INFORMATION TO USERS

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

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

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

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

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

Université d'Ottawa · University of Ottawa

Device Selection By Personal Proxy Agent In a Wireless Environment

Nabila Hadibi

An engineering report submitted to the School of Information Technology and Engineering at the University of Ottawa in Partial fulfillment of the requirements for the degree of Master in Engineering

Ottawa, Ontario (Canada) 2002

Université d'Ottawa
University of Ottawa

I hereby declare that I am the soul author of this engineering report. I authorize the University of Ottawa to lend this report to other institutions or individuals for the purpose of scholarly research.

Nabila Hadibi

I further authorize the University of Ottawa to reproduce this report by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Nabila Hadibi

Abstract:

A multitude of machines surrounds people at home, at the office or in public places. These machines range from small to big devices, small microprocessors to powerful ones. Each time the user changes location, he faces a new set of devices so the need to make those devices collaborate and interact with each other fascinates many researchers. We know this as a pervasive computing environment.

While many projects focus on the device and session mobility for telecommunication applications, few if any, take into account the multimedia Quality of Service (QoS) parameters in satisfying user requirements. This project proposes a new architecture for mobile communication services, which includes QoS management. Our architecture intends to achieve three main goals:

- The user is always connected, so he/she is always reachable;
- It allows the user to select and access services on devices in his personal working area; and
- It builds a composite/virtual machine by combining one or more services of several devices to give the illusion of being one single distributed machine.

To construct this architecture we introduce a new component that we call Personal Proxy Agent (PPA). This PPA is the heart of our architecture. It will be the medium between the user's Home Directory and the user himself. The user receives incoming requests/calls and sends outgoing requests/calls through his PPA. The PPA will act on behalf of the user to detect, select and build composite devices that satisfy his preferences and requirements. As a proof of concept, we built a prototype that uses Bluetooth components for wireless device interaction, the Jini protocol for service location and the SIP protocol for request processing.

Acknowledgements:

I would like to thank my supervisor Prof. G.V. Bochmann for his valuable advice, encouragements and directions. He was always available to help when I needed him. His time, support and patience are really appreciated.

I want to thank my co-supervisor Dr. D. Makaroff for his guidance and generous help.

Thanks to my colleagues in the Distributed Systems Research Group who collaborates and contributes to make our lab a pleasant working area. I really appreciate especially the help and advice of K. Elkhatib.

I am grateful to my family for their support, but especially my parents who made my wishes become true, without them it would never be possible to continue my studies.

Finally, I would like to thank especially Mohamed who was present when I needed him the most.

Table of Content:

List of Figures and Tables:

Chapter III

Chapter IV

Chapter V

Chapter VI

Appendix A

Appendix B

Appendix C

Appendix D

Chapter I
Introduction

With the explosive growth of small, portable and sophisticated communication devices that are part of our daily needs, tasks and behavior, such as pagers, cell phones and PDAs, the idea of making those devices communicate and exchange data and services to serve and meet the user needs and expectations was born. From this idea, a new vision is merging - one of being always connected, regardless of the location and the access device of the user. This vision has created several forms of mobility, mainly device mobility, session mobility and user mobility. We define device or terminal mobility as the ability of a device to keep network connectivity when moved from one location to another. Network layer technologies [PB94] can provide this form of mobility. We define session mobility [Hotdesk] at the application layer, allowing a user to stop a session on a certain device, and continue it on another device (possibly at a later time) from the same point where the session stopped. User mobility or personal mobility [App et al 99] is the ability of the user to receive calls and access telecommunication services on any terminal at any location, giving the network the ability to identify the end user as she/he moves. Many projects [Zou00], [App et al 99], [Pha et al 00] have treated the device and user session mobility.

Collection of complete comprehensive information concerning the user is stored in a single user profile at a directory server called the Home Directory Agent (HDA) [EHB00] of the user. The user preferences are not fixed data; on the contrary they are subject to changes. The user has the ability to customize his system by modifying his profile thus updating his information when acquiring a new device. In most existing systems, the user keeps a list of devices that are stored in his profile. This list is more fixed than dynamic. As the mobile user changes location, a set of different devices surrounds him, that he may be able to use/access to receive information. If the user wants

to use the nearby devices, he has to update the device list contained in his profile by contacting his HDA. These devices change with the mobility of the user creating a new personal area network. This way of updating the user's device list is not worthwhile, because the user may not stay long enough at the same place. It may also be annoying for the user to do it each time he moves from one place to another.

Our architecture proposes a solution for this problem. It defines a new way of accessing and using the surrounding devices without involving the user directly. Seeking to achieve the idea of "Always being connected", our project proposes a new way of keeping the user constantly in touch by allowing him to access and select devices present in his working area and receive incoming calls. At the same time, it may offer complete privacy if the user asks for it, because the home directory can mask his current address and receive all invitations through it and forward it to the user on the selected device. Additionally, it allows the user to have composite devices or to build a virtual machine, which combines two existing services in different machines to act like a single machine. The selection of devices is always based on the QoS defined in the user profile and the availability of services and capabilities of the device(s).

We have introduced a new element that we call a Personal Proxy Agent (PPA), which is the heart of our project. The PPA will be smart enough to detect the devices near the user, and then select the appropriate device offering the required service. The selected device should satisfy the user and session requirements. The basis of the selection algorithm is to merge the user preferences (user profile) and the device capabilities and properties. The information concerning the capabilities, properties and services that a device can offer are gathered and kept in a comprehensive and well-defined format called a Device Profile. This may cover the hardware platform (CPU model, memory size, microphone, speaker, camera...), system software (operating system, list of audio and video encoders...) and applications (JMF, Vic, Vat, whiteboard...) available on the device. This Device Profile is stored on the device itself, and sent to the PPA when needed (mainly during the selection process). It is the PPA that builds the virtual (composite) machine, by selecting required services on different devices and make them work as one machine to satisfy the user and session exigency/preferences.

We assume that a PPA has a permanent connection to the user's Home Directory, and as soon as the user switches on the PPA, the Home Directory sends the user profile (preferences) that it will keep during all its activation period. As soon as the PPA detects the surrounding devices, each nearby device will send its Device Profile to the PPA. Now the PPA has both profiles, and enough information to make an appropriate selection.

Let us consider the following motivating example that makes use of the infrastructure we deal with in this thesis. A user named Bob works as a professor at the department of engineering; he has a conference meeting with his colleagues to discuss some research projects. Bob always carries his Personal Digital Assistant (PDA) containing his Personal Proxy Agent (PPA) with him. Bob leaves his office to reach the conference room in the next building, half way there he remembered that he has an interesting audio/ video presentation done by one of his student stored in his lab, that he would like to show to the people at the conference. Bob calls his lab administrator to send him the presentation. Once at the conference room, his PPA (PDA) will detect the devices in the room; there is a projector, a laptop with no sound card, a speaker, a phone and headphones. Having no choice to select one device that can offer both a video and audio service, the PPA has to build a composite (virtual) device. The PDA having the PPA running on it, will select the laptop and projector for video and headphones for audio instead of the speaker simply because the headphones have a better audio quality service (meeting his preferences) than the speaker. Finally, Bob receives the audio/video presentation on the selected devices. This is basically how an incoming communication call is handled. But for Bob to reach his lab administrator (outgoing communication call), he asked his PDA (PPA) to call his lab's administrator while he was walking toward the conference building. The PPA (PDA) detected a phone in the corridor not far from Bob and selected it to make the call. So as soon as Bob reached the phone, it started ringing. The PDA has confirmed that this is his call. Bob answers the phone and speaks to the administrator. This small scenario shows how the PPA handles both incoming and outgoing requests.

Our proposed solution is an improvement to other personal mobility architectures for classical non-ubiquitous environments. It leverages technologies in short-range wireless communication, such as Bluetooth, that permits build a Personal Area Network

(PAN); it also leverages service discovery protocols to discover the services available just in the WPAN of the mobile user. Running a service discovery protocol over a WPAN restricts the services available for the use of the user to devices that are within the WPAN and hence close enough to the user. A Personal Proxy Agent (PPA) process running on a personal device carries out service discovery and service selection. This PPA also enforces the user's policies and preferences stated in the user profile. We have experimented a prototype using Bluetooth units for wireless detection, and used the Jini protocol for locating the services present in the detected Bluetooth devices. The SIP protocol was the one used for telecommunication requests over IP.

The rest of the thesis is organized as follows: Chapter II presents a literature review about user, session and device mobility; Chapter III discusses wireless networks; Chapter IV reviews different existing service location protocols; Chapter V presents our proposed architecture; Chapter VI shows our implementation prototype and in Chapter VII we present our conclusions.

Chapter II
Related Work on User and Session Device Mobility

What is mobility?

Mobility means different things to different people and what might be considered mobile in one context is quite immobile in another. We might find more than one definition for mobility. From [Leap], mobility is defined as the ability to send and receive communications anytime and anywhere. Mobility means that both source and destination devices, applications and people are free of the constraints imposed by physical location.

II. 1 User mobility

Personal mobility [Sch96] is defined as the ability of a user to get access to telecommunication services from any terminal (e.g. workstations, notebooks, Personal Digital Assistants (PDA), cellular phones) at any time and from any place based on a personal unique identifier, and the capability of the network to provide services in accordance with the user's service profile.

The key challenge today is to be able to reach and communicate with people personally rather than communicating with their possibly inaccessible machines, cell phones that are turned off, or PCs on faraway desktops. Some projects [App et al 99], [KCM01] propose an architecture that puts the person, rather than the devices that the person uses, at the endpoints of a communication session.

The Mobile People Architecture (MPA) [App et al 99] project describes a prototype that performs person level routing, allowing people to receive communications regardless of the network, device, or application they use, while their privacy is maintained. The role of person level routing is similar to that of an IP router, in that it takes a communication from any of a variety of interfaces and directs it out over one or more interfaces, based on the recipient's preferences and on characteristics of the communication itself. Person level routing uses an addressing scheme that uniquely identifies people; these addresses are Personal Online Ids (POID). The personal proxy is

the heart of MPA and consists of four components: the Tracking Agent, the Rules Engine, the Dispatcher, and a set of Application Drivers. The Tracking Agent in a user's Personal Proxy is responsible for keeping track of the user as she/he moves from an application on one device to another application possibly on another device. The Rules Engine uses the user's accessibility information and her/his preferences to direct the Dispatcher on how to route any communication that arrives at the Personal Proxy. In performing the routing the Dispatcher may call upon an Application Driver to convert the communication into a form understandable by the receiving application. The user's Personal Proxy receives a communication on her/his behalf and forwards it to her/him. The Personal Proxy acts as an enhanced online analog to the human assistant. For each type of application the user uses, her/his Personal Proxy will have a corresponding Proxy Application Specific Address (PASA). This address allows the Personal Proxy to intercept and redirect all communication with the user's applications, which are at undisclosed Application Specific Addresses (ASA). MPA is flexible enough to support location privacy for the sender as well as the receiver. When the Personal Proxy receives the communication, it must use the receiver's POID to obtain, from the directory service, the appropriate ASA or PASA if the receiver has also requested privacy, to use when forwarding the communication. This architecture provides support for instant and convenient communication between people as they move from one place to another and use multiple heterogeneous communication devices, including laptops, PDAs, or cellular phones. MPA makes it possible to protect people's location privacy.

This project focuses on finding or contacting the user, by tracking his working application; if he is working on the laptop, it will send the communication to his laptop; if he is working on his station, he will receive the communication on his station.

II. 2 User and session mobility

By saying mobile users, we imply two kinds of mobility: personal mobility and session mobility. Mobile user refers to the user who changes physical location by moving from one terminal to another, either in the same room or between rooms or buildings, while maintaining his application sessions. The term session mobility refers to an application session running on a certain device that stops and then continues execution on

another device, starting from the exact interruption point of the application session. Session mobility prevents disruption of active sessions while users change terminals. It follows the user's physical location and the network locates/tracks and configures the user's movements to provide him services. User and session mobility have been the focus of several projects [Zou00], [Bru et al 00] and products.

The Mobile Id protocol project [Zou00] implemented a badge-activated application-level handoff protocol supporting user mobility for multimedia streaming applications. The protocol has a location aware system to detect user movement on the client side; a mobility database to store all the interrupted sessions on the service side; and a Mobile ID client/server side manager-pair coordinates the events on either side. This project developed an application of multimedia session routines for the Mobile Id protocol to redirect the session from one location to another when the user physically moved.

The EasyLiving project [Bru et al 00] at Microsoft research is concerned with the development of architecture and technologies for intelligent environments. An intelligent environment is a space that contains diverse devices that work together to provide users with access to information and services. For example, it might be desirable for the system to provide light for the user as he moves around at night; to enable this, the system uses some form of perception to track the user, and must be able to control all of the different light sources. EasyLiving's goal is the development of an architecture that aggregates diverse devices into a coherent user experience. An example scenario could start with the user at home. He enters the living room and sits down at a PC in the corner, surfs through a selection of MP3s, and adds them to a playlist. He gets up and sits down on the couch and his session follows him to the large wall screen across from the couch (this screen is selected because it is available and in the user's field of view). The user picks up a remote control sitting on the coffee table and uses the trackball on it to request the room controls. They appear in a window on the wall screen, showing a small map of the room with the controllable lights. He uses this interface to dim the light. The user opens up his playlist and presses play. The music comes out of the room's large speaker system.

As the number of available networked devices for a given user interaction increases, the complexity of identifying and selecting the appropriate devices for that

interaction increases greatly. Additionally, to enable user specification of devices desired for a given task, a mapping between network and physical identity is exceptionally helpful. Sensing devices allow the system to have information about the state of the world, such as locations of people, places, things, and other devices in the space. Having this information makes the interaction with the user seem more natural. Stereo computer vision is currently used in EasyLiving to avoid failing to give the system information about the world state. EasyLiving enables coupling input/output device applications rather than having a separation of hardware device control, internal computational logic and user interface presentation. It gives the flexibility to change the interaction mechanism without requiring modification of the underlying application. EasyLiving uses its EZLGM geometric model that provides a general geometric service for ubiquitous computing, focusing on in-home or in-office tasks in which there are different I/O, perception, and computing devices supporting multiple users. It is aimed at geometry "in the small" that is, for applications which may require sub-meter localization of entities in the environment; localizing services for large scales remains a problem.

These projects focus on the idea of tracking the user through sensors (either a badge or other infrared sensors) to locate him, which makes the session mobility possible. There is a priori knowledge of the geometrical position of the devices. There is a centralized system, which senses the user and finds his position compared to the present devices in his PAN, and then selects to satisfy the session motion or to perform intelligent tasks like shutting down the light automatically as it senses the user passing the door.

II. 3 Device mobility and service selection

The intersection of several technologies, including embedded systems, wireless networking and personal computing technologies, and service discovery is Pervasive computing. The concept of an ad hoc network facilitates the ability of devices to connect to each other or to attach to existing networks and to use available services. Interactive workspaces are addressed by several research projects [HB00], [Her et al 01], [BAL01].

The RAUM [HB00] system proposes a new paradigm for local communication between devices in ubiquitous computing environments, assuming a multitude of

computerized everyday appliances communicating with each other to solve tasks. The basis of this paradigm is the concept that the location of devices is central. Devices define their communication scope by spatial criteria. In spatially aware communication, devices use location as their primary attribute for selecting communication partners. The RAUM architecture defines the communication stack and the protocol for artifacts and optional communication infrastructure. Its concept describes the communication of devices according to their spatial relationship in the physical world. Devices define regions of interest in which they communicate (RAUMs). RAUMs describe a corresponding geometric space in the physical world; the definition of geometric space is an envelope over all points defining the RAUM. Objects that are inside the space of interest are also inside the communication scope of the device that defines the RAUM. This is because the communication in the RAUM- concept relates to the physical location of the device taking part in the communication. Every device that is physically present inside the envelope of a RAUM is then a member of the RAUM. The complete stack of the RAUM system consists of four layers. The "RAUM-communication-layer" is the lowest layer and groups the physical and the DLL layer. The next layer, called the "RAUM-Layer" is responsible for the creation and dissolution of communication relationships, for device location and for routing of packets. The third layer is the RAUM-Event-Layer that provides reliable communication and eventually offers abstract event communication services to the fourth layer, the application layer. Every packet sent through RAUM-Layer, must be mappable to a geometric position, and every device has to know its own position in space and be aware of the defined RAUMs in which it is located. The Locator service whose instances communicate with the RAUM Location Protocol (RLP) handles Device position information. The RAUM themselves are named and defined geometrically by a set of points. The RAUM system accepts packets if the potential receiving device and the datagram packet itself are located inside the same RAUM. The RAUM system is an example of a location-based communication system. This architecture is layered and can be related to the one defined in the ISO/OSI model. Layer 3 "RAUM-Layer" plays an important role in the RAUM architecture; it handles the delivery and reception of packets according to spatial areas (RAUMs) and out-routing of packets.

The RAUM technology is proposed to solve location-communication problems. Its can be used to detect and select the nearest device in the user's personal area network.

The DEAPspace [Her et al 01] project provides a framework for interconnecting pervasive devices over wireless communication technologies. Such devices in proximity can form an ad-hoc transient alliance to create new types of collective distributed applications. This project presents a new service announcement and discovery protocol that is different from other existing service discovery protocols such as SLP [SLP], Jini [Jini], UPnP [UPnP] and Salutation [Salutation]. In DEAPspace project, the service discovery process is a push-based model, where each device broadcasts its own worldview of the nearby present services. Each node maintains a list of service descriptions that it broadcasts on a regular time basis. Each node processes a newly received service list by combining the remote (received) list services with its own (local) services, and updates the time-to-live values of the received (remote) services. The advantage of broadcasting the entire worldview is that it permits the node to incorporate new elements found in each broadcast into its own view, and then occasionally missing a broadcast will not cause a problem, because the next broadcast by any device that did not miss the first one will repeat and update the information. The DEAPspace service usage model is role based. An entity providing a service is a provider and conversely, the entity requesting a service is a requester. To provide its services, a provider may act as a requester making use of other services. When acting as a service provider, the service description format determines the precision for services advertisements whereas when acting as a service requester, it determines the selection capabilities for choosing a matching service offer. This forms a distributed system: that is, requesters and providers can live on separate devices. The DEAPspace proposal reduces the time for discovering available services, because when one device broadcasts its worldview it transmits more information than when each device broadcasts its own information.

Barbeau's group [BAL01] proposes two protocols sent using a wireless point-to-point infrared communication, defined as the IDAdvert and SetIrdLink protocols. These protocols are not concurrent with the existing service discovery protocols but extend the existing protocols' capability by leveraging the sending of a small amount of information over a simple transport like infrared. Both IDAdvert and SetIrdLink protocols assume

that the devices on the network are accessible using a common addressing scheme, such as Internet protocol (IP) or Bluetooth [BT] addresses. The IDAdvert Protocol is ideally suited to a low cost one-way infrared implementation to facilitate service selection. Implementation took place over an IrDA stack for simplicity and convenience. The basic functionality of the protocol allows a user, close to a resource that she/he wants to use, to send an IDAdvert message through her/his mobile device over an infrared medium to the wanted resource. This message contains the mobile device's network address. Once the resource receives the message, it verifies the validity of the address and then uses it to reply with a unicast advertisement message of its services to the consumer through the network. For implementation, it uses a PDA having an access network point and under the SLP [SLP] protocol for discovering devices. The SetIrdLink Protocol is slightly more complex than the IDAdvert in that it requires two-way infrared communication and relies on the user agent to first acquire a list of possible service identifications. A possible scenario can be a Bluetooth PDA's user agent that uses Service Discovery Protocol [BT] to retrieve any services available from resources present in its piconet or scatternet, and the SetIrdLink protocol over infrared to automate the selection amongst all the services returned to the closest printer. Because the IR enables short-range point-to-point communication, only the closest resource in the line-of-sight of the infrared beam will receive the SetIrdLink message. Resources that do not match the address list sent will remain silent; matching resources will return their network address and access point through SetIrdLinkConf. For implementation, this protocol uses SLP on a Bluetooth enabled palm device that supports the LAN access point and can open a PPP connection to the network. Implementation was on a phone having both an SLP service agent and an OBEX service routine that can process an IDAdvert request. This triggers the phone to send a SAAdvert message to the application on the PDA, which will extract the phone's DN after acquiring its service access URL. The PDA will use the phone's DN to use as a default choice for the creation of a forwarding rule for the PDA's Personal Policy Agent (PPA).

