# Service discovery for Personal Area Networks

CÉCILE AYRAULT


KTH Microelectronics
and Information Technology

Master of Science Thesis
Stockholm, Sweden 2004

IMIT/LCN 2004-07

# Service discovery for personal area networks

CECILE AYRAULT

Master of Science Thesis

Stockholm, Sweden, August 2004

**Abstract**

With the increasing use of electronic devices, the need for affordable wireless services specifically context-aware services, in a so-called Personal Area Network (PAN) is becoming an area with significant potential. Service discovery is a basic function.

Even though a number of service discovery protocols have been implemented, a specific protocol for a PAN environment may need to be developed, as the characteristics of a PANs differ from other networking environments. Thus, the specific requirements for service discovery from a PAN perspective were studied. Methods for service discovery will be described that take into account both local and remote services.

These methods will then be evaluated in a SIP telephony infrastructure to decide where a call should be delivered. The location of a person is done by using the implemented service discovery.

# Sammanfattning

Med en ökad användning av elektroniska enheter blir behovet av trådlösa tjänster, speciellt context-medvetna tjänster i så kallade Personal Area Network (PAN), ett område med betydlig potential. Service Discovery är en grundläggande funktion.

Även om flera service discovery protocols har implementerats finns det behov av ett specifikt protokoll för PAN-miljöer då egenskaperna hos ett PAN skiljer sig från andra nätverksmiljöer. Således studerades de specifika krav för service discovery från ett PAN perspektiv. Metoder för service discovery kommer att ta med i beräkningen båda lokala och avlägna tjänster.

Dessa metoder utvärderas i en SIP telephony infrastructure för att avgöra var en påringning ska levereras. Lokalisering av en användare sker genom det implementerade service discovery-protokollet.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This thesis concerns discovering services in a so-called Personal Area Network (PAN). It is part of the Adaptive and Context-Aware Services (ACAS) project [1], which is described in section 1.1.

## 1.1 ACAS project and research challenges

The purpose of this project is to create affordable and adaptive support for users to interact with services on the Internet. These services include more than just those available with existing and planned cellular infrastructures.

Context-aware user interaction is the first part of this project. Its aim is to create affordable and usable support for a mobile work environment. Dynamic and adaptive configuration of artifacts, services, and interfaces are to be used in both public and private environments. A ubiquitous computing scenario and heterogeneous computing and communication infrastructures are assumed. The ACAS project addresses the following:

- Self/ad hoc configuration support, e.g. service adaptation and negotiation mechanisms to avoid manual configuration,
- Modeling the context,
- Automated reasoning, and
- Human machine interaction (HMI) issues .

Context-aware *service delivery* is the second part of the project. It aims at creating affordable support for adaptive and automatic (re)configuration of local and global infrastructure, facilitating the interaction of end-users with their services in heterogenous (wireless and mobile) networks.

## 1.2 Properties of the communication environment

The Personal Area Networks (PAN) considered in this work has its own characteristics which must be taken into account. A PAN enables the various devices to be interconnected through the use of a short-range radio link. Both Bluetooth and Wireless LAN (WLAN), such as IEEE 802.11b, are the main examples of communication technology which we will consider.

The main characteristics of the environment to be considered can be summarize as follows:

- PANs are highly dynamic environments, as the available devices and services will vary over time and can change quickly.
- Devices should use the appropriate communication link (as necessary).
- Consideration must be given to additional security threats due to communication via a wireless interface and to varying devices and services.
- When using the Bluetooth discovery protocol [2], point-to-point communication occurs between devices directly communicating with each other .
- Communication with other devices separated by two or more hops can occur when utilizing the Bluetooth PAN profile [3].

## 1.3 Specific problem statement

This paper deals with the specific problem of the discovery of services via IP across wireless links to create affordable and usable support to enable users to discover services running on devices in the range of this current PAN. In particular this thesis considers the Bluetooth service discovery protocol and how it could be expanded to:

- Allow communication between devices separated by two or more hops,

- Create support at lower levels of the Bluetooth stack to inter-work with services based on IP to create (automatic) support for service allocation,
- Permit personal devices to discover services outside the PAN, and
- To discover services in the fixed network which can be accessed through access points (which may be one or more hops away).

## 1.4 Proposed method

The differences between networks is illustrated by using two different scenarios. First, a network environment composed only of Bluetooth devices is considered. Service discovery is based on a central coordinator. The Bluetooth devices in range only need to register with this coordinator to discover the devices and services in range. In the second scenario, one or more of these devices also have access to a Wireless LAN. The simple networks can be linked to a larger networks composed of WLAN devices and to the fixed networks. The coordinator is used to deliver IP packets inside the Bluetooth networks.

To evaluate the performances of the proposed solution, the native service discovery performance was first evaluated. Then, the proposed solution was considered for context aware call delivery. In a SIP telephony infrastructure, the location of a person was determined by using location information and the Proximity of a Person. To determine the Proximity of a Person, the Bluetooth service discovery was used. Knowing the environment surrounding the user, a better decision regarding where the data should be sent can be made.

# Chapter 2

# Background

## 2.1 Bluetooth Specifications

Bluetooth [2] is a wireless technology, which should have the same characteristics of a cable in terms of security and cost. Bluetooth technology also provides a universal framework for wireless communication of both data and voice. Devices utilize short-range radio links. Bluetooth operates in the license free 2.4 GHz ISM (Industrial, Scientific, and Medical) band and uses Frequency-Hopping Spread Spectrum (FHSS) technology. Bluetooth devices share 79 channels of 1 MHz bandwich within the 2.4 GHz band. The Bluetooth architecture is based on a protocol stack, shown in figure 2.1. This architecture is divided into seven layers.

A connection between devices sharing the same channel form a piconet. In each piconet, one device acts as a master whereas the others act as slaves. A connection

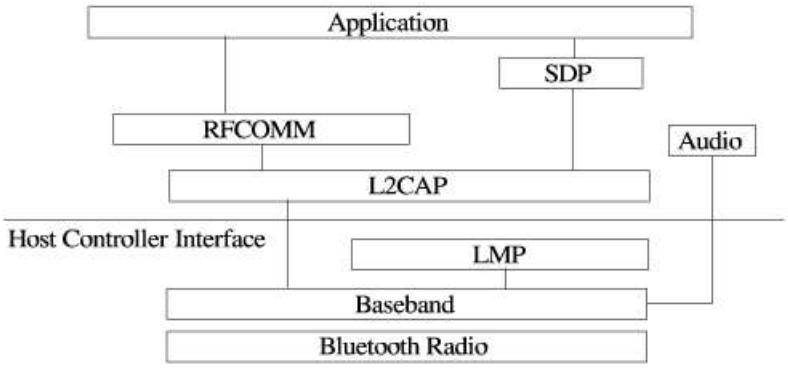


Figure 2.1: Bluetooth protocol stack

between two devices is set up through two successive stages, the inquiry and page process. They both use the baseband protocol.

### 2.1.1 Inquiry Process

The inquiry procedure detects when other devices are in range by broadcasting ID packets on 32 of the 79 hop frequencies at twice the normal frequency hopping rate. The inquiry only uses the Inquiry Hopping Sequence, 32 of the 79 frequencies, to overcome the initial frequency discrepancy between devices.

If any Bluetooth device is listening to the same frequency in the Inquiry Hopping Sequence, it responds by sending its address, clock information, Class of device (CoD), and other information in a Frequency Hopping Synchronize (FHS) packet. The CoD value has a length of 24 bits and can be assigned by the application. This CoD field is shown in figure 2.2, by providing an indication of the services offered by each device, permits the potential master to differentiate the available services, in order to find the one matching the requested service. The CoD specification can be found in the Bluetooth assigned numbers specification [4]. The CoD is divided into the major service class, the major device class, and the minor device class field. Two bits of the major service class field are reserved. The CoD can be assigned by the application; it is neither fixed by the hardware nor the Bluetooth stack. This CoD field is not used in native Bluetooth Service Discovery Protocol. A device enters inquiry scan in certains intervals. Thus, according to the specification, the inquiry process must last at least $T_w$=10.24 s to ensure that all Bluetooth devices in range respond to the inquiry in an error free environment.

The paging process can now establish a connection between the master and the slaves which provide the requested service(s). These slaves are now Active members of the piconet. Seven slaves can be active members at a time within each piconet. If more than eight devices take part in a piconet, some of them must be in one of the seven interim states. Three of them, SNIFF, HOLD, and PARK modes are power saving mode. In the SNIFF mode, the slave listens to the master at every $T_{SNIFF}$ interval, but is still considered as an active member of the piconet. In the HOLD mode, the master and slave don't commnicate with each other during a period which they agreed upon. Finally, in the PARK mode, the slave no longer participates to the piconet, but remains synchronized to the channel. The master can un-park the slave, whereas the slave can asked to be un-parked.

