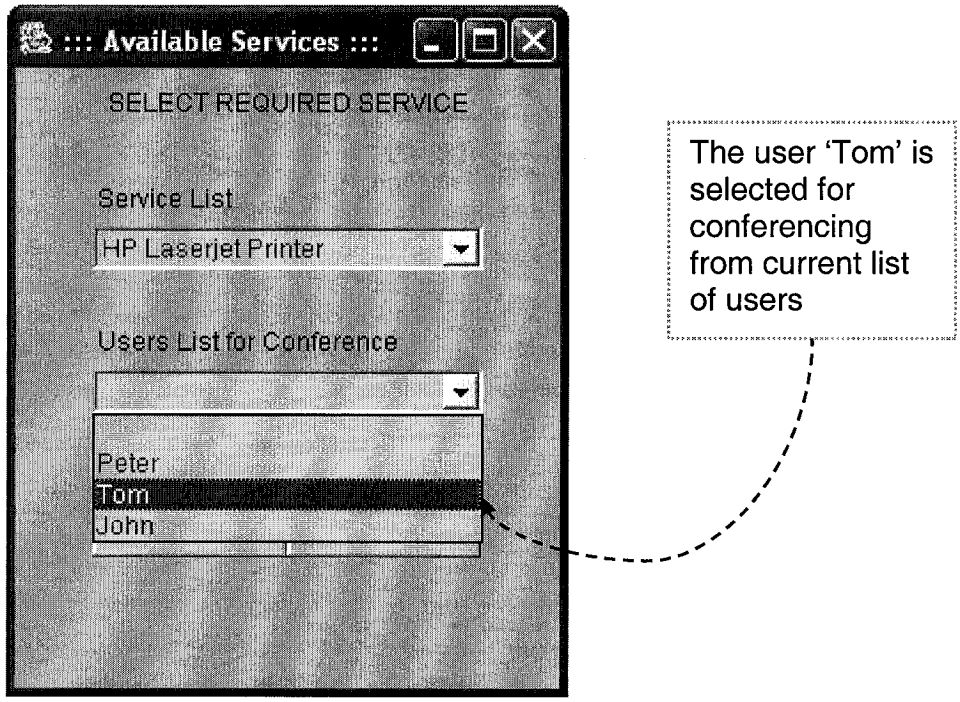

Figure 5.20 Snapshot Displaying Available Users

The snapshot displayed in figure 5.21 represents the printing service's GUI. The user can select the file to be printed by clicking the *browse* button and can also specify the number of copies to be printed.

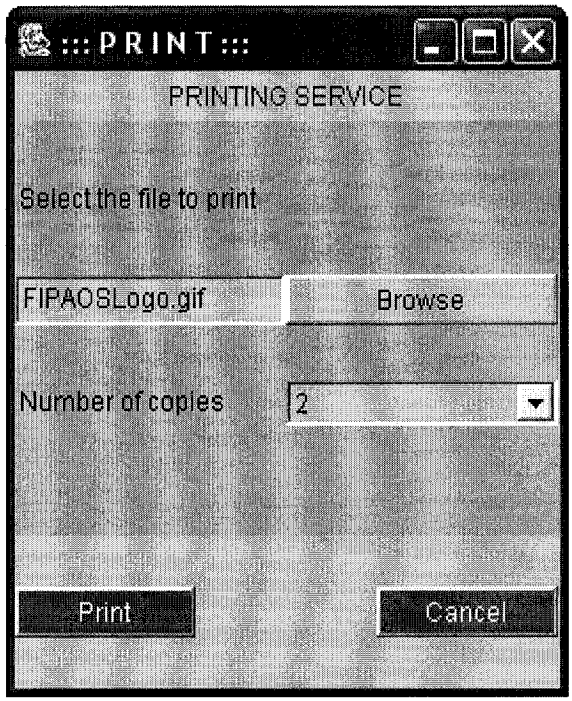

Figure 5.21 Snapshot Displaying the Printing Service

The snapshot displayed in figure 5.22 represents the PDF writing service's GUI. Similar to the printing service, the PDF writing service also involves file transfer. The user can select the file to be converted by clicking the *browse* button and can click the *Get PDF* button to start the PDF conversion. The current version of the PDF writing service supports DOC to PDF, PPT to PDF, and TXT to PDF conversion. Converted PDF files are saved at the location of the source file. Stored PDF files can be viewed using the *Adobe Acrobat Reader for Pocket PC.*

