Royal Institute of Technology (KTH), Dept. of Microelectronics
and Information Technology
Stockholm, Sweden

Wireless Center, Research Center at KTH
Stockholm, Sweden

Politecnico di Torino, Dept. of Telecommunication
Turin, Italy

Reconfigurable Application Networks through Peer Discovery and Handovers

Master of Science Thesis

June 2003

Student:	Roberto Gioacchino Cascella
Supervisor at Politecnico:	Prof. Fabio Neri
Supervisor at Politecnico:	Prof. Carla-Fabiana Chiasserini
Technical Advisor:	Dr. Theo Kanter
Supervisor at KTH/IT:	Prof. Gerarld Maguire

Abstract

This Master thesis work was carried out at the Wireless Center at KTH and it is part of a pilot project. This thesis is conducted for the Institute for Microelectronics and Information Technology (IMIT) at the Royal Institute of Technology (KTH) in Stockholm (Sweden) and for the Department of Telecommunications at Politecnico di Torino in Turin (Italy).

This thesis addresses an area with significant potential for offering services to mobile users. In such a scenario users should have minimal interaction with applications which, by taking into account available context information, should be able to make decisions, such as setting up delivery paths between peers without requiring a third party for the negotiation.

In wireless reconfigurable networks, the mobile users are on the move and must deal with dynamic changes of network resources. In such a network, mobile users should be able to contact other peers or resources by using the current route. Thus although manual configuration of the network is a possible solution, it is not easily used because of the dynamic properties of the system which would demand too much user interaction. However, existing discovery protocols fall short of accomodating the complexity of reconfigurable and heterogeneous networks.

The primary objective of this thesis work was to investigate a new approach at the application level for signaling by taking advantage of SIP's features. The Session Initiation Protocol (SIP) is used to provide naming and localization of the user, and to provide functionality to invite users to establish sessions and to agree on communication parameters. The Specific Event Notification of the SIP protocol provides a framework for the notification of specific events and I believed that it could be instantiated as solution to the problem for reconfigurable application networks.

This thesis proposes a method for providing localization information to SIP User Agents in order to establish sessions for service discovery. Furthermore, this method should consider context meta-data to design strategies effective in heterogeneous networks. A viable solution must support (re)location of users at the application layer when they roam between different wireless networks, such as GPRS and WLAN. An analysis of the implications of the proposed model is presented; in this analysis emphasis has been placed on how this model interacts with existing services.

Acknowledgments

This Master Thesis was performed at Wireless Center at KTH and it is part of a pilot project. I wouldn’t have been able to accomplish it without the help of many people who supported, advised me, and encouraged me. First I would like to express my gratitude to Professor Gerald Maguire Jr., my supervisor at KTH, for his patience, assistance, and for his continuous support during this thesis work; he gave me a lot of hints and helped me by proofreading this report. I was also very fortunate to have Dr. Theo Kanter as my technical advisor. He has helped me during these months by supporting me and suggesting to me how to finalize this thesis.

Many thanks to Professor Fabio Neri and Professor Carla-Fabiana Chiasserini for being my supervisors at Politecnico di Torino and for following my thesis work giving me comments and useful feedback.

I would also thank Andreas Wennlund and Asim Jarrar, with whom I participated in the Pilot project, for their ideas and feedback. I also acknowledge Vahid Mosavat for having helped me to understand the Java code of the SIP UA and for giving me advise on the implementation of the proposed solution. I would like also to thank the people that work at the Wireless Center, where I carried out this thesis.

Then, I would like to acknowledge the support of those who have helped me endured the stress and enjoy the happiness that took me to end my studies, such as Marco Cordella, Pietro Lungaro, Giulio Mola, Clarisse Adouard, Raffaele Digiovanni, and all the others who are not mentioned here by name.

I want to dedicate this thesis to my Parents for their unending support, encouragement, and patience during my studies.

Contents

List of Figures

List of Abbreviation and Acronyms

ALG	Application Level Gateway
AP	Access Point
BSS	Basic Service Set
BSC	Base Station Controller
CXDP	Context eXchange Data Protocol
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DS	Distribution System
ESS	Extended Service Set
FA	Foreign Agent
FTP	File Transfer Protocol
GENA	General Event Notification Architecture
GGSN	Gateway GPRS Support Node
GPRS	General Radio Packet Service
GSM	Global System for Mobile communication
HTTP	Hyper Text Transfer Protocol
IAPP	Inter Access Point Protocol
IBSS	Independent Basic Service Set
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISP	Internet Service Provider
JAXB	Java (TM) Architecture for XML Binding
JVM	Java Virtual Machine
LAN	Local Area Network
LSA	Location Service Area

LSG	Location Service GPRS
MCU	Multi-point Control Unit
NAT	Network Address Translator
PDA	Personal Digital Assistant
QoS	Quality of Service
RMI	Remote Method Invocation
RTP	Real Time Protocol
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SGSN	Serving GPRS Support Node
SIP	Session Initiation Protocol
SOAP	Simple Object Access Protocol
SRG	SIP Registrar for GPRS
SPDP	Service Peer Discovery Protocol
SRW	SIP Registrar for Wide area
SSDP	Simple Service Discovery Protocol
TCP	Transport Control Protocol
TLS	Transport Layer Security
UA	User Agent
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URI	Universal Resource Identifier
URL	Universal Resource Locator
VoIP	Voice over IP
WAN	Wide Area Network
WAP	Wireless ApplicationProtocol
WLAN	Wireless Local Area Network
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Overview

The increasing use of wireless infrastructure and portable technologies, such as laptops, cellular phones, and Personal Digital Assistants (PDAs), puts new demands on network infrastructures and network accessability. Ad hoc networks have changed the role of the user in the network by offering new kinds of services that allow users to communicate without any infrastructure. Furthermore, device mobility and user mobility creates new interesting services designed for reconfigurable networks. The need for connectivity along with this mobility puts high demands on the networks, as interruption of ongoing communications or transactions should be avoided.

Mobile users need to automatically discover services relevant to the their current location, without needing to have complete apriori knowledge of the communication environment. In such a network, the infrastructure should be self-configuring, and should take in account the context of both users and their communication.

1.2 Introduction to the problem area

The rapid growth of wireless-enabled environments has increased the necessity for service discovery protocols in order to dynamically provide users with the services they wish and require. The user is able to move among different wireless networks while using the same user-identifier via a multi-access mobile device or multiple devices; thus, when these (wearable) devices are equipped with multiple interfaces (GPRS, WLAN, Ethernet, IR, etc.), roaming and handovers must be possible between **any** combination of network types. Accessing each type of network requires different routines and, thus, making the changes in connectivity transparent to the user becomes an attractive feature for users.

The user should be able to ask for services, provided by many devices (e.g. printers), or to simply utilize software services, such as file, audio, or video access. A robust and self-configuring wireless network is essential to enable computational devices or users to communicate with each other both autonomously and in response to a user's request. "Ad hoc networking" enables wireless devices to establish communication with each other, anytime and anywhere, even if there is no central infrastructure to provide network connectivity. Ad hoc networks are characterized by dynamic links between nearby nodes, and thus the network structure may change based on the node's mobility and/or node failure. Since devices should adapt dynamically to these changes, they should be smart enough that the user only notices changes in communication speed and delay (for example, by buffering data before a hand over to a slower network or by routing packets via another network).

Furthermore, a user should be able to discover an instance of a service which depends on the user's location. This enables new location-based applications, based on service discovery and local peer-to-peer communication, to be performed more reliably and allows context-sensitive computing to create and exploit a user profile or environment profile by extracting knowledge form a user's context and prior actions. The user or device only has to ask for the service needed and need **not** know how the service will be delivered; furthermore, since the user might not have complete knowledge of the network topology or the location of the desired resource, this discovery should be performed automatically (i.e., with minimal or no interaction with the user).

Making devices and nodes in networks context aware is a first step toward solving the above problems. By gathering and propagating context information, decisions can be made so that networks and hosts dynamically adapt to the changing network environment in support of user needs.

In order to enable this context information sharing, the operations of sensing, storing, exchanging, and analyzing context information must be efficient. This can be useful to provide better user interfaces and user interactions and for selecting optimal end-to-end delivery paths. Much context information can be made available to mobile devices, but probably little of it will be of interest at a given point in time. Different elements of context will be needed in different situations. Selection of what context information to share with **whom** and **when**, becomes an important part of any solution.

Information must be exchanged between hosts, and propagated through the network (and perhaps between networks) in order for hosts to act on and react to this information. At the same time, exchanging this information must leave resources for other communication, and hence must not consume too much bandwidth. Context information must also, when exchanged between terminals, be represented such that it can be understood by all relevant parties. A common agreement of how to represent and request context information is needed.

Once a way of exchanging context information has been established, applications that act upon this "knowledge" can be developed. This information will be used by devices to make intelligent decisions regarding routing of user traffic and service delivery at the end points and at intermediate proxies.

In order to automatically adapt to the network's topology and to exploit context "knowledge", distributed provisioning of services seems to be a better solution for delivering services (i.e., peer-to-peer model), rather than having a central node in the network delivering services. This distributed solution reduces the risk of decreased performance of the system, due to failure of a single node. In such a (distributed) scenario, services are dynamically provided to wireless devices. A peer discovery protocol dynamically discovers peers when the user comes into a new area, in order to provide peer-to-peer delivery for applications preferring local delivery. This is essential, because in ad hoc networks static configuration of devices is unusable due to the dynamically changing topology.

1.3 Problem specification

The scenario described above introduces a new area with interesting opportunities for devices, network operators, and users. The users will play an important role in this; since they can offer services to other entities, both the services and entities need not be coupled to a network operator. Thus, users can cooperate with each other in such a network, this is part of the evolution to a user-centric networking model [32, 31]. Network operators will still play an important role, since in some settings there is a need for a network infrastructure. Furthermore, the users should be able to roam among different network topologies, hence there is a need for both user and device mobility. Devices need to achieve a higher degree of autonomy and utilize intelligence for taking decisions, in order to both meet the user's needs and to provide a simpler interface to the user.

As centralized systems evolve towards decentralized systems, their resources are made available and offered in a distributed fashion [39]. The user should be able to find resources in such a network and to use them without thirdy party negotiation. Distributed peer-to-peer networks can be a good solution to the problem while delivering services to users who are able to discover these services. Follow sections provide a detailed introduction to the problem that are investigated in this thesis.

1.3.1 Towards a Service Discovery Protocol

Technologies to form peer-to-peer networks are already available, but the current decentralized methods of service discovery are inadequated in large networks or ad hoc networks [36]. For example, UPnP [13] is a suite of protocols and system services for *device* discovery and **control** in small to medium size IP networks, but it does not scale well in large networks - due to the service announcement traffic that is generated when the number of services or clients grows [15]. Jini [27] is based on a directory service, which is queried when clients need to discover services. UPnP and Jini both lack efficiency and performance.

Pervasive environments are characterized by services and computational resources dispersed throughout the environment and are generally highly dynamic, since services and devices vary constantly. Thus, pervasive environments cannot be relied upon to have a device permanently present to act as a central server and there is a need for a dynamic method to discover available services at a given time. For instance in Jini, the service availability checking is limited to the lookup service in the federation; a centralized approach cannot have a global view of the available services in the network as the discovery protocol should be designed to work in wide area network. Moreover, the number of transmission (number of service updates) should be kept low as possible to reduce battery power consumption [12].

Ad hoc networks (see further section 3.2.2) change dynamically in structure based on the movement of users. This technology introduces new issues with respect to data reliability for discovery protocols due to the frequent disconnections of devices. A centralized directory discovery would lack consistency due to these interruptions in the communication, while an architecture based only on broadcast or multicast messages requires a large number of messages to keep data consistent [25].

1.3.2 Location and context based services

In order to offer location and context based services, users should be able to package this information as services [57]. Context information can also be of great interest to determine the best delivery path for content data according to the network conditions. Location information is useful in order to provide the user with nearby resources if available. However, current systems do not offer adequate solutions for these kinds of applications [15]; location and context based services need support access to data, such as location or sensor data. Some examples of these services are illustrated in [33].

1.3.3 Naming and localization of users

Discovery protocols rely upon an IP address for routing packets in the network. Using IP address to contact peers in the network rises the problem of how users learn the correspondent's IP address i.e., where the user is currently reachable. Thus, localization of users in ad hoc network, when the network changes dynamically and users change both their location and IP address (due to user mobility) is difficult. Furthermore, the ability of users to package services and deliver them to other is useless, if there is no naming to identify and locate users wherever they move.

The Session Initiation Protocol (SIP) provides a naming scheme and means for locating users; thus a user can be found independently of their location and current network device, thus the user is able to move among different networks and it can always be identified and reachable via their SIP URI. Since the SIP URI contains the identification of the SIP domain, other users can contact the SIP location server for the current mapping while the URI is resolvable to a node which is accessible from other.

1.3.4 Mobility and routing

Users can dynamically change their location and, as mentioned above, a single central service directory could not have consistent information of all the services in the network due to the fact that consistency requires constant updates. Central servers should react to changes in user location (macro mobility) and should track the entity's presence, as central servers are the service repositories. SIP addresses the presence issue by allowing protocol extension for registration and presence.

However, the problem of actually accessing the service has still not been solved and it is independent of locating the entity or the service which this entity is capable of providing. Service might include delivering of content data and there are different strategies to access it, at the application layer or at the network layer. The routing of the content between end points can be done according to the context information and type of content by choosing the best interface and next hop.

1.4 Goal of the thesis

The goal of the thesis is to define a method for how to locate mobile SIP users in reconfigurable networks. The solution may use the SIP Event framework for the design of strategies for dynamic negotiation between end-points. The method should take in consideration relocation (movements) of the "corresponding" user when this user moves to another network (device mobility) and/or adopts another SIP URI (personal mobility). Furthermore, context meta-data can be used to design strategies effective in heterogeneous networks.

1.5 Methods

In order to achieve the goal of this master's thesis, I proposed a discovery process to determine the location of services in reconfigurable networks, both local and wide area networks. This approach takes into consideration the possibly rapidly changing topology of the network and the resulting changes in availability of services. Thus, the client would be able to directly query its peers in the local area to ask for services, furthermore it can use the context server that serves a geographical area (the design of it is addressed in [29]), to determine the SIP URI of the peers in the network .

Although there are many different kinds of wireless networks, here we consider only WLANs and GPRS (described in sections 3.2 and 3.3). The proposed method should operate in both environments without any difference in performance for the discovery phase; although the user may subsequently detect delays in delivery of the service due to the current network conditions and the difference in maximum link bandwidth. Once the mobile user has discovered the peers in its new location, it can request services from them by sending a SIP message carrying a service specification to one or more of these peers based on context information and network connection.