Siemens Corporate Research proposed The Composite Device Computing Environment (CDCE) project architecture [Pha et al 00], which provides mobile users with access to rich multimedia information and services, where the surrounding available

computing resources are considered as another facet of situated computing. This architecture uses a PDA as a primary device through which requisition information, applications and services are possible. Based upon a user's request done through his PDA, the CDCE framework dynamically creates a unified composite, or virtual device composed of an appropriate mix of the surrounding resources. The gateway is one of the main parts of this architecture, intelligent enough to organize, synchronize and distribute requested information and services for interactive media access. It provides four basic functions. It manages the pool of services available to the users. It establishes the composite devices based on location dependent information received from the PDA - predefined knowledge about the environment and dynamic information about the current state of the device. It maps the request issued by the PDA to the applications, and corresponding output, to the appropriate nodes in the virtual device. Finally, it performs the dynamic conversions needed to present the information on the selected output node. The primary focus of this architecture is the intelligent information distribution. Hence, it needs methods such as splitting, conversion and filtering to adapt "the content" to multiple output devices. These methods respectively address three issues. These are: an intelligent content separation, e.g. separate audio and video parts from a video message to be redirected to a telephone for audio and a PC without sound card for video; media conversion techniques, such as text to speech; content extraction and delivery of the sub-content, e.g. delivery of only audio part of a video message to telephone. Use has been made of a Distributed Component Object Model (DCOM) to facilitate a bi-directional flow between the PDA and gateway environments. It enables the gateway to remotely invoke processes without any proprietary client code. The basis of this architecture is a PDA that detects and communicates information to the CDCE Gateway, which in turn constructs a composite device. A proposed scenario shows how a doctor equipped with a PDA can use a nearby TV to see a patient's x-ray. First, a doctor uses his/her PDA to request a patient's medical history including symptoms, diagnoses and x-rays. The PDA detects the presence of a TV and a telephone in the patient's room using the infrared interface, and then sends all this information with the doctor's request to the CDCE gateway server. After authorizing the doctor's access and verifying a secure connection, the CDCE gateway routes symptoms and diagnoses directly to the doctor's PDA and

transmits x-ray images to the TV via RF for viewing - recognizing the PDA's physical limitations. The fundamental idea is to avoid having a single PDA performing the whole job. IR serial interface adapter for detection and Cellular Digital Packet Data (CDPD) network for PDA and Smart gateway communication realizes the project.

In this project, due to its physical limitations, they have used the PDA mainly as the tool to detect devices, and send them for treatment in a centralized "smart gateway". The basis of selection of output devices is whatever resources are currently available, and not QoS parameters, where the user preferences are taken into consideration. This project focuses more on the local environment having distributed smart gateways, which is not very flexible in a sense that we always need a centralized third party.

In the UbicompBrowser [Bei et al 98] project, the authors investigated the use of a PDA to enable ubiquitous access to local resources as well as resources on the WWW. The proposed browser, UbicompBrowser, detects devices and resources in the environment of the user, and delegates the rendering of the requested resources to these devices in order to overcome the limitation of the PDA. However, the UbicompBrowser does not deal with the issue of content adaptation of the resources to fit the capabilities of the available devices, or with the issue of keeping the user always connected.

Most of these proposed architectures are aware about the user attitude and needs, as well as his requirements without involving QoS parameters. Some research projects [EHB00], [RKH00] propose a way to satisfy the user by taking into account his preferences, and consider the multimedia QoS parameters.

Quality of Service Negotiation Based on Device Capabilities and User Preferences [EHB00] defines a scheme that represents and stores the user preferences in a well-defined format called the user profile, and device capabilities in a device profile. This system architecture merges the user profile and device profile for all participants to make the selection of QoS parameters. The selection is based on the concept of maximizing the user's satisfaction. It uses all possible combinations of QoS parameters of all available services, and selects the combination that generates the maximum satisfaction within the hardware and software limitations of the device, and the personal

perception of individual multimedia QoS parameters, as well as the caller and callee preferences on the choice of the multimedia types.

A Framework for QoS and Mobility In the Internet Next Generation [RKH00] defines a new framework that integrates various existing QoS over IP architectures with protocols supporting mobility. It offers the user the flexibility to use and access a large variety of applications with different QoS requirements, as well as a choice between wireless and wired access technologies based on certain QoS parameters. The main advantage of this framework is the possibility to negotiate the QoS parameters through a session layer negotiation protocol (as SIP [SIP] or H323 [H.323]) before the actual network resources are reserved. It enhances the scalability and complexity problems, and reduces the waste of resources in the access network.

II. 4 Conclusion

As small device technology gains popularity with the user, mobility gains the interest of researchers, investigating the different concepts of mobility, which include nomadic users, device mobility, and session mobility. In this chapter, we have seen different projects involving mobility and service location concepts, and others that focus on QoS to satisfy the user. Our architecture is an extension to [EHB00] work. It proposes a new concept of mobility taking into consideration QoS parameters defined in the user preferences and the device capabilities.

Chapter III

Wireless Networks

The term wireless networking refers to technology that enables two or more computers to communicate using standard network protocols, but without network cabling.

There are two kinds of wireless networks as defined in [WLAN]:

An ad-hoc, or peer-to-peer wireless network consists of a number of computers each equipped with a wireless networking interface card. Each computer can communicate directly with all of the other wireless enabled computers. They can share files and printers this way, but may not be able to access wired LAN resources, unless one of the computers acts as a bridge to the wired LAN using special software. (This is called "bridging").

A wireless network can also use an access point, or base station. In this type of network the access point acts like a hub, providing connectivity for the wireless computers. It can connect (or "bridge") the wireless LAN to a wired LAN, allowing wireless computer access to LAN resources, such as file servers or existing Internet Connectivity.

III. 1 Introduction

Wireless communication exists in different forms, such as radio frequency, Infrared, satellite and others. One way to classify wireless communication technologies is by the geographic area they cover. Three classes have been defined as in [WLAN], wide-area, local-area and personal area networks.

Wireless Personal Area Networking (WPAN): describes an application of wireless technology intended to address usage scenarios that are inherently personal in nature. The emphasis is on instant connectivity between devices that manage personal data or which facilitate data sharing between small groups of individuals. An example is synchronizing data between a PDA and a desktop computer. Another example is spontaneous sharing of a document between two or more individuals. The nature of these types of data sharing scenarios is that they are ad hoc and often spontaneous. Wireless communication adds value for these types of usage models by reducing complexity (i.e. eliminates the need for cables).

Wireless Local Area Networking (WLAN): This focuses more on organizational connectivity not unlike wire-based LAN connections. The intent of WLAN technologies is to provide members of workgroups access to corporate network resources, be they shared data, shared applications or e-mail, but does so in a way that does not inhibit the user's mobility. The emphasis is on a permanence of the wireless connection within a defined region, like an office building or campus. This implies that there are wireless access points that define a finite region of coverage.

Wireless Wide Area Network (WWAN): While WLAN addresses connectivity within a defined region, WWAN addresses the need to stay connected while traveling outside this boundary. Today, cellular technologies enable wireless computer connectivity either via a cable to a cellular telephone or through PC Card cellular modems. The need addressed by WWAN is to stay in touch with the critical business communications while traveling.

III. 2 Bluetooth [BT]

Bluetooth specification technology was proposed by the Bluetooth Special Interest Group (SIG) in early 1998, which included leaders in the mobile telephony and computing field - Ericsson, IBM, Intel, Nokia and Toshiba. It is a short-range wireless technology allowing devices like PDAs, notebooks and phones to exchange voice and data in real time, covering an area range of 10 meters. It replaces connecting cables for electronic devices. Bluetooth is robust compared to other technologies in terms of complexity, low power and low cost. Its radio modules operate in the unlicensed 2.4GHz ISM (Industrial, scientific, medical) band at a gross rate of 1Mb/s, and avoid interference from other signals by using a fast frequency-hopping scheme (nominally 1,600 times per second). A single unit can support a maximum data transfer rate of 721kbits/second or a maximum of 3 voice channels. It is possible to support multimedia applications having a mixture of voice and data transfer.

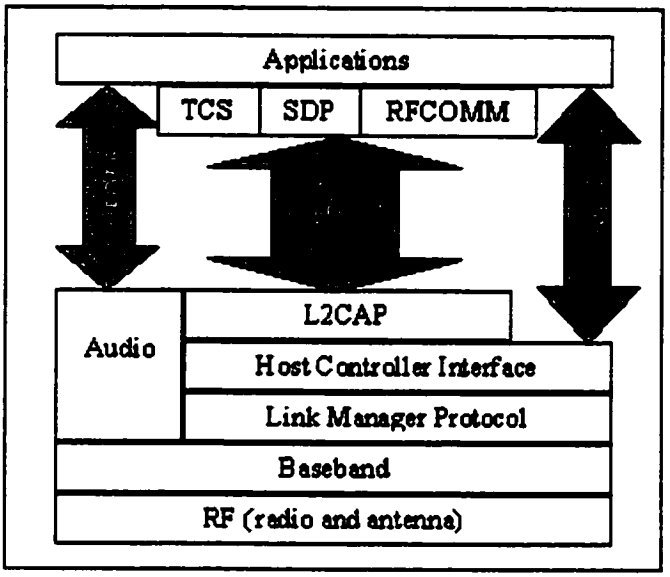

Figure III. 1: Bluetooth protocol stack[MB00]

In a Bluetooth network, all devices/units have the same hardware and software interfaces, but are differentiable by a unique 48bits Bluetooth address. At the start of a connection, we call the unit initiating the communication with the other unit(s) a master and the other unit(s) a slave(s). These assigned roles are valid only during this particular connection.

A master can have up to seven active slaves in one connection. The configuration of having a master connected to one or more slaves is a piconet. Time division multiplexing allows both a master and a slave to play a bridging role between several piconets, by forming a so-called scatternet as shown in Figure III.2. The role of a master in a piconet is the synchronization of the frequency hopping spread spectrum communication between the devices. The Bluetooth network supports both point-to-point and point to multipoint connections.

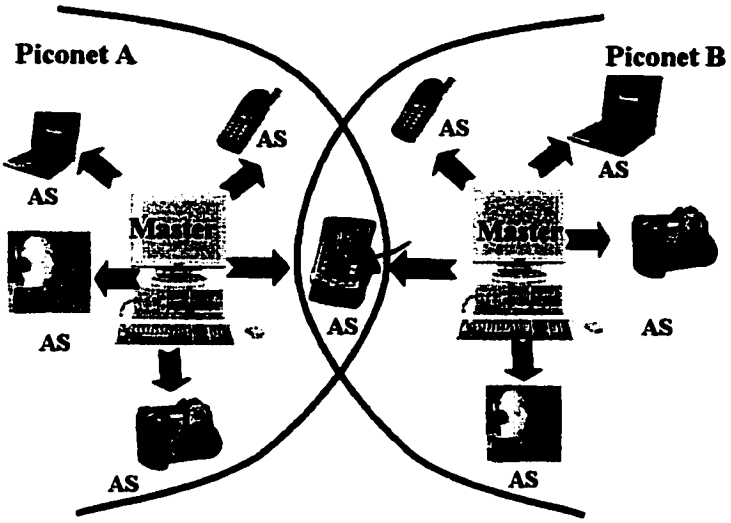

Figure III. 2: Scatternet example where an Active Slave (AS) can be in both Piconet A and Piconet B

Figure III.1 represents the Bluetooth protocol stack. The Bluetooth Radio (layer) is the lowest defined layer of the Bluetooth specification stack. It defines the requirements of the Bluetooth transceiver device such as in-band and out-of-band spurious emissions, frequency accuracy, co-channel etc. The Bluetooth radio accomplishes spectrum spreading by frequency hopping in 79 hops displaced by 1 MHz.

The Bluetooth baseband is the physical layer of Bluetooth. It supports two types of links over a piconet: asynchronous connection-less (ACL) and synchronous connection-oriented (SCO) links. It handles packets and does paging to access and make inquiries about the area's Bluetooth devices. The SCO link is a symmetric point-to-point link between a master and a single slave in the piconet. The master maintains the SCO link by using reserved time slots at regular intervals (circuit switched type). The SCO link mainly carries voice information. The master can support up to three simultaneous SCO links while slaves can support two or three SCO links. Retransmission of SCO packets does not occur. This uses SCO packets for 64 kB/s speech transmission.

The ACL link is a point-to-multipoint link between the master and all the slaves participating on the piconet. In the slots not reserved for the SCO links, the master can establish an ACL link on a per-slot basis to any slave, including the slave already

engaged in a SCO link (packet switched type). Only a single ACL link can exist. For most ACL packets, packet retransmission is applied.

The Bluetooth Link Manager carries out link setup, authentication, link configuration and other protocols. It discovers other remote Link Managers and communicates with them via the Link Manager Protocol (LMP). To perform its service provider role, the LM uses the services of the underlying Link Controller (LC).

The Host Controller Interface (HCI) provides a command interface to the baseband controller and link manager, and access to hardware status and control registers. Essentially this interface provides a uniform method of accessing the Bluetooth baseband capabilities.

The Logical Link Control and Adaptation Layer Protocol (L2CAP) is layered over the Baseband Protocol and resides in the data link layer. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. L2CAP permits higher-level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length. The definition of the L2CAP protocol is for only ACL links and provides no support for SCO links.

RFCOMM is a simple transport protocol, which provides emulation of RS232 serial ports over the L2CAP protocol. The ETSI standard (TS 07.10) is the basis of this protocol. This protocol supports up to 60 simultaneous connections between two Bluetooth devices. The number of connections potentially used simultaneously in a BT device is implementation-specific.

The service discovery protocol (SDP) provides a means for applications to discover which services are available and to determine the characteristics of those available services.

III. 3 IEEE 802.11[802.11]

The Institute of Electrical and Electronic Engineering (IEEE) specified the standard IEEE 802.11 from 1990 to 1997. The first phase of the standard IEEE 802.11 supports only 1 Mbit/s and 2 Mbit/s data rates. The first phase standard was followed by

an extension, IEEE 802.11b, which supports data rates up to 11 Mbit/s with the radio frequency technology direct sequence spread spectrum. It operates in the unlicensed 2.4 GHz ISM frequency band. It is a standard for wireless local area networking technology (WLAN), which covers distances from ten to a few hundred meters.

The IEEE 802.11 is composed mainly of the physical layer (PHY) and Medium Access Control (MAC) sublayer. In IEEE 802.11, the destination address does not equal the destination location like in wired LANs where an address is equivalent to a physical location - here a station (STA) is the addressable unit.

One of the requirements of IEEE 802.11 is to handle mobile as well as portable stations. A portable station is one moving from one location to another, but that is used while at a fixed location, whereas mobile stations actually access the LAN while in motion.

IEEE 802.11 architecture consists of a basic building block called the basic service set (BSS). We can think of the BSS as a coverage area having different stations (STAs) with the ability to communicate with each other directly. As soon as a station is out of the coverage of its BSS, it can no longer communicate directly with the other stations remaining in the BSS.

The independent BSS (IBSS) is the most basic type of IEEE 802.11 LAN, where it may contain a minimum of two stations as in Figure III.3.

IEEE802.11 Components

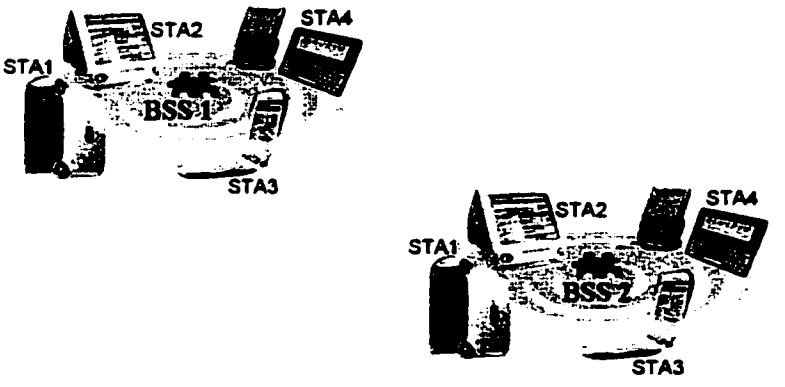

Figure III. 3: Basic Service Set

The association between STA and BSS is dynamic, where a station can be on, off, come within range or go out of range. To become a member of the BSS a station shall become "associated".

The need to interconnect multiple BSSs by forming an extended form of a network leads us to a new architectural component called a distribution system (DS). The DS provides the logical services necessary to handle address to destination mapping, enabling mobile device support and seamless integration of multiple BSSs.

Access point (AP) is the name given to the STA that provides access to the DS and its services. It permits the data to move between a BSS and the DS.

The DS and BSSs allow IEEE 802.11 to create a wireless network of arbitrary size and complexity. We refer to this type of network as the extended service set (ESS) network (see Figure III.4). Stations within an ESS may communicate, and mobile stations may move from one BSS to another within the same ESS transparently to LLC.

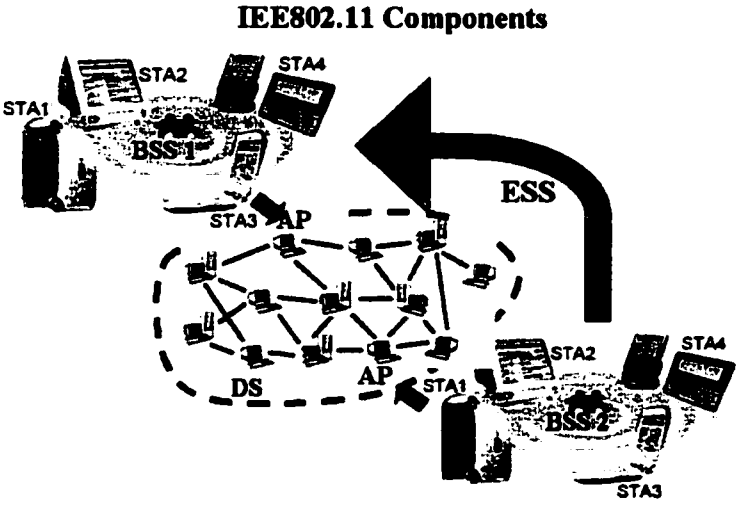

Figure III. 4: Extended service set

The IEEE 802.11 standard places specifications on the parameters of both the physical (PHY) and medium access control (MAC) layers of the network. The PHY layer handles the transmission of data between nodes; it uses a direct sequence spread spectrum. The MAC layer is a set of protocols, which is responsible for maintaining order in the use of a shared medium. In this protocol, when a node receives a packet to

transmit, it first listens to ensure no other node is transmitting. If the channel is clear, it then transmits the packet. Otherwise, it chooses a random "backoff factor" which determines the amount of time the node must wait until it is allowed to transmit its packet. During periods in which the channel is clear, the transmitting node decrements its backoff counter, but not when the channel is busy. When the counter reaches zero, the node transmits the packet. Because the probability of two nodes choosing the same backoff factor is small, minimal collisions occur between packets.