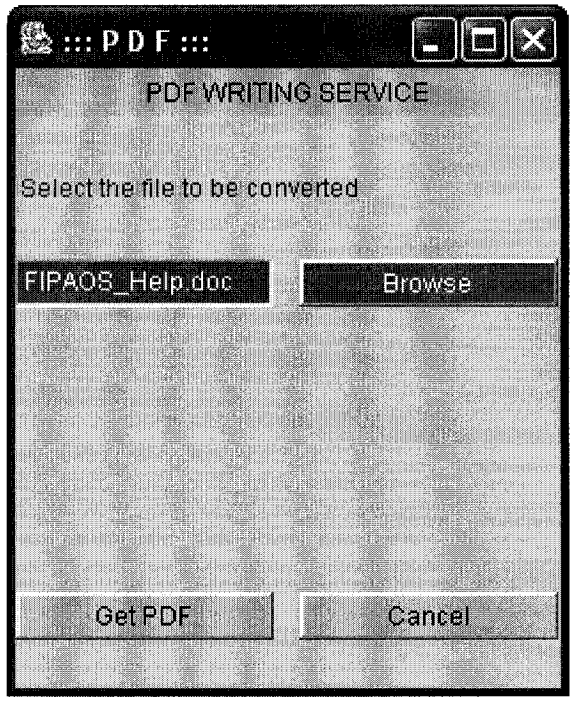

Figure 5.22 Snapshot Displaying the PDF Writing Service

The figure 5.23 shown below represents a snapshot of the MP3 service's GUI. The java-based mp3 player has full playback controls as in any media player. The user can play mp3 files from the local network or local memory. In the case of selecting mp3 files from the local network, the selected mp3 files would be buffered directly from the local network to save mobile devices' memory space.

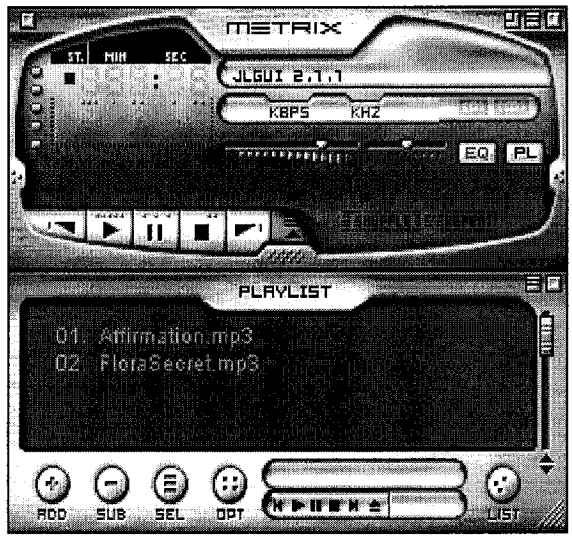

Figure 5.23 Snapshot Displaying the MP3 Service

5.3 System Flexibility and Performance Evaluation

The personal assistant's flexibility feature is discussed herein as follows.

- The personal assistant can successfully execute in any Internet enabled mobile device or workstation that supports the PersonalJava specification and virtual machine and has the lightweight MicroFIPA-OS agent platform installed on it [29]. The lightweight agent platform's components can be removed or added depending upon the memory capacity of the device.
- The personal assistant is tested in different pocket PC's (E.g., Compaq iPAQ 3600 series and 3800 series) operating on *Microsoft Windows Pocket PC 2002*, *Microsoft Windows CE*, and *Linux*. The personal assistant is also tested on different desktop PC's operating on *Microsoft Windows XP* and *Microsoft Windows 2000*.