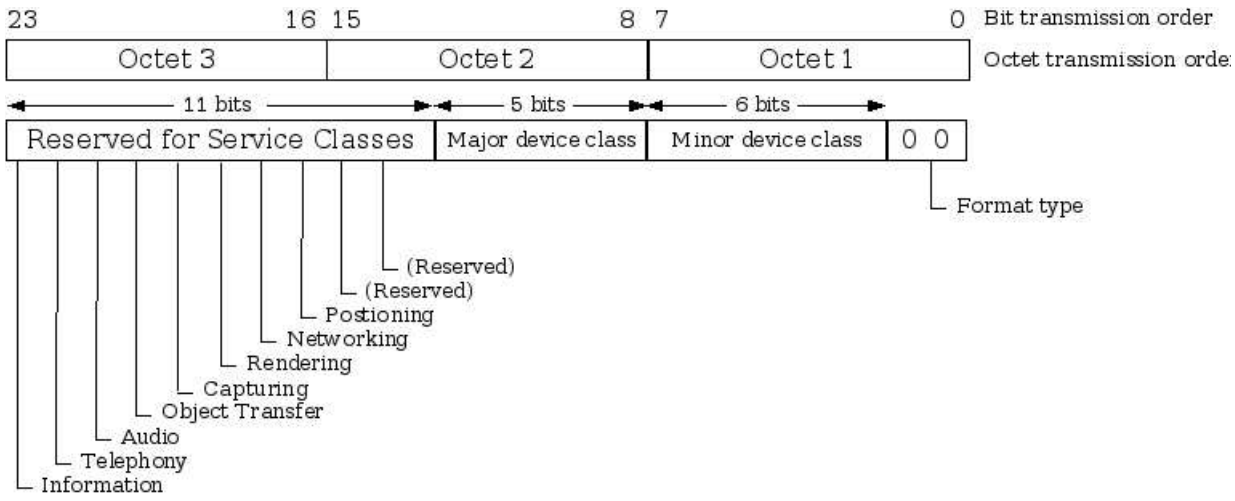


Figure 2.2: The Class of Device (CoD) field

### 2.1.2 Bluetooth Service discovery

The Bluetooth Service Discovery Protocol (SDP) provides a mean for applications to discover the services available and determine their characteristics. When the link is set up, the service discovery protocol is initiated. The service discovery protocol in the lower layers of the Bluetooth stack focuses on discovering services available from or through Bluetooth devices. This protocol fulfills these main requirements:

- Discovery of services based on service attributes and service classes.
- Browse for services without a priori knowledge of their characteristics.
- Discovery and use of services providing access to other service discovery protocols. The native SDP provides a means for *discovering* available services on remote devices, but there is no means to provide *access* to them. Thus, another service discovery protocol needs to be implemented for this purpose.

The service discovery protocol allows a user to:

- Establish a link layer connection using the Logical Link Control and Adaptation layer Protocol (L2CAP) protocol between the devices
- Search for specific services

- Browse for services
- Retrieve attributes stating information necessary to enable connection to services.
- Establish a non-SDP connection to use the service.

The Service Discovery Protocol relies on L2CAP links between the SDP server and client. For example, to query the SDP server about available services, a client uses a dedicated L2CAP channel, via a Protocol Service Multiplexer (PSM) reserved for SDP. To create an L2CAP channel between two devices, a connection request packet is sent. The PSM field is one of the data fields. Each Bluetooth protocol, such as the Service Discovery protocol, is assigned a value [2]. All devices know this channel beforehand. However, this L2CAP channel cannot be used to access the service.

The SDP protocol is based on a client/server model. Any Bluetooth device can generally act as either server or client, or even both at the same time. The protocol is a simple request-response scheme. The SDP client i.e., the Bluetooth unit utilizing a service issues requests on an existing connection, searches for services, and requests attributes of these services.

On the other hand, the SDP server, a Bluetooth device providing services, keeps a database of service information, and responds to requests on an existing connection i.e., via the specific PSM reserved for SDP over the established L2CAP. The description of a service's characteristics is stored in a single service record. The service record is a list of attributes as shown on figure 2.3. Each attribute consists of an ID and a value for a specific data type as shown in figure 2.4. The attribute value is represented as a data element. The type Universally Unique Identifier (UUID) or a set of UUIDs is the most common type. This service record is composed of at least the attributes called ServiceRecordHandle, a unique ID within the SDP server, and ServiceClassIDList, containing the path of service classes, and including the specific class the service belongs to as well as all the superior service classes.

### 2.1.3 Bluetooth device Address (BD_ADDR)

Each Bluetooth transceiver is allocated a unique 48-bit Bluetooth device address (BD_ADDR). This address is derived from the IEEE 802 standard and is shown in figure 2.5. This 48-bit address is divided into a 24-bit Lower Address Part (LAP) field, a 16-bit Non-Significant Address Part (NSAP) field and a 8-bit Upper Address Part (UAP) field.

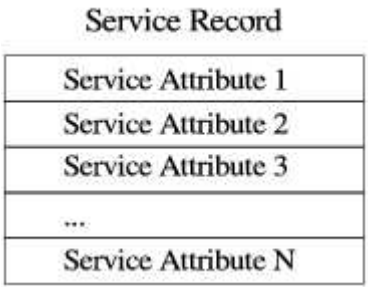


Figure 2.3: Service Record

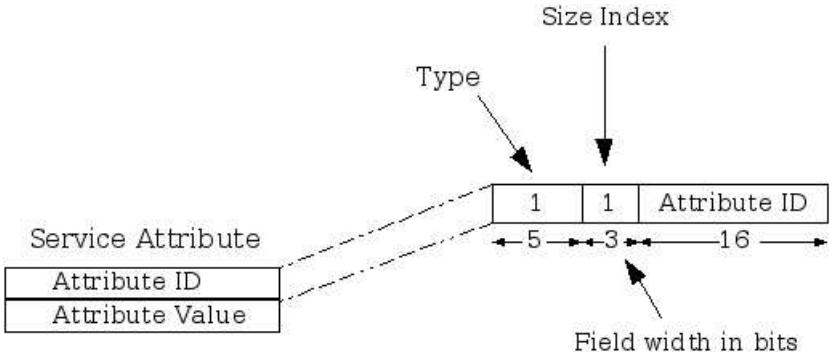


Figure 2.4: Service Attribute

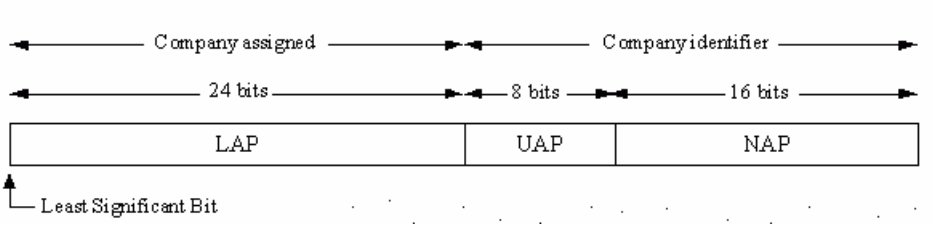


Figure 2.5: The Bluetooth device address

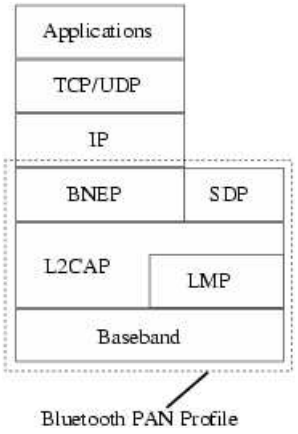


Figure 2.6: Bluetooth PAN protocol stack

The NSAP and UAP fields are assigned to the company by IEEE, whereas the LAP field is company assigned and corresponds to the device identifier.

## 2.2 Bluetooth Personal Area Network (PAN) profile

The Bluetooth PAN profile provides networking capabilities for Bluetooth devices. To achieve this, the Bluetooth Network Encapsulation Protocol (BNEP) is needed.

### 2.2.1 Bluetooth Network Encapsulation Protocol (BNEP)

The main purpose of the BNEP [5] is to define a packet format, to enable support of common networking protocols such as IPv4, IPv6, and other common networking protocols. The encapsulated packets are then transported over L2CAP, which furnishes a data link layer. BNEP is integrated, as shown in figure 2.6, between the IP-layer and the Bluetooth Logical Link and Control Adaptation Layer. The master Bluetooth device acts much as an Ethernet Bridge at the BNEP layer, forwarding packets that are not destined for itself.

BNEP defines the packet format used to transport common networking protocols over the Bluetooth media. The packet format is based on EthernetII/DIX Framing as defined by IEEE 802.3 [5]. Figure 2.7 shows BNEP being used to transport an Ethernet frame.

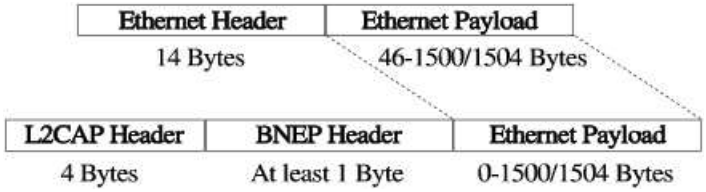


Figure 2.7: BNEP packet

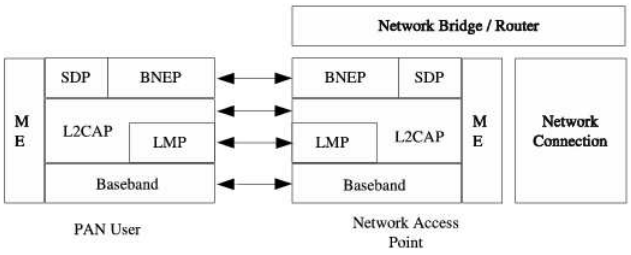


Figure 2.8: SDP Network Access Point scenario

### 2.2.2 Bluetooth Personal Area Network (PAN) profile

The Bluetooth PAN profile [3] defines three different scenarios. First, the Network Access Point (NAP) scenario, which concerns one or more Bluetooth devices and a Bluetooth device which is also attached to another network, this device acts as a bridge between a Bluetooth piconet and another type of network technology. The second scenario is Group Ad-hoc Networks (GNs), which consists of a single piconet with connections between two or more Bluetooth devices. Finally, the PAN User-PAN User (PANU-PANU) scenario, which offers point-to-point communication only between these two nodes. These three scenarios use the protocols and entities of the figures 2.8, 2.9, and 2.10. The PAN profile only defines Network Encapsulation for a single piconet.