The design of the discovery process considered the existing discovery protocols for peer-to-peer networks (described in sections 2.2 and 2.3), and the use of SIP for localization and event handling (described in chapter 4). Some extensions to SIP for registration and user presence can also be utilized (these are described in sections 5.5.2 and 4.7.2).

Chapter 2

Peer-to-Peer Technology for Service Discovery

The goal of a service discovery protocol is to allow users to dynamically find services in the network. Discovery should be performed in a smart fashion in order to provide the required service where and when it is needed. There are several service discovery protocols, each one with different features. Traditional service discovery protocols and delivery mechanisms have been designed for networks where users do not change quickly their point of attachment and address. Mobile users in such settings are able to contact peers in the network through discovery technologies, such as Universal Plug and Play (UPnP) [13] and Jini [27].

2.1 Peer-to -Peer

The term "Peer-to-Peer" is applied to networks where end users directly share files, computational power, and other resources. Users interact **directly** with each other to perform a service or to exchange data. The main purpose for Peer-to-Peer networks is to **directly** use resources (including storage capacity, processing power, and knowledge/content) of the peer device, resulting in a distributed environment without any central control point (section 1.3). This scenario suits mobile devices well, since they can take advantage of this solution by storing less content, as they might have smaller amounts of storage than fixed stations. In addition, they are able to utilize services else where the network as needed. Mobile users can have true global mobility while having access to a wide range of services that other nodes provide.

Peer-to-Peer technology has already been used for many years, but a client/server scenario has been preferred when the resources of the server where much greater than many clients. The client only needed to initiate a connection to a well-known server, download some data, and disconnect. This approach does not require the client to have a well-known address or even a permanent address. Furthermore, the increasing need for security (resulting in the introduction of Firewalls) and the growing demand for IP addresses (resulting in most networks being behind a NAT) have partitioned the network and have further limited the use of Peer-to-Peer connections.

However, the users can gain by using Peer-to-Peer technology, as it offers the possibility to enhance an individual device's capacity. So, solutions should be found to enable such peer-to-peer networking despite the impediments, such as Firewalls and NAT.

Fully utilizing Peer-to-Peer networking produces a completely decentralized network without central control points. This solution has rarely been applied to existing systems, most of which implement a hybrid solution with a central server, that lists all services available, the resources, and their locations. There are several approaches to implementing a Peer-to-Peer network, I will examine several of these below.

Many recent peer-to-peer applications (such as NAPSTER [1]) present a decentralized face even though they use a central server to store bindings between resources and locations. In a wireless environment, where reconfigurable networks may play an important role, a decentralized solution would be preferable. Due to the dynamic changes in users' locations, a centralized model can suffer from inconsistency of data and from scaling problems of handling all the updating required to preserve consistence (section 1.3.1).

Peer discovery services differ depending on how one locates other peers and how the peers interact [51]. I will examine three alternatives below.

2.1.1 Static configuration

Static configuration of the network means that every peer in a Peer-to-Peer application needs to be aware of other users it will interact with. When a peer enters an existing environment, it needs to learn the list of the peers currently there. With static configuration, each peer must be (pre)configured in order to know the address of all peers, with which this new peer **may** want to establish a dialog with. These correspondent peers might be organized in a list and this new peer might need to have presence information about them. In general, this approach does not scale well in large networks, and suffers from a difficulty in doing all the updates in dynamic networks about presence of peers currently available. It is the case of wireless networks, where the users may move and hence their network location may change; there are many trade-offs between accuracy and the amount of traffic, (both peak and average volumes of traffic) due to user location updates in order to keep the knowledge of the presence of peers consistent. Furthermore, if all peers are not properly preconfigured, the network will have blind spots for some peers.

However, static configuration assures a certain level of security. By preconfiguring all peers, the network will be safer from outside attacks, because each peer establishes dialogs only with the peers which it already knows about and each peer may establish secure relations (for example, based on key exchange).

2.1.2 Centralized model

In a centralized Peer-to-Peer (or Hybrid Peer-to-Peer) model connections are based on the use of a central system (Fig. 2.1), which directs the traffic between individual endpoints. To maximize scalability and to avoid single points of failure, a small number of replicated central servers serve a large number of users. For example, these servers might keep track of the location of shared files stored on the users' personal devices. Each time a user comes into the network, their device registers or updates information about their identity and which services they can provide with the central directory. To request a service, a user queries the central system, which responses with a list of the users currently connected to the network offering the requested service. The user directly contacts one of these peers to establish communication in order to request content or otherwise utilize the service which they offer without any further support by the central server. Thus, only the location of the central directory has to be configured into each peer.

This approach seems to scale well thanks to the presence of a robust central system, which is constantly updated. However, the centralized server is the single point of entry, and it can become congested due to the traffic, i.e. it lacks efficiency, flexibility, and performance to be effective in very large networks. An example of this centralized model is NAPSTER [1]. It has been configured with centralized management and

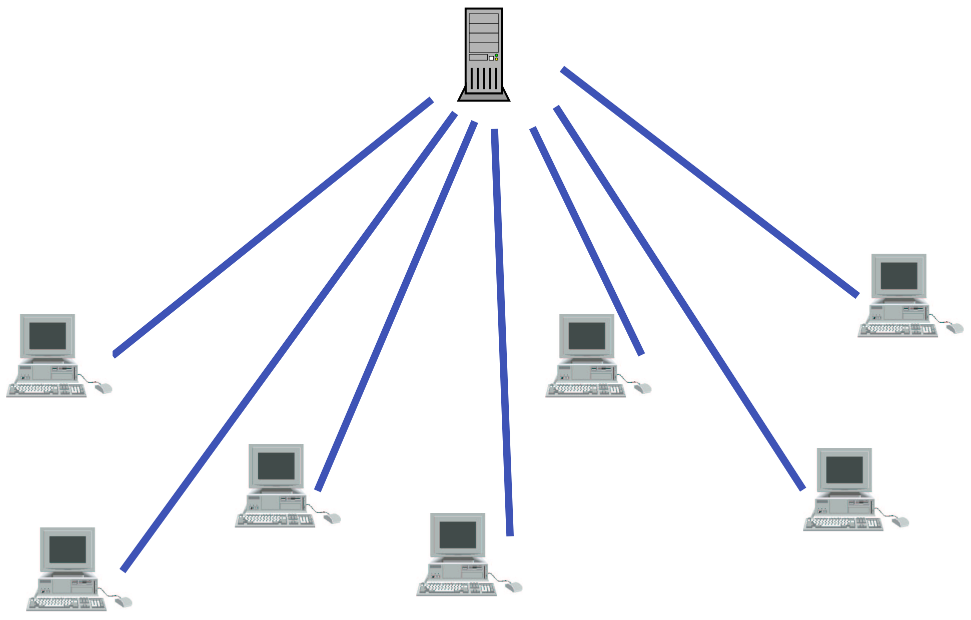

Figure 2.1: Centralized model

administration, which ensures quality of service, but at the same time, made blocking the service easy (in this case for legal reasons [9]).

2.1.3 Decentralized model

A decentralized model consists of a collection of peers, which can dynamically move around the network. There is no (common) central control point, managing or providing content services for the network, and no single entity knows the structure of the network or the identity of all peers located in the network (Fig. 2.2). In such a situation, there are several strategies to discovery other peers.

A purely distributed Peer-to-Peer network is built on top of the IP network, and can be considered as an overlay, or virtual network. Virtual networks are characterized by establishing virtual paths between users, which are addressed using an id that is temporarily mapped to the IP address. Virtual paths are used for both discovery and for transferring content. Furthermore, since no central control point exists, the routing is done entirely by the peers. Strategies such as flooding should be used to find peers and learn about their services. Flooding increases the traffic load in the network exponentially with the number of nodes [47, 49].

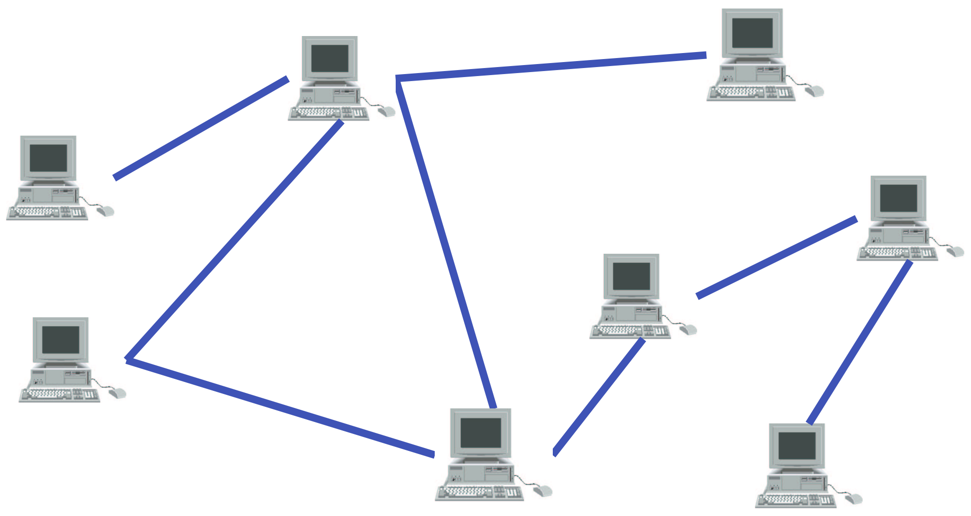

Figure 2.2: Decentralized model

Gnutella [3] proposed and uses a method to discover services based on collaboration between peers. The starting point is to discover another Gnutella peer in the network, when the user connects to the network. The protocol specification does not describe how to do this, but a solution could be to use the Gnutella peers discovered during the last login or to download a list of potentially connected users from preconfigured hosts. In such a situation, the peer will notify all its neighbors of its presence.

To obtain services, a peer sends a request to all its neighbors, which try to satisfy the request; if they can not do so, they forward the request to their neighbors, and so on. When the service is located, the two peers (requestor and supplier) become neighbors and they establish a direct channel to exchange content. Thus, the dimension of the network can increase quickly, therefore a peer can quickly reach another peer even at the edge of the network. A time-to-live (TTL) field and hops counter are attached to the search message to prevent the network from being flooded; each user decrements the TTL and increments the hops counter before passing a request further. When the counters reaches a predefined value, the message is not forwarded further and the message is removed from the network.

This model seems to be more robust than the centralized one because it eliminates the reliance on central servers and the performance of the network does not depend on the load on one particular node. Furthermore, a single point of failure is avoided; this makes Denial of Service attacks nearly impossible.

Another decentralized model is based on multicast messages [51]. A user sends a request on a well-known multicast address asking for the service(s) or to notify others of its presence in the network. In this model the sender does not need to know how many receivers exist and each user does not need to assist with the discovery, unlike in Gnutella where users must forward incoming requests if they can’t satisfy them. However, this technique is limited by the lack of underlying multicast technology and it often can only be used in broadcast subnets, due to the difficulties in routing multicast message across subnets (since few ISPs support multicast).

2.2 Universal Plug and Play

Universal Plug and Play (UPnP) [13] is an architecture for pervasive peer-to-peer network connectivity of intelligent devices. It was developed by Microsoft as an extension to their plug and play peripheral model by adding new features. UPnP defines a set of rules for joining a network, obtaining an IP address, communicating a device’s capabilities, and learning about the presence and capabilities of other devices. It has been designed for home and small business networks to control content transfer and to provide seamless networking by utilizing the TCP/IP protocol and Web technologies, specifically: IP, TCP, UDP, HTTP, and XML. UPnP is media and implementation platform independent, thus, devices can be implemented using any programming language and run on any operative system, offering interoperability while not restricting users unnecessarily.

2.2.1 UPnP network components

The main components of an UPnP network are: devices, services, and control points. Here follows a short explanation of each of these components.

2.2.1.1 Devices

UPnP requires all devices to have an IP address to be located and addressed on the local network. A device can consists of nested devices and services, which, as well as the device properties, are specified in a XML description hosted by the device. The description contains information about device specifications and a list of all embedded devices and services, such as vendor-specific information, a list of all services, and a URL for control and eventing (see sections 2.2.3.4 and 2.2.3.5 for further explanation).

2.2.1.2 Services

A service is the smallest unit of control, and it is described through state variables and actions. As for devices, a service is described in a XML description, standardized by the UPnP Forum; the description includes all actions that can be performed, a list of parameters for each action, the service state table, and it is pointed to by a URL within the service description contained in a device description. The service state table is always updated when the service executes actions after a request and as service state changes, thus generating events that can be published to interested subscribers.

2.2.1.3 Control Points

The UPnP Control Point is a control node in a local network that is capable of discovering new UPnP devices and controlling other devices. When discovery is performed, a control point retrieves device descriptions, device basic configuration parameters, and services associated with the device. If interested in services, the control point can also retrieve their service description(s). After learning more about a service, the control point can invoke actions on the service and it can register itself for eventing. Thus, when a service state changes, a notification will be sent to the control point.

2.2.2 Protocols used

UPnP introduces several protocols to format the messages according to the different steps in the UPnP protocol: the Simple Service Discovery Protocol (SSDP) [24] for services discovery, General Event Notification Architecture (GENA) [28] for eventing, and Simple Object Access Protocol (SOAP) [18] for control.

SSDP defines how services and devices can be discovered within a local area network. SSDP implements an hybrid approach to the discovery process; it allows for both announcement messages and discovery messages. SSDP uses HTTP messages, both multicast (HTTPMU) and unicast. HTTPMU is used to allow control points to discover services by sending search messages (M_SEARCH) on the reserved local administrative scope multicast address 239.255.255.250. SSDP services that match the request respond (NOTIFY) using a HTTP unicast message. Similarly, a device can send a multicast message to announce its presence and its services when entering a new network by issuing a NOTIFY message, or when it intends to leave the network. Services are identified in a message by a service URI and a Unique Service Name. The service is also associated with location information in order to contact it, and with a living time, which indicates for how long the service is available.

GENA provides the ability to send and receive notifications using HTTP over TCP/IP and multicast UDP. GENA also defines the concepts of arbiters, subscribers, and publishers of notifications to enable events. GENA allows users to accept and keep track of subscriptions and to send notifications when a change in service state occurs.

SOAP defines how to format and deliver control messages to devices and return results or errors back to control points.

2.2.3 UPnP protocol overview

UPnP features can be summarized as consisting of five steps, as described in the UPnP architecture specification, along with a preliminary step for address configuration.