- The personal assistant can also enable audio conference while operating on *Linux* operation system [50] by using the *Robust Audio Tool* (RAT) developed by UCL Network and Multimedia Research Group, London, UK [51].
- The personal assistant can enable a mobile device to support unrecognized file formats without additional software requirements by utilizing the proxy and adaptation agents. For example, the adaptation agent converts a PDF file's content into a string of bytes, wraps the string in the ontology or content ACL field, and finally forwards the ACL message to the personal assistant's components in the mobile device.
- The personal assistant enables file transfers between agents by using either sockets or by converting the file into a string of bytes. The method employed for file transfers depends upon network restrictions of the mobile device.

Performance Evaluation:

A quantitative evaluation of the personal assistant model is carried out in order to improve the personal assistant's performance. The performed evaluations helped to analyse the following.

- Does the format and size of an ACL message's content field widely affect the ACL message's transfer time?
- Does the ACL message filtering widely affect the ACL message's transfer time?
- Does the proxy agent's performance degrade if it acts as a gateway of communication for more than one personal assistant?

- The effects of using a wireless connection for communication between the personal assistant's components.
- Does the services offered by the personal assistant is efficient than commercial services available in mobile devices?

The evaluations are performed on a *Compaq iPAQ 3850* pocket PC operating on *Microsoft Pocket PC 2002*. The pocket PC is powered with 206 MHz Intel StrongARM processor, 64 MB memory, 32 MB flash ROM, and expansion slots for CF (Compact Flash) and SD (Secure Digital) cards [52]. The desktop PC which executes the proxy and adaptation agents operates on *Microsoft Windows 2000* and features 2.4 GHz Pentium4 processor, 512 MB RAM, and 10/100 Ethernet LAN card. The wireless connection between the pocket PC and desktop PC is established using an 802.11 wireless LAN CF card. The personal assistant is also evaluated through a wired connection between the pocket PC and desktop PC. This wired connection is made via USB ports.

The evaluations are performed by employing scenarios similar to those described in section 4.3.

- The pocket PC executing most of the personal assistant's components took approximately 4 seconds to load and execute agents.
- An ACL message transfer from the pocket PC to the desktop PC and vice-versa through a wireless connection took approximately 7.5 seconds.
- The same ACL message transfer through a wired connection took approximately 6 seconds.

- The same ACL message transfer after filtering by the adaptation agent took approximately 6.5 seconds through a wireless connection, while it took approximately 5.5 seconds through a wired connection.
- Direct file transfers between the personal assistant's components in the mobile device and workstation are performed using sockets. The transfer of a 50 KB file took approximately 6 seconds through a wireless connection, while it took approximately 3.5 seconds through a wired connection.
- The proxy agent took approximately 4 seconds to process, adapt, and forward the received data to the personal assistant's components in the mobile device.
- When the proxy agent acts as a gateway of communication for two personal assistants, the proxy agent took approximately 4.5 seconds to process, adapt, and forward the received data to appropriate personal assistants.

The above stated results are summarized and the distribution of time taken for the personal assistant to communicate with an external agent is illustrated below in figure 5.24.