Whenever it wants to transmit a packet, the transmitting node first sends out a short ready to send (RTS) packet, containing information on the length of the packet. If the receiving node hears the RTS, it responds with short clear to send (CTS) packet. After this exchange, the transmitting node sends its packet. On successful receipt of a packet, as determined by a cyclic redundancy check (CRC), the receiving node transmits an acknowledgment (ACK) packet. Sending those control frames is necessary to avoid the "hidden node" problem, shown in Figure III.5, where node A can communicate with node B, and node B can communicate with node C. But node A can not communicate with C, thus even if node A senses a channel to be clear, node C may be transmitting to node B. The protocol described above tells node A that node B is busy, and thus it has to wait before transmitting its packet.

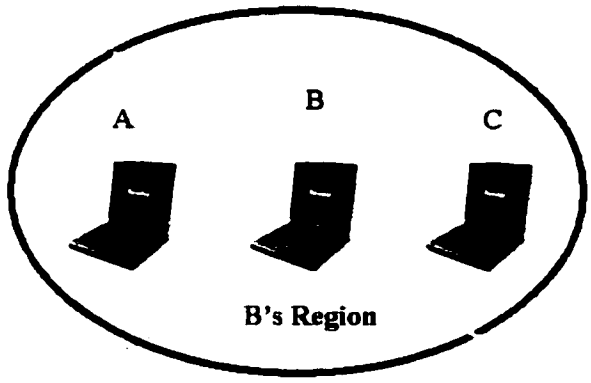

Figure III. 5: The hidden node problem

III. 4 Conclusion

The WAN's coverage area is typically measured in kilometers or at least hundreds of meters. Communication over such distances requires high power transmission; this is why it needs a licensed frequency band. The data rate is typically relatively slow.

WLANs typically cover distances from ten to a few hundred meters. This smaller distance does not require high power transmission, so often permits the use of unlicensed frequency bands. The usual use of LANs is for high capacity data communications. They have quite high data rates. The devices used in this technology need a robust computing platform and power supply. It tends to be better suited for stationary networks or at least in networks where the participants move infrequently. WLAN applications permitting a degree of mobility and eliminating the wires used to connect to networks widely deploy IEEE 802.11 technology. It has a nominal range of 100 meters and data rates up to 11Mbps, it usually needs a powerful receiver/transmitter, which makes it less suited for small hand held devices like pagers, PDAs and mobile phones. This is where WPAN comes into play. The WPANs cover distances in the order of 10 meters and typically connect various personal portable devices without using cables. This peer-to-peer device communication does not usually require fast data rates. Bluetooth technology is more suited for short range, low power communication networks. It has a nominal range of 10 meters with a data rates up to 1 Mbps. Its low power consumption makes it more portable for small, mobile, battery-powered devices as PDAs, pagers and mobile phones. The fact that IEEE 802.11 and Bluetooth operate over the same 2.4 GHZ band, does not make them competitive but more complementary to each other. Optimization occurs for different purposes in different domains; Bluetooth is not a robust networking technology and IEEE 802.11 is not the preferred technology for WPAN applications.

Chapter IV

Service Location Protocols

A service discovery protocol is used by devices to disseminate and discover services within a network. It allows devices to advertise the services they provide, while allowing other devices to inquire and use these services. For instance, it can be used to find a printing service in a network without a previous configuration of the printer or the computer.

IV. 1 Introduction

With the emergence of wireless ad-hoc networks, specialized devices/services information is taking on considerable importance. Devices come and go frequently in ad-hoc networks, which results in a need for a new mechanism to replace the old resource configuration and management. Service discovery protocols have been developed for that purpose, all in favor of the mobile user. In ad hoc networks, where services can appear temporarily, service discovery protocols are indispensable for the incorporation of dynamic changes in the available services.

Many existing service discovery protocols provide the means to announce the presence of a service to the network, to discover services in the neighborhood, and to access those services. Among emerging service discovery protocols, Jini [Jini], Bluetooth Service Discovery Protocol (SDP) [BT], Service Location Protocol (SLP) [SLP], Salutation [Salutation], Universal Plug and Play (UPnP) [UPnP]. They all address those aspects but with different perspectives. We describe in the following paragraphs the most popular and widely used protocols including Jini, Bluetooth (SDP) and SLP protocols.

IV. 2 The Jini protocol

Jini was developed by Sun Microsystems in 1998. It is one of a large number of distributed systems architectures such as CORBA and DCOM. Jini simplifies distributed systems development and deployment taking care of some difficult parts of DS like the one dealing with discovery and look-up of distributed services. A Jini system is a collection of clients and services all communicating by the Jini protocols - usually applications written in java that use the Java RMI (Remote Method Invocation) mechanism to communicate.

When we talk about Jini architecture, it involves three parties: a service such as a printer, a microwave, a travel agency, etc.; a client, which would like to make use of this service and thirdly a look up service (service locator) which acts as a broker/locator between services and clients.

These three parties move this code around by marshalling the objects, involving serialization to permit them to move through the network.

IV.2. 1 Look-up service

A look-up service finds and resolves services. The look-up service is one of the central mechanisms or nodes of the Jini system, which maps interfaces indicating the functionality provided by a service to sets of objects that implement the service.

In order for the client to locate a service, it queries the look-up service/service locator. To do so, the client has to determine where the look-up service is. On the other hand, a service must register itself with the look-up service, and in order to do so it has first to locate the service locator.

Thus, the initial phase for a client and a server is to discover the look-up service, which is simply another Jini service that stores services and sends those services to clients asking for them. Either unicast or multicast can perform service locator discovery.

When a look-up service receives a request, it returns an object known as a registrar to the server. It acts as a proxy to the look-up service and runs in the server JVM (Java Virtual machine). All the requests made by the service provider to the look-up

service must go through this proxy registrar. The server provider uses it to register services with the look-up service. The client has to go through the same process to get the registrar but does different things with it. The client mainly uses the registrar to request service objects for copying across to it.

IV.2.1. 1 Unicast discovery

With advance knowledge of the location of the service locator, Unicast discovery is used. It directly addresses the machine where the look-up service resides by using its publicly known address or link using the class LookupLocator [Jini] in package net.jini.core.discovery. It has two constructors: the first passes a URL string as a parameter formed as jini://host/ or jini://host:port/ and the second one passes a host string and an integer port. It also has methods as:

String getHost();

int getPort();

that return information about the host the look-up will use and the port it will connect on.

In Figure IV.1, a UnicastRegister object makes a *new()* call to create a LookupLocator and this call returns a Lookup object. Then the *getRegistrar()* method is invoked on the lookup object, resulting in the creation of a *ServiceRegistrar* object which is returned by the method as the registrar.

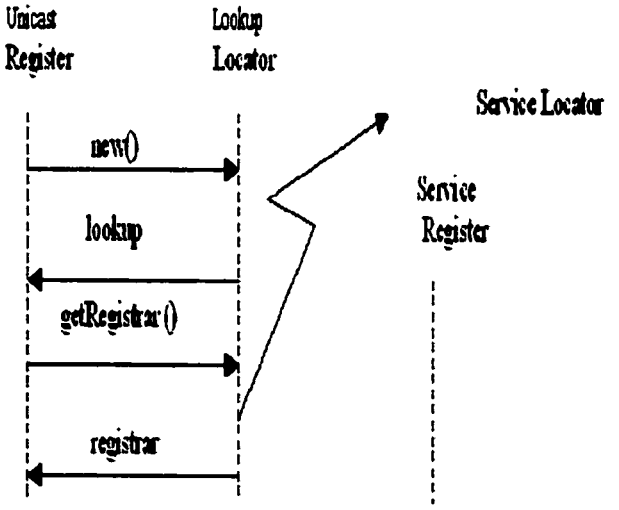

Figure IV. 1 UML sequence diagram for lookup[Arn et al 99].

IV.2.1. 2 Broadcast discovery

We use broadcast discovery when the look-up service location is unknown and we want to search for it. There may be any number of look-up services running on the network accessible to broadcast search and each one of these may choose to reply to a broadcast request, where some services may be available to any user but some may be more restricted in applicability. There could be lookup services restricted to a particular group of services such as Engineering department services, etc.

When starting a look-up service, it may receive a list of groups to act for and a service can give a list of groups information it belongs to. Broadcasting a look-up service discovery may take time in the sense of the number of look-up services that exist and can reply in a network. For notification of the discovery of a look up service, the application must register a listener with the *LookupDiscovery* object using events for notification.

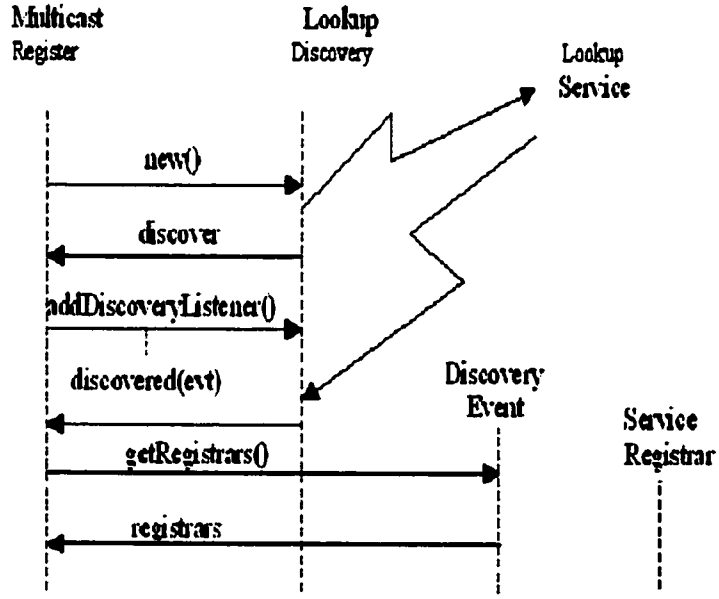

Figure IV. 2: UML sequence diagram for discovery [Arn et al 99].

In Figure IV.2, creation of a LookupDiscovery object starts the broadcast search returning the discover object. The *MulticastRegister* adds itself as a listener to the discover object. The search continues in a separate thread, and as soon as a look-up service replies, the discover object invokes the *discovered()* method in the

MulticastRegister by passing it a newly created DiscoveryEvent. The MulticastRegister can call the method *getRegistrars()* on a DiscoveryEvent object returning a suitable *ServiceRegistrar* object.

IV.2. 2 Service registration

One of the important concepts of Jini is that of service. A service is an entity used by a person, a program, or another service. It is a logical concept such as a toaster, a chat, and a travel agency. A Java interface usually defines a service, commonly identified by this interface. A Jini system consists of systems collectable together to perform another or a particular task. A service may make use of other services and a client for a service may be a service itself to other clients. A service provider creates a service. The service provider plays different roles such as creating the objects that implement a specific service, registering the service object with the look-up service downloaded to the client, and keeps the service "alive" in a server role.

A pair of protocols called discovery and join adds a service to a look-up service. First, the service provider finds or locates the lookup service using a discovery protocol (see section IV.2.1), and then it joins it by using the join protocol, which is the process of adding a service to a Jini system.

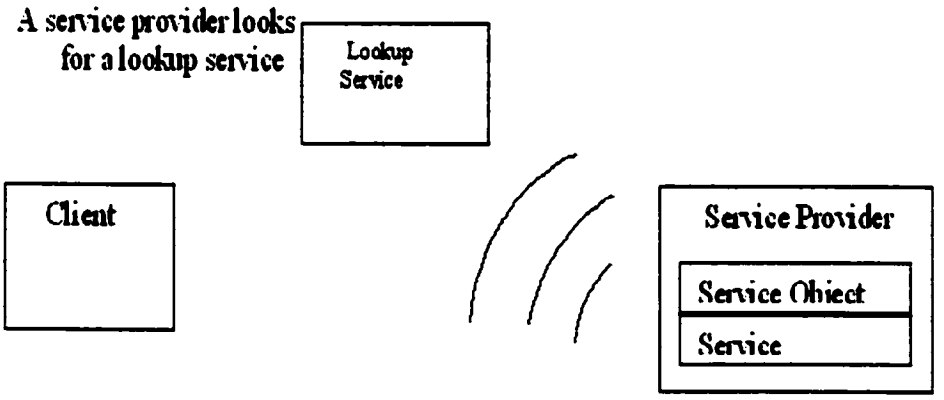

Figure IV. 3: Discovery [Arn et al 99].

The return of a ServiceRegistrar object to act as a proxy for the lookup service occurs on finding a look-up service. The server then registers the service with the lookup service through the *register()* method of the ServiceRegistrar object, passing a *ServiceItem* object as a parameter. The ServiceItem object contains the *ServiceID* used as a globally unique identifier for the service, and an *Entry[]* object that defines a set of attributes as a location and comment.

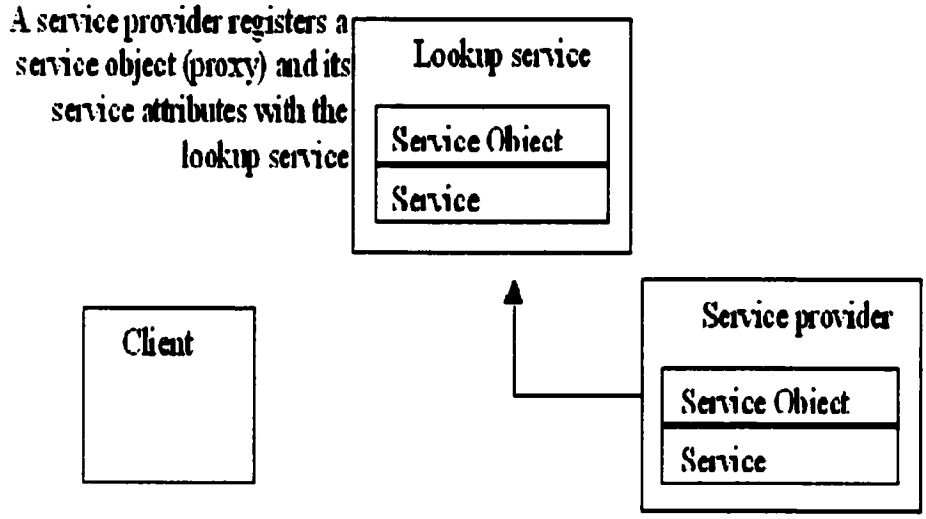

Figure IV. 4: Join [Arn et al 99].

IV.2. 3 Client search

A client locates a specific service by its type along with descriptive attributes. A java interface defines these specifications, so the client needs to ask using this interface. The client will usually request an interface object. This interface defines the set of methods that usable by the client to interact with the service.

After the client performs a look-up service discovery using one of the methods defined in section IV.2.1, it receives a ServiceRegistrar object which is used to search for a service stored in that look-up service using the *lookup()* method .This method finds a service that matches the requested service using a class of type ServiceTemplate to specify the service looked for. A ServiceTemplate object is a list of class objects and a list of entries, and has three fields:

```
ServiceID          serviceID;
Java.lang.Class[]  seviceType;
Entry[]            attributeSetTemplates;
```

The *serviceID* is a "universally unique identifier" used to identify a service. The service locator can use this as a filter to easily discard other services. A serviceType may be a collection of classes or interfaces that can be associated with the same service; for example, a client wants to use a service that implements a Toaster and FireAlarm. Finally the *Entry[]* can be a set of attributes that must be satisfied by each service. Each client required attribute is compared and matched against those defined by the service. For example, in addition to requesting a Toaster and a FireAlarm the user can specify a location "Engineering building". A single match is good enough.

If the match is successful, it returns an object castable into the class required, and invokes a service method on this object.

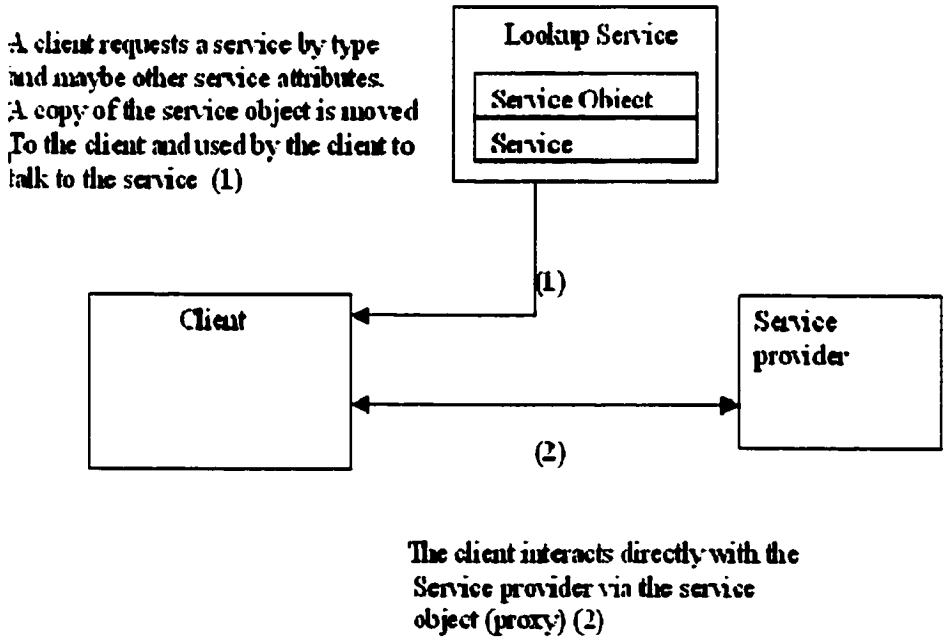

Figure IV. 5: Client search service [Arn et al 99].

IV.2. 4 Leasing

A Lease is a way to guarantee access to a service during a time-period. It is a way for services to register that they are alive. The look-up service acts as the granter of the lease and decides for how long it will create the lease. The lease is returned to the service, and is accessible through the *getLease()* method of the *ServiceRegistration* object.

When the lease expires, it does so silently in the sense that the lookup locator will not send a notification to inform the service provider that the lease has expired; instead it is up to the service provider to renew the lease by calling *renew()* method before the lease expires if it wants the lease to continue.

Leases are either exclusive or nonexclusive. Exclusive leases ensure the exclusivity of one user to take a lease on the resource during the period of the lease; whereas nonexclusive leases allows the sharing of a resource by more than one user.

IV.2. 5 Events

One of the features of the Jini system is distributed events support. Objects may be interested in a certain event's occurrence; these objects may register their interest with the event's object and receive a notification as an event occurs.

IV. 3 The Bluetooth Service discovery protocol (SDP)

Service discovery protocol (SDP) is one of the Bluetooth[BT] protocols. Devices and services in networks can discover and gather information about other available services in the network by this process. Its intention is to allow devices in Bluetooth environments to locate available services.

In traditional networks such as LANs, a network administrator statically configures and manages these services. This is not suitable for ad hoc networks like the ones formed by Bluetooth wireless technology, as they need a more flexible and dynamic

way of connection for devices that spontaneously and predictably join and leave the network.

SDP focuses primarily on discovering services and leaves more sophisticated service functions (such as accessing or using the service) and operations to other protocols, possibly used in combination with SDP.

SDP uses a request/response model for a specific transaction, where a transaction consists of one-request protocol data unit (PDU) and a one-response protocol data unit (PDU). When using SDP with the Bluetooth L2CAP transport protocol, a client must receive a response to each request before issuing another request on the same L2CAP connection.

IV.3. 1 SDP Client–Server interaction

The service discovery protocol provides a mechanism that helps the client applications to search for and discover existing services and attributes of those services provided by the server applications.

SDP involves communication between the SDP client and SDP server entity; see Figure 4.6, where the SDP server holds a list of registry records, each describing a single service and its characteristics and the SDP client is the entity requesting or looking for services and their attributes. There is a maximum of one server in each Bluetooth device, but devices acting only as a client do not need to have an SDP server. A device containing multiple service providers can use an SDP server acting on their behalf to handle requests for services they provide, and similarly to multiple client applications where an SDP client can be used to request servers instead of the applications.