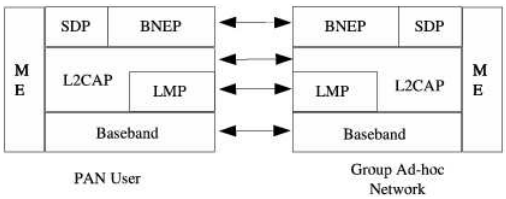


Figure 2.9: SDP Group ad hoc Network scenario

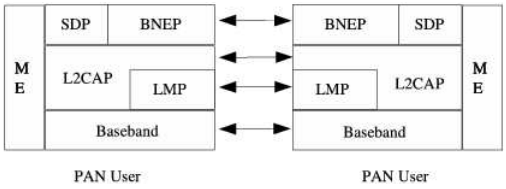


Figure 2.10: SDP PAN User scenario

Each device, such as those capable of providing NAP, GN, or both services shall indicate one or more of these service class in its PAN profile service. One instance of each of these Service Classes (NAP, GN, and PANU-PANU) may be provided by each device. The available services are made public via the SDP server of this device.

## 2.3 Network mobility

### 2.3.1 Mobile Ad Hoc Networks

The Internet Engineering Task Force (IETF) working group Mobile Ad Hoc Networks (MANET) [7] addresses the routing specification of ad hoc networks, i.e. networks formed on a temporary basis, easy to set up, operating without preexisting infrastructure, and characterized by untethered multihop communication. This means that an ad hoc routing protocol is suitable for wireless routing within both static and dynamic topologies. The working group has concentrated on a number of core protocols such as Ad hoc On-Demand Distance Vector routing (AODV), Dynamic Source Routing (DSR), Optimised Link State Routing (OLSR), and Topology Dissemination Based on Reverse-Path Forwarding (TBRPF).

The AODV routing protocol utilizes On-demand route creation with a two dimensional routing metric: the *sequence number* and the *hop distance*. A node, which wants to send a packet to a destination node, floods Route Request (RREQ) messages in order to find a route. A node who receives this message can send a Route Reply (RREP) message either if it is the destination node or if it has a “fresh enough” route to the destination node. Finally, when a node detects a route break in active sessions, it sends a Route Error (RERR) message to upstream nodes and the source.

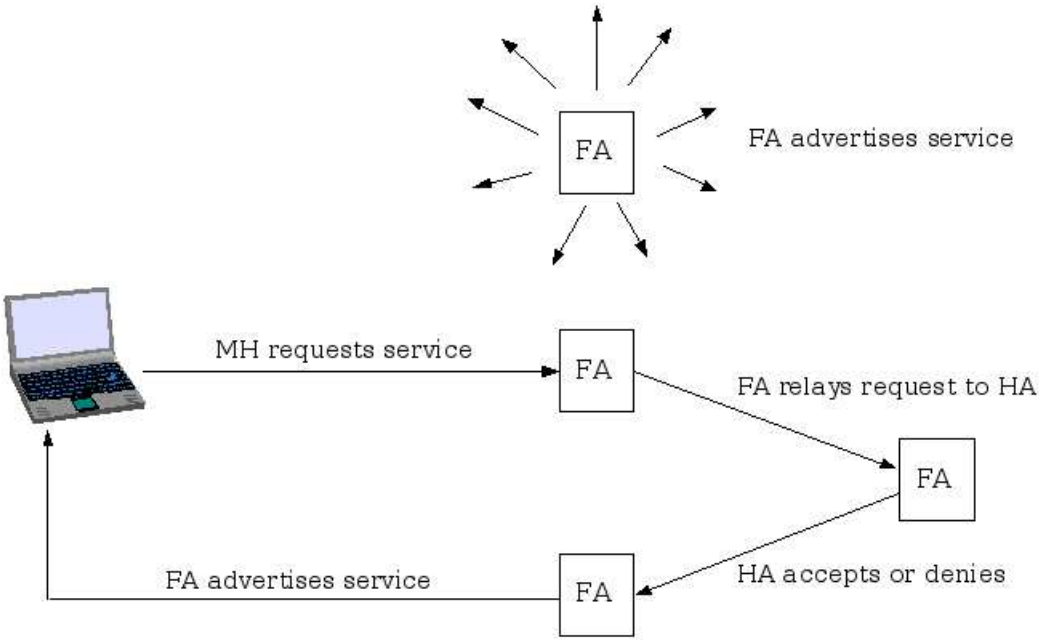


Figure 2.11: Operations in Mobile IP

### 2.3.2 Mobile IP

The IETF Mobile IP working group [8] has defined routing support to permit IP nodes, hosts and routers, using IPv4 or IPv6 to seamlessly "roam" among IP subnetworks and different media types. The protocol defined three entities shown in figure 2.11. The Mobile Host (MH) node may change its location without changing its IP address. The Home Agent (HA) is a router on the mobile node's home network that tunnels datagrams for delivery to the mobile node. Finally, the Foreign Agent (FA) is a router on a mobile node's visited network that provides routing services to the mobile node while registered.

The protocol is divided into some main components. The mobility agents periodically send agent-advertisement messages. A mobile host node receives one of these agent advertisements and determines whether it is on its home network or a foreign network. On its home network, a mobile node works without mobility services; on a foreign network, a mobile node must on the contrary obtain a care-of address by either utilizing the IP address of the foreign agent, or getting a co-located care-of address. At this point, the mobile node registers its care-of address with its home agent. Finally, datagrams to the mobile node's home network address are intercepted by its home agent, are tunneled to the care-of address, there they are received at the tunnel endpoint and delivered to the mobile

node.

As personal devices should be able to discover services outside the PAN and especially if one of the personal devices in the PAN has other wireless interfaces for accessing services on Internet, then this personal device could share access to services on Internet with other devices in the same PAN. However, since a PAN is a very dynamic environment, this personal device might have to use a different access point. Mobile IP, provides a way to seamlessly “roam” among IP subnetworks and media types, might enable this personal device to share access to services on the Internet.

# Chapter 3

# Method

## 3.1 Network environment

We consider a network scenario involving Personal Area Networks (PANs). Our goal is to allow much faster discovery and service negotiation than otherwise possible with Bluetooth. A service is an entity that can provide information, perform an action, or control a resource on behalf of another entity [9]. To discover the available services faster is important because of the long discovery time which may be required to find a new Bluetooth device; this is especially important when the user may be moving relatively quickly past this device. Information about devices, optionally with their Bluetooth capabilities, can be distributed via WLAN over a much larger area than covered by direct Bluetooth to Bluetooth communication makes this information more accessible. Thus we can now know their existence, their MAC address, and possibly the services they provide in **advance** of actually encountering this device. Knowing their MAC address can significantly speed up the inquiry phase. Information regarding Bluetooth devices in this WLAN cell are provided as context information to the mobile user or mobile device. Furthermore, additional service providers are available, since the WLAN access point is attached to an IP infrastructure any service provider in the fixed network is available.

## 3.2 Problem statement

### 3.2.1 Personal Area Network properties

The properties of the PAN which differ from a LAN can be summarized in three points. A PAN is a highly dynamic environment. Thus, service discovery will have to be carried out frequently or when a service is required in order to maintain an up-to-date picture of the available services of the local network. Furthermore, a service discovery protocol should minimize the use of communication. Bluetooth is a technology optimized for portable devices with constrained power.

### 3.2.2 Service discovery properties

Support for service discovery is available in the lower layers of the Bluetooth stack. However, the native service discovery protocol is not adequate for PAN, since this discovery process is only point-to-point between Bluetooth devices that are directly connected to each other, whereas the Bluetooth PAN profile allows devices in a PAN to communicate with other devices; even if they are multiple hops away. Moreover, because a Bluetooth unit can only transmit or receive in one piconet at a time, a bridging unit must switch between piconets on a time division basis. Due to the need to resynchronize its radio from one piconet to another and perform the necessary signaling, a Bluetooth unit necessary loses some time while switching, this represents an important performance constraint in building scatternets [10]. Thus, a service discovery protocol based directly upon Bluetooth scatternets does not meet our requirements.

The native Bluetooth SDP is unefficient for several reasons. Bluetooth only provides broadcast capability **after** connection establishment of the devices within a piconet. Furthermore, because of the highly dynamic nature of Bluetooth piconets, changes in the topology would require new service discovery whether it is warranted or not. Furthermore (since routing is out of the scope of the Bluetooth specification) every node has to establish a connection with every other node in order to carry out service discovery. Thus requiring large expenditures of energy and causing significant delays in service discovery. Then, nodes which have very limited memory and computational resource called “restricted nodes”, cannot support native Bluetooth SDP. Only the devices that already know the access parameters of the restricted node can retrieve its service parameter(s) [11].

Therefore, Bluetooth service discovery protocol should be enhanced to inter-work with services on top of IP to create (automatic) support for service allocation. However, BNEP is simply an adaptation layer which introduces additional

overhead and limits the PAN's dynamics as devices can appear and disappear at any time.