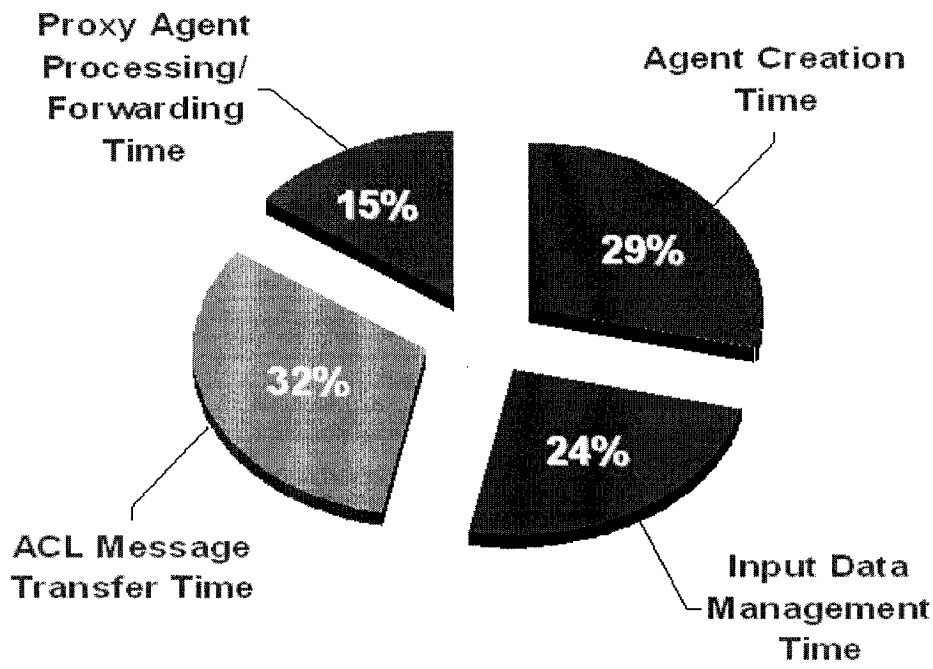

Figure 5.24 Distribution of Time Taken for a Personal Assistant Communication

As seen from figure 5.24, 29% of the overall time is consumed for the personal assistant initialization, creation, and registration with platform-specific agents. After the user inputs the necessary information, 24% of the time is consumed for converting, managing, and storing the inputted information. The processed data is then sent to the proxy agent as an ACL message through a wireless link, which consumes 32% of the overall time. Finally, 15% of the time is consumed for further processing and forwarding by the proxy agent in the workstation. The results represent that most of the time consumed is in the mobile device side, where the necessary agents are created and necessary data is processed. The proxy agent in the workstation performs a similar job for very less amount of time. The excess time consumption in the mobile device side is mainly because of limited resources and nascent execution environments in current mobile wireless devices.

The personal assistant is capable of receiving e-mails on behalf of the user. This service is compared with a commercial e-mail service that is already available in the pocket PC. The personal assistant retrieved the user's new e-mails (approximately 100KB) and displayed the same to the user in approximately 3 seconds. The same e-mail retrieval process using the e-mail client pre-installed in the pocket PC took approximately 6.5 seconds. The time consumed while using the personal assistant is comparatively less, and this is mainly because of the use of the proxy and adaptation agents separately in the workstation. The proxy agent periodically checks for new e-mails and if any is received, the adaptation agent converts the same into a string a bytes so as to wrap the string in the content ACL field. This ACL message is in advance received by the personal agent whenever a wireless interface is established between the two devices. The e-mail could then be viewed by the user as and when necessary. Here, only the proxy agent periodically connects to the e-mail server, thereby reducing the amount of processing performed in the mobile device side. The significant advantage of using the personal assistant model is that no other third-party applications are required in the mobile device in order for the user to perform services.

From the performed evaluations the following conclusions are made.

- The time consumption to transfer an ACL message is directly proportional to the size of the ACL message's fields.
- The ACL message content filtering performed by the adaptation agent has more effect when the ACL message transfer is performed through a wireless connection instead of a wired connection.

- The communication between the personal assistant's components is efficient when a wired connection is used instead of a wireless connection.
- The proxy agent's performance degrade level is negligible even if it acts as a gateway of communication for more than one personal assistant.
- The performance of the personal assistant model can further be increased by performing most of the agent processing in the proxy agent side.
- The personal assistant's e-mail service has less time responsiveness and software requirements when compared with a commercial e-mail service.

Certain technical challenges faced while implementing the personal assistant model for mobile devices were discussed in section 3.5. It is believed that the performance of the current version of the personal assistant could further be improved by making use of FIPA's new mobility concepts, meliorating the lightweight agent platform to be as efficient as the FIPA-OS agent platform, and enabling voice recognition in the personal assistant.