2.2.3.1 Addressing

An important feature of UPnP is the automatic configuration of a device's IP address, via AutoIP [53]. This enables a device to auto-configure its IP address. Since each device must have an IP address to be addressed on the network, UPnP requires all devices to have a built-in DHCP client to search for the DHCP server that manages this network. If a server is not present, AutoIP enables a device to auto-configure its IP address by choosing an address from a range reserved for a local network. Following one of this methods of address assignment, the device sends an ARP message to check if the address is already in use.

A host name can be associated with devices to solve problems that may occur in dynamic networks when a device changes its IP address. This can easily be addressed by using a DNS server, however, small networks might not be served by a DNS server for name and address mapping. UPnP solves this problem by introducing Multicast DNS using link-local multicast for queries.

2.2.3.2 Discovery

UPnP discovery allows a device to advertises its presence and its (embedded) services to the control points via multicast SSDP NOTIFY messages. Similarly, when a control point is added to the network, it searches for devices and services of interest via the SSDP M_SEARCH discovery message. The messages exchanged each contain a device or service type, an identifier, and a pointer to more detailed information.

The search message contains an identification of the device or service type, which the control point is looking for. The message has a limited Time-to-Live, the default value is 4, in order to limit network congestion. To be found, all the devices on the network must respond using unicast messages if the request matches their devices or services.

2.2.3.3 Description

Once the control point discovers a device, it needs to retrieve more information about the device's capabilities in order to communicate and to use the services handled by the device. As introduced in section 2.2.1.1, the device is described in a document expressed in XML.

2.2.3.4 Control

After the control point has retrieved a description of the device, it is able to send requests for actions to the device's various services. To invoke a specific action, the control point sends a suitable control message to the service control URL, specified in the device description document. Control messages are expressed in XML using the Simple Object Access Protocol (SOAP). In response to a control message, the service returns an action-specific value after completing the request.

2.2.3.5 Eventing

As described above, a service is defined by a set of variables that model its state. As the state can change dynamically a control point may be interested in modifications to the service state. UPnP eventing describes event subscription and the format of the event messages. These are expressed in XML and formatted using the General Event Notification Architecture (GENA). The control point subscribes to receive service information that is published when variables change. The first event message is sent when the control point subscribes, which indicates the initial state of the service. In order to keep information consistent in all subscribed control points, this message is sent to all subscribers. The subscriptions to a particular event are limited and they can be renewed or canceled at any time.

2.2.3.6 Presentation

If a device indicates in its description a URL for its presentation, the control point can control the device and view its status. To get the presentation page the control point submits a request to the device, which returns the presentation page, written in HTML. A control point can present this page to a user.

2.3 Jini

The purpose of the Jini architecture [27] is to group users and resources into a single dynamic distributed system, named a Jini federation. Within this federation, Jini services can represent both hardware devices or software components which can be used in a flexible fashion by human or computational clients. When a client wants to use a resource, it needs only download the necessary programming in order to communicate with the resource, each object can delivery this software when ask for it. Jini is designed to make the network more dynamic, so that services can flexibly be added or deleted.

The Jini system can be segmented in three parts: services, infrastructure, and programming model. These three categories, which I will describe in later sections, are separable and they are not required to be part of the Jini system. However, for complete functionality, the three parts should be tied together.

2.3.1 Jini requirements and components

Jini is completely built on top of Java. The communication between entities is accomplished using Java Remote Method Invocation (RMI). This gives to the system a complete distributed model. Jini requires that the devices in the network have some memory and some computational power in order to compile and execute the Java code. Hence, the device should have a Java Virtual Machine (JVM) or use a proxy, which controls the device and contains processing power. For further reading or documentation about Java code and RMI see [4, 5].

2.3.2 Services

Jini introduces a new way to consider and use services from the client side, for instance a service is seen as a Java interface, where all methods that are possible to invoke for that service are specified. Thus, services can be anything, and they are the entities that forms the Jini federation. They can dynamically be added or removed from a federation, they can require other services or they can contribute to a particular task. Once the device has discovered via the Lookup Service (see section 2.3.3.2) the new service, it uploads the service proxy to the Lookup Service by using the Join Protocol (see section 2.3.3.3).

Therefore, this approach enables clients to use services; hence, both software or hardware can be accessed in a uniform fashion. This means that a client only needs to download the service proxy to contact the original service and to invoke its methods remotely.

2.3.3 Infrastructure

The infrastructure defines the core of Jini. It consists of a security remote system for the definition of the access of the services, a discovery/join protocol for services to discover, join, or advertise services provided to others in a federation, and a lookup service used as a repository for services and to find other services.

2.3.3.1 Remote Method Invocation (RMI) and security

RMI allows communication between services; it enables clients to execute the java code of the method of the service remotely, where the resource resides, and to upload the result. RMI is a fundamental part of the Jini infrastructure, as it enables move around the code describing the service functionality, encapsulated as an object.

Jini also defines a mechanism to access an object remotely based on a access control list. This method gives a certain level of security and its implementation defines the right to invoke a service object on behalf of the requestor or on behalf of others.

2.3.3.2 Lookup Service

The main component of Jini is the Lookup Service. It can be considered as a directory service, since it maintains dynamic information about the currently available services in a Jini federation. For every service that the client wants they must join a federation, and discover the Lookup Service and must register with it, by using the Discovery and Join protocols. However, the Lookup service is not a requirement for Jini to work, even if the functionalities of the system are reduced.

Services are represented as Java objects in the Lookup service. Entries in the Lookup service can be associated with other objects (hierarchical lookup), that provide access to the service, and with attributes which define a service according to a language that people can understand easily. Furthermore, a Lookup Service can contain objects that refer to other directory services.

2.3.3.3 Discovery and Join Protocols

Discovery and Join protocols identify the process of finding a Jini federation and adding a service. The Discovery protocol is used to find the Lookup Service, to which to register the service. This protocol is based on multicast or unicast messages depending on the knowledge and configuration of the client. Here, the different approaches are listed:

- Multicast Request Protocol is the multicast based discovery protocol. It is used to find the Lookup Service.

- Multicast Announcement Protocol is used by the Lookup Services when they want to announce their presence to clients, that may have an interest in the community.

- Unicast Discovery Protocol occurs when the client knows the location of the specific Lookup Service to establish communications with it. Mainly this protocol is used when the device wants to contact its manufacturer.

Once the device has discovered the service via the Lookup Service, the new service downloads a lookup service object, which is then used to communicate with and invoke the methods the Lookup Service discovered. Using this method, the client can upload the service proxy to the Lookup Service to register itself through the Join Protocol . This protocol uses RMI to load into the Lookup Service the service object, which contains the Java programming interface for the service and methods that can be invoked, as well as it can contains descriptive attributes which can be used in a user interface.

Hence, the client has joined the Jini federation. When a client is interested in a service, it queries the Lookup Service, which performs service discovery based on Java attributes and interface matching. Once the service has been found and the proxy object has been downloaded, the client is able to interact with a service only by using the methods specified by the original service provider of the service. Thus, the developer of the service handles the way the service and the client communicates, and furthermore, this approach enables clients to know how to use the service, because that use is defined by the methods.

If a client can not locate any Lookup Service, it uses the same kind of messages, that the directory service uses to announce its presence, to ask other peers to register their services with it. Then, the providers will register with the client, that can subsequently select the services it needs.

2.3.4 Programming model

The Jini programming model can be seen as a collection of Java interfaces, that are used in the Jini implementation to work in a distributed environment. These interfaces define the communication setting to enable the Jini service to communicate with the Jini infrastructure, and, as cited in the specification, "*these interfaces, taken together, make up the distributed extension of the standard Java programming language model that constitutes the Jini programming model*"[1].

[1]Taken from the Jini Architectural Overview [27].

The main interfaces are:

- The leasing interface defines a model to allocate resources based on release time.
- The event and the notification interfaces handle the events that might occur due to dynamic changes of the network.
- The transaction interface provides coordination between services to provide distributed computing.

2.3.4.1 Leasing

The leasing interface defines for how long a resource is valid. This kind of mechanism guarantees a certain level of reliability; in a dynamic network, where disconnections are frequent, the information about service availability may lack consistency. Leasing means that a resource is available for a limited time. For instance, the registration of the service in the Lookup Service is valid for a limited time period. Furthermore, when a client requests a service, the resource is released for the negotiated time. Thus, if clients need the service for longer time, the release time should be refreshed before it expires. If the client leaves the federation, all resources allocated to provide services are released when the service periods expire.

Jini also allows for lease delegation. This entail a third party negotiating or renewing the lease time on behalf of the service provider. Furthermore, the lease interface defines in which way the resource should be shared; a lease can be either exclusive or non-exclusive. The former indicates that the resource is strictly allocated to only one client, while the latter allows for resource sharing among clients.

2.3.4.2 Events and notification interfaces

The event and notification interface is an extension to Java's eventing by considering the distributed case. Jini provides eventing to notify subscribers when a desired change occurs in the system. For instance, a device can subscribe with a Lookup Service for eventing about particular services in a federation.

As we have seen for the lease time, Jini offers a delegation mechanism for eventing. An entity can handle events on behalf of another device accepting subscription for events of a service. The third party will subscribe for eventing with the service provider and, when a notification occurs, it will send to its subscriber the notification. The subscriber considers the third party as the originator of the notification.

2.3.4.3 Transaction

The transaction interface is used in Jini to coordinate all distributed operations performed in the system. When computation needs to be done distributively, a mechanism that ensures reliability and consistence is required.

An operation performed by different services can fail due to a minimal error, that can occur during the processing time in a service. In this kind of situation, the transaction manager does not consider the case of a partial failure. Thus, a task can succeed atomically or it can fail completely ; in case of failure all the processes involved in the task recover to the previous state. This ensures an easier mechanism to recover from failure than analyzing what was wrong.

The transaction interface consists of two steps. The first is the voting phase where the transaction manager queries the objects to know if they are ready to change state, i.e. they have performed all the operations. While during the second phase the transaction manager, after having received the confirmation from the objects, sends the authorization to change their state. Jini does not specify how to handle this control of the distributed computing, leaving the implementation of the transaction semantics to the service provider.

Chapter 3

Mobile networks

3.1 Introduction

A mobile user should be able to roam between different wireless systems and different technologies. If a Personal Data Assistant (PDA) is provided with multiple interfaces active at a particular time, for instance GPRS and WLAN interfaces, then a device can be connected to both networks and it can switch between them according to parameters, such as the best signal quality or better area coverage.

3.2 Wireless LAN

The IEEE 802.11 standard [26] defines how wireless communication can be provided to portable, fixed, and moving stations within a local area network. There are several extensions to the standard that specify the frequency, coding, media access and control, and the maximum data rate. The standard defines two modes of operation: an infrastructure network and the ad hoc network mode. This section describes the infrastructure mode, while the ad hoc network mode is discussed in section 3.2.2.

The infrastructure mode is defined as a set of stations, forming a Basic Service Set (BSS), controlled by a coordination point, called an Access Point (AP), which connects the BSS to the backbone Distribution System (DS). Several BSSs, interconnected by a DS, form the Extended Service Set (ESS); this enables the APs to communicate among themselves and to forward the traffic enabling movement of stations between different BSSs (Figure 3.1).

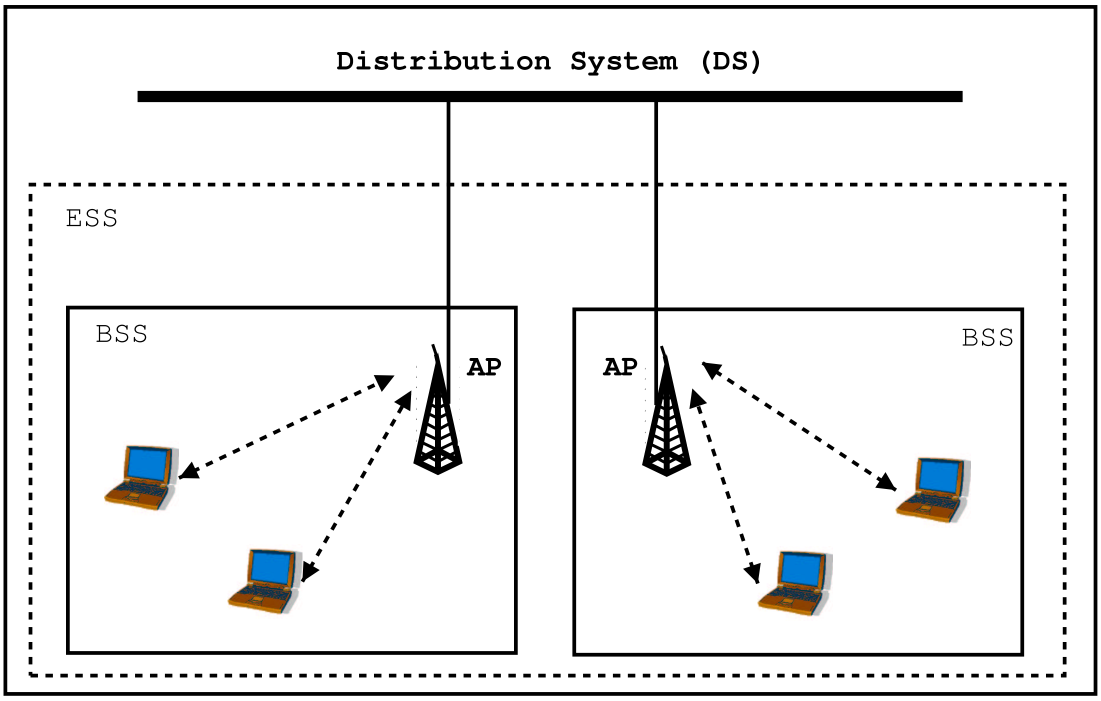

Figure 3.1: WLAN - Infrastructure Mode

The last logical entity defined in the standard is the portal, which provides integration of the wireless LAN with the wired LAN or other wireless LANs, enabling traffic flow between mobile stations, even those not inside the WLAN. Some installations integrate the AP and the portal in the same device.

3.2.1 Registration in WLAN

In order to communicate, a mobile station needs to be associated with an AP. The registration phase is initiated by the device which initiates the scanning process to detect the presence of an AP. The device can act in two modes:

- Passive scanning is performed when a mobile station listens in promiscuous mode to the channel. The wireless interface detects the presence of an AP from the synchronization messages that the AP periodically broadcasts.

- In active scanning mode, a mobile station sends a request in order to find an AP, then waits for a period of time for the response from the AP.

Once the mobile station has discovered the APs in its communication range, it tries to associate with one access point, preferably the one with highest Signal-to-Noise Ratio (SNR). The device exchanges authentication information with the AP to be authenticated and registered.