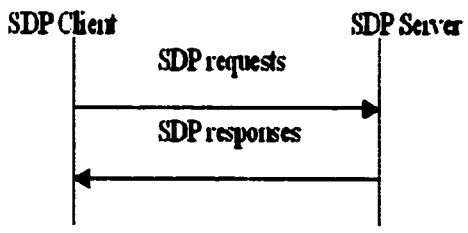

Figure IV. 6 SDP Client-Server interaction.

IV.3. 2 SDP Services

IV.3.2. 1 Service record

SDP defines two entities: a client that asks for or looks for services, and a server that provides services. Any device can be either a client or a server at a given time, acting sometimes as a service client and sometimes as a service provider. Each server needs to maintain a list of service records describing the services it provides, which is called a service registry. A service record is a standard way to represent a class of service (e.g.: faxing, printing, audio, etc) and its attributes (e.g.: for a printing service the attributes can be color, duplex and finishing capability).

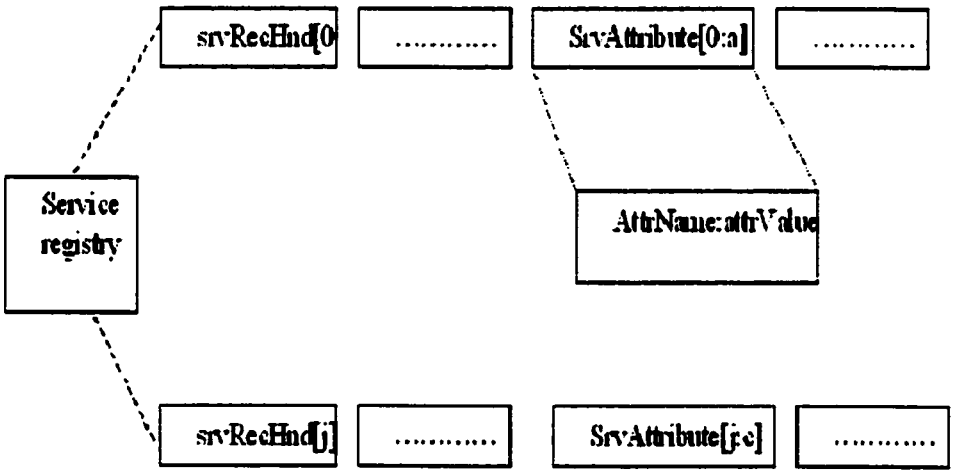

Figure IV. 7: General SDP service registry structure [MB00].

Figure IV.7 illustrates the general structure of a service registry with its constituent service records, containing a set of services each with a service record handle (srvRecHnd[0] through srvRecHnd[j]) and a set of attributes per service. Service records consist of both universal service attributes (applicable to all types of services, such as the service class) and service specific attributes (relevant only for a specific class service). The universal service attributes are all optional except two: the service class attribute

defining the class or type of service, and the service record handle, which serves as a pointer to the service record and is used by the client to access the server's service record.

A service record handle is a 32-bit number that uniquely identifies each service record within an SDP server; this handle is unique only within each SDP server. There is one record handle consistent through all SDP servers and this handle represents the SDP server itself, it has the value 0x00000000.

IV.3.2. 2 Service attribute

Each attribute describes a single characteristic of a service. Examples of service attributes are *ServiceClassIDList*, which identifies the type of service represented by a service record, and *ProviderName* specifying the organization or individual that provides the service.

A service attribute consists of two components:

- An attribute ID is a 16-bit unsigned integer that distinguishes each service attribute in a service record from other attributes within that service record. It identifies the semantics of the associated attribute value. All services belonging to the same service class assign the same meaning to each particular attribute ID.
- The attribute value is a variable length field whose meaning is determined by the associated attribute ID and the service class of the service record in which the attribute is contained. In a service discovery protocol, a data element represents an attribute value.

Example:

Attribute Name	Attribute ID	Attribute Value Type
ServiceRecordHandle	0x0000	32-bit unsigned integer

Table 1: Example of a service attribute.

IV.3.2. 3 Service class

Each service is an instance of a service class. The service class definition provides the definitions of all attributes contained in service records that represent instances of that class. A service record contains attributes that are specific to the service class as well as universal attributes common to all services. Each service class receives a unique identifier contained in a service record as an attribute value for the *ServiceClassIDList* attribute, represented by a UUID. A UUID is a universally unique identifier that is uniquely assigned across all space and all time. A UUID is 128-bit value. The *ServiceClassIDList* attribute lists the service class identifiers in order from the most specific class to the most general class. For example, a color postscript printer with duplex capability service may have a service class listed as follows [BT]:

DuplexColorPostscriptPrinterServiceClassID,

ColorPostscriptPrinterServiceClassID,

PostscriptPrinterServiceClassID,

PrinterServiceClassID

IV.3. 3 Data representation

SDP describes the data contained in an attribute value by defining a simple mechanism known as a data element. A data element is a typed data representation, and it contains two fields:

- A Data element header field, which in its turn is composed of two parts: a type descriptor and a size descriptor. A type descriptor is the (high order) 5-bits of the first byte of the data element header describing the meaning of the data. The size descriptor is the (low order) 3-bits of the first byte of the data element specifying the size of the data sequence. 0, 8,16, or 32 bits can follow it.

- A Data element data field which is a sequence of bytes whose size is specified in the size descriptor and its meaning in the type descriptor.

IV.3.3. 1 Protocol Data Unit (PDU)

A single SDP transaction consists of one PDU request and one PDU response. A PDU consists of a header followed by PDU-specific parameters (see Figure IV.8). The header contains three fields:

- A *PDU ID* identifying the type of the PDU, for example: the value 0x02 represents an *SDP_ServiceSearchRequest* PDU.
- A *Transaction ID* uniquely identifies a request PDU and is used to match it with the response PDU for that specific request. The request transaction ID and the response transaction ID must be the same.
- A *ParameterLength* field which specifies the length in bytes of all the parameters contained in the PDU.

A PDU-specific parameter may contain a continuation state parameter.

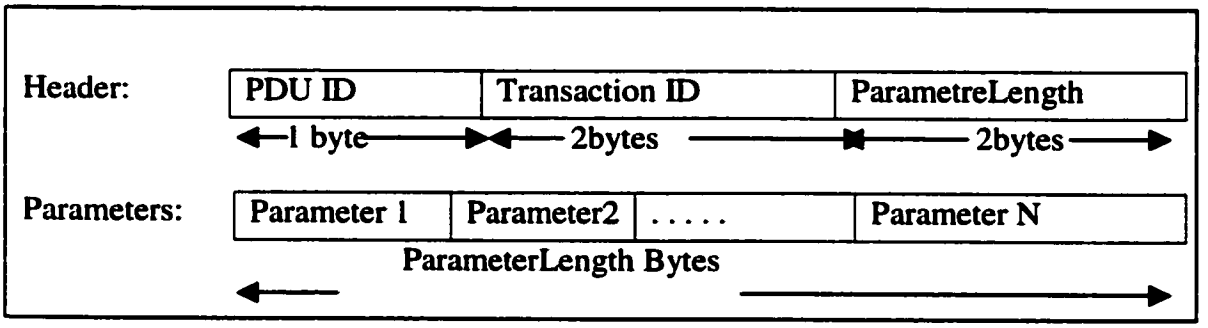

Figure IV. 8: PDU format [MB00].

IV.3. 4 Discovering a service in Bluetooth wireless communication

Discovering a service in Bluetooth wireless communication requires a simple operation; the client specifies the service(s) needed and the server replies with any available services that match the requested service. The sending request and reply in SDP are in the form of a SDP protocol data unit (SDP_PDU), where services match the

requested service sent by the client. In order to accomplish that matching, the client and the server need a standard way to represent the service(s), which is why SDP introduces universally unique identifiers (UUIDs) for representing services. A client looking for a service just specifies the UUID associated with that class of service, and the service provider matches that UUID against those services that it has, to generate a response.

The general SDP protocol flow requires only two basic transactions plus a third, which is a combination of those two. A typical SDP transaction consists of:

1. Client sends a request to search for the needed services; service provider replies with handles to the matching requested services
2. Client uses the handle(s) obtained in step 1 to generate a request to get additional service attributes for the requested service(s).

Figure 4.9 summarizes the general SDP transactions, where only services and service attributes described by UUIDs are searchable. The non-described attributes are retrievable only after the service has been located using the UUID attribute.

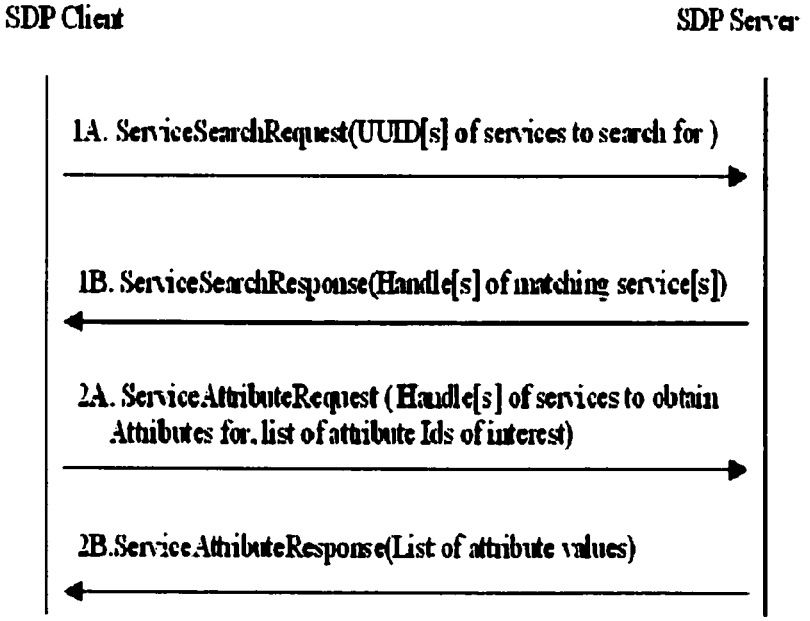

Figure IV. 9: The general SDP transaction summary [MB00].

The third transaction is a combination of those two transactions, where the SDP client requests specific attribute values from a specific service record by sending an *SDP_ServiceAttributeRequest ()* with *ServiceRecordhandle* and *attributeIDLists*

parameters. The server replies with the attribute list through *SDP_ServiceAttributeResponse()*.

IV.3.4. 1 Example of *ServiceSearchRequest* PDU

Suppose a client requests a PrinterServiceClassID represented as a data element (32-bits UUID) having a value of 0xqqqqqqqq. It sends an SDP_ServiceSearchRequest specifying a maximum of three service record handles for return by the server. The *transactionID* has a value of 0xtttt. This PDU will look like Figure IV.10:

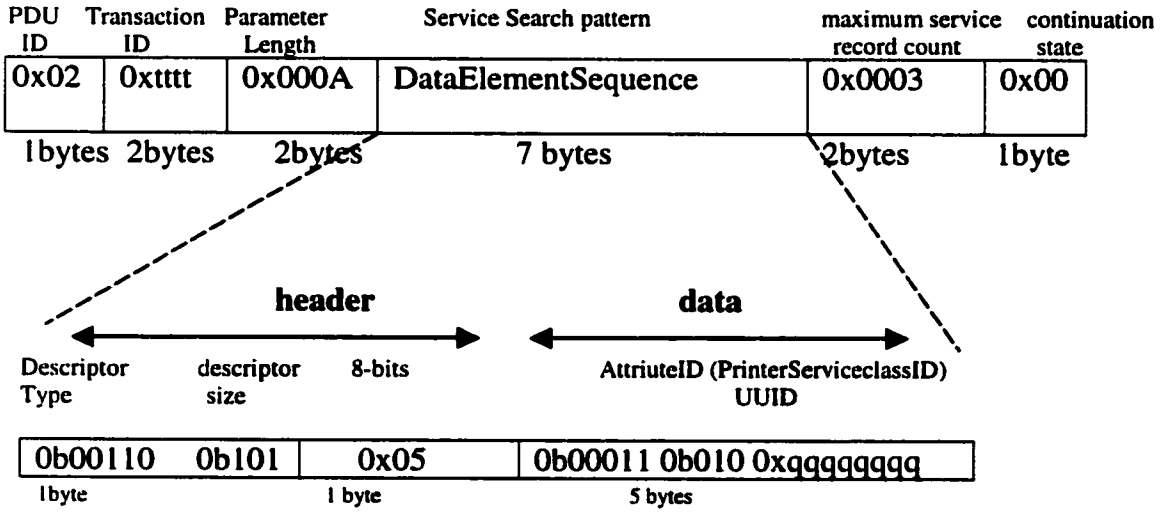

Figure IV. 10: SDP_ServiceSearchRequest PDU.

IV. 4 Service location protocol

Service Location Protocol (SLP) is an Internet standard developed by Internet Engineering Force Task (IETF) in 1997 with a new version 2 published in 1999 [SLP].

SLP provides a framework to discover and locate services in a network. It applies to Internet based networks under cooperative administrative control, which permits multicasting, and grouping clients and service providers. However, SLP is not scalable for the global Internet.

Service Location Protocol proposes a framework where a client application looking for services is modeled as a "User Agent", the entity offering and advertising the service is modeled as a "Service Agent" and a third entity acting as a repository or a cache for available services is modeled as a "Directory Agent".

Service Location Protocol has a concept of scope, which is a way to group services into administratively controlled domains. People in the Art department, for example, do not like it when someone from the Engineering department uses their printer. People in general like to reserve access to services for people in their own department. Administrative domains permit the User Agent (UA) belonging to an administrative entity to seek and see only services available in that entity domain. A UA and SA can belong to more than one scope.

IV.4. 1 Finding a service

The client known as a User Agent (UA) is the entity seeking for services by type or properties formulated as attributes. This User Agent belongs to an administrative domain defined as a scope. The purpose of having a scope concept is mainly to limit access to resources and for scalability reasons in order to reduce overloading the network with the number of replies to a single request.

SLP supports two kinds of interaction mode: one with a Directory Agent where services are stored with a URL to the service provider (User Agent), and one without a Directory Agent where the User Agent has to communicate directly with the User Service(s).

In a network supporting the presence of a Directory Agent, the UA sends a Unicast UDP or TCP service request message to the DA(s), supporting its scope, directly by having their addresses or after discovering them. The directory(s) listening to this request will respond with a Unicast reply message after finding a match between the requested service and the registered ones (see Figure IV.11).

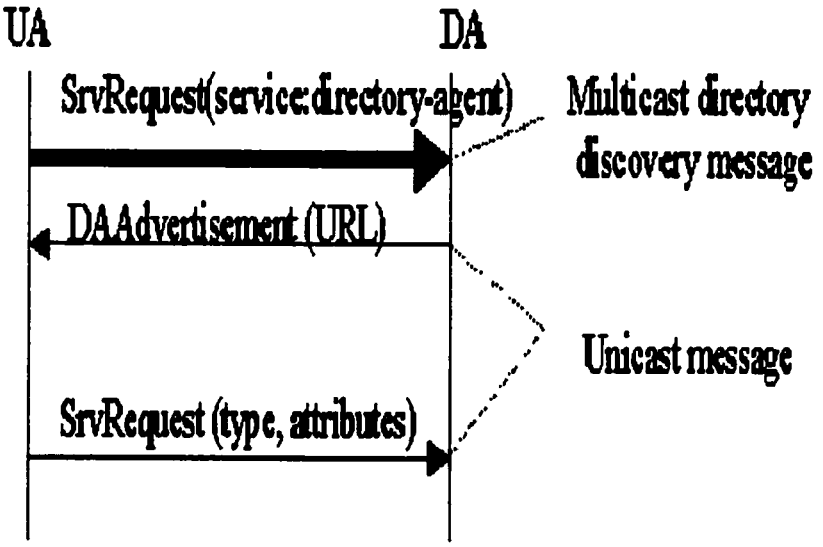

Figure IV. 11: Discovering services through DA.

If a network does not support the presence of a Directory Agent, the User Agent has to find services through the service provider, called Service Agents. In this case the User Agent sends a multicast or broadcast service request message to Service Agents specifying the characteristics required. The Service Agents supporting its scope, and listening to the request through a well-known port, will reply by a unicast message containing the requested service location; on finding a service match, the SA(s) advertise (see Figure IV.12). In SLP, each service type needs a corresponding service template. A Template defines attributes for the specific service type, intended to be readable by humans and parsed by browsing software that will match against the registered services defined by the corresponding service template.

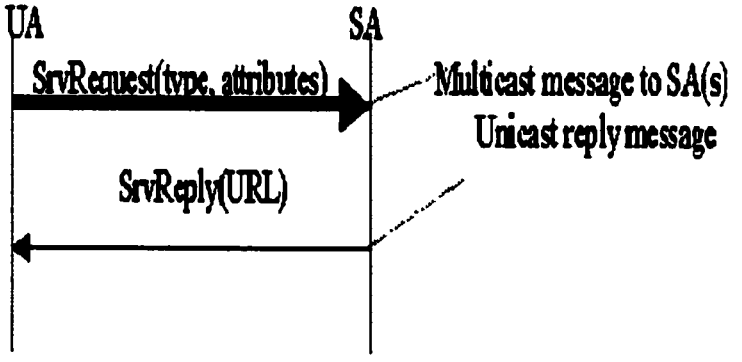

Figure IV. 12: Discovering services in absence of DA.

IV.4. 2 Directory agent discovery

Directory Agent (DA) implements a repository or a cache for service location and attribute information. It is a centralized database for service information. It exists for scalability purposes and decreases the network use thus increasing the UA's speed to obtain a requested information service. The user sends a unicast service request to the DAs (when they exist) that supports its scope. The DA in turn replies with a list of URLs obtained from the Service Agents (SA) registration. Both UA and SA need to locate the Directory Agent supporting their scope(s) through its IP address. There are two ways to find an existing DA - by active or passive discovery.

IV.4.2. 1 Active discovery

The User Agent and Service Agent multicast or broadcast a service request (*SrvRequest*) to find a Directory Agent (see Figure IV.13). Scoping occurs for all requests and services, except for a request looking for either a DA or a SA (in case of a UA). This broadcast service request will set its Type field to "service: directory-agent", which may have a zero scope-list field. DAs receiving this multicast request must respond with a unicast reply called a *DAAdvertisement* containing a URL that indicates the DA's location, for example: "service:directory-agent://foobawooba.org", service(s) SLP URLs; as well as the location information and it also gives the time to live (TTL) for the returned service(s).

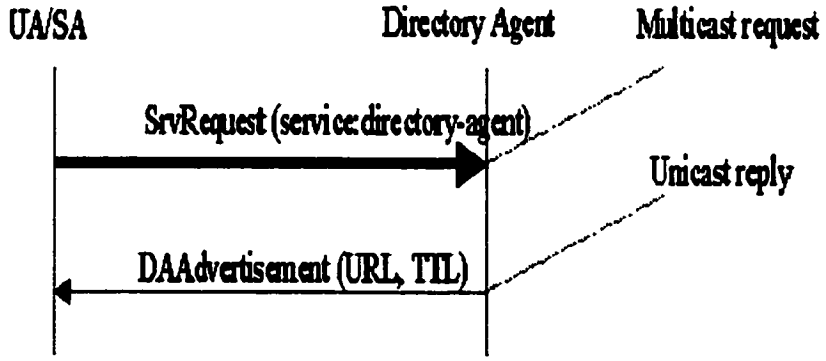

Figure IV. 13: Directory agent active discovery.

IV.4.2. 2 Passive discovery

In passive discovery the Directory agent sends a multicast or broadcast on a periodic basis to announce its presence and location through a *DAAdvertisement* message. When the User Agent and Service Agent hear the unsolicited message on a well-known port, they extract the URL included in the advertisement message and keep it for future use or when they need to contact a DA. Now the Service Agent is able to register its service with the directory, and a User Agent can request services cached by the Directory Agent through a unicast message. The DA timestamp should not be zero otherwise is not taken into a consideration by the UA and SA, because this means that the directory can no longer handle any incoming messages (see Figure IV.14).