5.4 Summary

This chapter first discussed the implementation details of the personal assistant model in detail including class diagrams, code snippets, and user interface snapshots. Next, the *mobile agent-based ad-hoc communication system* is described including the ad-hoc system's integration details with the personal assistant model. Various snapshots representing integrated services were also illustrated in this chapter. Snapshots displayed in this chapter are captured from a desktop PC running Windows XP operating system instead of a mobile device. This is done to achieve better clarity and visibility in the displayed

snapshots. The chapter finally evaluates the personal assistant's flexibility and performance levels including comparison with commercial e-mail service. The next chapter concludes the thesis and discusses future research directions.

CHAPTER 6

Conclusion

The personal assistant thus makes an effort to simplify the tasks performed by mobile device users. This chapter first summarizes the thesis and then discusses future research directions.

6.1 Summary

The Internet and mobile devices' usage are growing rapidly. Much research is being conducted to realize the presence of mobile devices in every day human life activities. The design and implementation of a personal assistant model for mobile device users is an effort taken to develop a new application for mobile devices. Though the Internet is accessible in mobile wireless devices, Internet applications for mobile devices are complex to use. The agent-based personal assistant framework for mobile devices proposed in this thesis allows mobile device users to perform tasks in mobile devices with least effort and intervention. The personal assistant semi-autonomously acts on behalf of a particular mobile device user

and provides assistance to him/her while performing tasks such as e-mail retrieval and file/ media transfer.

The agent-based personal assistant for mobile wireless devices has several advantages when compared with other personal assistants stated in section 2.7. The agent-based personal assistant model has less software and hardware requirements, less bandwidth requirements, reduces network latency, and is preferred in devices which severely lack in resources. Also, the personal assistant's GUI represents information in an appealing way to the mobile device user and is interactive.

The personal assistant is broken down into different agent components. The user, management, and personal agents reside in a mobile wireless device and are implemented on the MicroFIPA-OS agent platform and *Java programming language*. These agents are mainly used to process user input, manage data, and store data by applying user-specific policies. Information required for future use is stored in XML files. The proxy and adaptation agents reside in a workstation and are implemented on the FIPA-OS agent platform and *Java programming language*. The proxy agent acts as a gateway of communication for the personal assistant to enable direct communication between the personal agent and external agents. It also provides proxy services to the personal assistant's components in the mobile device. By doing so, the proxy agent approach requires limited bandwidth for agent communication (compared to other approaches stated in section 2.7), overcomes constraints faced by agent components in mobile devices, and resolves problems in mobile devices such as unstable execution environments, overloading, and insufficient resources. The adaptation agent adapts the received data to formats supported by mobile devices. Therefore, the personal assistant enables mobile devices to support unrecognized

file formats without additional software requirements. The adaptation agent thereby reduces compatibility problems in mobile devices. Control agents reside in both devices in order to establish wireless communications between the personal assistant's components.

The agent-based personal assistant communicates with external agents using the ACL in order to perform certain tasks. This personal assistant model can be integrated with other agent-based systems, thereby utilizing services offered by external systems as well. The personal assistant is integrated with the *mobile agent-based ad-hoc communication system* as a part of this thesis and imports services including printing service, PDF writing service, and MP3 service.

6.2 Future Research Directions

The personal assistant discussed in this thesis paves way for numerous research directions in agent technology and high-tech multimedia services for mobile devices, which would help to design a better performing personal assistant with many more capabilities.

Currently, the main challenge faced while designing an application for mobile devices is to deal with mobile devices' compatibility and capability restrictions. This thesis uses the adaptation agent to resolve compatibility restrictions in mobile devices. The proxy agent is used to resolve resource limitation problems in mobile devices. The restrictions in mobile devices could further be effectively handled and resolved by improving the performance of the adaptation and proxy agents. Designing an adaptation agent which adapts various data formats to appropriate formats suitable for mobile devices is a real challenge. One way to improve the performance level of the personal assistant's components is by fully utilizing agent technology i.e., by complying with significant FIPA specifications. For example, the

agent-based personal assistant can be made mobile by adopting FIPA's mobility concepts [4, 39]. Several significant issues regarding mobile agents are open to future research. This includes security, reliability, failure recovery, and mobile agent management [32].