## 3.3 Proposed solution

We consider two different scenarios. Either the PAN is only composed of Bluetooth capable devices, or the PAN is composed of Bluetooth and WLAN capable devices. In the later case, context information about Bluetooth devices and additional service providers, attached to an IP infrastructure, are known in the WLAN and distributed via the WLAN to PAN gateway; thus this information can be known in both networks.

### 3.3.1 Service discovery protocol requirements

In the Bluetooth environment, the service discovery protocol requires devices to:

1. Perform a service inquiry for the available devices and services within radio range of the device, this consists of:
    - Starting an inquiry process,
    - Search for services which are located locally,
    - Search for remotely located service(s),
    - Select a service which matches the inquiry.
2. Listen to other devices within radio range. Therefore, the device should be able to listen to the inquiry process from other devices when the device is in range of other devices requesting services, thus:
    - Respond to an inquiry search when the device's services matches the query.
    - Send useful information, such as IP address, to the query device.

### 3.3.2 Bluetooth capable device environment

The PAN is composed of Bluetooth capable devices. Several possibilities were considered. A brief description of each solution considered is given below.

Service discovery protocols like Jini [12], UPnP [13], and salutation [14] rely on IP broadcast and multicast to find services. Two ways to achieve IP over Bluetooth while providing broadcast capability are PPP over RFCOMM or BNEP over L2CAP. However, both lead to a high communication overhead since a high level broadcast requires a properly pre-established piconet. Furthermore, this type of solution leads to long discovery delays which imply high power consumption. [11]

An other solution would be to enable IP communication in the scatternet directly on the L2CAP protocol. This will reduce the time and energy needed to discover the available service as compared to utilizing TCP/IP over RFCOMM. IP packets are directly encapsulated in L2CAP packets, thus the inefficiencies of IP over RFCOMM, such as the connection setup, are avoided. The assignment of an IP address within the scatternet can be done by considering this network as an ad hoc network. Thus, dynamic configuration of IPv4 link-local addresses (Autonet) can be considered [15]. Autonet may enable a host to automatically configure an interface with an IPv4 address. This address is only valid for communication with other devices connected to the same physical (or logical) link, thus within the ad hoc network. However, even if this solution has less overhead than running TCP/IP over RFCOMM, so less delays are involved, a high communication overhead remains.

To enhance the native Bluetooth SDP, a coordinator-based service discovery [11] can be used. To overcome the missing broadcast capability in non-connected state, the inquiry process is completed by using the Class of device (CoD) field of the FHS packet in the response to the ID packet. Coordinator-based service discovery relies on a central coordinator node which holds the complete service information of all Bluetooth nodes in range. As described in section 2.1.1, the CoD field of the device can be assigned by the application. Thus, the CoD field is used to locate the coordinator; one of the two reserved bits of the major service class field of the CoD locates the coordinator, as the other bits defined in the Bluetooth assigned numbers correspond to different specific services. This method is more effective with one central/fixed and many distributed/mobile nodes than with a peer-based service discovery.

When a device enters the coordinator's radio range, the coordinator is discovered during the inquiry process by interpreting the CoD field. Furthermore, as the coordinator manages the service information of all devices in communication range, a device can request service information about available devices or register its own services after having discovered the coordinator. However, the coordinator must be a device with sufficient computational, power, and memory resources. A device entering a PAN must only establish one connection and perform only one service discovery process. The proposed algorithm is divided into three parts

which are described below.

First, the coordinator election algorithm, an algorithm based on the lowest-ID is used. As the first part of the BD_ADDR is company assigned as described in section 2.1.3, the device with the larger BD_ADDR has more computational, power and memory resources. The variable VOTES of every node is set to 1 when the device is powered up; as soon as the initialization of the device is over, the device enters the states INQUIRY and INQUIRY SCAN. Then, the coordinator election starts if there is no coordinator in range. The coordinator election is initiated by one of the node in range. After two nodes discover each other, they compare the value of their VOTES variable; if the two nodes have equal VOTES variable, the node with the larger Bluetooth address increases its VOTES variable by the value of the VOTES variable from the other node. Otherwise the node with the larger VOTES variable increases its VOTES variable in the same way. The loser node also sends all the FHS packets (i.e. identity and clock) of the nodes it has won so far to the other node, and enters the PAGE SCAN state. Thus it will not be able to hear inquiry messages, but only the page messages. The winner node continues in the leader election process. For N nodes within the PAN, N-1 iterations are required. The process is linear since a node within range starts the election. The winner node of the N-1 iterations will be the coordinator; one bit of the reserved Major Services Classes area of the CoD field is set to 1. All other nodes will be in the PAGE SCAN. [16] Any devices entering the PAN are now able to locate the coordinator during the inquiry process. As described in the section 2.1.1, the time of the inquiry process is enough to discover all devices in range, even if a device is dividing its time between multiple piconets. This election occurs when the previous coordinator is no longer in communication range or during network initialization.

When the coordinator is elected, each device can register its services through the service registration and provision algorithm. The coordinator is aware of all available services from devices in range. A device which does not answer the request is marked as unknown. The native bluetooth SDP has thus been enhanced to discover existing services. Finally, to enhance the native Bluetooth SDP, three new signaling parameters are added to the SDP_ErrorResponse PDU and included in the Healing and error handling algorithm, these are: SuccessfulRegistration, TooManyCoordinators, and NoCoordinator. For the community to be self-healing, the nodes work collaboratively. When a device discovers a failure, it informs all nodes to switch to the required state.

### 3.3.3 Bluetooth and Wireless Local Area Networks (WLAN) capable devices networks

The PAN is composed of Bluetooth and WLAN capable devices. Thus, context information about Bluetooth devices and additional service providers, who are attached to an IP infrastructure, are known in the WLAN. Two scenarios were considered.

In the first scenario, the fact that a device has to discover the IP address of other devices was considered as the main focus. Thus, as there is no hierarchically assigned IP address and because of the dynamic environment, a search must be done periodically. A device needs to perform address resolution to determine the link layer address of the device using the discovered IP address. We assume that Bluetooth devices are present within the WLAN with links to some of the devices within the scatternet. Thus, different nodes in the scatternets might have to share the same external IP-address as devices directly connected to the Internet due to the scarcity of IPv4 addresses. For the IPv4 protocol, this can be implemented using Network Address and Port Translation (NAPT) on the access device, thereby allowing other Bluetooth devices to use private IP addresses. However, if the scatternets become important this solution can lead to a complicated structure, not suitable for the networks considered in this paper. For the IPv6 protocol [19], the autoconfiguration feature could be used to resolve this problem and the problem of the scarcity of IPv4 addresses as well.

However, due to the highly dynamic environment, the use of MANET, described in section 2.3.1 was considered. The WLAN devices will form an ad hoc network using Ad hoc On Demand Distance Vector routing protocol. The connection between the Bluetooth device's environment and the WLAN device(s) are done with one or more devices supporting both Bluetooth and WLAN technology. To access the services provided by the WLAN devices, a Bluetooth device sends its request to the coordinator. If the coordinator is not able to answer its request, it forwards it to a node with access to other networks. In the same way, a WLAN device who wants to request a service in the Bluetooth network forwards it to the node supporting both Bluetooth and WLAN technology. This node will then forward it to the coordinator of the Bluetooth network.

MIPMANET [17] is a way to connect a Mobile Ad Hoc Network to the Internet. It provides nodes in ad hoc networks with access to the Internet and allow a user to use mobilty services of Mobile IP, described in section 2.3.2. This is possible due to the mobile IP foreign agents. The foreign agents work as access points to the internet for an ad hoc network. To access the internet, a node in an ad hoc network uses his home address for all communications and registers with a foreign agent. Then, this node can send a packet to a host on the internet by

tunneling the packet to the foreign agent with whom the node is registered. This node can also receive packets from hosts on the Internet; packets are routed to the foreign agent by ordinary Mobile IP mechanism, which will then deliver it to the node in the ad hoc network by using the ad hoc routing protocol. When a node in an ad hoc network doesn't require access to the Internet, this node will see the ad hoc network as a stand-alone network, i.e., this node will not be aware of routes to destinations outside of the ad hoc network.

MIPMANET can provide the Mobile ad hoc Network (MANET) access to the internet. It can thus provide access to the internet to the Bluetooth networks as well. The mechanisms to provide access between the Bluetooth devices environment and the WLAN device will still work.

Instead of using MANET, the possibilty of using JXTA [18] was considered. JXTA is a peer-to-peer (P2P) framework. It enables the discovery of other computers running JXTA, the exchange of messages between them, and the forwarding of messages through peers. However, JXTA assumes that the network topology does not change or changes very slowly. Thus, the MANET approach is preferred.

Due to the autoconfiguration of IPv6 addresses, it could be interesting to consider them with the use of MIPMANET. The use of IPv6 will only change the mechanisms inside the Mobile ad hoc networks. The mechanisms to provide access to the Bluetooth device network, since BNEP as described in section 2.2.1 supports IPv6, will remain the same. The mechanisms to provide Internet access to the Mobile ad hoc networks will remain the same as well. Auto-Networking Technologies for IPv6 Mobile Ad Hoc Networks [19] address this issue. The purpose of using IPv6 addresses is to be able to communicate more easily through zeroconfiguration [20] for users in MANET. Four auto-networking technologies for automatic networking in IPv6 Mobile ad hoc networks are considered. First of all, IPv6 unicast address autoconfiguration through which a unique unicast address is configured in mobile node is considered. Then, the IPv6 multicast address allocation is taken into account; a unique multicast address is allocated to an application that needs a new multicast address. Next, the secure multicast Domain Name System (DNS) is considered. Every ad hoc node takes part in DNS service, such as name-to-address translation. Finally, service discovery based on multicast DNS is considered. Thus, ad hoc users can discover the service information that is necessary to connect to or join the service when the name, transport protocol, and domain for the service are given.