The mobile station can change its point of attachment (i.e., roams between different BSSs or ESS) if it detects that the signal is weak. If the movement was inside the same ESS, then the mobile station performs a reassociation with the new AP. This new AP will inform the other APs on the same Distribution System (DS) of the new association.

A protocol that provides coordination between APs when the user performs a handover is the Inter-Access Point Protocol (IAPP). This protocol supports coordination between access points of wireless LAN systems of different vendors. In order to provide support for real time traffic, Ren [41] has proposed an extension to IAPP introducing the concept of differentiate handover to discriminate the handover procedure based on resource availability. Her model defines a method to provide and exploit knowledge of neighboring access points.

3.2.2 Ad hoc network mode

An ad hoc network consists of a self configuring wireless network without any infrastructure. As discussed above, the mobile station needs to be associated with an access point in order to transmit a frame to the public Internet. However, even in the absence of an AP, the mobile station is able to communicate with other devices in communication range forming an Independent Basic Service Set (IBSS) (Figure 3.2). Within an IBSS mobile stations can connect to each other, thus organizing an ad hoc network.

Since the packet are not long *only* routed via the AP, in order to reach nodes which are *not* directly reachable each node needs to have routing capabilities in order to rely packets. With such routing the communication is completed distributed, and multihop techniques enable devices out of range to communicate with each other via one or more hops between devices that are directly reachable. Without a network layer ad hoc routing protocol lacks routing capabilities, to provide such features to the network we can either introduce an ad routing protocol or define an overlayer routing scheme for relaying packets between hosts at the application layer.

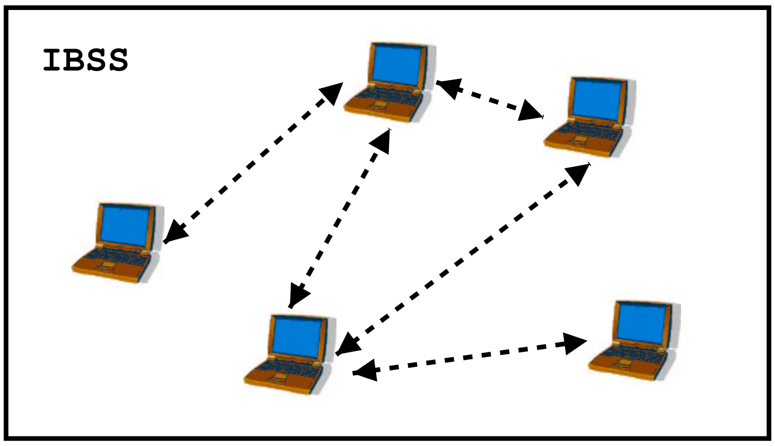

Figure 3.2: Ad Hoc Network

Ad hoc networks are well suited for environments where there are difficulties planning or installing an infrastructure mode network. The use of such an ad hoc network does not scale to a large number of terminals, since the links established between mobile stations to route the traffic are not stable; as the network topology changes, signal traffic will increase with the number of the nodes within the network [23].

3.3 GPRS network

General Packet Radio Service (GPRS) is a standard for packet delivery on a GSM network. Some nodes are added to the GSM system in order to provide functionality to route packet traffic. GPRS does not use circuit switched technology to route data, but rather packet switched technology; which avoid resources being allocated for the duration of a call, thus making more efficient use of the radio channel. Because of this, the user is only charged for the data that he/she transmits on the link, and not for the reservation of resources as in circuit switched networks.

In order to provide such a service, the GPRS system adds to the GSM system two nodes: the Gateway GPRS Support Node (GGSN) and the Serving GPRS Support Node (SGSN). GPRS also requires changes at the Base Station Controller (BSC) (Figure 3.3). The main difference between the two systems for sending data concerns the use of a frequency channel. GSM ordinarily only uses one of the eight timeslots, in which the channel is subdivided, to send the data, while GPRS and HSCSD standards can use several timeslots to send the data, and in GPRS these timeslots can be quickly allocated and deallocated.

This section does not provide any discussion about the GSM network, but it only gives a general overview of the GPRS system. Details about GSM can be found at [30].

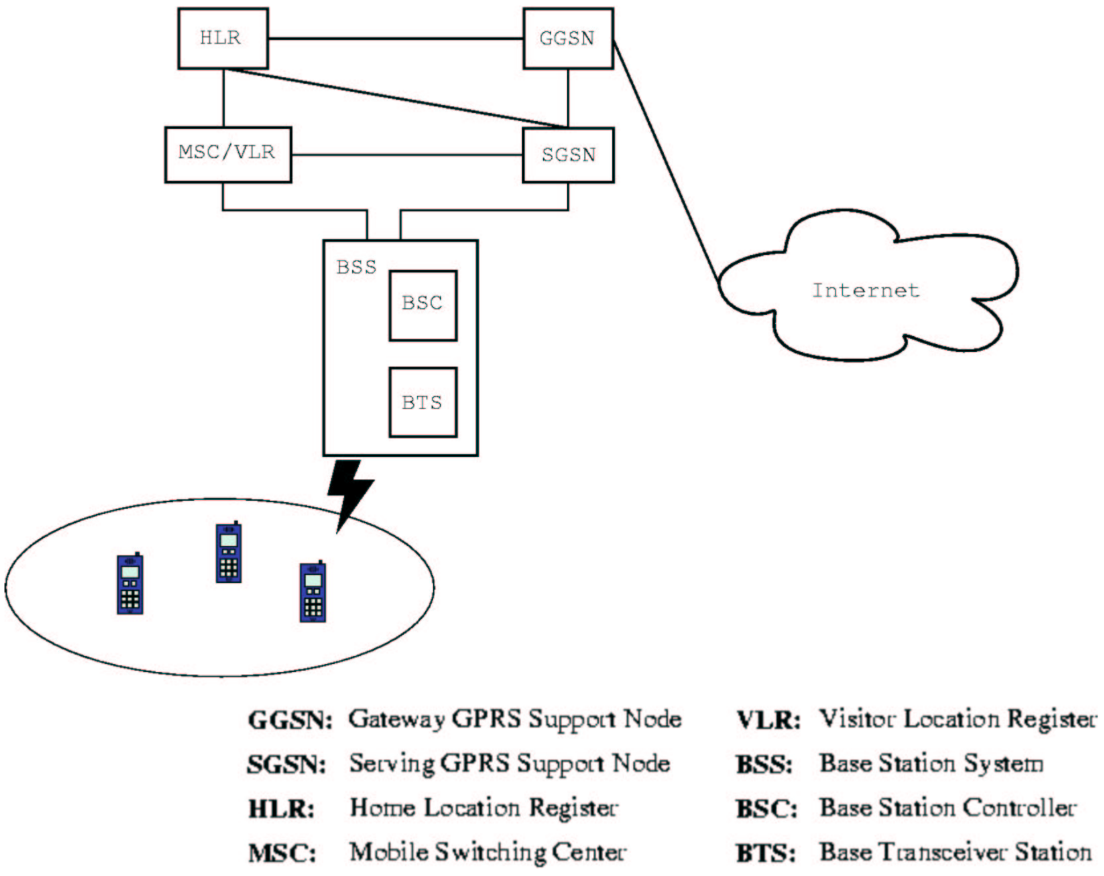
HLR
GGSN
MSC/VLR
SGSN
BSS
BSC
BTS
Internet
GGSN: Gateway GPRS Support Node
SGSN: Serving GPRS Support Node
HLR: Home Location Register
MSC: Mobile Switching Center
VLR: Visitor Location Register
BSS: Base Station System
BSC: Base Station Controller
BTS: Base Transceiver Station

Figure 3.3: The GPRS Network

3.3.1 Registration in GPRS

The registration of a device within the GPRS network consists of an association with a Serving GPRS Support Node (SGSN) and exchange of authentication and authorization information for gaining access to the network. When a mobile device detects a GPRS network, it sends a Packet Data Protocol (PDP) request to the Serving GPRS Support Node (SGSN) in order to instantiate and validate the context information to be able to communicate with the Gateway GPRS Support Node (GGSN). The SGSN responsible for the mobile device forwards the packets to the GGSN node, which registers the user's GPRS context information. The GGSN acts as a Mobile IP Foreign Agent (FA) managing and setting up all the communication for the user. Furthermore, the GGSN node associates the mobile station to the right SGSN node and collects charging information.

Once the mobile station has been granted access to the network, the station is assigned to a SGSN. While the device is on the move inside the same GPRS network (i.e., roaming between cells), it needs to inform the GGSN of each new association. The new association is done via the new SGSN node, that forwards a PDP context to update location information and that informs the old SGSN of the new association, requiring the transfer of the user's information. However, if the association has not yet been established, the traffic addressed to the device that reaches the old SGSN node is forwarded to the new SGSN. The SGSN provides user mobility in addition to authentication and collecting charging information related to the use of GPRS resources.

The GGSN node acts as an external router for all devices registered within a GPRS network. When a user wishes to send data packets to the public Internet, the packet is received by the responsible SGSN node which then encapsulates the packet into another packet and sends in on to the GGSN node. The GGSN node unpacks the data and routes it to the public Internet.

Thanks to the functionalities provided by the GGSN node, a user within a GPRS network is reachable by other users located outside the GPRS network. However, since GPRS providers charge the user for the **incoming** and **outgoing** traffic, full IP connectivity for the user is not necessarily an advantage. Furthermore, saving IP address is also an important issue to consider for the design of the GPRS network. Thus, most operators add a Network Address Translator (NAT) in order to limit the number of IP addresses used and to block unwanted connections destined to the user.

3.4 Mobile IP

Mobile IP [40] is built upon IP and provides network layer mobility to applications and higher protocols regardless to the network attachment point of the host. Since the traffic addressed to a host is routed according to the destination IP address, this must some how correspond to the current location of the host, to do this Mobile IP introduces the use of two IP addresses, a home address and a care-of-address, in order to support host mobility. The home IP address is allocated logically on the home network of the mobile, while the care-of-address is a temporary IP address associated with the current attachment point of the mobile while roaming in foreign networks.

Mobile IP defines two types of mobility agents, the home agent and the foreign agent, the latter is used by the mobile while it is away from the home network. The home agent attracts all packets sent to the mobile at its home address and tunnels them to the care-of-address. The foreign agent keeps track of the mobiles currently visiting its network and cooperates with the home agent to deliver packets to the mobile. The mobile station detects the presence of mobility agents because they broadcast advertisements to announce their presence. Because the mobile listens for these advertisements it can learn of the local foreign agent.

When a correspondent node wishes to communicate with the mobile node, it sends packets to the mobile's home address. Then, the home agent tunnels the packets and forwards them to the foreign agent which detunnels the packets before sending them to the mobile node. This routing scheme is called triangle routing. In such a scenario the mobile does not lose any packets and the only operation that it has to do is to notify the home agent of its current care-of-address. However, the delay introduced by the triangle routing can affect real-time communications and the fact that the packets are tunneled adds to each packet additional overhead.

In order to reduce delays, a more efficient scheme has been proposed, called the route optimization. The mobile notifies the correspondent nodes of its current IP address. However, this implies that the correspondent host must keep track of the mobile's care-of-address and home agent, and that the old foreign agent should forward packets to the new foreign agent.

3.5 Vertical handover

Due to the large overlap of WLAN and GPRS networks, a mechanism is needed to select the best interface to provide better connectivity and to guarantee reliable communication. While the WLAN network might offer a high throughput over a limited geographic area, a GPRS network would provide a low bandwidth service over a wide geographic area (Figure 3.4). In this project, we consider that the user is equipped with a PDA with multiple interfaces, specifically with two interfaces: one for GPRS and one for WLAN. A mechanism providing vertical handover can be a good solution to this problem of sending data using heterogeneous networks [35, 16].

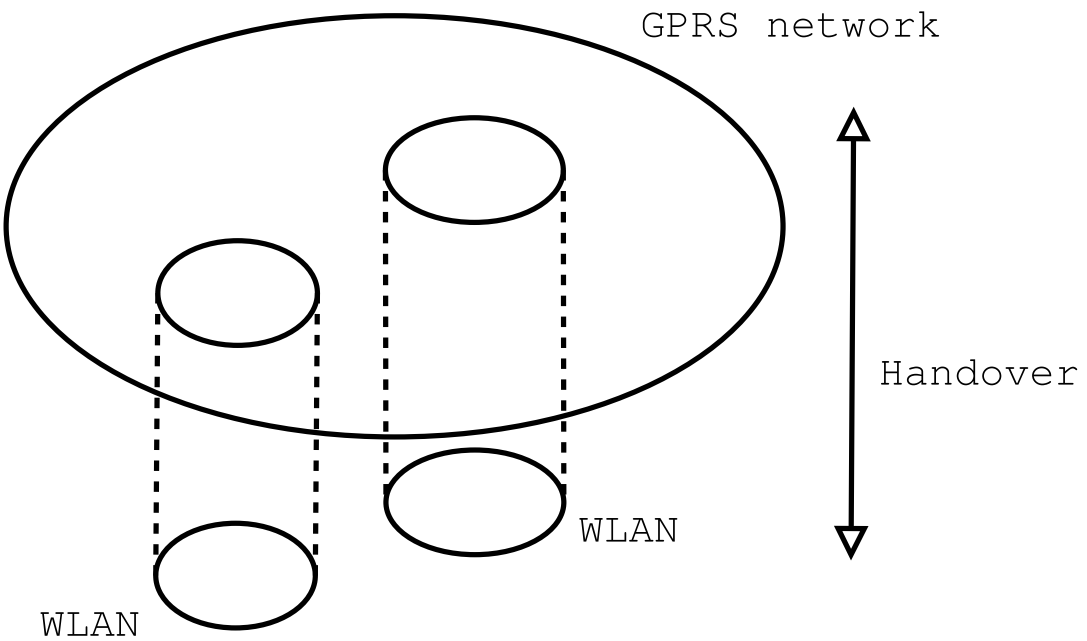

Figure 3.4: Vertical Handover

3.5.1 Handover GPRS to WLAN

The problem of performing a handover between a WLAN and GPRS network, in order to roam outside the coverage area of a WLAN, is analyzed and addressed in Ervenius and Tysk's thesis work [16]. They define two kinds of handover: soft and hard. As discussed above, a GPRS system covers a large area that might include many WLANs. They assume that there is always GPRS connectivity, but there are some spots not covered by any WLAN area. Thus, if a device is using the GPRS interface for transmitting data, the handoff to the WLAN interface in order to have better throughput can be considered soft, since the connection need not suddenly be interrupted. However, the handoff from WLAN to GPRS can be both hard and soft. Soft handoff is performed when a device roams to the GPRS network when it detects that the SNR is becoming weak, but has not yet lost the WLAN signal. Hard handoff is caused by a abrupt interruption of the WLAN communication, the user completely loses wireless connectivity **before** roaming to the GPRS network.