The concept of virtual assistant can be popular and widely used if user-interfaces are more appealing and easy to use compared to current personal assistants. The personal assistant's GUI can further be improved and modernized by utilizing state-of-the-art technologies for designing user interfaces. One example is to enable *voice recognition* and *text-to-speech* technologies in user interfaces. The *Video-conferencing* service could also be enabled when the personal assistant is integrated with the *mobile agent-based ad-hoc communication system.* There is a need for further research to employ video-conferencing for mobile device users as there are not many tools and devices that are currently in existence. Some currently available tools and devices to perform video-conferencing in mobile devices are as follows. The *Videoconferencing Tool* (VIC) [53] developed by UCL Network and Multimedia Research Group, London, UK is a fully functioning tool to perform video-conferencing. The Winnov Videum PCMCIA compact video camera [54] is designed specifically for Compaq iPAQ pocket PC's [52] to perform video-conferencing.

An execution environment is required to execute any agent or program code. There are not many efficient standards, execution environments, and operating platforms currently in existence for mobile devices. This decreases the performance level of mobile device applications. Hence, there is a need for advanced research to develop execution environments for mobile wireless devices. Currently available execution environments for mobile devices should further be improved to support new protocols and concepts without consuming considerable memory.

References

[1] *The World Internet Project Surveys*, Review and Report, available at http://media.asaka.toyo.ac.jp/wip/Chapt3.pdf

[2] *Men Say It's Strictly Business, Women Say They Can't Live Without It*, Siemens Survey, Siemens AG, Nov 2002, available at http://www.my-siemens.com/MySiemens/CDA/Index/0,2730,US_en_1_news%253A_0_8434_0_0,FF.html

[3] K. Englmeier and J. Mothe, *Trustworthy personal assistance: a design objective for interactive agents*, tech. meta-track, 7th Americas Conference on Information Systems, Association for Information Systems (CD-Rom), Boston, MA, USA, Aug 2001.

[4] *FIPA* (Foundation for Intelligent Physical Agents) Home Page, available at http://www.fipa.org

[5] P. Buckle and R. Hadingham, "FIPA and the Internet Revolution", *2nd International ACTS Workshop*, Singapore, Sep 1999.

[6] S. J. Poslad, S. J. Buckle, and R. Hadingham, "The FIPA-OS agent platform: Open Source for Open Standards", *Proc. of the Fifth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* (PAAM 2000), Manchester, UK, Apr 2000, pp. 355-368.

[7] *FIPA-OS and MicroFIPA-OS agent platforms*, Emorphia Research, Emorphia Ltd., available at http://fipa-os.sourceforge.net 'and' http://www.cs.helsinki.fi/group/crumpet/mfos

[8] *Java 2 Platform Standard Edition*, v. 1.3.1, Sun Microsystems, available at http://java.sun.com/j2se/1.3

[9] T. Koda and P. Maes, "Agents with Faces: The Effects of Personification of Agents", *Proc. of Human-Computer Interaction* (HCI 96), London, UK, Aug 1996, pp. 239-245.

[10] H. Lieberman, Neil W. Van Dyke, and Adriana S. Vivacqua, "Let's Browse: A Collaborative Web Browsing Agent", *Proceedings of the 1999 International Conference on Intelligent User Interfaces* (IUI '99), Los Angeles, CA, USA, Jan 1999, pp. 65-68.

[11] *AgentLand – Get smart, get an Agent*, Review and Report, available at, http:// www.agentland.com

[12] K. Raatikainen, et al., "Monads – Adaptation Agents for Nomadic Users", World Telecom '99, available at http://www.cs.helsinki.fi/research/monads

[13] Y. Lashkari, M. Metral, and P. Maes, "Collaborative Interface Agents", *Proc. of the Twelfth National Conf. on Artificial Intelligence* (AAAI-94), AAAI Press, Seattle, WA, USA, Aug 1994, pp. 444-450.