# Chapter 4

# Analysis

## 4.1 Scenario

To evaluate the performance of the service discovery protocol discussed above, we consider the case of context aware call delivery. In this scenario there is a SIP telephony infrastructure (based mainly on the Vovida Open Communication Application Library (VOCAL) [21]), the location of a user is determined by using location information (e.g., from MICA Motes [22] or RFID) and the Proximity of a Person (based on Bluetooth Discovery). Thus, the type of devices in range are known. A better decision about where the call should be delivered can be made according to the type of data (phone call, video) and the available devices (mobile phone, note book, etc.).

First, Bluetooth Discovery was tested to measure the discovery performance of the Service Discovery Protocol originally implemented. The test environment was composed of a PC and HP iPAQ which all have embedded or added Bluetooth support.

The proposed work fits into the ACAS project, as we can identify a person and his presence through detecting available Bluetooth devices in the vicinity. Location can be done through a mapping database, based on Bluetooth MAC address discovered corresponding to devices at fixed locations. This mapping service can also be used to learn device types and who the owners of these devices are.

## 4.2 Implementation

### 4.2.1 Software used

The implementation was done in java. The Impronto developer kit [23] was used, as it implements the standard Java/Bluetooth Application Program Interfaces (APIs) to facilitate creating wireless applications. The Impronto Developer Kit implentation uses Java SDK 1.4.1 (J2SDK 1.3.1 can be used as well). I also used Red Hat 9.0, Apache ANT [45], and the default kernel [linux 2.4.20] (a later version can be used as well).

The Java implementation of the Ad hoc On Demand Distance Vector routing from the University of Bremen, UoB-JAdhoc AODV implementation [24] was used. Their java implementation of Mobile IP [25] was used as well.

### 4.2.2 Implementation

The implementation of an service discovery mechanism based on a central coordinator in a Bluetooth capable device environment was implemented as a java package named a package serviceDiscovery. It fulfills the requirements described in the section 3.3.2. This module interacts with the java class library for Bluetooth wireless technology. The implementation is divided into two parts, as described in the section 3.3.

The implementation for the Bluetooth capable device environment follows the diagram shown in figure 4.1. This diagram provides a graphical view of the different objects of the implementation, their grouping, and their inter-relationships. When the device is powered up, the server side of the device starts running along with the Graphical User Interface (GUI) application. The data store is initialized by the Data class.

The Data class handles all the data needed by both the client and server. It contains five fields useful to the application. The addressCoordinator field contains the address of the coordinator the device should connect to in order to retrieve the available services that are in range. The vector “coordinator” is used during the coordinator election to store the address of the coordinator(s) discovered during the inquiry process. Then, depending on the number of coordinators a decision is made. The Hashtable named “devices” stores the address(es) of the available devices in range using their MAC address as a key and the corresponding service record as a value. The Hashtable “neighbours” stores the available devices in range (again using their MAC address as keys) and their CoD as values. Finally, the integer named “VOTES” is used during the coordinator election.

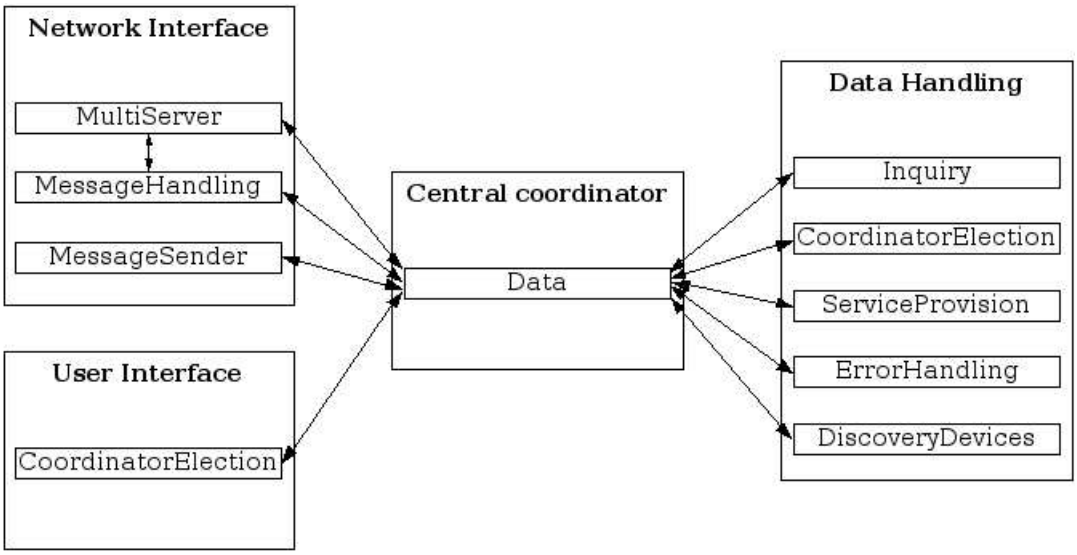


Figure 4.1: Class architecture of the coordinator-based service discovery

The application's GUI first tries to discover the coordinator in range during the inquiry process. According to the number of coordinator(s): if zero, then the device sends a "NoCoordinator" message to the devices it discovered and then enters the coordinator election application, if one, then retrieves the available services from the coordinator and then registers its own services, or if more than one, then sends a "TooManyCoordinator" message to the coordinators discovered and enters the coordinator election process. Once the devices in range and their available services are discovered, the GUI is automatically updated to display them. The user can request the data to be refreshed, if so the device starts an inquiry process again.

During the coordinator election process, the device makes a point-to-point connection to each of the devices in range as stored in the hashtable "neighbours". It then retrieves the integer "VOTES" from each device to compare it to its own variable "VOTES". Depending on the comparison, the device may win or lose. It then sends a message to the other device with the new VOTES variable of the other device and, if the device lost, with the devices it already beat and the devices still in the coordinator's election process. The device may continue the coordinator election or end it. In both cases, it sets its own VOTES variable to the new value and updates the hashtable of devices.

The class MultiServer handles all the messages sent to the device by the clients and creates a new thread for each client. Then, the pieces of information contained in each message are read and those particular to this application are handled. Two

types of OBEX messages could be received: “get” and “put”. Either the client tries to get an specific object from the server, or it tries to put an object on the server. The HeaderSet received from the client distinguishes the different cases based on the fields TYPE and NAME.

A client can get from the server the Hashtable of devices. This is only possible when the server is the coordinator. The client device will discover the coordinator during the inquiry process and can then retrieve the devices in range and their services with a “get” OBEX message. When a client sends this message, the server sends the required file and also adds the address and the services of this client in its hashtable of devices. A client can also get the “VOTES” variable from the server as explained above.

A client can also sends different objects to the server using an OBEX put message. When a device notices the lack of coordinator, it starts a coordinator election process and then sends a “NoCoordinator” message to the devices in range. In the same way, a client can send a “TooManyCoordinators” to the devices in range after starting a coordinator election process when he notices there is more than one coordinator. During the coordinator election, the client can send two types of objects. During the election, when the client wins, he sends the new “VOTES” variable of the server. On the other hand, the client sends the new “VOTES” variable of the server as well as the devices it has already beaten and the devices still in range in an “election” message. If the client lost, it receives a coordinator message with the Bluetooth address of the new coordinator.

When a device looks for a specified service that can’t be found in the Bluetooth environment, then if one of the devices in range has a WLAN service, the device can query for the requested service via the WLAN. This implementation is based on the draft “Service Discovery in On-Demand Ad Hoc Networks” [43]. This protocol provides a way to discover services along with routes to those services. The AODV routing protocol, described in section 2.3.1, is extended to provide service discovery ability within the ad hoc network. The query for a service is done via an extension to the RREQ message, while the node with a service matching the query answers back via an extension of the RREP message.

Although the java implementation of the AODV routing protocol from the University of Bremen [24] was used, few changes had to be made to extend their implementation to support has proposed service discovery in on-demand ad hoc networks. The class architecture is shown on figure 4.2.