As shown in their studies, Mobile IP was a good solution since it does not require any modifications to the existing WLAN and GPRS protocols. Furthermore, the handoff introduces a delay for packet transmission when TCP is used for the communication; the delay is high when the user switch from WLAN to GPRS since the GPRS network has a Round Trip Time (RTT) lower than a WLAN causing unnecessary retransmissions of the data even if it is still on the way to the destination.

Chapter 4

Session Initiation Protocol (SIP)

4.1 Introduction

The Session Initiation Protocol [20] is an application layer control protocol, that can be used to establish sessions between users who are able to move between different endpoints. SIP provides the necessary signaling for initiating communication and supports user **and** device mobility by using SIP servers. These servers can operate in either proxy or redirect mode. SIP is independent of the lower layer transport protocol, for instance it can use TCP, SCTP (Stream Control Transmission Protocol), or UDP as a transport protocol, and it can operate in conjunction with other application layer protocols, such as the Real Time Protocol (RTP) or Session Description Protocol (SDP), to build multimedia sessions. The default port for SIP depends on the transport protocol in use. It is 5060 for UDP, TCP, and SCTP; and 5061 for Transport Layer Security over TCP (TLS).

SIP does not provide any control of the subsequent data flow between endpoints, nor does it provide any resource reservation mechanism, because SIP messages are carried independently of the subsequent session content. SIP only contacts the peer with which a user would like to establish a session and relies on other protocols for any subsequent user data exchange. During a session, SIP allows users to modify the session's communication parameters and to keep track of all on going sessions.

4.2 Entities

SIP is composed of several types of entities which play different roles:

- A *User Agents* is a logical entity that can act as both a user agent client (UAC) and user agent server (UAS). The client is able to send invitations for a session to a peer and acts as a client for the duration of the session. If it receives a request it assumes the server role.

- A *Registrar* is a server that receives REGISTER requests and processes those belonging to the domain it handles, by updating a location database based on the information carried by the message.

- A *SIP Server* is an entity that receives SIP requests. It is able to process these requests and to send replies. A SIP Server can operate in a redirect or in a proxy mode. In the former case the server sends back a response with the address of the server to contact. In the latter it forwards the request to the next SIP Server. A Proxy server is able to modify messages and acts as a client on behalf of the requesting user. Thus, it contacts external location servers to determine the target user's location. In addition, a Proxy server can operate in a stateful fashion (i.e. maintaining state information) or in a stateless fashion.

4.3 SIP Messages

4.3.1 Methods

The method is carried in the request and identifies the action that the requestor wants to invoke on the server. The method also determines the form of the message and depending on the method there are specific fields which are mandatory in the header.

INVITE, ACK, and CANCEL are methods to set up, acknowledge, or abort sessions respectively. INVITE is the most important method in SIP and is used to establish dialogs to peers or servers. ACK (acknowledge) is the response to an INVITE request. CANCEL is only used to cancel a previous INVITE request sent by the client; the server that receives the CANCEL request immediately stops processing the INVITE request and sends back a final response. A CANCEL request does not have any effect if the server has already answered.

The BYE method is used to terminate a session; it terminates the communication session without requiring any acknowledgment. The OPTION method allows a user agent to query the peer or a server about its capabilities to process messages, for instance to learn what methods and extensions are supported.

The REGISTER request is used to register the user with a registrar. The registration binds a SIP URI to the correspondent's current location. A user agent is able to update and modify bindings stored in the location service by sending a new REGISTER message with a new contact address indicating the new location of the user.

To facilitate extensibility SIP allows the definition of new methods (for the protocol itself) or definition of extensions to the SIP protocol. These definitions need not be supported by all SIP User Agents (UAs), hence UAS can use the OPTION method to check if a given UA supports the desired extension.

4.3.2 Response messages

The SIP response codes are an extension of the HTTP response codes. They can be provisional or final and they consist of a three digit numeric status code. The first digit tells the class the code belongs to, and the other two digits define the response message.

A provisional response, 1XX, is issued by a User Agent Server to notify the sender that the INVITE has been received and that it is processing the message to call the invited party. A provisional response may only be sent in response to an INVITE request.

Final responses are divided into different classes, and they terminate the transaction and may establish a dialog or indicate a failure. The 2XX response indicates a successful dialog establishment and it requires an acknowledgment. The 3XX redirects SIP requests and gives information about the user's new location, which can be permanent or temporary with an associated expiration timer. The 4XX indicates failure, for example the request can be malformed or the callee can be busy or the caller is not authorized to perform the requested method. The response code 5XX is only sent in case of server error; the client can retry the server later. The response code 6XX indicates global failure in the request.

4.3.3 Header field

The SIP header has a similar structure to HTTP. It is written in a human readable format. The header field consists of a field name and a value separated by a colon. SIP defines header fields and rules to use them in a SIP message according to the method and type of message, such as response or request. A full description of the header field and where it is allowed to appear is shown on pages 162 and 163 of [20]. Multiple header values can be used in one field, but it is recommended to order them to simplify message parsing. In this section I present a brief description of the most common header fields.

Accept indicates which formats, i.e., Content-Types, are acceptable in the response. The default value is “application/sdp”. Two other two Accept header fields specify the encoding and the language supported, these can be used in the response for coding files or for a session description.

Allow lists all the supported methods (by the generator of the request). If no Allow header is present, no specific information is given. However, if a user wish to specify all supported methods, an Allow header field is present in an INVITE request.

Call-ID field uniquely identifies a particular invitation or registration of a client. It is always copied in the response to a particular request and it must be a globally unique identifier.

Contact provides a URI where the user would like to be contacted. In case of a REGISTER request, the Contact URI is associated with a preference value among the given locations, named "q", and the expiration timer for this registration is specified. The same parameters can be used in a 3XX response, when an invitation is redirected.

Content-Disposition indicates how the message body should be interpreted. There are several types of Content-Disposition. The default value for application/sdp is "session", which specifies that the body describes a session. Otherwise, the default value for other kinds of Content-Type is "render", that indicates that the body should be displayed to the client. The type "icon" indicates that the body contains an image which could be used as an icon of the caller or callee (depending on who sent it).

Content-Encoding indicates the format chosen to code the body and which decoding should be applied to display the body without losing information. Several encryption layers can be present, and they should be listed in the same order as they were used.

Content-Length is a decimal number, which indicates the size of the message body in octets. If no message is included, the field value must be zero.

Content-Type indicates the media-type of the message

CSeq Command Sequence orders transactions and distinguishes new requests from request retransmissions. It consists of a decimal sequence number and the request method for a request inside a dialog, otherwise it is expressed as a random number.

Expires specifies how long the message content is valid. The meaning of this field depends on the method used. Expires for a registration method indicates the time after which the registration is no longer valid. An Expires header field can be added to an INVITE request to limit the validity of an invitation or for a 302, "Moved Temporarily" response, to indicate for how long the Contact header field is valid.

From indicates the initiator of a request. This must contain a SIP URI and could also include a display name. If the user wish to apply processing rules to a request, the From header should not contain an IP addresses. The URI can be different from the Contact URL, if the initiator prefers to establish a dialog via another location. The From header field must contain a tag parameter, this is appended to the field value and identifies a dialog.

In-Reply-To mentions the Call-IDs that this message referred to.

Priority determines the urgency to display a message to the user. It is not used to reserve resources on the path to deliver the message or to indicate priority in router forwarding. There are four possible values, " urgent", "non-urgent", "normal", and "emergency". The default is "normal".

Require indicates the options that the client expects the server to support in order to process the request. The require field includes a list of option tags and defines SIP Extensions, which must be supported to understand the request.

Retry After indicates the number of seconds that a requestor should wait before attempting the request again. In general this is used in the event of a server error or when the called party is busy.

Supported enumerates all the extensions supported by the sender of the message. This header field differs from the Require field, because it does not require that the destination supports an extension to understand the current message.

To specifies the logical recipient of the request. The To header field may contain an IP address (as for the From header), which can identify a user or a resource. It is associated with tag parameters appended by the callee to distinguish sessions in case of a forking request.

Via header field indicates the path followed by the request to reach the recipient, and the reverse of this path should be followed by responses. For example, this may be necessary when traversing a firewall or a NAT. Via contains the transport protocol, used to send messages, the host name or network address that processes the message, and can also contain the port to which it prefers to receive a response. A "branch" parameter can be appended by the proxies, this used it to detect loops (how every this is no longer the preferred method for loop detection).

4.4 SIP Registration

SIP provides a registration mechanism. A user needs to register their SIP Universal Resource Identifier (URI) and current location in order to be reachable via this URI. SIP defines a particular type of SIP Server, known as registrar, that is able to accept incoming REGISTER requests and processes them in order to update a location service database.

Bindings between a SIP URI, known as address-of-record, and contact address are maintained in the location service responsible for that particular domain. In such a scenario, a SIP user is reachable by looking in the location service database, and the SIP server, that accepts requests for that domain, redirects or proxies invitations to the new location.

A REGISTER request can add or modify the bindings in the location service database. Registration can be done by a third party, that acts on behalf of the address-of-record user. The registration does **not** establish a dialog between the SIP User Agent and the registrar, but each registration is identified by a Call-ID that should be the same for all registrations from a given User Agent.

4.4.1 Registration

A SIP User Agent includes in the REGISTER request the Contact header, which indicates the new location. The request can contain more than one Contact value and parameters for each value. The "q" parameter indicates the preference for the Contact header value, and the "expires" parameter determines the time the binding is valid. If there is no such parameter, but the request contains a Expires header field, then this value is the requested expiration for all bindings.

```
REGISTER sip:wireless.kth.se SIP/2.0
Via: SIP/2.0/UDP proxy.wireless.kth.se:5060
From: sip:roberto@student.wireless.kth.se; tag=d4728b4350c95f7e
To: sip:roberto@student.wireless.kth.se
Call-ID: 31d1edd7ac49d1ae@student.wireless.kth.se
CSeq: 2 REGISTER
Date: Fri, 20 Sep 2002 14:25:37 GMT
Contact: <sip:roberto@student.wireless.kth.se:5060>; q=0.8
Contact: <mailto:roberto@wireless.kth.se>
Contact: <sip:roberto@lappis.home.com>; expires=3600
Expires: 7200
Content-Length: 0
```

In the example above a message is sent by "sip:roberto@student.wireless.kth.se" to the registrar "sip:wireless.kth.se" via the Proxy Server "proxy.wireless.kth.se". The registrar is identified by the domain part and receives a REGISTER request to create bindings with the Contact header values for "roberto". The expiration time for "sip:roberto@lappis.home.com" is defined by the "expires" parameter, while the Contact "sip:roberto@student.wireless.kth.se:5060" is valid for the duration of the Expires header value 7200 (in seconds).

The registrar first inspects the URI to determine if it is responsible for the address-of-record of the specified domain, and if so processes the message. If the registration succeeds, the registrar returns a 200 response message which contains all current bindings and their status.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.wireless.kth.se:5060
From: sip:roberto@student.wireless.kth.se; tag=d4728b4350c95f7e
To: sip:roberto@student.wireless.kth.se; tag=d4728b4350c95f7e
Call-ID: 31d1edd7ac49d1ae@student.wireless.kth.se
CSeq: 2 REGISTER
Date: Fri, 20 Sep 2002 14:25:37 GMT
Contact: <sip:roberto@student.wireless.kth.se:5060>; q=0.8;expires=7200
Contact: <mailto:roberto@wireless.kth.se>;expires=7200
Contact: <sip:roberto@lappis.home.com>; expires=3600
Content-Length: 0
```

4.4.2 Update registration

Bindings expire if they are not refreshed. A client can update bindings by using the same message, as it used to register (shown above). Properly setting the Contact values and the expiration timers, it can also modify locations.

4.4.2.1 Delete locations

To trigger the immediate expiration of a location, a client specifies an expiration timer of "0" for the Contact address, that it would like deleted. SIP also provides a mechanism to delete all the bindings for an address-of-record without knowing their exact value; in this case the client must set the Contact value to "*" and the Expires header to "0". An example of the first method is shown below.

```
REGISTER sip:wireless.kth.se
SIP/2.0 Via: SIP/2.0/UDP proxy.wireless.kth.se:5060
From: sip:roberto@student.wireless.kth.se; tag=d4728b4350c95f7e
To: sip:roberto@student.wireless.kth.se
Call-ID: 31d1edd7ac49d1ae@student.wireless.kth.se
CSeq: 3 REGISTER
Contact: <sip:roberto@lappis.home.com>; expires=0
Content-Length: 0
```

4.4.2.2 Refreshing locations

A client might refresh bindings before they expire or it might add a new one. It sends a REGISTER request for each binding, that it would like to refresh or to add. The registrar's response is a complete list of bindings and their status for each request.

```
REGISTER sip:wireless.kth.se
SIP/2.0 Via: SIP/2.0/UDP proxy.wireless.kth.se:5060
From: sip:roberto@student.wireless.kth.se; tag=d4728b4350c95f7e
To: sip:roberto@student.wireless.kth.se
Call-ID: 31d1edd7ac49d1ae@student.wireless.kth.se
CSeq: 4 REGISTER
Date: Fri, 20 Sep 2002 16:15:20 GMT
Contact: <sip:roberto@student.wireless.kth.se:5060>; q=0.8
Contact: <sip:roberto@barletta.home.com>; expires=3600 Expires: 7200
Content-Length: 0
```

Response:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.wireless.kth.se:5060
From: sip:roberto@student.wireless.kth.se; tag=d4728b4350c95f7e
To: sip:roberto@student.wireless.kth.se; tag=d4728b4350c95f7e
Call-ID: 31d1edd7ac49d1ae@student.wireless.kth.se
CSeq: 4 REGISTER
Date: Fri, 20 Sep 2002 16:15:20 GMT
Contact: <sip:roberto@student.wireless.kth.se:5060>; q=0.8;expires=7200
Contact: <mailto:roberto@wireless.kth.se>;expires=617
Contact: <sip:roberto@barletta.home.com>; expires=3600
Content-Length: 0
```

4.4.2.3 Discovery of a registrar

SIP provides different methods to discover the registrar, leaving the choice upto the implementor. The simplest way to contact the registrar is to manually configure the user agent with the SIP registrar's address; this is suitable for a static scenario, but is probably not be acceptable in a dynamic environment. The registrar can be also contacted via an IPv4 multicast address. In this case a client sends the registration request to the well-known multicast address "sip.mcast.net" (224.0.1.75). Note that the SIP user agent can learn the location of other users by listening to this multicast address. However, this multicast solution only works if the link supports multicast. Unfortunately, privacy problems can occur due to this multicast if a user does not want his, her, or its location to be public, since end-to-end encryption is not applicable to the registration request. The final possibility is to contact the SIP registrar by using the host part of the address-of-record as the address of the request.