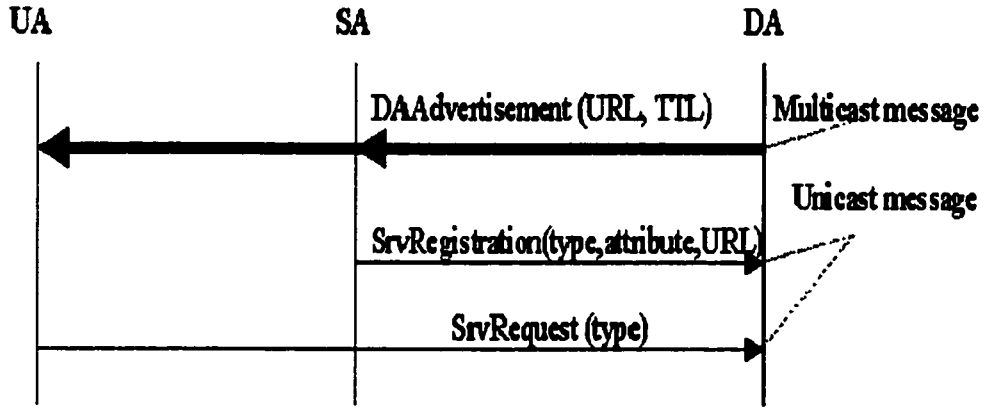

Figure IV. 14: Directory agent passive discovery.

IV.4. 3 Registering a service

Once the Service Agent finds or discovers the DA in its supported scopes, it will send a *SrvRegistration* message containing the type and attributes of the service this agent advertises or offers. The registration message will contain an important field where it defines the URL for the service that the Service Agent wants to register. Each service has a lifetime defining the availability of the service. In order to offer continuously advertised services, Service Agents should restart a new registration process before the expiration of the lifetime they used in the registration.

An example of a service registration (valid for 2 hours) is as follows:

Lifetime: 7200
URL: service:lpr://site.dsgr.ftp.ca:428/draft
Attributes: (SCOPE=ENGINEERING),
(PAPER COLOR=WHITE),
(PAPER SIZE=LEGAL),
UNRESTRICTED_ACCESS,
(LANGUAGE=POSTSCRIPT),
(LOCATION=4 FLOOR)

On receiving the registration message, the Directory Agent replies to the service user with an Acknowledgment (*SrvAck*) message. Once a DA acknowledges a service registration, it makes the information available to clients. As soon as the service is no longer available for use, the Service Agent has to deregister itself from Directory Agents with which it is registered. When the Service Agent receives an acknowledgment indicating success, it can assume that the Directory Agent no longer advertises the service.

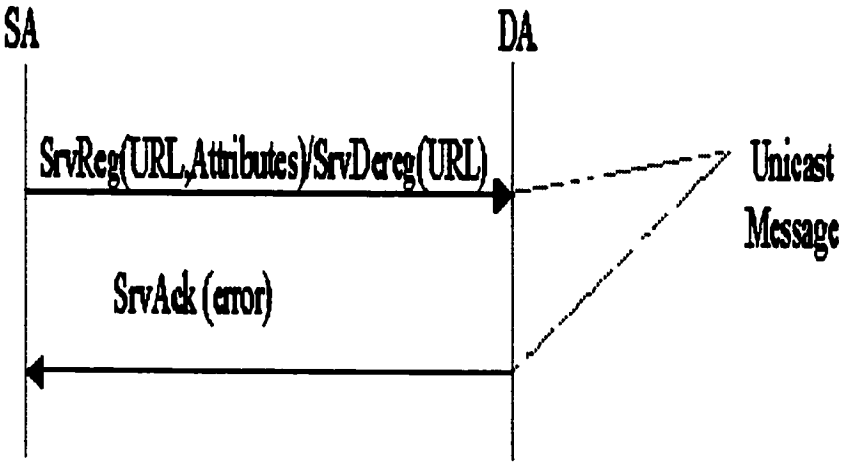

Figure IV. 15: Service Registration/Deregistration.

IV. 5 Other existing technologies

The Universal Plug and Play (UpnP) protocol [UPnP] is the product of the UPnP Forum, headed by Microsoft. The design was originally to be an extension to the Microsoft "Plug and Play" feature. In the UpnP architecture, service points are functional units and the clients are control points. Each functional unit must run a web server as an entry point for the control points. UpnP works at lower level protocol suites assuming an IP type network underneath, but it can also work with other transport technologies such as IEEE 1394 and USB. When plugged into the network, a device uses the Dynamic Host Configuration Protocol (DHCP) to acquire an IP address. In case the network does not have a DHCP server, the device uses AutoIP, which in turn uses the Address Resolution Protocol (ARP) to find an un-used IP address.

Once the device has an IP address, it starts to discover other services in the network and announce its own services. Both service announcement and discovery use the Simple Service Discovery Protocol (SSDP). The service announcement or discovery returns the URL of the service location that points to an XML document, called Description Document. This contains: the description of the service including the vendor, manufacturer and serial number; a list of actions or commands that can be called on the service; a list of state variables and, the most important, a list of URLs for controlling, eventing and presenting the service. Using the control URL, the control client can query for a list of all actions supported by the service. It can also register for receiving events using the eventing URL, and retrieve a web page from the presentation URL that might grant permission to the control point for sending actions, registering for event monitoring, and checking the device state. UpnP is language independent. In contrast to Jini, there is no central service registry. UpnP does not address the security issue, leaving it to the upper layer to take security measures when needed.

Salutation [Salutation] is the Salutation Consortium's architecture for service discovery and utilization among networked entities. The main component of the Salutation architecture is the Salutation Manager (SM) that runs on each machine, and acts as a

service broker for the services running on the machine. Salutation Managers use the point-to-point Salutation Manager Protocol to exchange the capabilities of the registered service. The Salutation architecture is network independent; the Transport Manager isolates the Salutation Manager from the underlying transport protocol. A disadvantage of the current Salutation architecture is that it supports only point-to-point brokerage service between Salutation Managers, requiring each Salutation Manager to keep track of all SMs in their network. It is language independent. Salutation can operate in any network; this transport independence is its strongest feature.

IV. 6 Conclusion

We have proposed different service discovery protocols to enhance dynamic interaction among devices/services with less human intervention. The comparison is not really fair in sense that each uses a different approach for service discovery and gives a different weight for the different addressed concepts. Because Bluetooth SDP supports limited functionality compared to other protocols, it is not considered a suited candidate. Table 2 presents a quick summary of the service discovery protocols presented above.

Feature	Bluetooth	UPnP	SLP	Jini	Salutation
Main Entities	Bluetooth server, and client	Control point, devices	Directory Agent, Service Agent and User Agent	Lookup service, client and server	Salutation manager, client and server
Service Repository	No	No	Directory Agent	Lookup service	Salutation manager
Service Announcement and multicast	No	Advertisement	Discovery/ Service Registration	Discovery /join	Registering with local manager
Service Discovery and access	Service class, service attribute or browsing. No access	Contact control point/ listening. Invokes actions on the service	Contact Directory Agent/ multicast SUs. No clear defined access	Contact Lookup service. Use service proxy based on RMI	Query Local manager. Service session management
Security	Yes	No	Yes	Yes	Yes
Service description	Service record attributes	XML description	Template: Service type and attributes	Interface type	Functional unit and attributes

Table 2: Some features of the Resource Discovery Technologies.

Chapter V
The Proposed Architecture

Small and powerful end-user devices are getting more and more ubiquitous and many new sophisticated multimedia services are also becoming widely available. The requirements of the user in communications in terms of flexibility, availability and quality of service (QoS) are increasing as well, making it necessary to provide a user with new telecommunication infrastructures that permit the user to access any available device next to him.

V. 1 Introduction

Experiencing the different concept of mobility, which includes nomadic users, device mobility, and session mobility, we have seen in Chapter II different projects [Bei et al 98], [Pha et al 00] involving mobility and service location concepts. But few, if none of them, have invoked the idea of having the user always connected and using QoS concept for session negotiation. Our architecture is an extension to the one proposed in [EHB00]. We propose a new concept of mobility taking into consideration QoS parameters defined in the user preferences and the device capabilities.

In previous work [EHB00], they suggested that each user uses the service of the Home Directory to store his profile. This profile contains information about how to handle incoming call requests for communication sessions and the set of possible devices, which could serve for receiving these calls. This information could also be used for making the best choice of devices depending on the user's preferences, the session requirements and the capability of the available devices. But as the set of available devices becomes more dynamic, automatic update of the user profile becomes a necessity. A mobile user that moves from one location to another will always find himself in the presence of a set of different telecommunication devices that he may use to

receive a call or access information. A user would have to contact the home directory and change his profile if he wants to make use of new devices.

Imagine for example an office telephone present in the user's temporary new location. To receive calls through this telephone, the user has to contact his Home Directory (HD) and add the new phone number to his user profile. Now the user can receive calls in the new location.

We propose a new scheme that permits the user to access services or information through any device that is present in his (dynamically changing) working area, by giving him a simple and flexible way to do so. We introduce a personal proxy agent, which may be included in any telecommunication device with the capability to participate in an ad hoc network, such as a PDA. The Personal Proxy Agent (PPA) will be the bridge between the user and the HD, so that authentication and profile update are transparent to the user. The PPA contains the user profile obtained from the user's HD and the device profiles for all devices in area around the user.

The Personal Proxy Agent detects the set of devices present in a specific location, so that it can receive information about their (device) profiles. When the PPA receives an incoming call for the user, it will select the appropriate device based on the user's profile and the profiles of the available devices, to receive the call.

V. 2 Home directory and user profile

The home directory is a personalized database, which may include the information of user location, contact information, the current user access information, user preferences [EHB00] for the sessions, policy for call forwarding, presence service, user authentication, authorization and accounting. The home directory is an important component for providing personalized network services. It stores the customized user profiles for the registered users. By registering with the Home Directory, the user can define the policy of call processing depending on the class of the caller, the time period and even the payment method. By this way, the incoming call will be processed in different ways according to policies defined by the caller and callee. The user can also define his preference for QoS, cost, reliability and security levels for every kind of

service mentioned in the user profile. Therefore, his preferences will be considered for the QoS negotiation when he receives calls or when he requests a service. An example of a typical user profile is show in Figure V.1.

Personal Identification Name: Peter Alleyene Employer: University of Ottawa Email peter@site.uottawa.ca Phone Number 613-562-5800	Callees Callee rami@wam.umd.edu QoSselectionWeight 7.5 PriceCeiling_CentsPerMinute 30 AudioPreferences AudioWeightFactor 7 ReceiveAudio Yes SendAudio Yes Min Acceptable Telephone Quality Ideal CD Quality VideoPreferences VideoWeightFactor 1 ReceiveVideo Yes SendVideo Yes FrameRate Min Acceptable 10 Ideal 30 FrameRateWeightFactor 5 FrameResolution Min Acceptable 320x240 Ideal 800x600 FrameResolutionWeightFactor 2 Device to try : Home telephone, Cellular Phone, Answering Machine Callee john@nortelnetworks.com Publishing Preferences ..... Receiving Preferences ..... Device to try : Office Telephone, Cellular Phone, Answering Machine 1 Callee www.sportsnews.com ..... Callee www.newsondemand.com .....
Devices information Home telephone Network Address: 1-613-521-5555 Permission to: ALL Office Telephone Network Address: 1-613-562-5800 Permission to: ALL Cellular Phone: Network Address: 1-613-286-1052 Permission to: rami@wam.umd.edu, john@nortelnetworks.com Answering Machine Network Address: 1-613-521-5556 Permission to: ALL	

Device usage Preferences Time: 8 am- 5pm Office-Phone : 613-5625801 WorkStation: 137.122.20.102 :7512 Time: 7pm- 11pm Home-Phone: 819-7702266 Laptop : 137.122.40.30 :7512 Time: any Cell-phone: 613-2236598	

Figure V. 1: User profile.

Each user in the system is required to create a profile containing his customized information stored in his Home Directory. When a user subscribes to a domain, he enters all the basic information in the profile. The HD provides a template profile, and users can put additional information as they acquire new devices and change their preferences. Security keys for authorization and authentication purposes are also assigned at this point, though they can be also updated later. The user will then be assigned a universal distinguishable identifier to identify the user. When the HD supports the SIP [SIP] protocol, the user will be assigned a universal SIP address that would then be used to contact the user.

V. 3 SIP overview

The Session Initiation Protocol (SIP) [SIP] is IETF's standard peer-to-peer signalling protocol for multimedia conferencing over IP. It is an application-layer control that creates, modifies and terminates sessions with one or more participants, for applications such as Internet multimedia conferences, Internet telephone calls and multimedia distribution.

Caller participant initiates a session by sending a SIP invitation to a callee. This session may carry session descriptions allowing participants to agree on a set of compatible media types. SIP supports user mobility by proxying and redirecting requests to the user's current location. Callers and callees are identified by their unique SIP addresses having a similar form as a mailto or telnet URL, i.e., sip:userID@host. The SIP

protocol has two components: a user agent client application (UAC) that initiates a SIP request, and a user agent server application (UAS) that contacts the user when a SIP request is received and that returns a response on behalf of the user. A SIP end point can play both roles UAC and UAS, but not at the same time for a single transaction. It groups the SIP network into clients and servers. A user registers with a registrar server using their assigned SIP addresses. A successful SIP invitation made by a caller to a callee, is when receiving a "SIP:200 Ok" message back from callee. The invite request is send to a SIP server asking a callee to join a particular two-way conversation or a session. The request includes the address of the caller and the address of the intended callee. After the callee has agreed to participate in the call, the caller sends an Ack message to confirm the call. And if the caller no longer wants to participate in the session, it sends a Bye request. A single conference session (or call) involves one or more SIP request-response transactions. SIP messages are coded in text format, which facilitates implementation in languages like Java.

V.4 Device profiles

In order to make the selection of the device for a communication session that best fits the requirements of the user, the properties and capabilities of each device have to be collected and represented in a certain well-defined format. Work on describing the properties and capabilities of devices attached to the Internet are underway in the World-Wide Web Consortium (W3C) [W3C] and the Wireless Application Protocol (WAP) Forum [WAP]. The work defines mechanisms for describing and transmitting information (metadata) about the capabilities and properties of resources on the World Wide Web. The Composite Capabilities/Preference Profile (CC/PP) describes the way to represent these capabilities/properties using the Resource Description Framework (RDF). The H.245 [H.245] protocol used by H.323 [H.323] defines messages that cover receiving and transmitting capabilities, so that the capabilities of each terminal to receive and decode will be known to the other terminal.

A profile for a device connected to the Internet may cover the hardware platform (CPU model, the size of the memory, microphone, speaker, camera,), system software

(operating system, list of audio and video encoders,...) and applications (JMF, Vic, Vat, whiteboard,...) available on the device. An example of a typical device profile is show in Figure V.2.

Hardware		
CPU		AMD Athlon
MemorySizeinMB		128
VoiceInputCapable		Yes
VoiceoutputCapable	Yes	
VideoInputCapable		Yes
VideooutputCapable	Yes	
Max Resolution	1600x1200	
Software		
VideoCodecs		
		MPEG-1
		MPEG-2
		H.261
		H.263
AudioCodecs		
		MP3
		G.711
Applications		
		JMF
		Vic
		Vat
		WhiteBoard

Figure V. 2: Device profile.

V. 5 Proposed architecture

The previous architecture for personal mobility [EHB00] is based on the concept of a Home Directory. The Home Directory Agent (HDA) executes the logic in the user profile every time the user receives an incoming call through the HDA. The user can also invoke the HDA in order to find (and establish) the best way to place an outgoing communication request.

In a ubiquitous computing environment, the set of available devices for the user may change continuously as the user changes his location. Updating the information about currently available devices is not a feasible solution since in ubiquitous environments, devices do not have to be visible to the user, and the set of available device

changes continuously as the user moves from one location to the other, making it unfeasible to the user to manually update his/her profile.

To overcome these limitations, we propose to run a modified version of the HDA on a hand-held device, such as a PDA, that is carried by the user. We call this modified version of the HDA the Personal Proxy Agent (PPA), and it is responsible for detecting devices in the vicinity of the user as well as his/her managing communication. We require that the hand-held device, on which the Personal Proxy Agent runs, has access to the Internet in order to retrieve the user profile and send/receive communication requests to/from the HDA. The PDA is also supposed to be able to construct a Wireless Personal Area Network (such as Bluetooth [BT]) in order to be able to detect and communicate with other wireless devices and services just around the user.

At any one time, either the HDA or the PPA is providing personal mobility service to the user. When the PDA is switched on, the PPA starts and contacts the HDA to retrieve the user profile. From that point on until the PDA is switched off, the PPA is responsible for executing the logic in the user profile. The HDA would act only as a proxy for incoming call requests between the caller and the PPA, forwarding all requests to the PPA. The HDA can detect that the PPA is not running or the PDA is currently out of reach when a reply to a forwarded call is not answered after a certain time-out period. The HDA would then switch into active mode, and handle the request according to the rules specified in the user profile.

Our architecture is designed to be open and not restricted to a specific technology or protocol, as shown in the reference model of Figure V.3. The Personal Proxy Agent can run on top of any location service protocol (Jini [Jini], Bluetooth SDP [BT], SLP [SLP], etc ..) to find the services surrounding the user, and use any wireless communication technology (IEEE802.11, Bluetooth, etc..) to build a WPAN.

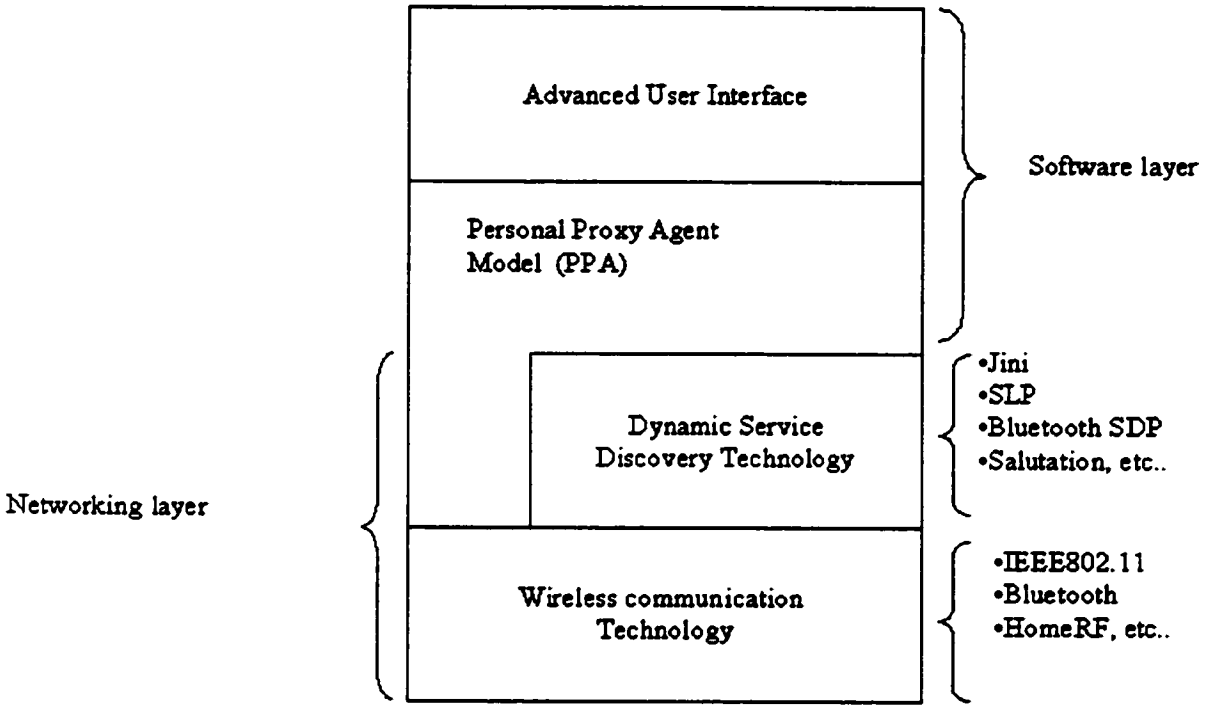

Figure V. 3: Reference model.