When extending their original code, all new functionality has been placed in a separate unit shown as AODV unit (AODVU) in figure 4.2. The class msg was extended with two new types of messages, the Service Request extension (SREQ) message and the Service Reply extension (SREP) message. Thus, an intermediate

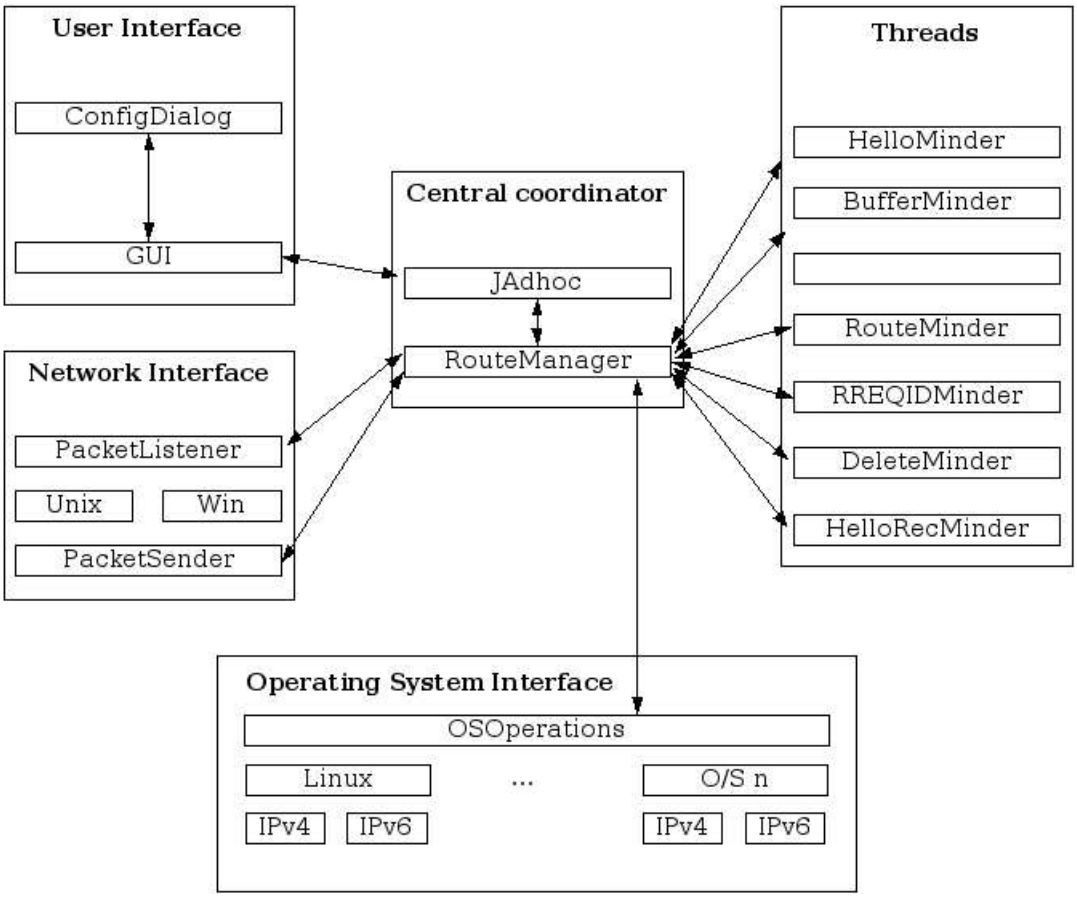


Figure 4.2: Class architecture of JAdhoc

processing node extending the class “net” treats the RREQ as a SREQ message since the service request extension is present.

As stated earlier, the implementation of MIPMANET could be based on the java implementation of MIP from the University of Bremen [25]. As described in [17], the new functionality was also placed in a separate unit called MIPMANET Implementation Unit (MIU). As shown in figure 4.3, MIU extends the foreign agent itself. As this extension has been well described and evaluated, this part was **not** implemented in our application.

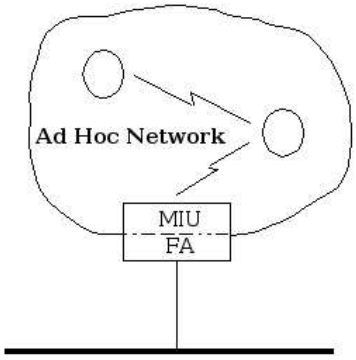


Figure 4.3: Location of the MIPMANET implementation unit (MIU) in the MIP foreign agent

## 4.3 Results

### 4.3.1 Tests

The main test measured the time a device takes to discover both the other devices in range and their services. A comparison was made between the native Bluetooth service discovery protocol and the coordinator-based service discovery protocol as implemented. Additionally, the time needed to retrieve the available services in different situations is compared. First, when the network has no coordinator present, then, when a coordinator is already within range.

To realize the tests, some pratical details need to be addressed according to the device used. To activate Bluetooth on a device using bluez, the command line on a shell, $hciconfig\ hci0\ up$ , is used. Local services need to be run as well. For example, to add the LAN service, the two command lines $sdpd$ then $sdptool\ add\ LAN$ have to be given. On the iPAQ, which supports Bluetooth natively, the command line $hciattach\ /dev/tts/1\ any\ 921600$ sets up Bluetooth; services need to be set as well on the iPAQ. On any device, the command line $hciconfig\ hci0\ class\ 0x4000$ sets the bit #14 of the major service class field to 1. In this implementation, the command $exec$ is used to set this bit from within a program. Thus, the absolute path of the command $hciconfig$ needs to be known and is implementation specific.

To send the data from device to device, serializable objects are needed. Thus, string elements are used to send the pieces of information from device to device. Three hashtables are sent to identify the services available in range. Their key values are addresses of the devices in range as strings. Their values are

| Native SDP | one device | two devices | three devices |
|---|---|---|---|
| inquiry time (ms) $t_{INQ}$ | 10262 | | |
| service time (ms) $t_{SDPC}$ | 3571/2818 | 7343 | 9138 |
| application time (ms) | 13256 | | |

Table 4.1: Test result of the native SDP

| implemented SDP | iPAQ | PC |
|---|---|---|
| initialization time (ms) | 2762 | 760 |
| inquiry time (ms) | 10592 | 10469 |

Table 4.2: Test results of the implemented SDP

respectively the name, the string of services stored in a vector, and the respective URL of each of these services (also stored as a vector).

All the tests were made one hundred times. The results presented here are the average time for each of these tests. Table 4.1 shows the tests corresponding to the native service discovery protocol. The other results in tables 4.2, 4.3, and 4.4 are the tests of the coordinator-based service discovery protocol implemented. The tests of the coordinator-based SDP were made with Bluetooth devices closed to each other, i.e., less than 50 centimeters apart. The influence of the distance between the devices wasn't taken into account.

The definition of the time described in the tables are quite intuitive. The inquiry time is the time needed to discover all the devices in range. The service time defines the time the application takes to discover the available services of the devices discovered during the inquiry. The applicaton time corresponds to the time the application takes without the initialization time. The initialization time is the time the application takes to initialize all the common values stored in the Data class. Finally, the election time is the time the election of a coordinator takes.

| implemented SDP | PC coordinator, iPAQ client | iPAQ coordinator, PC client |
|---|---|---|
| service time (ms) $t_{SDPC}$ | 3087 | 3591 |
| Application time (ms) | 13727 | 14086 |

Table 4.3: Test results of the implemented extended SDP, when a coordinator is in range

| imlemented SDP, no coordinator in range | device becoming coordinator | device becoming client |
|---|---|---|
| Election time (ms) | 2155 | 3906 |
| service time (ms) | 2179 | 3983 |
| Application time (ms) | 12807 | 14758 |

Table 4.4: Test results of the extended implemented SDP, election time

### 4.3.2 Analysis

In the native Bluetooth SDP, a device has to establish a peer-to-peer connection with all the devices in range to retrieve their available services. The time to discover n devices in range and their services is the sum of the inqury time $t_{Inq}$, the time needed to connect to a device $t_{Con}$, and the time to retrieve the list of services $t_{NSDP}$ [11].

$t = t_{Inq} + \sum_{i=1}^{n-1}(t_{Coni} + t_{NSDPi})$

On the other hand, for the coordinator-based service discovery protocol, the coordinator provides a table of all devices in range and their services. Thus, a device entering an already established network only needs to establish a peer-to-peer connection with the coordinator to retrieve all the available services. The time to discover n devices in range is the sum of the inquiry time $t_{Inq}$, the time needed to connect to the coordinator $t_{Con}$, and the time to retrieve the list of services $t_{SDPC}$. This process is **independent** of the number of devices.

$t = t_{Inq} + t_{Con} + t_{SDPC}$

As shown in table 4.1 and 4.2, the inquiry time corresponds to the value of the Bluetooth specification, 10240 ms, as explained in section 2.1.1. In the native SDP, the time needed to connect to a device $t_{Con}$ and the time to retrieve the list of services $t_{NSDP}$, as defined above, are, as expected, almost of the same order. As seen in table 4.1, the time of discovery of one, two, or three devices is between three and four seconds.

The tests of the coordinator-based SDP implemented in tables 4.3 and 4.1 show that the application time, when there is already a coordinator in range, is almost the same as the time of the native SDP to discover *only one device*, respectively 14086 ms and 13256 ms. Similarily, the time to elect the coordinator between two devices is of the same order as the time for a device to retrieve the services from another device using the native SDP. This time is approximately three seconds as shown in tables 4.4 and 4.1. Thus, during the election of the coordinator between n devices, there will be (n-1) connections. It takes about 3 seconds to retrieve the information from each device. However, following this, each device entering

the established network only needs to connect to the coordinator. Decreasing the application time is only possible if there is already a coordinator in range. If not, the time required to learn of all the nodes and their services is almost the same time as the native SDP.

If there is two coordinators in range, the election of the new coordinator is done between these two coordinators. Since the time of election and the effective time of the application after the inquiry time are almost the same, the election of a new coordinator between two devices already known to be coordinators and between two devices who are non-coordinators was expected to be the same. The results of the tests done are stored in Appendix B. In the application, first the device checks the CoD of all the devices found during the inquiry. The coordinator(s) found are stored in a vector. Based on the length of this vector, a decision is made. If there is no coordinator or too many coordinators in range, the application starts the election of a coordinator respectively either between all the device in range, either between the coordinators in range. The tests done show that the time of election when there is two coordinators in range and when there is no coordinator in range is approximately the same.