[14] Hyacinth S. Nwana, "Software Agents: An Overview", *Knowledge Engineering Review*, Cambridge University Press, vol. 11, no. 3, Sep 1996, pp.1-40.

[15] L. Kagal, T. Finin, and A. Joshi, "A Policy Language for A Pervasive Computing Environment", *IEEE 4th International Workshop on Policies for Distributed Systems and Networks* (POLICY 2003), Lake Como, Italy, Jun 2003, pp. 63-76.

[16] S. Poslad and M. Calisti, "Towards improved trust and security in FIPA agent platforms", *Proc. of Autonomous Agents 2000 Workshop on Deception, Fraud and Trust in Agent Societies* (AGENTS 2000), Barcelona, Spain, Jun 2000, pp. 87-90.

[17] T. Käpylä, I. Niemi, and A. Lehtola, "Towards an Accessible Web by Applying PUSH Technology", *4th ERCIM Workshop on User Interfaces for All*, Stockholm, Sweden, Oct 1998, pp. 133-147.

[18] M. Millier, "Software Agents", *Conference on Human Factors in Computing Systems* (CHI 97 Electronic Publications: Tutorials), available at http://www.acm.org/sigs/sigchi/chi97/proceedings/tutorial/mm.htm

[19] P. Maes, "Agents that Reduce Work and Information Overload", *Communications of the ACM* (CACM), vol. 37, issue 7, Jul 1994, pp. 31-40.

[20] *Software Agents, Intelligent Agents, and Bots*, Review and Report, available at http://www.istis.unomaha.edu/itc/meetings/itc-agents.pdf

[21] *First International Conference on Autonomous Agents*, Introduction, Marina del Rey, California, USA, Feb 1997.

[22] Knoblock and C.A., Ambite, J.L., *Agents for Information Gathering*, In J. Bradshaw, editor, Software Agents. AAAI/MIT Press, Menlo Park, CA, USA, 1997, pp. 347-374.

[23] *CRUMPET* (Creation of User-friendly Mobile services Personalised for Tourism), Information Society Technologies, London, UK, available at http://www.ist-crumpet.org

[24] *FIPA-OS*, Nortel Networks Announcements, Nortel Networks, available at http://www.nortelnetworks.com/products/announcements/fipa

[25] *FIPA-OS*, Emorphia Research, Emorphia Ltd., available at http://www.emorphia.com/research/about.htm

[26] *FIPA-OS user guide,* version 2.2.0, Emorphia Research, Emorphia Ltd., available at http://sourceforge.net/project/showfiles.php?group_id=3819

[27] S. Poslad, et al., "CRUMPET: Creation of User-friendly Mobile services Personalised for Tourism", *Proceedings of Second International Conference on 3G Mobile Communication Technologies*, London, UK, Mar 2001, pp. 28-32.

[28] *PersonalJava Application Environment*, v. 1.1.2, Sun Microsystems, available at http://java.sun.com/products/personaljava

[29] *MicroFIPA-OS user guide*, Creation of User-friendly Mobile services Personalised for Tourism (CRUMPET), Information Society Technologies (IST), Dept. of Computer Science, University of Helsinki, available at http://sourceforge.net/project/showfiles.php?group_id=3819

[30] *FIPA ACL Message Structure Specification*, FIPA specification no. 61, FIPA, available at http://fipa.org/specs/fipa00061

[31] *FIPA Agent Message Transport Service Specification*, FIPA specification no. 67, FIPA, available at http://fipa.org/specs/fipa00067

[32] D'Agents tutorials on mobile agents, available at http://agent.cs.dartmouth.edu/tutorials

[33] O. Shehory, et.al., "Agent Cloning: An Approach to Agent Mobility and Resource Allocation", *IEEE Communications*, vol. 36, no. 7, Jul 1998, pp. 58-67.

[34] I. Chatzigiannakis, S. E. Nikoletseas, and P. Spirakis, "An Efficient Communication Strategy for Ad-hoc Mobile Networks", *In Proc. of 15th Symposium on Distributed Computing* (DISC 2001), Informatics Dept., Faculty of Sciences, University of Lisbon, Portugal, Oct 2001, pp. 285-299.