V.5. 1 Fundamental concepts

To control and manage the incoming and outgoing communication requests of the user, the PPA is required to perform a certain number of functions. Those functions consists of retrieving the user's profile from his Home Directory, building a local device/service registry that contains user's surrounding devices capabilities offering specific service or services and finally, select from those devices the appropriate device or devices that satisfies the session requirements. To provide the appropriate selection, a decision has to be made concerning (a) the service(s) (camera, speakers, microphone...)

that should be used among the discovered services that meet the requirements of the communication session, and (b) what Quality of Service (QoS) parameters (frame rate, frame resolution, audio quality...) should be selected for the quality of the presentation of the media. These decisions are affected by many factors including the media type of the exchanged content, the hardware and software profile of the devices where the services are running (also called device limitations), the status of the user, and finally the properties, preferences, and privileges of the end user (included in the user's profile).

The PPA is aimed to satisfy three main objectives:

- *The user is always connected:* We want to keep the user in reach wherever he will be and whatever his PAN is composed of. As soon as the PPA is launched, it sends a request to the HDA asking for the user's profile. The HDA changes then its state to a passive mode in which it will play a role of a proxy between a caller and the PPA by forwarding the accepted incoming call requests to the PPA. This leads us to the conclusion that the PPA is preferred to be running on a handheld device (PDA, smart phones, ets..). But because of the limitations of those small devices, we propose to keep only the information from the user's profile for accepting calls and the QoS information to determine the quality of the presentation of the media (frame rate, frame resolution, audio quality,..etc) , instead of having the whole user profile content copied from the user's HD. Therefore an incoming call that should be rejected based on the user profile, could be directly rejected by the HDA without being forwarded to the PPA. The HDA does not switch to active mode unless the PPA has failed or is out of reach and does not reply to a forwarded message for a certain time period. In this case it will handle the request according to the rules specified in the user profile.
- *Discovery of services and devices:* Independently of the underlying network technology and the service discovery protocol, the Personal Proxy Agent builds a local device/service registry that contains information about the capabilities of the devices present in the vicinity of the user. There are two extreme strategies to collect information (a) collect all the time, (b) collect when needed. The agent may also use intermediate strategies. The device/service registry is used to merge

the capabilities of different devices and connect to those devices that match a specific session. To build the device/service registry, the PPA has to interact with the surrounding devices and detect/retrieve their addresses and capabilities. To do so, a wireless communication technology should be used. Now, the PPA may use these addresses to request a specific service(s) required by a new communication session.

- *Selecting devices and building compound machines:* When the PPA receives an incoming call through the HDA, and assuming that the user is willing to accept calls now, the first step in the selection procedure consists of selecting the services that are required for establishing the communication session. Based on the session requirements. For instance, a communication session that includes the receipt of audio media requires the discovery and selection of an audio playing service.

  The second step in the selection procedure consists of selecting the QoS parameter values for each service, such as the audio quality for an audio service and the video frame rate and resolution for a video service. This selection is based on the hardware/software limitations of the device, and the preferences of the user. For instance, the network access line of one of the devices may limit the maximum throughput (e.g. modem or wireless connection). Also, a video content that requires a high-resolution display eliminates the possibility of using the display service of a monochrome device, especially if the user expressed his desire to receive only high quality video. This selection process also deals with the case when multiple similar services exist in the environment of the user. While these limitations depend on the capability of the device where the service is running, the user may impose other limitations by establishing a maximum cost per minute (which may depend on the effective network throughput used).

  We base the selection of QoS parameters for each service on the concept of maximizing the user's satisfaction as defined in [EHB00], where the concept is using all possible combinations of QoS parameters of all available services, and select the combination that generates the maximum satisfaction within the restrictions of the device and the preferences of the user. The PPA might also

require mixing and matching several services to satisfy the requirements of the session.

V.5. 2 Architecture of the PPA

The Personal Proxy Agent is composed of three major components: a Service Discovery Agent, a User Activity Monitor Agent (UAMA) and a QoS Selection and Negotiation Agent (QSNA). Figure V.4 shows the architecture of the Personal Proxy Agent with its components. Following is a detailed description of each of these components.

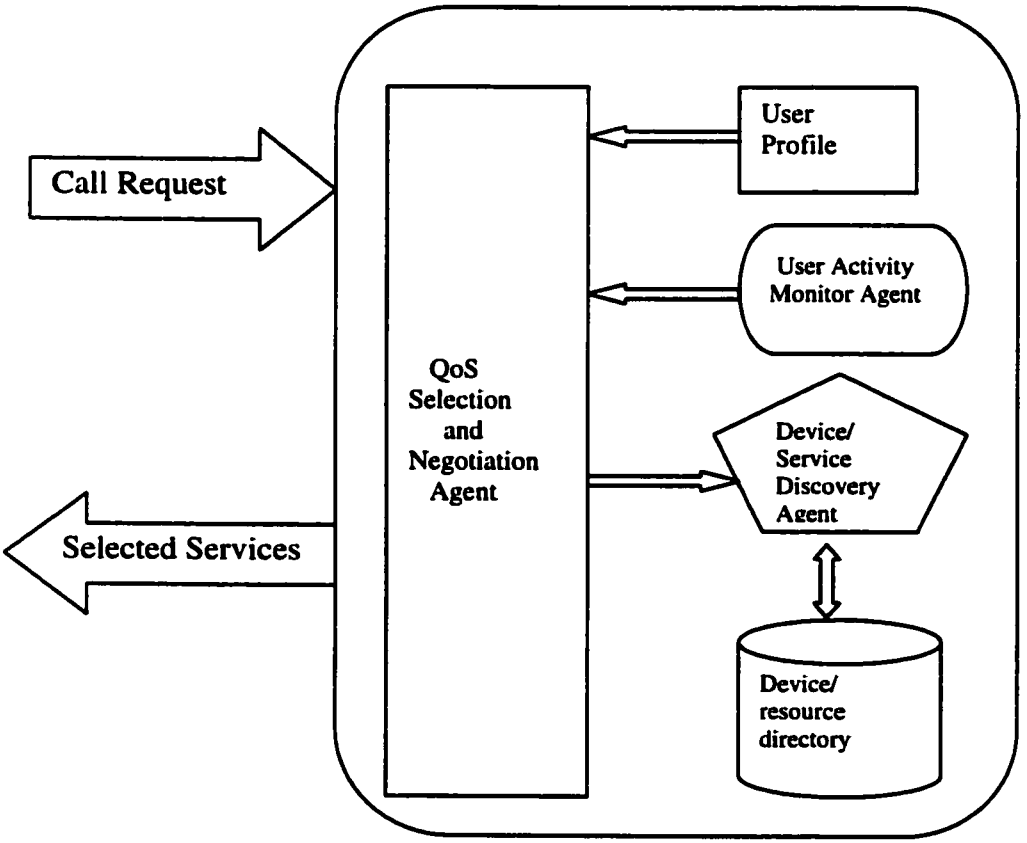

Figure V. 4: Personal Agent Architecture.

- *Device/Service Discovery Agent*: the Device/Service Discovery Agent (DSDA) plays two roles: First, it detects devices in the user's Personal Area Network. We suppose that the surrounding units have a wireless communication technology as Bluetooth [BT] or IEEE802.11 [802.11]. Second, it acts as a service discovery client that searches for specific service(s) in the WPAN on QSNA demand. The DSDA provides the QoS Selection and Negotiation Agent (QSNA) (discussed below) with the list of currently available requested services to the user, which are stored in a *Device/Resource Directory*. This directory contains information about different devices present in the user vicinity, their capabilities and services they provide. This information is updated each time the QSNA needs specific service(s) to satisfy the session requirements and user preferences. This way, we limit the registered services to the needed resources instead of having all resources, most not needed, stored in the directory. It is a way to keep the directory small in case there are a lot of devices and services in the user's PAN. Since different services might be discovered using different service discovery protocols (JINI [Jini], SDP [BT], SLP [SLP], UPnP [UPnP]), the DSDA acts as a client in all service discovery protocols. For service discovery protocols that support a central lookup service, such as JINI, the DSDA may query the lookup service for requested service(s) available for the user. For service discovery protocols that do not support a central directory service, such as the Service Discovery Protocol (SDP), the DSDA broadcasts a service discovery message to find information about the requested service(s).
- *User Activity Monitor Agent (UAMA):* The UAMA keeps track of the current activity of the user (idle, busy,...) as well as the services he is currently using. The tracking process assists the QSNA (discussed below) during the service selection phase by providing up-to-the-minute information about the user's status. The UAMA can also incorporate information about the context of the user, such as his location, and whether he is by himself or surrounded by other people, as suggested in [Bei et al 98].
- *QoS Selection and Negotiation Agent (QSNA):* based on information in the user profile, the service availability provided by the DSDA, the user's current activity

provided by the UAMA, and the session requirements, the QSNA selects the best services that satisfy the session requirements, and complying to the preferences of the user. The QSNA might combine and match several services to satisfy the requirements of the session making them act as one service.

V.5. 3 Usage scenarios

In this section, we present a usage scenario as an illustration of the function of the Personal Proxy Agent (PPA). We will assume that the SIP [SIP] signalling protocol is used to establish and maintain a communication session. And the user profile will contain only the needed information as explained before. Therefore an incoming call that should be rejected based on the user profile, could be directly rejected by the HDA without being forwarded to the PPA. On the other hand, we assume that all devices present in Alice and Bob's environment are Bluetooth devices. Security and authentication are out of the scope of this paper, however, the approach described in [DBC01] could be used. During all the period when the PPA is active or responding, the HDA should be in inactive state, allowing the PPA to be the session manager. As long as the Personal Proxy Agent is active, the user's HA acts as a SIP proxy server.

In the scenario of Figure V.5, Alice is working in her office, surrounded by her laptop, cell phone, station, headphone, ..etc, and she tries to reach Bob who decided to stay home. Bob in his turn is surrounded by a TV, stereo system, PC, fixed home phone, speakers, ..etc. Both, Alice and Bob, carry their PPA running on a handheld device. As soon as the PDA is switched on, the PPA will send a request (messages 1, 3) to the HDA asking for the user's profile. The HDA, in turn, replies (messages 2, 4) returning the necessary information contained in the profile. Alice's PPA detects and searches for services that can satisfy the requirements of a multimedia session and Alice preferences (messages 5, 6). Once it is found, the PPA updates the registry (message 7), then selects the device(s) offering the service (message 8) and finally sends a SIP invite message to Bob's HA (message 9). Bob's HA forwards the SIP invite (message 10) to Bob's PPA, which will search for devices/services (message 11,12) that satisfy the requested multimedia session and Bob's preferences. The PPA selects (message 14) the required service(s) that satisfies the requested multimedia session and Bob's preferences, and then

replies with SIP 200 Ok (message 15) that will be forwarded to Alice's PPA (messages 16, 17). Finally data can be exchanged between Alice and Bob through the respectively selected device(s) (message 18).

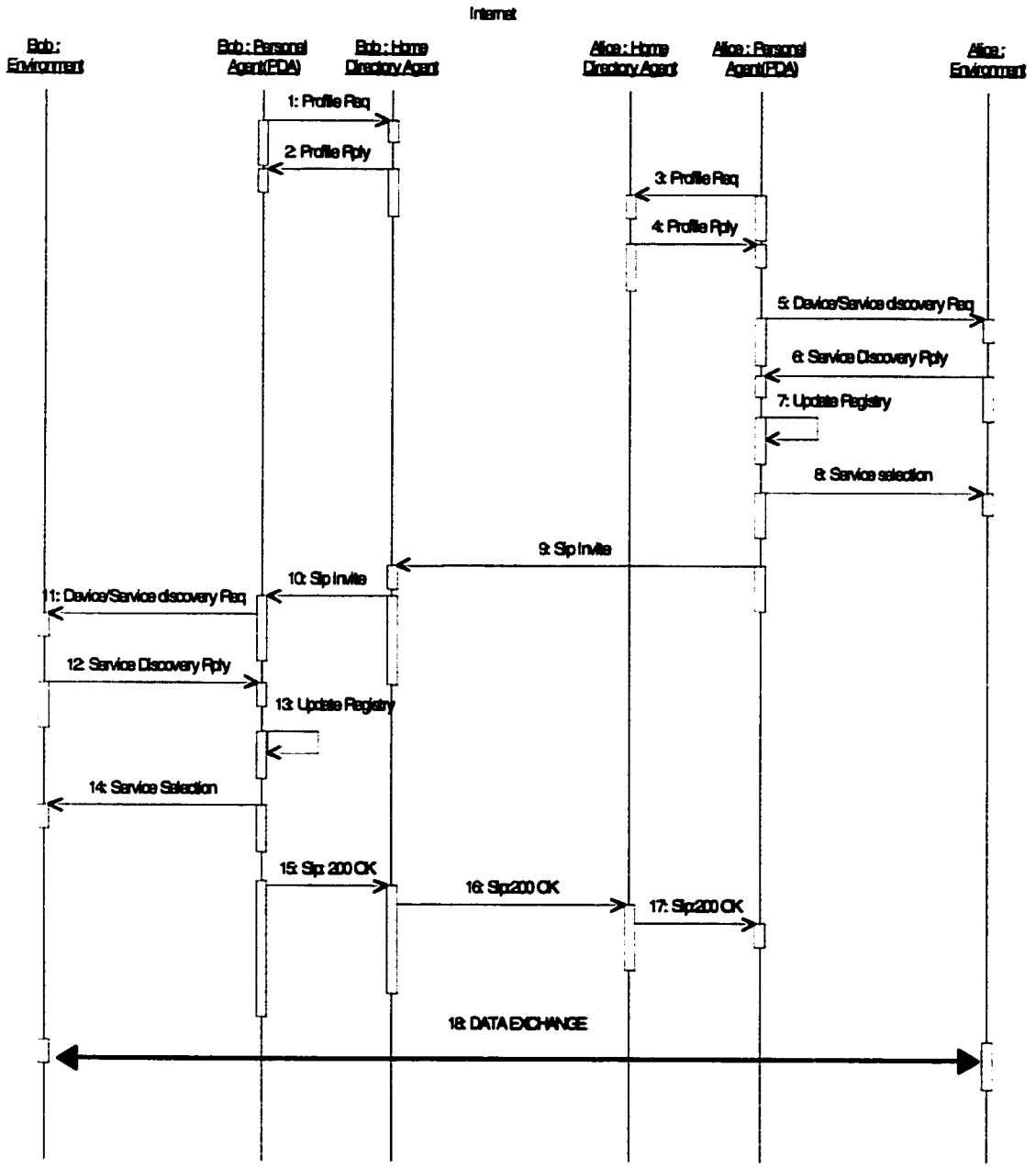

Figure V. 5: Session establishment based on the Personal Proxy Agent.

The above scenario gives the basic idea of how Personal Proxy Agents play their role in establishing a multimedia session. The next paragraphs and sequence charts will

explain in details the operations at the two different sides: the call requesting side and the call receiving side.

V.5.3. 1 Incoming request scenario

In this scenario (see Figure V.6) Bob receives a call from an outside caller Alice. Messages (1,2) are for retrieving the user's profile as explained before. When Alice sends a SIP Invitation for a multimedia conference session (message 3), Bob's Home Directory (currently in passive mode) forwards the invitation to his Personal Proxy Agent (PPA) (message 4). At the reception of the invitation request, the PPA checks the user status, i.e if busy or not. If the user is idle, the Device/Service Discovery Agent (DSDA) in the PPA will be queried to find out about the devices around Bob and the services they offer (message 5, 6, 7, 8), and then update the registry (message 9). Once the wanted service has been found, in this scenario Bob's PPA selects a TV and a computer to satisfy the requested multimedia session requirements and Bob's preferences, as frame rate, audio quality, ..etc (message 10, 11). In this scenario we assumed that Bob's TV does not have an IP connection; this is why the computer has been selected to play a role of a router. Now the selection has been made, the PPA is ready to use the service(s) and informs Bob's Home Directory by sending a SIP 200 OK reply that will be forwarded to Alice (message 12, 13). Finally the data can be exchanged between Bob and Alice though the selected device(s) (message 14) where the computer and the TV communicate through wireless technology, such as Bluetooth, IEEE802.11 or HomeRF [HomeRF] (message 15).

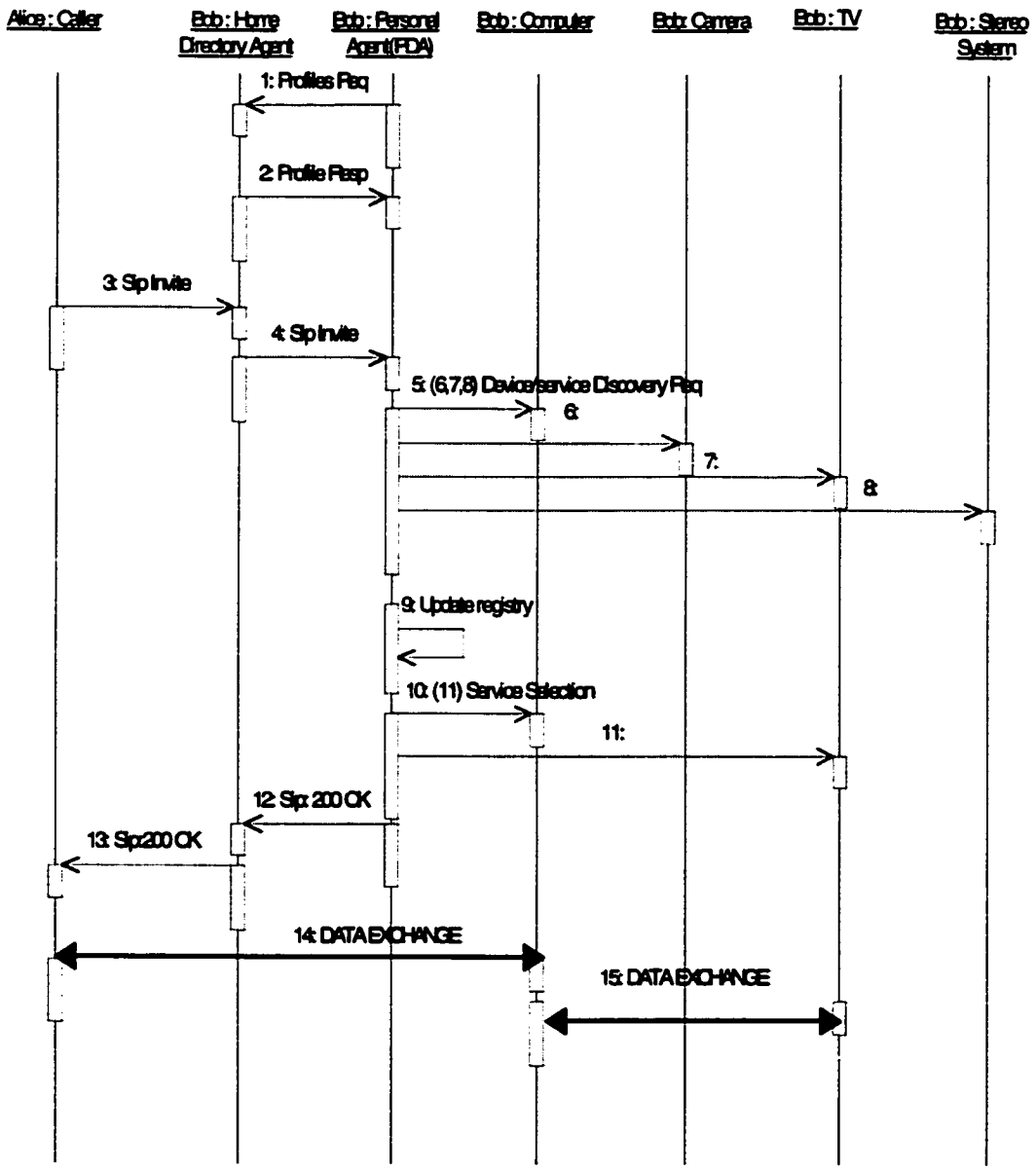

Figure V. 6: Session establishment for incoming request based on PPA.