4.5 Dialog and Session

SIP defines two approaches to exchanging messages, inside or outside a dialog. A dialog represents a SIP relation between two peers that persists for some time. It is specified by a dialog ID, which is defined by the Call-ID and the tag parameter in the From and To header fields. For instance, a dialog is established through the generation of non-failure responses to an INVITE request, such as 2XX or a provisional response. Thus, registration is considered a message exchange **outside** a dialog.

A user agent can open different sessions to negotiate parameters or can exchange meta-data in the same dialog, these sessions are distinguished by the CSeq header field.

4.5.1 Dialog

A dialog is created whenever the initiator receives a non-failure response. A given dialog is identified by each involved peer with a dialog ID, which is associated with all responses and requests with a given tag parameter in the To header field. Depending on the role played in the communication, the user agent sets the values in the dialog ID. Thus, this value need not be the same at each peer involved in the dialog. The dialog ID consists of the Call-ID header value, and a remote and local tag. The User

Agent Client sets the remote tag to the tag parameter of the To header field and the local tag to the one appended to the From header field of the request message. While the User Agent Server sets the remote and local tag in reverse order.

Once a dialog has been established, user agents can initiate sessions. The user agent that sends a request assumes the client role in the transaction. The client sets the parameter in the SIP request based on the dialog ID. The tag parameter of the From header is set to the local tag and the tag of the To field to the remote tag. The CSeq number is incremented for each transaction and it is set locally. Thus the local sequence number would be different for requests initiated by different user agents. For a CANCEL or an ACK method, the CSeq is the same as the corresponding request which they cancel or acknowledge.

In the same call more than one dialog can be generated. Each dialog is distinguished by the dialog ID. This scenario frequently happens when an INVITE request is processed by an intermediate server which acts as proxy. By forking the request, the proxy server generates multiple INVITE requests for the same call through different paths. A dialog can be modified by sending the target a refresh request, which modifies the dialog parameters.

The dialog is terminated by sending a BYE method. SIP allows client, server, or both to send BYE messages. Receipt of the BYE method will close all open sessions within a dialog and terminate the dialog itself.

4.5.2 Session

A session is established when a 2XX response is generated in response to an INVITE request. The final response is always acknowledged with an ACK which can be generated in several ways according to the type of response. Optional fields can be added to the INVITE request, such as an Expires header to indicate the validity of the invitation, a Subject can be specified for the invitation, and a message body can follow.

The message body may contain a session description, which is identified in the Content-Disposition field. SIP uses an offer/answer model [44] via the Session Description Protocol (SDP) to describe sessions. This model specifies the rules to place offers and answers in a transaction.

Once a session is established, peers can modify the existing session, by sending a new INVITE request, known as RE-INVITE, within the same dialog. A full description of the session should be sent in a RE-INVITE request in order to recover from failures in the preceding session description.

A session is terminated by sending a BYE method for a particular session inside a dialog, or by a BYE method for a dialog. In the latter case all open sessions will be terminated.

4.5.3 SIP message transaction

This section presents an example of SIP messages for a transaction between two users, <sip:theo@ericsson.se> and <sip:roberto@wireless.kth.se>. For clarity the message body carrying the SDP message is omitted.

```
INVITE sip:theo@ericsson.se SIP/2.0
Via: SIP/2.0/UDP wireless.kth.se:5060
From: sip:roberto@wireless.kth.se:5060; tag=adc715ad24f094db
To: sip:theo@ericsson.se
Call-ID: 2d32dd7b7dbe43bb@wireless.kth.se
CSeq: 234 INVITE
Date: Thu, 03 Oct 2002 15:07:08 GMT
Subject: hello
Contact: <sip:roberto@student.wireless.kth.se:5060>; q=1.0
Supported: 100rel
Content-Type: application/sdp
Content-Length: 203
```

Roberto has invited Theo to an Internet call. The dialog and the session are not yet established, Theo has not yet answered.

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP wireless.kth.se:5060
From: sip:roberto@wireless.kth.se:5060; tag=adc715ad24f094db
To: sip:theo@ericsson.se
Call-ID: 2d32dd7b7dbe43bb@wireless.kth.se
CSeq: 234 INVITE
Date: Thu, 03 Oct 2002 15:07:09 GMT
Content-Length: 0
```

Theo's User Agent has received the call and acknowledges Roberto's INVITE by sending a provisional response, Theo's device is ringing. The dialog and the session will be established when Theo accepts the call.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP wireless.kth.se:5060
From: sip:roberto@wireless.kth.se:5060; tag=adc715ad24f094db
To: sip:theo@ericsson.se; tag=42c3bd0ff1868ac
Call-ID: 2d32dd7b7dbe43bb@wireless.kth.se
CSeq: 234 INVITE
Date: Thu, 03 Oct 2002 15:07:22 GMT
Contact: <sip:theo@research.ericsson.se.:5060>; q=1.0
Content-Type: application/sdp
Content-Length: 203
```

Theo accepts the call and sends a final response to Roberto. The dialog is established, and the dialog ID can be set by Theo's User Agent and Roberto's User Agent. Theo's dialog ID is {Call-ID:2d32dd7b7dbe43bb@wireless.kth.se; local tag=42c3bd0ff1868ac; remote tag=adc715ad24f094db}. Roberto's dialog ID is {Call-ID:2d32dd7b7dbe43bb@wireless.kth.se; local tag=adc715ad24f094db; remote tag=42c3bd0ff1868ac}.

When Theo hangs up, this generates a BYE request. In this case is up to Theo to set the CSeq number in the BYE message, while the From and To fields are swapped.

```
BYE sip:roberto@wireless.kth.se:5060 SIP/2.0
Via: SIP/2.0/UDP ericsson.se
From: sip:theo@ericsson.se; tag=42c3bd0ff1868ac
To: sip:roberto@wireless.kth.se:5060; tag=adc715ad24f094db
Call-ID: 2d32dd7b7dbe43bb@wireless.kth.se
CSeq: 25 BYE
Date: Thu, 03 Oct 2002 15:07:56 GMT
Content-Length: 0
```