The initialization time of the iPAQ is higher than for the PC. This is due to the clock speed difference between the two devices. However, the time to retrieve the services from a coordinator is not influenced by this difference as seen in table 4.3. The application time is nearly the same for the iPAQ and the PC. The time to elect a coordinator is different if the iPAQ starts the application than if the PC does it. This difference is due to the fact that the iPAQ becomes the coordinator and the PC doesn't.

Some problems remain in the application. One of the first devices creating a new network will be the coordinator of all this future network. However, this new coordinator may not be the most appropriate coordinator in terms of efficiency, due to the difference in performance and availability of resources. In the same way, if two established networks are brought together, some devices may not be able to connect to all the Bluetooth devices discovered.

The version of Linux on iPAQ used to run the tests was not stable. The iPAQ never turns off and needs to be always plugged in. In the same way, the Bluetooth capability of the iPAQ shuts down from time to time. The iPAQ then needs to be rebooted. Thus, the application is quite unstable. The test results shown in Appendix B illustrate the differences among the measurements.

The version of Linux used on iPAQ, the "familiar" release from handhelds.org, has two others problems. The GUI doesn't work due to a problem of java on the iPAQ. In the same way, the command line $exec$ also doesn't work. Those problems didn't prevent the tests from running. They simply caused more manual

configuration. These kinds of problems may be solved when Linux on iPAQ and java on iPAQ become stable.

# Chapter 5

# Conclusion

## 5.1 Conclusions

The implementation of the method proposed to enhance the native SDP reached its goals; the implemented service discovery allows faster service discovery than otherwise possible with the native SDP. The tests showed the performance of the application in different cases. If there are no coordinators in range or only one device is in range, the application takes just as long time as the native SDP. If there are many devices in range and there is already a coordinator elected, the application is more effective than the native SDP.

Furthermore, the inquiry time, fixed by the Bluetooth specifications, remains the most consuming part of the application. However, if a user entering the PAN discovers the coordinator before the end of the inquiry process, the application could be much faster. If a second coordinator is in range, then the user can send a TooManyCoordinators message to this node to trigger a new election for a single coordinator.

However, the application has some limitations. The use of linux on the iPAQ is the base of this. Linux on iPAQ is not yet stable and the implemented SDP faced some problems due to this instability. Futhermore, java on iPAQ takes lots of memory space. Thus, little memory remains to add new applications. This may be a problem for smaller devices.

## 5.2 Future work

Connections via wireless interfaces on personal devices pose additional security threats, which were not considered in this paper. The security threats can lead to

passive or active attacks. Due to the nature of radio, it may not be possible to limit the range of a signal propagation to within the desired volume of space. Thus, packets can be susceptible to wireless packet sniffers, which monitor the airwaves, capture data, compute the encryption key, and allow unauthorized users entrance into the wireless networks. Thus, passive attacks such as eavesdropping or traffic analysis, or active attacks such as masquerading, replay, message modification, and Denial-of-Services can occur. Moreover, security features are sometimes not enabled, even when installed, cryptographic keys may be too short, shared or not updated automatically and frequently. All of this makes the networks more vulnerable to attacks.

The implemented service discovery can face Denial-of-Services. A user entering the PAN pretending to be a coordinator can get all the pieces of information of the devices in range and hide them to the devices in range. In the same way, a user entering the PAN with a “VOTES” variable sets to a high number can become the coordinator of the PAN.

In the Bluetooth and WLAN capable devices networks, the Bluetooth PAN will seem transparent to the WLAN networks. The node supporting both the WLA and Bluetooth technology will forward all the requested services to the coordinator. Thus, the performance of the proposed solution should be as efficient as the one of the AODV routing protocol used to access the services on the WLAN.

Restricted nodes with very limited memory and computational resources cannot accomplish native SDP. Thus, only the devices which have access to the restricted nodes can communicate with it. To make restricted nodes discoverable by other devices a additional protocol data unit (PDU) called SDP_ServiceRegister PDU can be used [11]. Thus, restricted nodes may be enabled to register their services. Those nodes don’t need to support their own SDP server. Even these resources can be retrieved from the coordinator if one of the discovered nodes does hava access to the restricted node.

# References

[1] Adaptive and Context-Aware Services (ACAS) project, Wireless center, Royal Institute of Technology, 21 March 2003. http://psi.verkstad.net/acas

[2] Bluetooth consortium, “Bluetooth Core 1.0B”, January 2003.https://www.bluetooth.org/docman2/ViewProperties.php?group_id=53&document_content_id=330

[3] Bluetooth consortium, “Personal Area Networking Profile”, February 2003.

[4] Bluetooth consortium, “Bluetooth Assigned Number”, February 2003. https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers

[5] Bluetooth consortium, “Bluetooth Network Encapsulation Protocol (BNEP) Specification”, February 2003.

[6] Kuijpers G., Nielsen T. T., and Prasad R. “Optimizing Neighbor Discovery for Ad hoc Networks based on the Bluetooth PAN Profile”, in: 5$^{th}$International Conference on Wireless Personal Multimedia Communications (WPMC 02), October, 2002, pp.203-207.

[7] Chairs: Macker J. and Corson S., “Mobile Ad Hoc Networks, IETF working group”, June 2004. http://www.ietf.org/html.charters/manet-charter.html

[8] Perkins C. E., “RFC 2002: IP mobility support”. Oct. 1996, updated by RFC 3220. Status: PROPOSED STANDARD.

[9] Md. Asraful Islam, “Development of a portable implementation of the bluetooth service discovery protocol”, May 2003. http://www.wireless.mnsu.edu/2003%20symposium/Presentations/Development%20of%20a%20Portable%20Implementation.pdf

[10] Miklòs Gy., Rácz A., Turányi Z., Valkó A. and Johansson P. “performance aspects of bluetooth scatternet formation”, in: MobiHoc, the 1$^{st}$ annual workshop

on Mobile Ad Hoc Networking and computing, August 2000, Boston. http://comet.ctr.columbia.edu/~zoltan/BTscatternet_mobihoc00_abs.pdf

[11] Sedov I., Preuss S., Cap C., Haase M., and Timmermann D. "Time and energy efficient service discovery in Bluetooth", in: Proceedings of the 57$^{th}$IEEE vehicular technology conference, Jeju, Korea, 2003. http://citeseer.nj.nec.com/570807.html

[12] "Jini technology core platform specification", Sun MicroSystems, Inc., 2004. http://wwws.sun.com/software/jini/specs/jini1.2html/coreTOC.html

[13] "Universal Plug and Play Device Architecture", June 2000. http://www.upnp.org/download/UPnPDA10_20000613.htm

[14] Salutation technologies, 2002. http://www.salutation.org/techno.htm

[15] Cheshire S., Aboba B., Guttman E., IETF working group, "Dynamic Configuration of IPv4 link-local Addresses (Autonet)", work in progress, July 2004. http://www.ietf.org/internet-drafts/draft-ietf-zeroconf-ipv4-linkloca
txt

[16] Salonidis T., Bhagwat P., Tassiulas L., and LaMaire R., "Distributed topology construction of Bluetooth Personal Area Networks". *Infocom.* 2001. http://www.ieee-infocom.org/2001/paper/785.pdf

[17] Jönsson U., Alriksson F., Larsson T., Johansson P., Maguire Jr. G. Q., "MIPMANET - Mobile IP for Mobile Ad Hoc Networks". *IEEE Press*, 2000, pp. 75-85. *http://ce.sejong.ac.kr/~dshin/Papers/MOBILE/pdf/p2-1.pdf*

[18] JXTA project, 2003. http://www.jxta.org/

[19] Jeong J., Park J., and Kim H., "Auto-Networking Technologies for IPv6 Mobile Ad Hoc Networks", February 2004. www.adhoc.6ants.net/~paul/publications/international-conference/icoin2004-jaehoon.pdf

[20] Chair: Guttman E., "Zero Configuration Networking, IETF working group", January 29, 2004.http://www.ietf.org/html.charters/zeroconf-charter.html

[21] The Vovida Open Communication Application Library (VOCAL), March 2004. http://www.vovida.org/

[22] MICA Motes, developed by UC berkeley research group, 2004.http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf

[23] Rococo, Impronto Developer Kit, 2004. http://www.rococosoft.com/products/

[24] University of Bremen. UoB JAdhoc - AODV Implementation in java, July 2004. http://www.aodv.org/modules.php?op=modload&name=UpDownload&file=index&menu=6

[25] University of Bremen. Mobile IP Implementation in Java, July 2004. http://www.aodv.org/modules.php?op=modload&name=UpDownload&file=index&menu=6

[26] Kanoksri S. “IEEE 802.11b “High Rate” Wireless Local Area Networks”, March 2001. http://alpha.fdu.edu/~kanoksri/IEEE80211b.html

[27] “IEEE 802.11b Wireless Networking Overview”, Microsoft TechNet, The Cable Guy, March 2002. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/columns/cableguy/cg0302.asp?frame=true

[28] Jonvik T.E., Engelstad P. and Thand D.V., “Ad-Hoc Formation of Bluetooth Piconet and IP Allocation in PAN”, *IEEE Press*, 2002, pp.489-493.

[29] Nordbotten N.A., Skeie T. and Aakvaag N.D., “Service Discovery in Bluetooth Scatterents”, Workshop on Mobile Ad Hoc Networking and Computing (MADNET), Sophia-Antipolis, France, March 2003.