V.5.3. 2 Outgoing request scenario

In this scenario (see Figure V.7) Alice wants to make an outgoing request through her PPA running on her smart PDA. Her PDA requests the profile from her HD as explained before as soon as it is switched on (message 1, 2). Alice requests an Audio/Video session, for example an oral presentation stored in a specific server from her PPA. Her PPA's DSDA will detect devices and services they offer, and store them in the registry (message 3,4,5,6). The QSNA will select the service(s) that satisfies the Audio/Video session requirements, based on Alice preferences and the queried devices capabilities. The selection function in [EHB00] can be used. The PPA is ready to send a SIP Invite message to the Audio/Video server requesting the presentation (message 7). The server replies with a SIP 200 OK message (message 8). Now, that we know the QoS the stored video provides, the selection process can take place. Alice's computer has been chosen for this requested Audio/Video session (message9). Finally the data can be sent between the server and Bob's computer (message 10).

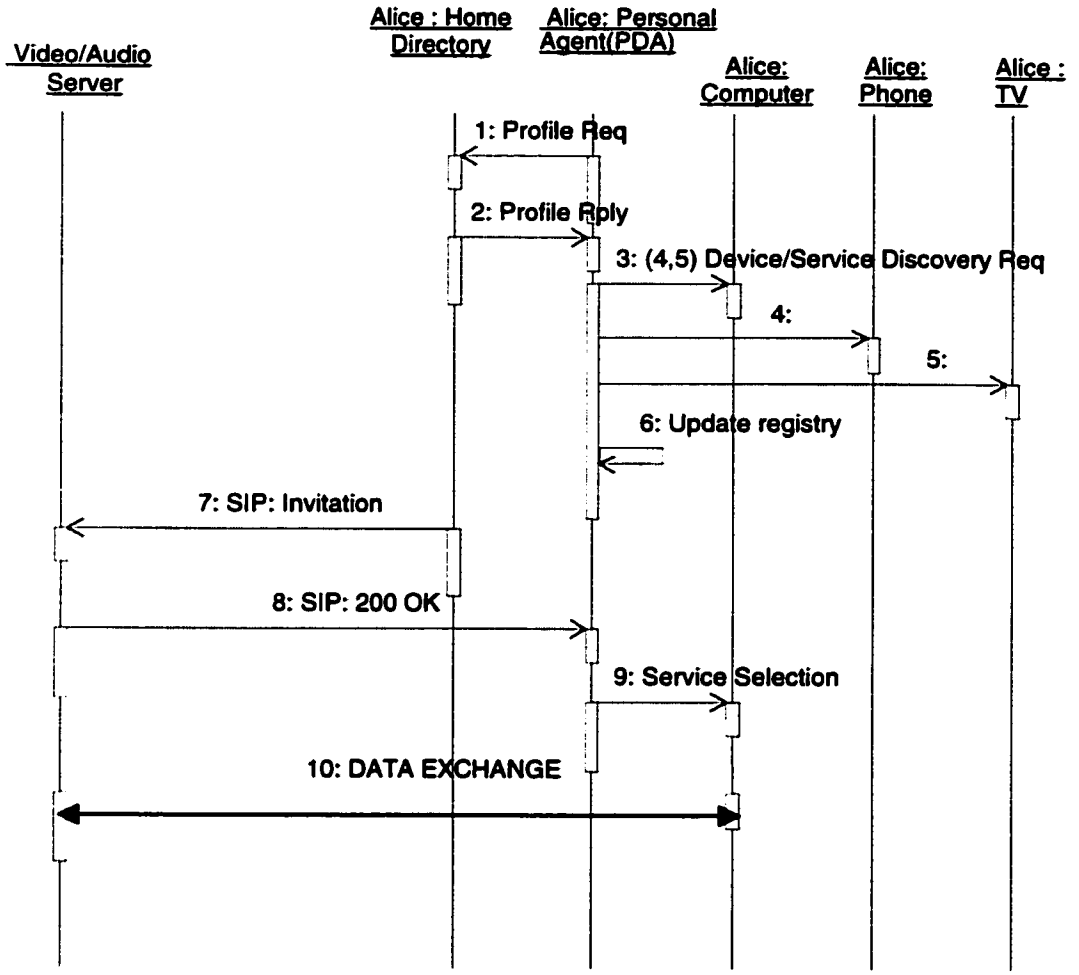

Figure V. 7: Session establishment for outgoing request based on PPA.

V. 6 Conclusion

Providing personal mobility in ubiquitous environment has a great impact on the way we perceive personal communication. We have presented here our work on supporting personal mobility in ubiquitous environment. The architecture presented here is at the intersection point of several research areas including mobile multimedia application, personal mobility, wireless communication and ubiquitous computing. This

architecture allows nomadic users to benefit from the availability of large number of devices and services in a ubiquitous environment to establish communication sessions.

Chapter VI
Prototype Implementation

As a proof of concept of our proposed architecture explained in chapter V, we implemented a prototype that can realize a basic scenario that demonstrates how our proposal works. We used Bluetooth [BT] kits as a wireless network technology and the Jini [Jini] protocol for service location.

VI. 1 Implementation choices

VI.1. 1 Protocol for service location

Different service discovery protocols exist and they all use different approaches for service discovery and give different weights for the different concepts addressed. Some of those protocols have been described in Chapter IV. We compared mainly three protocols: Service Location Protocol (SLP)[SLP], Jini [Jini] and Bluetooth [BT] Service Discovery Protocol (SDP), in term of the application programming interface (API) they offer for service representation and service registration, to see which one is more suited for our architecture. Bluetooth SDP [BT] has been designed specially for Bluetooth environments, it supports limited functionality compared to the other protocols and addresses primarily the discovery service. It does not provide access to services, does not allow service advertisement, or service registration. Therefore we had to look for another protocol that can overcome these limitations. Comparing SLP and Jini, the two widely used protocols, we finally choose Jini for the following reasons. Jini technology is based on the Java language, it uses very simple techniques to overcome problem of distributed

service discovery. It is simple, open source and ready to be used compared to SLP. The developers already provide the Jini main classes, which is not the case for SLP.

VI.1. 2 Technology for wireless networks

Wireless communication exists in different forms, as radio frequency, Infrared, satellite and others. One way to classify wireless communication technologies is by the geographic area they cover. In our architecture we need to build a wireless Personal Area Network for a user, covering distances on the order of maximum 10 meters or less and used typically to connect various personal portable devices without using cables. We looked to the different technologies that exist, as explained in Chapter III to find a one that satisfies our requirements. We focused mainly on Bluetooth and IEEE802.11 [802.11]. IEEE802.11 technology is being deployed widely for WLAN applications permitting a degree of mobility and eliminating the wires used to connect to networks. It has a nominal range of 100 meters. And because WLANs is used for high capacity data communications, they have quite high data rates, up to 11Mbps. The devices used in this technology need a robust computing platform and power supply. It tends to be better suited for networks where the participants move infrequently, which makes it less suited for small hand held devices like pagers, PDAs and mobile phones. Bluetooth technology is more suited for short ranges and low power communication networks. It has a nominal range of 10 meters with a data rates up to 1 Mbps. Its low power consumption makes it more portable for small, mobile, battery-powered devices as PDAs, pagers and mobile phones. It is true that Infrared technology covers even smaller ranges, between 10 to 80 cm, but it is not really suitable for ubiquitous networks mainly for two reasons: first, two devices or units have to be almost directly aligned to communicate (each device is aware of the other). Secondly, common materials as walls, plants or anything else can block or affect the transmission. This made our choice simple and easy: we adopted Bluetooth as the wireless technology.

VI.1. 3 Telephony protocol

In our proposed architecture, we assumed that all requests are through a telecommunication protocol for architectures that support multimedia conferencing over IP. SIP [SIP] has been proposed for its simplicity and for its availability as free source code in C, C++ and Java. It is a protocol that uses requests and responses to establish communication among various components in the network and establishes conferences between one or more endpoints.

The only free SIP implementation source code found in Java was from NIST[1], that implements the JAIN SIP [Jain] (standard Java Interface to the Session Initiation Protocol) specifications from Sun Microsystems. They provide SIP examples that explain how the SIP user agent client and SIP user agent server interact. The user agent client is the application that initiates the SIP request (invite), and the user agent server is the one that receives a request and returns a response in behalf of the user. We used the user agent examples, since we try to reuse as much code as possible.

VI.1. 4 Implementation language

For simplicity, we wanted to use the same and uniform programming language over all points of the implementation. As we had already chosen the protocols to be used in our implementation, the language that may be compatible for almost all the reused components for our architecture is Java, since Jini and also JAIN SIP use it.

VI. 2 Main components of the implementation

To implement our architecture, we used two Bluetooth kits[2] attached to two different computers, and a Bluetooth card[3] connected to a laptop. See Figure VI.1.

[1] National Institute of standards and technology
[2] See Appendix A
[3] See Appendix B

Figure VI. 1: The used units.

The laptop has a Personal Proxy Agent (PPA) running on it. The PPA includes a *SIP user* Agent for handling the telecommunication requests over IP, a *BT client* (Bluetooth) that gets the IP addresses of the Bluetooth devices in the nearby (detection process) and a *Jini client* that uses these addresses for service search process through the Internet using a unicast lookup discovery[4]. Whereas the two other computers run two Jini components: a *Jini server* that provides the service and a *Jini Lookup* service that stores the service. See Figure VI.2.

We used a Lookup service in each computer, that stores its provided services, to solve the problem of centralized unit network, and permit ubiquitous communication between devices.

[4] See Chapter IV for details

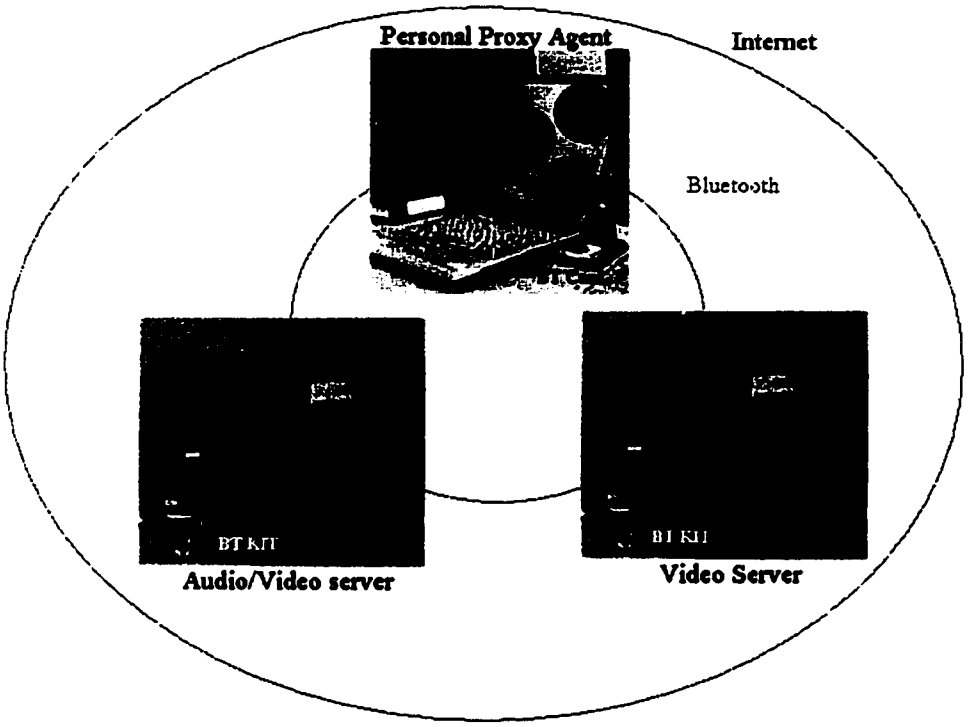

Figure VI. 2: The main components of our implementation.

VI. 3 Implementation scenario

We implemented a scenario where one of the computers of Figure VI.2 runs a Jini Audio/Video server and the other a Jini Video server. The laptop runs the Personal Proxy Agent (PPA). Figure VI.3 explains the message sequence diagram of the implemented scenario.

The Personal Proxy Agent (PPA) receives a SIP Invite request send by an outside caller (message 1). As soon as the SIP Invite request has been received defining the session description (in this example an Audio and Video request), the Personal Proxy Agent detects the surrounding devices present in the PPA's vicinity through its Bluetooth client (message 2). The Bluetooth client is responsible for detecting Bluetooth devices and getting their IP addresses and then passes them to the PPA's Jini Client representing the Jini lookup service (message 3). Now that the Jini client has the Jini lookup service's IP addresses, it sends unicast messages to the Jini lookup servers at the IP addresses found looking for Audio and Video services (message 4, 5). Each service is represented

in a template defining its attributes. Each Jini lookup service will look for registered services that matche the templates defining the Audio and video services. The servers offering those services will return to the PPA the interface objects ("stubs" in the sense of Corba) supporting the functions that can be called by a client for interacting with the service (message 6, 7). Now the PPA's Jini client is able to interact with the services. The Personal Proxy Agent's SIP User Agent will send a Sip 200 Ok message (message 8). In this implementation, to simulated a real data transfer between the different components, the PPA invokes the two functions Audio() and Video() which, open Audio and Video windows respectively (message 9, 10). Finally, the PPA ends the session by sending a SIP Bye message (message 12) after it receives the SIP Ack message from the outside caller (message 11).

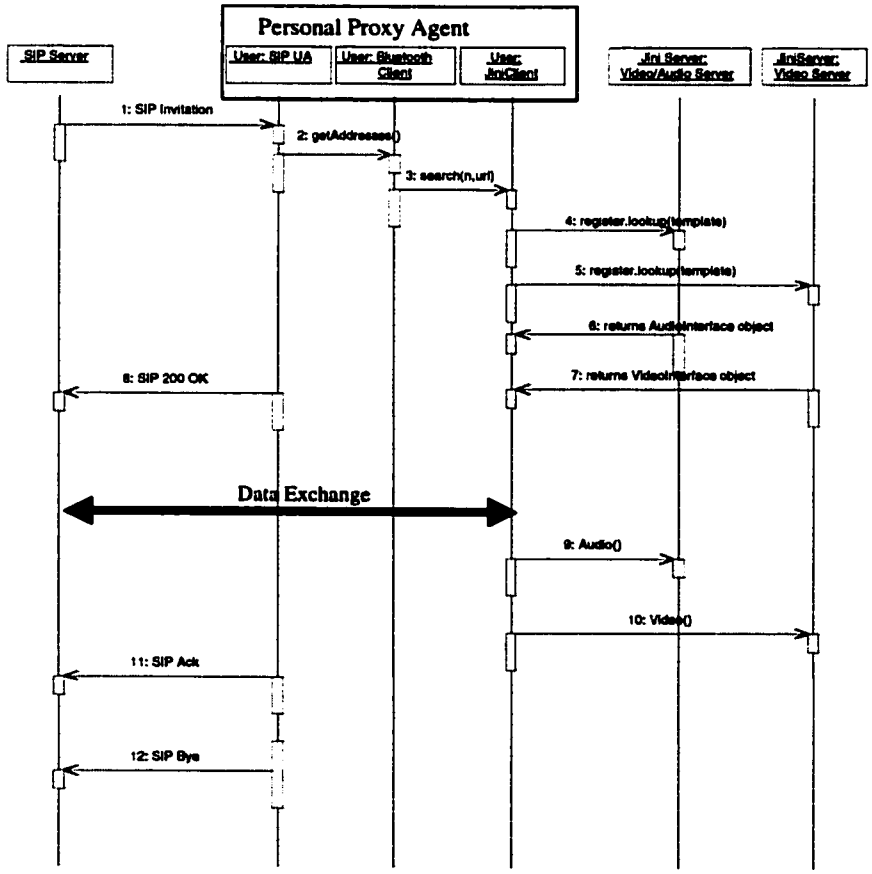

Figure VI. 3 The implemented scenario[5]

[5] Only messages exchanged through the Internet are shown.

VI. 4 Explanation of the implemented code

In this section we explain mainly how the scenario of Figure VI.3 was implemented. When the PPA receives the SIP request from the outside caller for Audio/Video, the Bluetooth client returns the IP addresses of the present Bluetooth devices through the *getAddresses()* function. These addresses are passed to the Jini client for the service search process.

The search process goes mainly through the *search(n,url)* function, where "n" variable represents the number of Bluetooth devices detected by the Bluetooth client, and "url" represents a list of URLs[6] containing the IP addresses of those devices. See Figure VI.4.

[6] Uniform Resource Locators

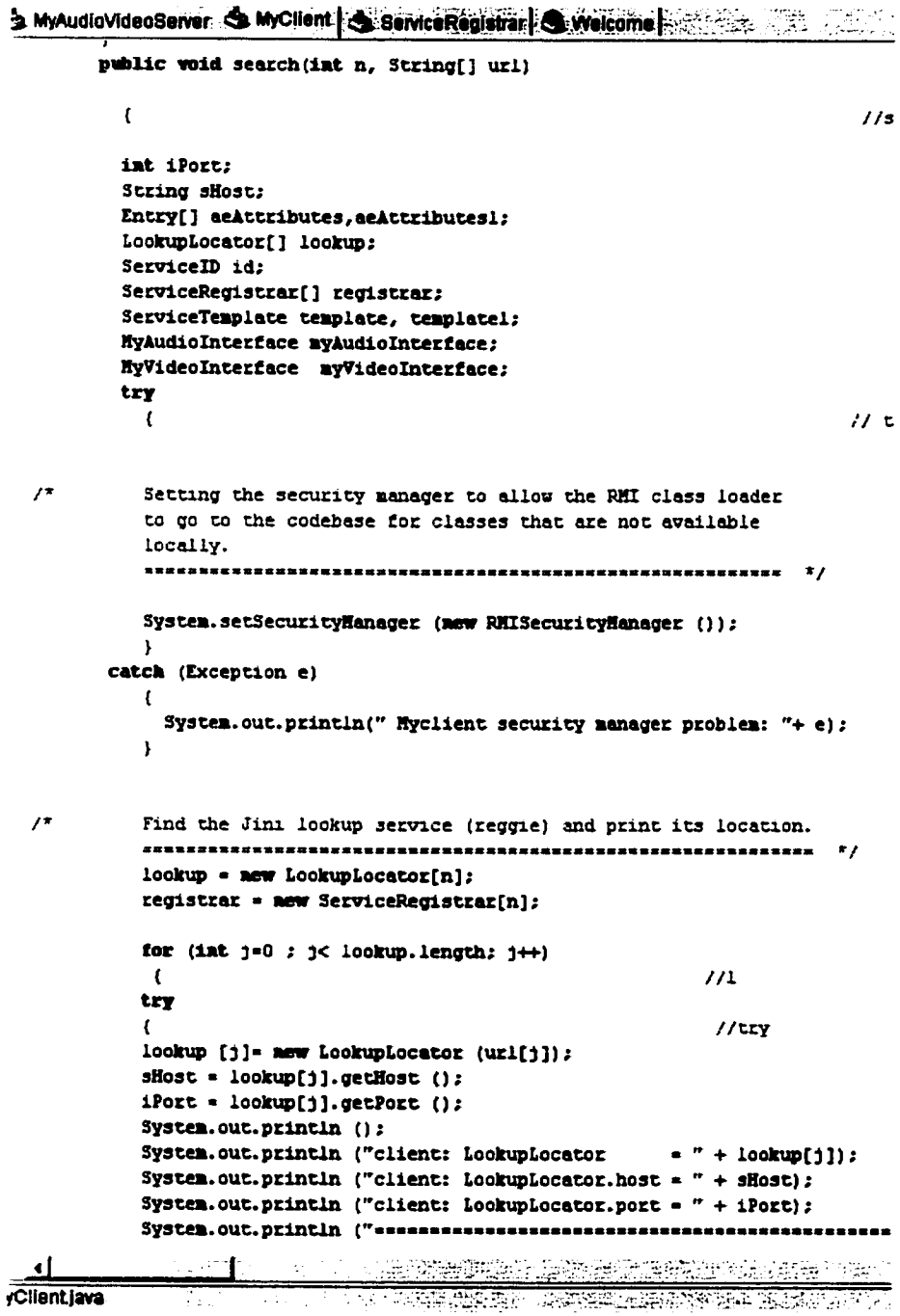

Figure VI. 4: Jini Client source code showing *search()* function.