[35] B. Hughes and V. Cahill, *Towards Real-time Event-based Communication in Mobile Ad Hoc Wireless Networks*, tech. report TCD-CS-2003-25, Distributed Systems Group, Dept. of Computer Science, Trinity College, Dublin, Ireland, Jun 2003.

[36] J. Jing, A. Helal, and A. Elmagarmid, "Client Server Computing in Mobile Environments", *ACM Computing Surveys*, vol. 31, no. 2, Jun 1999, pp. 117 – 157.

[37] P. Klark and U. Manber, "Developing a Personal Internet Assistant," *Proceedings of ED-Media 95 - World Conf. on Multimedia and Hypermedia*, Graz, Austria, Jun 1995, pp. 372-377

[38] A. Moukas, "Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem", *Applied Artificial Intelligence: An International Journal*, vol. 11, no. 5, 1997, pp. 437-457.

[39] *FIPA Agent Management Support for Mobility Specification*, FIPA specification no. 87, FIPA, available at http://fipa.org/specs/fipa00087

[40] *FIPA ACL Message Representation in Bit-Efficient Specification*, FIPA specification no. 69, FIPA, available at http://www.fipa.org/specs/fipa00069

[41] T. Finin et al., "On Agent Domains, Agent Names and Proxy Agents", *Proc. of the ACM CIKM Intelligent Information Agents Workshop* (CIKM-95), ACM Press, Baltimore, MD, USA, Dec 1995.

[42] Chun-Nan Hsu, et al., "Reconfigurable Web Wrapper Agents for Web Information Integration", *Proceedings of IJCAI-03 Workshop on Information Integration on the Web* (IIWeb-03), Acapulco, Mexico, Aug 2003, pp. 15-20.

[43] P. Maes, "Modeling Adaptive Autonomous Agents", *Artificial Life: An Overview*, vol. 1, no. 1 & 2, MIT Press, 1994, pp. 135-162.

[44] K. Decker, K. Sycara, and M. Williamson, "Intelligent Adaptive Information Agents", *Proc. of the AAAI-96 Workshop on Intelligent Adaptive Agents* (IAA-96), AAAI Press, Portland, Oregon, USA, Aug 1996, pp. 239-260.

[45] F.M.T. Brazier and N.J.E. Wijngaards, "Automated servicing of agents", *In Proc. of the AISB-01 Symposium on Adaptive Agents and Multi-Agent Systems*, Mar 2001, pp. 54–64.

[46] B. Sheth, *A Learning Approach to Personalized Information Filtering*, master's thesis, Dept. of Electrical Eng. and Computer Science, MIT, Cambridge, MA, USA, 1994.

[47] *Jeode Runtime Environment*, Insignia, available at http://www.insignia.com

[48] *Xerces Java Parser*, v. 1.2.1, The Apache XML Project, available at http://xml.apache.org/xerces-j

[49] H. Helin, H. Laamanen, and K. Raatikainen, "Mobile Agent Communication in Wireless Networks", *European Wireless'99/ITG'99*, Oct 1999, pp. 211-216.

[50] *Linux for Compaq iPAQ*, Review and Report, available at http://www.ipaqlinux.com and http://mstempin.free.fr/linux-ipaq

[51] *Robust Audio Tool* (RAT), UCL Network and Multimedia Research Group, London, UK, available at http://www-mice.cs.ucl.ac.uk/multimedia/software/rat

[52] *Compaq iPAQ Pocket PC*, Description and Specification – Model 3800 Series, Compaq, available at http://www.compaq.ca

[53] *Videoconferencing Tool* (VIC), UCL Network and Multimedia Research Group, University College London, London, UK, available at http://www-mice.cs.ucl.ac.uk/multimedia/software/vic

[54] *Winnov Videum PCMCIA Compact Video Camera*, Description and Specification, Winnov, available at http://www.winnov.com