Roberto sends a final response.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ericsson.se
From: sip:theo@ericsson.se; tag=42c3bd0ff1868ac
To: sip:roberto@wireless.kth.se:5060; tag=adc715ad24f094db
Call-ID: 2d32dd7b7dbe43bb@wireless.kth.se
CSeq: 25 BYE
Date: Thu, 03 Oct 2002 15:07:57 GMT
Content-Length: 0
```

4.6 Locating SIP Servers

SIP is a protocol used for initiating communication between two peers. In order to establish a dialog, the calling party needs to forward the call to the right domain. To do so, the user agent contacts a proxy server in its home domain, which will forward the request to a proxy server of the callee's domain. More than one intermediate server, acting as a proxy server, may process the message before it reaches the callee.

A user agent can address SIP messages using an IP address or the URI of the callee. In the latter case the user agent client first delivers SIP messages to a proxy server (called the "out-bound proxy") in its own domain. By querying a DNS server, the client learns the IP address of the SIP Server to which the call should be sent. In the same domain one can have several SIP out-bound proxies, that can accept calls; hence, the user agent client determines the right one by checking the DNS SRV Record, which lists the available servers for a service ordered by weight. A similar process is performed by all the intermediate SIP servers until the call is forwarded to a proxy server in the callee domain (called an "in-bound proxy"). (Fig. 4.1).

A SIP entity performs a DNS query whenever it contacts another entity located by SIP URI. This method is fully described in [45].

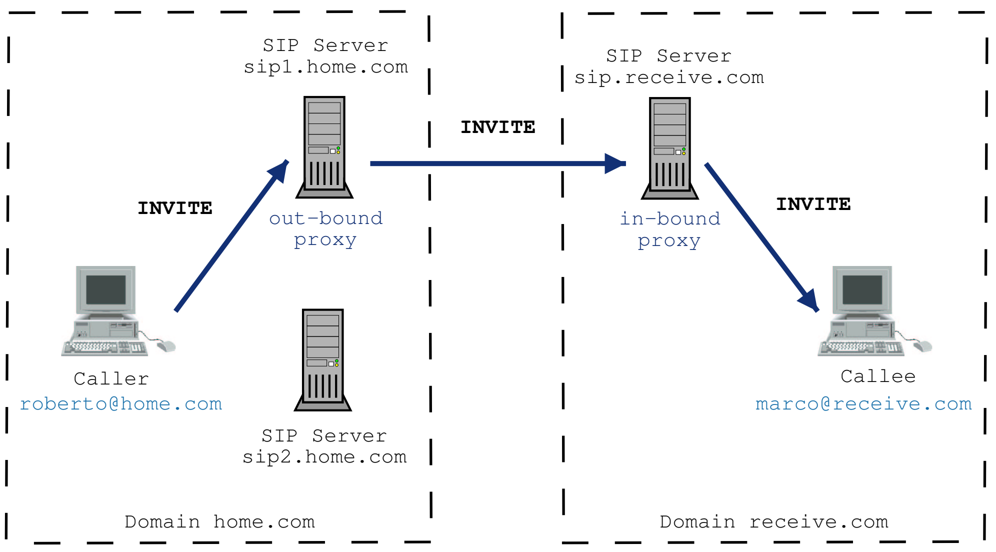

Figure 4.1: Locating SIP Servers

Another method to locate a SIP proxy is based on the use of DHCP, using the DHCP standard definition for SIP Servers [50]. The standard defines a set of options to locate a SIP server when the user is in a foreign domain and needs to locate an outbound proxy to rely the messages. DHCP configuration provides the client a list of name servers and the client needs to follow the same pattern defined above for locating one or more SIP servers in the peer's home network using the DNS SRV.

4.7 Extension

SIP is designed to be extensible. Additional methods and header fields can be defined by SIP Extensions. The SIP standard [20] defines user agents and proxies behaviour if they don't support the extension in use. A user agent simply ignores the parameters, that it does not understand, unless they are indicated in the Require header field. A proxy should not refuse to forward a message formed by an extension. If a proxy needs particular features to handle the message, these should be specified in the extension definition.

SIP specifies a way to let an user agent become aware of the capabilities of a peer. The user agent can use the OPTION method to discover all the extensions understood by a particular user agent, which are described in the Supported header field in the response. This header lists all tags supported, as defined in the SIP RFC [20].

4.7.1 Specific Event notification

The Event notification extension [42] provides additional capabilities to SIP entities, which can request notification of a particular event. The event can be quite general and defined in several packages. An event package is a document which specify the event extension and defines the syntax and the semantics of the event information that should be exchanged between a notifier and a subscriber. Each package is identified by a name, this is specified in a header "Event" field that can be associated with a "id" field to identify subscriptions within a dialog.

This extension defines two method, SUBSCRIBE and NOTIFY, in order to ask for notification and to obtain notification when the state change occurs respectively. Furthermore, user agents or proxy servers can act as State Agents, so that they can accept subscriptions for events and operate on behalf of the resource.

4.7.1.1 SUBSCRIBE

The SUBSCRIBE method is used to request notification of events, and to establish a subscription. The duration of the subscription is indicated by the value of the Expires header or by a default value defined in the package. A subscription is always associated with a dialog, that is specified by the dialog ID, to a “event package” name and it can be identified with a “id” parameter if there are multiple subscription in the same dialog. This is confirmed with a 2XX response which indicates the actual expires time, that can not be longer than specified in the request. A subscription should be periodically refreshed by sending another SUBSCRIBE request pointing to the same dialog with the same values for the “Event” header and “id” parameter. If the subscriber wants to unsubscribe before the timer expires, it sends a SUBSCRIBE request with Expires value of zero. No BYE message needs be sent to close the dialog as it will occur when the subscription expires and when the notifier has no more any state associated with the subscription.

4.7.1.2 NOTIFY

When a subscription is accepted, the notifier sends a NOTIFY request, which contains the event package name, an identification of the subscription (if it is present) and an optional body, that can contain the state of the subscribed resource. The event package definition defines when a notification should be sent and the state information when an event occurs; a new request subscription causes a immediate notification with complete state information of the resource, while the next notification can contain state deltas (i.e., only changes in state) according to the package definition.

4.7.1.3 Event

An user agent can subscribe for events, which are identified by the Request-URI, the event-type, and an optional message body. The “Event” header identifies the event package registered with the IANA, to which the user agent should refer for the semantics of the event. The Request-URI contains information to identify the location of the resource.

As noted above, the event package defines the state that can be associated with an event. The Event notification extension also introduces the concept of an eventtemplate package, which defines the semantics that can be applied to other packages.

4.7.2 Extension for Presence

The event package Presence [21] defines the use of the Session Initiation Protocol for providing information about changes in the communication state of an user. The base SIP specification already provides registration and naming of users, and it also defines strategies to route calls to a user's location. The Presence package has not yet been standardized, but it makes use of the SUBSCRIBE and NOTIFY methods already defined in the Event framework [42]. The "Event" header is set to "presence".

An user agent interested in the presence information of a particular user (called a “presentity”) makes a SUBSCRIBE request to a presence agent, that processes the request as defined in the event framework specification. The presence agent is a user agent able to handle presence subscriptions and has presence knowledge of a presentity. Thus, the presence agent can be co-located with the presentity (called an edge presence server), or with the registrar responsible for the presentity (called a presence server), or in any SIP server. In the latter case the server must learn the presence state of the presentity by subscribing with other entities.

The SUBSCRIBE message establishes a dialog between the presence agent and the initiator of the request. It determines the duration for the subscription via the Expires header field; if it is not present, then the package defines a default value of 3600 seconds. As described above, each SUBSCRIBE message triggers a notification, which are sent periodically. The period is defined in the event package. The package defines a lower limit of 5 seconds for the period in order to avoid network congestion. A NOTIFY message is also issued when the state changes.

The package defines rules for a user agent to apply filters to subscription requests. The description is placed in the body of the SUBSCRIBE request and it influences the behaviour of the presence agent. Presence information is carried in the body of NOTIFY messages and the default content type is "application/cpim-pidf+xml". Presence information can be described by XML, which must be understood by all user agents that wish to utilize the “presence” package.

4.7.2.1 Presencelist Extension

Other packages define similar strategies for presence information. The Presencelist package [43] introduces the use of a list of presentity, which reduces the number of subscriptions. A user agent creates the list and represents it by a SIP URI, and the UA only subscribes to presence information of the presence list, instead of subscribing to each presentity in the list. The specification, which is not yet an RFC, does not explain the creation and maintenance of these lists.

4.7.3 Extension for Instant Messaging

Instant Messaging [22] defines the rules for the transfer of messages between SIP users. SIP already provides support for session based communication and events, such as presence of users, but the SIP specification [20] does not provide any mechanism for instant messaging. The Instant Message extension proposes a new method MESSAGE; MESSAGE requests carry the message content in the body in MIME type, text/plain or message/cpim. The use of such instant message requests should be considered outside a dialog and each message is stand alone. However, it is possible to send messages inside a dialog preventing the separation of the signaling end point and the instant message end point.

Since [22] defines the use of an instant message inbox for a SIP user, im:user@domain, messages might be stored and acknowledged with a 202 (accepted) response, even when the message has not been delivered to the user. A 200 response indicates that the user has received the message, but it might not be displayed to the user. The Expires header field defines the bounded validity of the message; a Date header field might specify the starting validity time for the message. A limit of 1300 bytes to the size of the body is proposed by the specification to avoid congestion and to avoid instant messaging traffic interfering with call signaling traffic. However, it is possible to send larger payloads when the message is part of a media session.

Chapter 5

SIP in Mobile Networks

5.1 Introduction

This chapter will present the handoff from one network to another at the application layer, when Session Initiation Protocol (SIP) based mobility can take care of the movement between heterogeneous networks.

In a GPRS (General Packet Radio Service) network, mobile devices are allocated and addressed at the Network layer. In order to provide subscribers localization using a Session Initiation Protocol (SIP) Universal Resource Identifier (URI), new functionality and tasks should be added to the Gateway GPRS Support Node (GGSN) or to the GPRS network. As described in section 3.3, the GGSN already acts as router for the subscribers' traffic, and allocates public or private IP addresses to these users when they register with the GGSN of the GPRS network. In the latter case (private IP addresses), either the GGSN should act also as a Network Address Translator (NAT), or it should be present another that serves as a NAT for the GPRS network. This is necessary in order to route packets to and from the public Internet and to enable subscribers to be reachable from users outside the GPRS network.

5.2 SIP's mobility support

As described in section 3.4, Mobile IP provides mobility for hosts when they are away from their home network. In this section an application layer mobility solution is described that uses the mobility features provided by the Session Initiation Protocol (SIP) [20]. Handover handled at the application level could give more flexibility to operate in heterogeneous network when the user could be moving between different types of access networks, that might be managed by different Internet Service Providers (ISPs). Application layer mobility might improve performances for real-time traffic, as the application is aware of the mobility and might use knowledge about the traffic to handle mobility among heterogeneous networks.

The solution is based on the SIP registration mechanism which establishes a binding between the host IP address and the user's URI [10, 48]. However, SIP based mobility is less suitable for TCP applications, as these applications need to maintain the TCP connection alive, but the socket is specified by the *IP addresses* of the involved parties and the *ports* in use. Thus, traditional mobile IP with triangular routing may be more suitable for TCP based data delivery as the packets are routed via the home address hence there is no observable change in IP address or port number.

Supporting mobility using SIP raises new issues, since the application layer should be aware of new networks (i.e., movement detection) and needs to know the address of the current (active) interface. The follows sections present some solutions which provide mobility for terminals, users, and sessions.

5.2.1 Terminal mobility

Schulzrinne and Wedlund [48, 55] proposed several solution to support terminal mobility when the mobile host is invited to a call. They described two mobility scenarios: pre-call mobility and mid-call mobility.

In the pre-call mobility case, the user moves before receiving or making a call. It needs to register its new location with the home registrar in order to receive the call to the new IP address. However, the registration had not been completed when the user received a call, thus the INVITE request will be routed to the user's most recent location. To solve this problem, Schulzrinne and Wedlund define the use of a scoped IP multicast address to route the call to the user; the proxy of the most recent registration sends a multicast message and if it there is no answer to the invitation within a short time interval, the proxy reports a failure to the caller. In [48], this solution assumes that the SIP proxies and location servers are organized in a hierarchical structure to reduce the traffic overhead and to support mobility of the user among a subset of SIP proxy.

In case of mid-call mobility, the user sends to the correspondent host a new INVITE request that contains an updated session description with their new IP address. This end-to-end renegotiation can cause same delay as the new INVITE request must reach the correspondent node before there can be a new media path. A micromobility approach could be used to reduce the time of the renegotiation; if the correspondent node were to send data to a proxy that handles the host mobility, the mobile host could notify this proxy that it has moved to another location, and the proxy can forward the data to the new location. This solution assumes the new location is close (in a network sense) to the old location and the mobile host updates their location to the proxy that is responsible for host mobility. However, if this solution is not supported, the renegotiation will be further delayed since the new INVITE needs to reach the correspondent node.

Another solution to support handover uses on the conferencing feature of SIP [10]. Since SIP supports multi party sessions, the communication between two users are handled as a two party conference. If a mobile user wishes to contact a correspondent party, the mobile user sends an invitation to the other party specifying the SIP URI of a Multi-point Control Unit (MCU) that provides the capability for participating in multipoint conference. Both users need to join the conference, and, when the mobile user changes its location, it sends an INVITE request to the conference address from the new location. This solution benefits from the fact that no rerouting needs to be performed by the correspondent node, but it is at the price of wasting resources for the two connections to the MCU.

5.2.2 Personal mobility

Using the REGISTER method, a user may register more than one location for its SIP URI, thus causing the invitation to be routed to different terminals or applications, if the latter are running on the same host and identified by different SIP URIs. Personal mobility is the ability for a user to be reachable under the same identifier while using different terminals. In this scenario the user might login into different terminals at the same time, but be reachable at the current location of any of these terminals, since, by using SIP’s forking proxy, the invitation can be forked to all the different locations.

5.2.3 Session mobility

SIP feature enables session mobility when the user wish to change terminal while maintaining a media session. A user might have access to different terminals with different functionalities and intends to use them according to the kind of call.

The session is described in the body of the SIP message and it is associated within a dialog. SIP enables third party control, i.e., an intermediate node initiates and negotiates the session between two users at the SIP level; thus, a third party control allows a user to initiate a session and to redirect to different terminals or to split into different components the same session according to the media type. This solution enables a user to participate to a session and at the same to use several hardware components; however, the third party must be contacted to terminate or to change the session.