[30] Greede A. and O’Mahony D., “A Service Driven Protocol For Bluetooth Scatternets”, in: proceedings of 5$^{th}$European Personal Mobile Communications Conference, Glasgow, Scotland, April 2003, pp.307-311.

[31] Avancha S., Joshi A. and Finin T., “Enhanced Service Discovery in Bluetooth”, *Computer*, Volume 35, June 2002, pp. 96-99. http://www.computer.org/computer/co2002/r6096abs.htm

[32] Famolari D. and Agrawal P., “Architecture and Performances of an Embedded IP Bluetooth Personal Area Network”, in: proceedings of the international conference on personal wireless communications, India, 2000, pp. 75-79.

[33] Baatz S., Frank M., Kühl C., Martini P. and Scholz C., “Adaptive Scatternet Support for Bluetooth using Sniff Mode”, in: proceedings of the 26$^{th}$annual conference on local computer networks, Tampa, Florida, November 2001, pp. 112-120.

[34] Robert M., “Bluetooth: a technical overview”, 3 parts.

[35] Kardach, James, “Bluetooth architecture overview”, *Intel Technology Journal*, 2nd quarter 2000. http://www.intel.com/technology/itj/q22000/articles/art_1.htm

[36] Karlsson J. and Persson A., “Device and Service Discovery in Bluetooth Network”, Blekinge Institute of Technology, Dpt of Telecommunications and Signal Processing, June 2002. http://www.inst-informatica.pt/v20/cid/biblioteca_digital/telecomunicacoes/200206BluetoothNetworks.pdf

[37] “UPnP, Jini and Salutation - A look at some popular coordination frameworks for future networked devices”, TechCenter - Systems Programming and Network Programming. Http://www.cswl.com/whiteppr/tech/upnp.html (November 2003)

[38] Bhagwat P., Rao S.P., “On the characterization of bluetooth scatternet topologies”, Technical report, Dpt of CS, univ. of Maryland, USA., 2001. http://www.cs.umd.edu/~pravin/publications/publist.htm

[39] Overview of wireless security threat and risks. http://www.acns.fsu.edu/network/pdf/Overview%20of%20Wireless%20Security%20Threats%20and%20Risks.pdf (December 2003)

[40] Davies J., “Introduction to IP version 6 White Paper”, informIT, November 15, 2000. http://www.informit.com/content/index.asp?session_id={DCE2A13E-A7D0-47D0-A43E-0D3433185628}&product_id={D003910C-BA0D-4

[41] Rekhter Y., Moskowitz B., Karrenberg D., de Groot G.J. and Lear E., IETF working group, “Address Allocation for private Internets”, February 1996. http://www.ietf.org/rfc/rfc1918.txt

[42] Internet Protocol version 6 (IPv6). http://www.ncs.gov/n2/content/tibs/html/tib97_2/content.htm

[43] Koodli R., Perkins C.E., MANET working group, “Service Discovery in On-Demand Ad Hoc Networks”, Internet draft, October 2002. http://www.ietf.org/internet-drafts/draft-koodli-manet-servicediscovery-00.txt

[44] Guttman E., Perkins C.E., Veizades J. and Day M., “Service Location Protocol, version 2”, Request for comments (Proposed Standard) 2608, IETF working group, June 1999. http://www.ietf.org/rfc/rfc2608.txt

[45] The Apache ANT project, “Apache ANT 1.6.2 Manual”, june 2004. http://ant.apache.org/manual

# Appendix A

List of acronyms and abbreviations

| Acronym or abbreviation | Writing out in full |
|---|---|
| ACAS | Adaptive and Context-Aware Services |
| ACK | (Packet) acknowledgment |
| AM_ADDR | Active Member ADDRess |
| AODV | Ad hoc On-demand Distance Vector |
| AODVU | AODV Unit |
| API | Application Program Interface |
| AR_ADDR | Access Request ADDRess |
| BD_ADDR | Bluetooth Device ADDRess |
| BNEP | Bluetooth Network Encapsulation Protocol |
| BSS | Basic Service Set |
| CAC | channel Access Code |
| CCA | Clear Channel Assessment |
| CCK | Complementary Code Keying |
| CoD | Class of Device |
| CRC | Cyclic redundancy Check |
| CSMA/CA | Carrier Sense Multiple Avoidance/ Collision Avoidance |
| CTS | Clear To Send |
| DAC | Device Access Code |
| DCF | Distributed Coordination function |
| DIFS | Distributed IFS |
| DNS | Domain Name System |
| DSR | Dynamic Source Routing |
| ESS | Extended Service Set |

| | |
|---|---|
| FA | Foreign Agent |
| FHS | Frequency-Hopping Synchronize |
| GIAC | General IAC |
| GN | Group ad-hoc Networks |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |
| HA | Home Agent |
| HCI | Host Controller Interface |
| HMI | Human Machine Interaction |
| IAC | Inquiry Access Code |
| IBSS | Independent Basic Service Set |
| IETF | Internet Engineering Task Force |
| IFS | Inter Frame Space |
| IP | Internet Protocol |
| ISM | Industrial, Scientific, and Medical |
| JXTA | Sun Microsystems' set of open-source peer-to-peer networking protocols for JuXTAposition |
| LAN | Local Area Networks |
| LAP | Lower Address Part |
| LLC | Logical Link Control |
| L2CAP | Logical Link Control and Adaptation layer Protocol |
| LMP | Link Management Protocol |
| MAC | Medium Access Control |
| MANET | Mobile Ad hoc NETworks |
| MH | Mobile Host |
| MICA Mote | a commercial wireless sensor networks |
| MIP | Mobile IP |
| MIU | MIPMANET Implementation Unit |
| NAP | Network Access Point |
| NAPT | Network Address and Port Translation |
| NOS | Network Operating System |
| NSAP | Non-significant Address Part |
| OBEX | OBject EXchange |
| OLSR | Optimized Link State Routing |
| OSI | Open Systems Interconnections |

| | |
|---|---|
| PAN | Personal Area Networks |
| PANU | PAN User |
| PCF | Point Coordination Function |
| PDU | Protocol Data Unit |
| PHY | PHYsical layer |
| PLCP | Physical Layer Convergence Protocol |
| PM_ADDR | Parked Member Address |
| PMD | Physical Medium Dependent |
| QPSK | Quadrature Phase Shift Keying |
| RERR | Route Error |
| RFID | Radio Frequency Identification |
| RFCOMM | Serial cable emulation protocol based on ETSI TS 07.10 |
| RREP | Route Reply |
| RREQ | Route Request |
| RTS | Request To Send |
| SDP | Service Discovery Protocol |
| SIFS | Short IFS |
| SIP | Session Initiation Protocol |
| SREP | Service Reply Extension |
| SREQ | Service Request Extension |
| TBRPF | Topology Dissemination Based on Reverse-Path Forwarding |
| TCP | Transmission Control Protocol |
| UAP | Upper Address Part |
| UDP | User Datagram Protocol |
| UMTS | Universal Mobile Telecommunications System |
| URL | Uniform Resource Locator |
| VoIP | Voice-over IP |
| VOCAL | Vovida Open Communication Application Library |
| WLAN | Wireless Local Area Networks |

# Appendix B: Test results

The definitions of the time described in the tables below can be found in section 4.3.1. All times in milliseconds.

| Implemented SDP | Average | Minimun | Maximum |
|---|---|---|---|
| Time of initialization | 2762 | 2513 | 3099 |
| Time of inquiry | 10592 | 10279 | 10828 |
| Time of services retrieval | 3087 | 2438 | 5160 |
| Time of application | 13727 | 13037 | 15816 |

Table 2: Retrieval of services to a coordinator, iPAQ client

| Implemented SDP | Average | Minimum | Maximum |
|---|---|---|---|
| Time of initialization | 760 | 649 | 892 |
| Time of inquiry | 10469 | 10276 | 11212 |
| Time of services retrieval | 3591 | 1839 | 5992 |
| Time of application | 14086 | 12320 | 16512 |

Table 3: Retrieval of services to a coordinator, PC client

| Implemented SDP | Average | Minimum | Maximum |
|---|---|---|---|
| Time of election | 3907 | 2383 | 4971 |
| Time of error treatment | 3983 | 2390 | 4988 |
| Time of application | 14758 | 12939 | 16501 |

Table 4: Election of a coordinator between two devices, PC client

| Implemented SDP | Average | Minimum | Maximum |
|---|---|---|---|
| Time of election | 2155 | 1792 | 3330 |
| Time of error treatment | 2179 | 1835 | 3351 |
| Time of application | 12807 | 12343 | 13993 |

Table 5: Election of a coordinator between two devices, iPAQ client

| Implemented SDP | Average | Minimum | Maximum |
|---|---|---|---|
| Time of election | 3622 | 1941 | 5858 |
| Time of error treatment | 3746 | 2002 | 5982 |
| Time of application | 14318 | 12582 | 16840 |

Table 6: Two coordinators in range, election of a new coordinator, PC client

| Native SDP | Average | Minimum | Maximum |
|---|---|---|---|
| Inquiry time | 10262 | 10252 | 10344 |
| Application time, one device | 13256 | 11374 | 19186 |
| Service time, one device | 3571 | 2202 | 10316 |
| Service time, two devices | 7343 | 3579 | 19325 |
| Service time, three devices | 9139 | 4886 | 15295 |

Table 7: Native SDP tests result

IMIT/LCN 2004-07

www.kth.se