Each template in Jini represents a specific service. This template contains a list of attributes defining the service characteristics. So the Jini client searches services by templates.

Figure VI.5 shows the defined templates (template and template1) that have one attribute entry representing the name of the service. The variable "j" represents the number of a particular Jini lookup service (device).

MyAudioVideoServer | MyClient | ServiceRegistrar | Welcome

```
        {                                                                    // catch
          System.out.println(" MyClient security manager problem: "+ e);
        }                                                                    //catch


/*      Find the Jini lookup service (reggie) and print its location.
        ------------------------------------------------------------  */
        lookup = new LookupLocator[n];
        registrar = new ServiceRegistrar[n];

        for (int j=0 ; j< lookup.length; j++)
         {                                                  //1
        try
        {                                                    //try
        lookup [j]= new LookupLocator (url[j]);
        sHost = lookup[j].getHost ();
        iPort = lookup[j].getPort ();
        System.out.println ();
        System.out.println ("client: LookupLocator      = " + lookup[j]);
        System.out.println ("client: LookupLocator.host = " + sHost);
        System.out.println ("client: LookupLocator.port = " + iPort);
        System.out.println ("-------------------------------------------------------

/*      Get the lookup service's ServiceRegistrar (the class by which
        interaction with the lookup service is possible).
        ------------------------------------------------------------  */

        registrar[j]  = lookup[j].getRegistrar ();
        id  = registrar[j].getServiceID ();
        System.out.println ("client: ServiceRegistrar   = " + registrar[j]);
        System.out.println ("client: ServiceID          = " + id);
        System.out.println ("=======================================================

/*      Perform a search on the lookup server to find the services
        that have the name attribute of "MyAudioVideoServer" and "MyVideoServer".
        The lookup service returns an interface object to the services.
        ------------------------------------------------------------  */

        aeAttributes = new Entry[1];
        aeAttributes[0] = new Name ("MyAudioVideoServer");
        template = new ServiceTemplate (null, null, aeAttributes);


        aeAttributes1 = new Entry[1];
        aeAttributes1[0] = new Name ("MyVideoServer");
        template1 = new ServiceTemplate (null, null, aeAttributes1);
```

Figure VI. 5: Jini Client source code showing the templates.

Each lookup server returns an interface object of the requested service when it is found. In our implementation, we defined two functions *Video()* and *Audio()* that are implemented by Video and Audio servers respectively. See Figure VI.6 and VI.7.

MyAudioInterface | MyAudioVideoServer | MyAudioVideoServer | MyClient | MyVideoInterface

```
import java.rmi.*;

/**************************************************************************

    MyVideoInterface -- The Server's Methods that are Available Remotely

 **************************************************************************/

    public interface MyVideoInterface extends Remote
      {
      public String Video () throws RemoteException;
      }
```

Figure VI. 6: MyVideoInterface source code.

MyAudioInterface | MyAudioVideoServer | MyAudioVideoServer | MyClient | MyVideoInterface

```
import java.rmi.*;

/**************************************************************************

    MyAudioInterface -- The Server's Methods that are Available Remotely

 **************************************************************************/

    public interface MyAudioInterface extends Remote
      {
      public String Audio () throws RemoteException;
      }
```

Figure VI. 7: MyAudioInterface source code.

VI. 5 Executing the scenario

Referring to Figure VI.3, each component is running on a different computer. The Personal Proxy Agent runs on a laptop, and the SIP server, the Video/Audio server and Video server run on three different PCs.

To run the simulation, we open two SIP User Agent windows on two different machines. One is the SIP user agent client running on the laptop, being part of the Personal Proxy Agent. The second is the SIP user agent server running on a PC, being the one initiating the SIP Invite request and handling the SIP reply messages.

From the PPA (SIP client) window, we see how the PPA receives and sends the SIP messages as SIP Invite and SIP Bye, as well as the result of searching for services

defined in the Invite message as explained in Section VI.2 (See Appendix C for output) The addresses shown in the Output represent the addresses of the Bluetooth devices in the vicinity mapped to their IP addresses. A template represents a service and its attributes, so the search process uses this template to find a specific service.

From the Caller side window, we see mainly how it handles the SIP reply messages as explained in Section VI.2 (See Appendix D for output).

The time it takes to establish the session and receive the Audio and Video data is about 25 seconds, of which about 10 seconds is required to detect Bluetooth devices and receive their IP addresses, and about 15 seconds to locate the lookup services, find the requested services and finally receive the multimedia data. Of these 15 seconds, about 6 seconds are spent to find services. One way to reduce the whole session establishment time is to consider continuous updating of the local directory, i.e. continuously collect this information as the user changes location, instead of collecting it only when needed.

At the reception of the multimedia data stream on the different devices, the separate audio and video data streams are not synchronized. Usually it takes about 6 seconds before the video data stream arrives after the beginning of the audio stream. One way to reduce this synchronization problem would be to send the video stream first and then the audio stream with a certain delay.

VI. 6 Conclusion

The main difficulties faced in this implementation is the mapping of Bluetooth addresses to the IP addresses of the devices having a Jini lookup service running in each device, because Bluetooth Kits do not support IP. The second issue was the synchronization of the sent multimedia data (audio and video). And finally, software conflict between Bluetooth PC card and Bluetooth kit. Instead of using a portable PC, this architecture could be realized using smart PDAs[7] that can access the Internet (example Trium Mondo[8] PDA) and integrates Bluetooth technology.

[7] Personal Digital Assistant
[8] http://www.zdnetindia.com/supercentre/mobiles/stories/30432.html

Chapter VII
Conclusion

In a ubiquitous computing environment, the set of available devices for the user may change continuously as the user changes his location. Updating manually the information about currently available devices is not a feasible solution since in ubiquitous environments, devices do not have to be visible to the user, and the set of available device changes continuously as the user moves from one location to the other, making it unfeasible to the user to manually update his/her profile.

To overcome these limitations, we propose a new architecture for mobile communication that leverages the existing service discovery protocols to discover services in the vicinity of the user. This architecture runs a modified version of the Home Directory Agent (HAD) on a hand-held device, such as a PDA, that is carried by the user, which we call a Personal Proxy Agent (PPA). The PPA runs a QoS negotiation and selection algorithm to select the most appropriate available service(s) to be used for a given application, and their configuration parameters, based on the user preferences and the constraint of the devices that provide the service(s)

This work leads us to new issues that we may propose as future work:

- We may think about the problem of authorization where the user may not be allowed to use all devices around him but be restricted to public-use devices only. In this case we have to find a way to detect or select only those devices that are for public-use and the PPA will not receive information from restricted devices.
- Geographical location of a device may be an issue. Since the detection of a device is made through radio frequency and the distance covered is about 10 meters, some units may be detected but not reachable (e.g. being in the next room). So the problem of detecting a device that is close to the user but not present in his/her WPAN should be solved. One way to solve this problem is to introduce signal

processing to detect obstacles, such as walls, determining the physical limits where the user could be.

- We may extend our architecture by supporting service hand-off, where services might be replaced by better services, or when the life span of a service is shorter than the duration of the session. This requires continuous periodic checking for services similar to the ones that are already in use, and providing a seamless handoff from one service to the other.
- Session mobility can be an extension to our architecture, where the PPA keeps a reference handle to the machine chosen during the previous session. As soon as the session finishes its execution, the result can be accessed or displayed on any other device/machine present in the user's new PAN.
- Service caching can be added to our architecture as well, where a PPA can keep in a cache the most often or usually requested services. As the user changes location to form a new PAN, the PPA's DSDA will continuously search for the cached services, anticipating the request.
- Security and authentication were not considered in this project, but we may apply the security concept proposed by [DBC01] to secure an infrastructure that provides telecommunication services on the Internet for mobile users. It establishes a trust relationship between any pair of the parties with a password-based user access.
- We should study and measure the usability based on our architecture in terms of the user's interest and ease of use. This information helps us to see how the user interacts with the system and how easy it is to learn, as well as if the system is relevant and meets the actual needs of the user.
- In building a composite/ virtual device, the separate reception of video and audio data streams is not synchronized. We have noticed a certain delay on receiving the video data stream compared to the audio data stream. The problem of synchronization is well known in the multimedia literature and there is no one and unique defined solution. To solve this problem we should consider synchronization across different levels, i.e. at network technology, software architectures, the operating system used, etc.

BIBLIOGRAPHY:

Papers and Books

[App et al 99] G. Appenzeller, K. Lai, P. Maniatis, M. Roussopoulos, E. Swiek, X. Zhao and M. Baker, "Mobile People Architecture". Technical Report: CSL-TR-99-777, Stanford University, January 1999.

[Arn et al 99] K. Arnold, B. O'Sullivan, R.W. Scheifler, J. Waldo, A. Wollrath, "The Jini Specification". ISBN 0-201-61634-3, Copyright Sun Microsystems, Inc., 1999.

[BAL01] M. Barbeau, V. Azondekon, R. Liscano, "Service Selection in Mobile Computing Based on Proximity Confirmation Using Infrared". Mitel MICON Conference, 2001.

[Bei et al 98] M. Beigl, A. Schmidt, M. Lauff, H.W. Gellersen, "The UbicompBrowser". Proceedings of the 4th ERCIM Workshop on User Interfaces for All, Stockholm, Sweden, October 1998.

[Bru et al 00] B. Brumitt, B. Meyers, J. Krumm, A. Kern, S.Shafer, "EasyLiving: Technologies for Intelligent Environments". Handheld and Ubiquitous Computing, 2nd International Symposium, pp. 12-29, Bristol, UK, September 2000.

[DBC01] I. Dupré la Tour, G.V. Bochmann, J.Y. Chouinard, "A Secure Authentication Infrastructure for Mobile Communication Services Over the Internet". Submitted to CMS 2001 Communications and Multimedia Security, 2001.

[EHB00] K. El-Khatib, X. He, G.V. Bochmann, "Quality of Service Negotiation Based on Device Capabilities and User Preferences". Technical Report, University of Ottawa, Canada, May 2000.

[HB00] F. Hupfeld and M. Beigl, "Spatially Aware Local Communication in the RAUM System". IDMS 2000, Spinger- Verlag Berlin, LNCS 1905, pp. 285-296, 2000.

[Her et al 01] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, A. Schade, "DEAPspace- Transient ad hoc networking of pervasive devices". Computer Networks 35, pp. 411-428, 2001.

[KCM01] R. Kravets, C. Carter and L. Magalhaes, "A Cooperative Approach to User Mobility". Computer Communication Review, a publication of ACM SIGCOMM, volume 31, number 5, October 2001.

[KPP01] S.J. Kang, J.H. Park and S.H. Park, "ROOM-BRIDGE: Vertically configurable Network Architecture and Real-Time Middleware for Interoperability between Ubiquitous Consumer Devices in the Home". Middleware 2001, LNCS 2218, pp. 232-251, Berlin, 2001.

[MB00] B.A. Miller, C. Bisdikian, "Bluetooth Revealed". The Insider's Guide To An Open Specification For Global Wireless Communications, ISBN 0-13-090294-2, Prentice Hall, Inc., 2000.

[PB94] C.E. Perkins and P. Bhagwat, "A mobile networking system based on the Internet protocol". IRRR Personal Communications, vol. 1, pp. 32-41, First Quarter 1994.

[Pha et al 00] T.L. Pham, G. Schneider, S. Goose and A. Pizano, "Composite Device Computing Environment: A frame for Augmenting the PDA Using Surrounding Resources". CHI 2000, Workshop on Situated Interaction in Ubiquitous Computing, April 2000.

[RKH00] Rexhepi, V., Karagiannis, G., Heijenk, G., "A Framework for QoS & Mobility in the Internet Next Generation", Proceedings EUNICE 2000, Sixth EUNICE Open European Summer School, University of Twente, Enschede, the Netherlands, September 13 - 15, 2000.

[Sch96] H. Schulsrinne, "Personal mobility for multimedia services in the Internet". European Workshop on Interactive Distributed Multimedia Systems and Services (IDMS), (Berlin, Germany), Mar. 1996.

[Zou00] B. Zou, "Mobile Protocol ID: badge activated application level handoff of a multimedia streaming to support user mobility". Thesis, University of Illinois at Urbana-Champaign, 2000.

Links

[BT] Specification of the Bluetooth system, http://www.bluetooth.com

[HomeRF] The HomeRF home technology, http://www.homerf.org/learning_center/

[Hotdesk] The Hotdesk technical information, Sun Microsystems, http://www.sun.com/products/sunray/techinfo/hotdesk.html

[H.245] Introduction to H.245 protocol, from the H.323 Information site http://www.packetizer.com/iptel/h323/

[H.323] International Engineering Consortium, definition and overview of the H.323 protocol, www.iec.org/online/tutorials/h323

[IR] An Introduction to Infrared Technology: Applications in the Home, Classroom, Workplace, and beyond, http://trace.wisc.edu/docs/ir_intro/ir_intro.htm

[Jini] The community resource for Jini technology, http://www.jini.org

[Leap] Lightweight & Efficient Application Protocols (LEAP) Forum, LEAP: An Effective Alternative to WAP, http://www.leapforum.org

[Salutation] The Salutation Consortium, http://www.salutation.org

[SIP] Session Initiation Protocol, Network Working Group Request for Comments: 2543, http://www.ietf.org/rfc/rfc2543.txt (Sip)

[SLP] An API for Service Location, Network Working Group Request for Comments: 2614, http://www.ietf.org/rfc/rfc2614.txt

[UPnP] Universal Plug and Play Forum (UPnP), http://www.upnp.org

[WAP] Wireless Application Protocol forum (WAP), http://www.wapforum.org/

[WLAN] Computer Networking, http://compnetworking.about.com

[W3C] The World Wide Web Consortium (W3C), http://www.w3.org/

[802.11] General information and short description of the standard IEEE802.11 technology, http://grouper.ieee.org/groups/802/11/main.html

Appendix A

Bluetooth tool Kit board

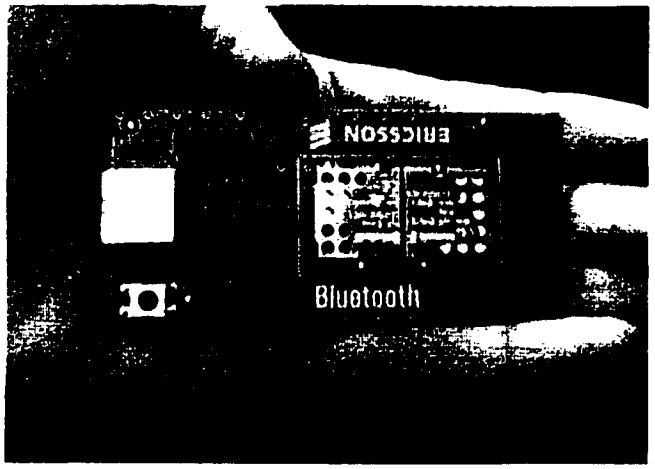

Figure A. 1: Bluetooth Module of Ericsson (ROK 101 008)

A.1 Description:

ROK 101 008 is a short-range module for implementing Bluetooth functionality into various electronic devices. The module consists of three major parts; a baseband controller, a flash memory, and a radio that operates in the globally available 2.4"C2.5 GHz free ISM band. Both data and voice transmission is supported by the module. Communication between the module and the host controller is carried out via UART and PCM interface. ROK 101 008, which is compliant with Bluetooth version 1.1, is a Class2 Bluetooth Module (0 dBm) and is type-approved.

A.2 Key Features:

- Qualified to Bluetooth 1.1
- RF output power class 2
- FCC and ETSI approved

- 460 kb/s max data rate over UART
- UART and PCM interface
- I2C interface
- Internal crystal oscillator
- HCI firmware included
- Point to Point (PtP) connection
- Built-in shielding

A.3 Supported Bluetooth Profiles

- Generic Access Profile
- Service Discovery Application Profile
- Serial Port Profiles
  - Dial-up networking
  - Fax
  - Headset
- Generic Object Exchange Profiles
  - File transfer
  - Object Push
  - Synchronisation

A.4 Suggested Applications

- Computers and peripherals
- Handheld devices and accessories
- Access points

Appendix B

Bluetooth PC Card from IBM

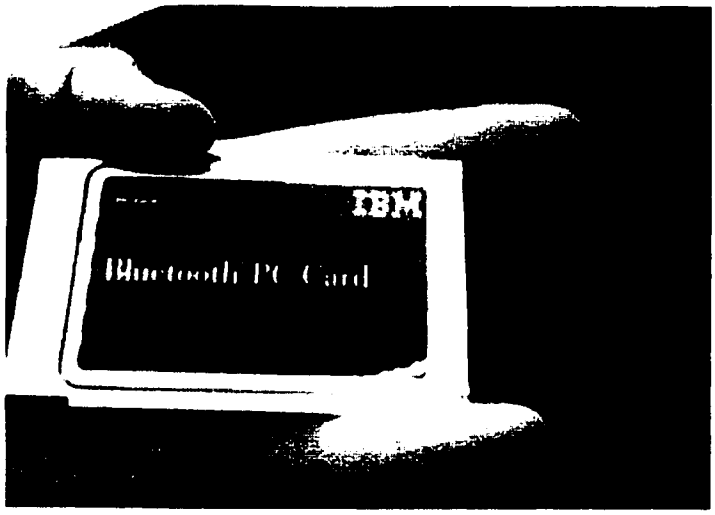

Figure B. 1: Bluetooth IBM PC Card

B.1 Technical specifications:

Type II PCMCIA card, 5 volt 16-bit PC Card 3.0 standard: Firmware upgradable Bluetooth 1.0B compliant, supporting the following profiles:

- Generic Access Profile
- Service Discovery Application Profile
- Serial Port Profile
- Headset Profile
- Dial-up Networking Profile (as Data Terminal)
- FAX Profile (as Data Terminal)
- LAN Access Profile (as Data Terminal)
- Generic Object Exchange Profile
- Object Push Profile
- File Transfer Profile

- Synchronization Profile (supported by the API)
- Integrated diversity antenna
- User application software including profile management and control center.
- Bluetooth Standard transmit power mode

B.2 Performance Specifications:

- Up to 10m range
- Power draw: average standby current less than 95 mA
- Transmit: less than 200 mA
- Receive: less than 120 mA
- Receive sensitivity -83 dBm

B.3 Physical Specifications:

Height:

inches: 0.2
mm: 5

Width:

inches: 2.1
mm: 54

Depth:

inches: 4.6
mm: 118

Operating temperature:

Between -20 degrees celsius and 29 degrees celsius
Between –5 degrees ` ahrenheit and 85 degrees ` ahrenheit

Relative humidity:

Operating: 93% maximum

Appendix C:

Personal Proxy Agent Output

Appendix D:

SIP Server Agent Output