Furthermore, SIP enables renegotiation of an ongoing session by sending a RE INVITE request introducing the new session parameters or indicating the "Contact" header to specify another location where the user would like to be contacted. In this case, the session will be completely redirected to the location.

5.2.4 Hierarchical registration

In order to support micro mobility of users when they move in a foreign domain, a hierarchical structure for registration is proposed in [48]. SIP registrars are organized in a hierarchical structure considering the home registrar at the top of this structure. When the user changes location, it updates the registration with the home registrar **through** the SIP server in the foreign domain, called outbound proxy, in order to redirect the invitation to the new location within this domain. This updating can be frequent (depending on the user's movement) and it might need to traverse several hops before reaching the home registrar hence introducing some delay in the registration.

Since the user registers their new location with their home registrar **through** the outbound proxy, which is local proxy in the foreign domain, then if the local proxy and the local registrar are on the same host, then the client might locate them using DHCP for SIP Server [50], as described in section 4.6. The outbound proxy, while processing the first registration request (**1** in fig. 5.1), changes the contact header value inserting a SIP URI corresponding to the foreign domain before forwarding the message to the user home registrar (**2** in fig. 5.1). Thus, a temporary SIP URI for the foreign domain is created in order to identify incoming requests for this user. For instance, SIP URI “sip:roberto@home.se” might generate a SIP URI in the foreign domain “sip:roberto%home.se@foreign.se” (Fig. 5.1). Now, all invitations for the user are directed by the home registrar to the outbound proxy, then forwards them to the user. When the user changes IP address **within** the foreign domain (**3** in fig. 5.1), it sends an update to the home registrar via the outbound proxy (**4** in fig. 5.1), however, this is not forwarded to the home registrar since all invitations will be sent to the local outbound proxy, it simply notes the new user location for locally forwarding these invitations. This solution reduces the number of registration updates at the home registrar and it can be further extended with a hierarchical structure of outbound proxies deep as desired.

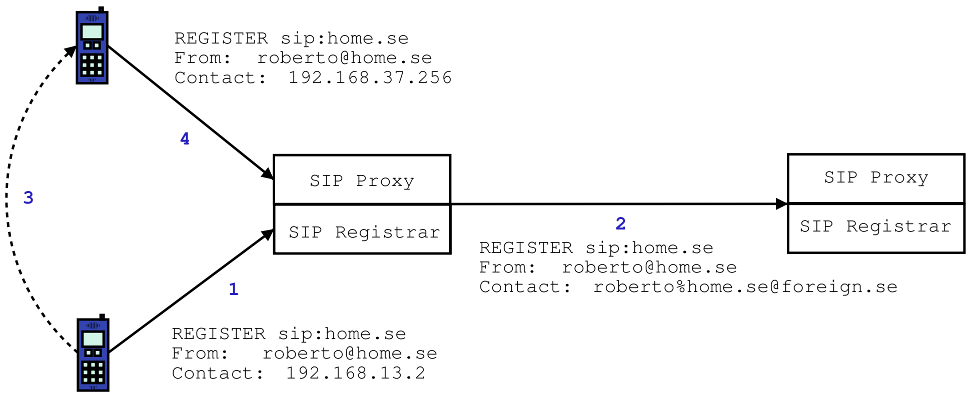

Figure 5.1: Hierarchical Registration in a foreign domain

5.3 Architectural alternatives using SIP for mobility

This section discusses some of the architectural designs for heterogeneous networks where SIP is used for application layer mobility, mainly for a Local Area Network (LAN) and a Wide Area Network (WAN), when the mobile user has ongoing sessions with a correspondent host and uses SIP for signaling and control of multimedia sessions [19].

In [19] different scenarios are analyzed; the session is redirected to the active interface when the mobile terminal is moving in a subnet within Local Area Networks. However, if the mobile terminal switches interfaces for sending traffic, it needs to redirect the call on the active interface with a new invitation. Furthermore, if the interface used for communication is completely unable to communicate, then the multimedia session is interrupted.

For Wide Area Access, [19] proposes a model where the device is connected to two different networks. It is shown that if the device applies different routines to access the network, it will take different amounts of time to establish the communication. The user might use both interfaces and then choose the one to continue to communicate with a correspondent node based on internal policies. When changing locations, the mobile node might need to reconfigure both interfaces. During an ongoing communication session it needs to take account of the IP address of the interface currently chosen for communication, as it needs to specify this as the contact address in the RE INVITE request.

The use of SIP for handling mobility of sessions in heterogeneous network raises some new issues, that need to be considered while designing a solution. Since, the device might have multiple interfaces active that can each be associated with one SIP URI; when a user moves, it needs to consider which is the active interface and not immediately update the location of other addresses. Furthermore, the use of private address used within a LAN behind a NAT should be taken into account.

5.4 SIP and NAT

As discussed above, several operators use a Network Address Translator (NAT) to protect their networks from unwanted connections. Since the Session Initiation Protocol (SIP) is used for signaling and for communication between users, this section investigates some of the solutions proposed to avoid service blocking when the users use SIP for controlling and establishing sessions.

NAT does not support an end-to-end communication since it rewrites the IP address and the port number of IP packets, causing packets from many host to come from the same IP address to correspondent nodes on the public Internet. Furthermore, for security reasons some NATs allow communication with users behind a NAT **only** if these users have already sent a packet to the correspondent node, thus limiting the communication between users in different private networks.

If the users involved in the communication are located inside a single private network, the session is established as usual and the two users are able to communicate.

A user located behind a NAT that initiates a session with others located in the public Internet, is also able to establish the session and to traverse the NAT. However, other protocols used in conjunction with SIP will not be able to traverse the NAT. For example, the Session Description Protocol (SDP) used by SIP to negotiate the parameters for establishing the communication between entities specifies as the IP address for the end-point a private address, which is not a routable address in the public Internet: thus the media session can not be set up. Furthermore, users located behind a NAT are not reachable by others, as SIP servers will map the SIP URL to the private IP address.

This problem is addressed in more details in Thernelius's Master Thesis [52]. Thernelius proposed and designed an Application Level Gateway (ALG), which directly manipulates the fields in the SDP body of SIP messages, thus allowing communication from a correspondent node to the subscriber to traverse networks behind firewalls and NATs, while using SIP as a signaling protocol.

Another solution adds a SIP Proxy Server which provides local SIP registration for the users inside the private IP address space by using an extension to the SIP protocol, as described below.

> "When used with UDP, responses to requests are returned to the source address the request came from, but from the port written into the topmost Via header of the request. This behavior is not desirable in many cases, most notably, when the client is behind a NAT. This extension defines a new parameter for the Via header, called rport, that allows a client to request that the server send the response back to the source IP address and port where the request came from." [56]

5.5 Proposed design for SIP enabled networks

This section discusses a proposed design for SIP enabled networking in heterogeneous networks, such as WLANs and GPRS networks, despite the existence of a NAT which limits the inter-network communication. In order to provide a handle for the user at the application layer using SIP naming, some existing solution are analyzed and extended in our proposed network. Since the GPRS network can be behind a NAT prevents mobile users from being directly reached by user on another network, I propose to have an external SIP server, acting as an outbound proxy, outside the GPRS network and the NAT, with a globally routable IP address to enable communication with the rest of the Internet. This SIP server acts as a SIP proxy for the GPRS subscribers. These subscribers need to be registered with a location server co-located with the SIP proxy in order to record and know their current location. Furthermore, these subscribers need to maintain an open connection with the SIP proxy in order to traverse the NAT (Fig. 5.2).

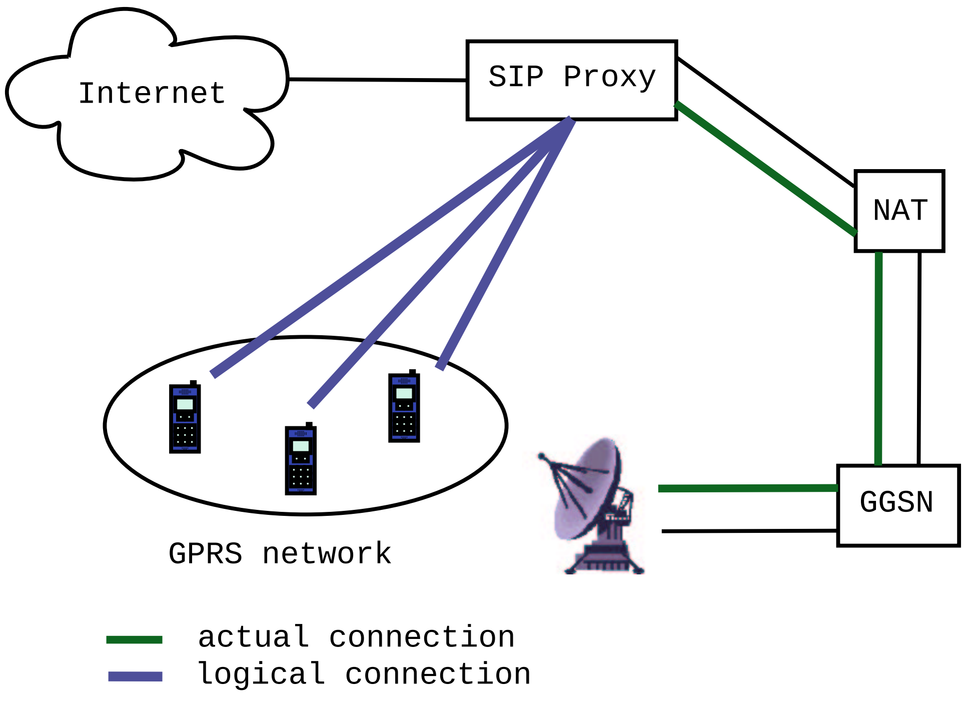

Figure 5.2: SIP proxy for connectivity through NAT for GPRS network

Söderstrom [49] demonstrated in his report that it is possible to guarantee communication through a NAT at the application layer even for virtual private networks. Using SIP for signaling and services is similar to building a virtual private network with an independent addressing scheme and some functions to map the temporary IP address to/from a SIP URI. As he pointed out in his report, we need constant connectivity between the GPRS subscribers and the SIP proxy outside the GPRS network, since some NATs do not allow communication unless an *outgoing* packet has already been sent to the outside address by the internal user. Following this outgoing packet, these users are able to be contacted and receive data using SIP, by specifying in the Record Route field of the SIP message the IP address of the SIP proxy. The solution takes advantage of having a constant connection through a GGSN node which makes the internal nodes reachable despite the network designs of some GPRS operators.

However, the SIP proxy needs to have open connections with all the GPRS subscribers that use SIP for signaling. This might not be a good solution since the users have to pay to keep the connection alive by polling the channel (even if the monthly cost for this extra traffic is low [49]). A typical GPRS network can provide services to a huge number of users (200 000 - 400 000), simultaneously attached and served by the a GGSN node. Maintaining open connections with the SIP proxy causes an overall increase in the traffic average and consumes battery power for the mobile terminals.

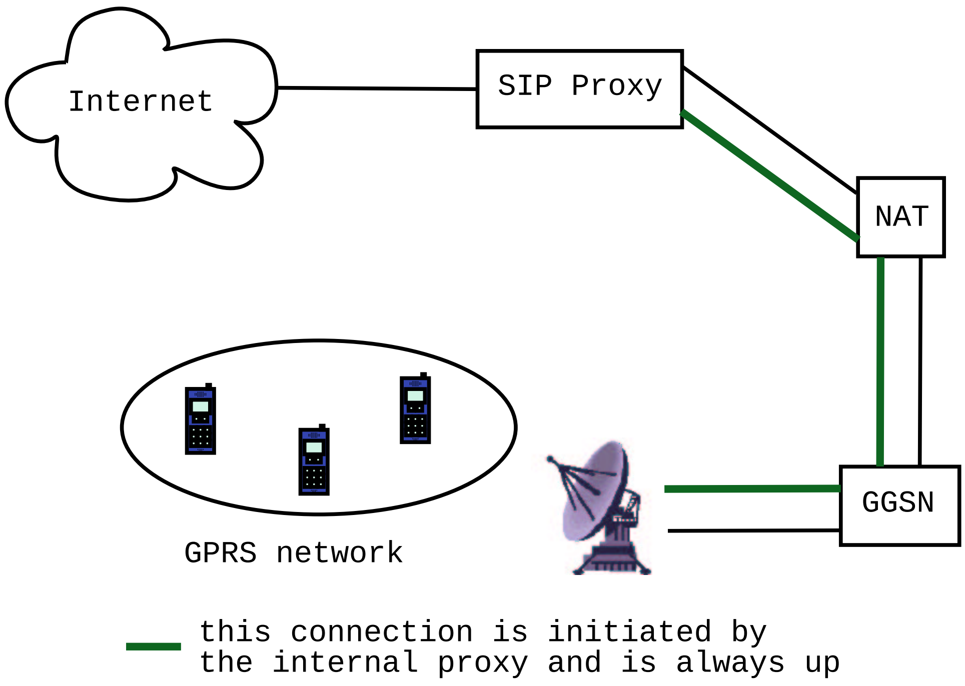
Internet
SIP Proxy
NAT
GGSN
GPRS network
this connection is initiated by the internal proxy and is always up

Figure 5.3: GPRS network enabled for constant SIP connectivity using SIP proxies

In order to reduce the number of connections that must be maintained through the NAT, an internal SIP server can be located behind the GGSN. This SIP server can be directly contacted by all GPRS subscribers, since it has a private IP address and internal connections can always be established behind a NAT. Furthermore, a registrar and a location server might be collocated with this SIP server to accept the SIP URI registrations for internal users and to keep tracks of their temporary location. However, an external SIP proxy needs to be present as it guarantees the communication with entities in foreign networks. This solution takes advantage of the fact that the internal users do not need to maintain connection with the external SIP proxy, and that the number of connections to be maintained is limited to the one between the two SIP proxies (Fig. 5.3).

A similar approach can be used for a WLAN network behind a NAT in order to guarantee communication between users in heterogeneous networks with different topologies (Fig. 5.4). The users are able to communicate at the SIP level thanks to the SIP Proxy between the two NATs, as shown in figure 5.4. Furthermore, several SIP proxies might be used to serve the same network using a hierarchical structure, discussed further in section 5.5.2.

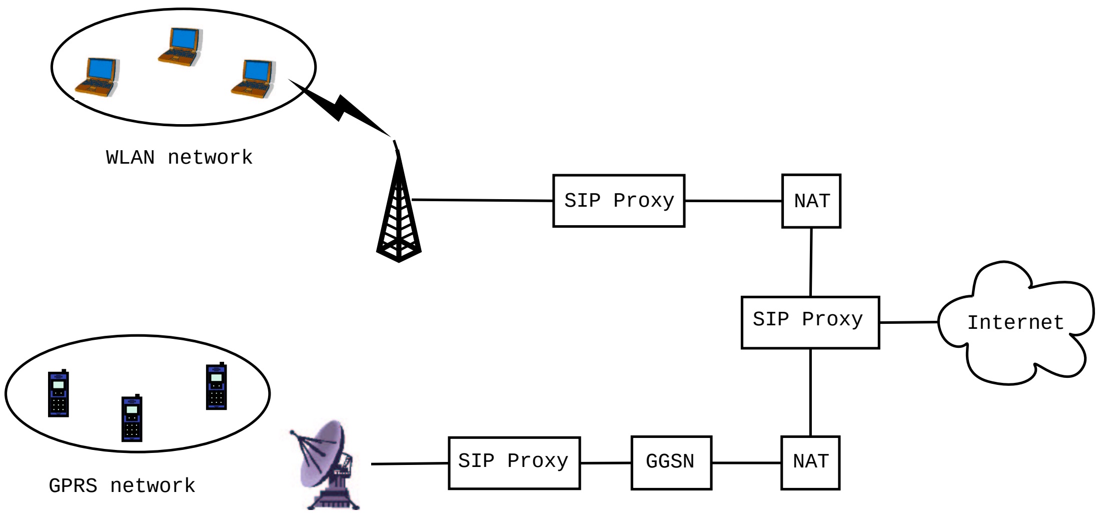

Figure 5.4: GPRS and WLAN networks for SIP communication

5.5.1 SIP URIs association in foreign and home domain

This section considers the registration process with the GGSN of a GPRS network behind a NAT, since registrations in foreign domain, for instance while roaming between WLANs must be considered. The SIP Server inside the GPRS network accepts REGISTER messages from SIP users within the GPRS domain and updates a location database, located at the Location Server, with the user's contact information specified in the request. Furthermore, this SIP Server also acts as SIP proxy because of the use of private addresses inside the GPRS network.

After the association with the GGSN node we have GPRS connectivity, hence the device's interface has been provided with a private IP address. The IP address determines the temporary location of the user and it is used for SIP registration of a new SIP URI, which might be "sip:user@phone_number.GPRS-domain" using the phone number as identifier for the terminal. The SIP Registrar, co-located with the internal SIP Server, updates or creates a new binding (Fig. 5.5). The user might already have a personal SIP URI, that does not identify the device, which is assigned by their (home) SIP provider. Thus, the user needs to inform their (home) SIP registrar of their temporary location in order to redirect calls to their new location.

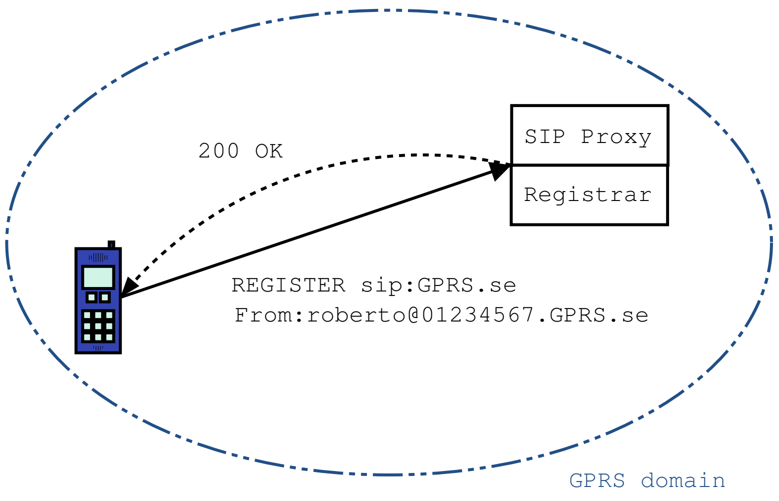

Figure 5.5: SIP registration within a GPRS domain

A user is now reachable inside the GPRS network and via the SIP proxies, see figure 5.3, from external users. The SIP external proxy redirects all calls addressed to SIP URIs of the GPRS domain to the internal proxy. Having global connectivity, i.e., the possibility to contact other SIP users outside the GPRS network and to receive messages, the user contacts their home SIP domain to register their new Contact address with their SIP home Registrar. This Registrar might be co-located with a SIP Server that can act as proxy or as a redirect server for invitations (Fig. 5.6).

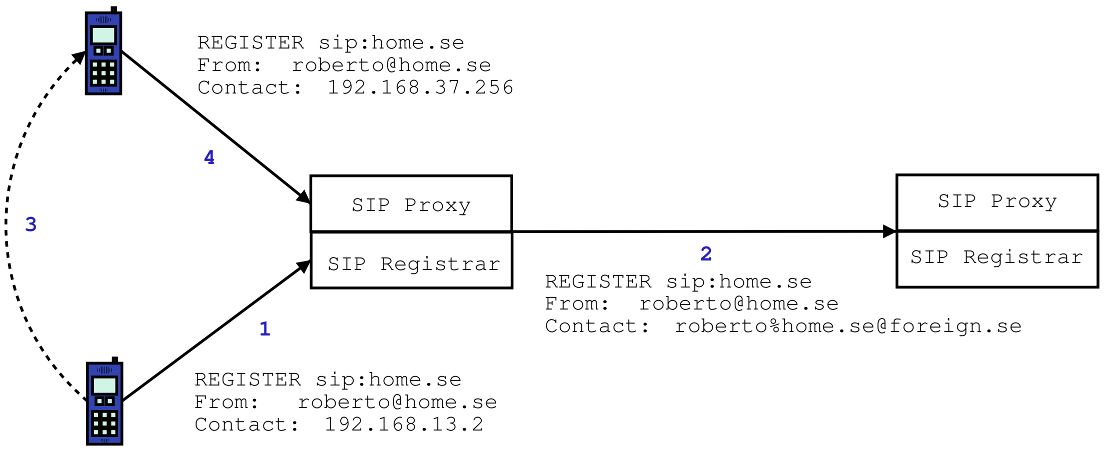

Figure 5.6: SIP registration within the home domain

Because GPRS operators have roaming agreements to guarantee the service while their user travels in another country, when a user is roaming in a foreign domain, the user needs only to send an update registration to their own GPRS Registrar or to the their home SIP Registrar. The updating might be seen as a registration within a home SIP domain, as illustrated in figure 5.6. This scheme is normally based on mobile IP to redirect the GPRS traffic and at the application level it is as if the SIP user did not move; however, a registration update enables user to use SIP for redirecting the call to the current location at the application level.

During the SIP registration, the user needs to discover a SIP registrar in the current domain. The possible discovery solutions, defined in the SIP standard [20], were described in section 4.4.2.3. However, the technology infrastructure might limit the use of a multicast scoped messages to locate the registrar, hence a DHCP configuration of the mobile device might be a better solution. In the latter case, the client issues a DHCP message using the SIP option defined in [50], and receives the host name of the outbound SIP proxy.

The client reaches this SIP registrar possibly via intermediate SIP proxies, each resolved via DNS lookup (see section 4.6). In order to route the registrar's response back to the user to complete the registration, the intermediate proxies inserted their own SIP URIs in the Record-Route header field before forwarding the register request.

A solution to provide mobility and other functionalities in the home and foreign network using IPv6 is designed for the third generation cellular networks by 3GPP. This requires the use of multiple outbound proxies for controlling the access to the GPRS subscribers. SIP registration and SIP call control are defined in the specification, more details can be found in [2].

5.5.2 Hierarchical registration in SIP

A hierarchical structure for SIP registration in a heterogeneous network might reduce the need to send global bindings updates across the network in order to inform the home network or the correspondent node about a node's movements, thus reducing the mobility signaling load. The solution, introduced in section 5.2.4, utilizes a hierarchical structure for SIP Servers in order to reduce the updates while moving in a foreign domain. A hierarchical structure for binding updates might reduce the number of retransmissions, since the retransmissions also occurs between levels in the hierarchy **not** end-to-end. Furthermore, the probability of packet loss increases as the number of intermediate hops increases [14] and these packets are going via the fixed network where the probability of loss should be very low.

This section proposes an extension to the hierarchical structure for SIP registration in order to be effective in heterogeneous networks. I have already introduced in previous sections an outbound proxy and a local SIP registrar for the GPRS subscribers. Furthermore, the GPRS network needs to provide a SIP location service database for SIP users, called here a Location Service GPRS (LSG), which handles the location of this GPRS network's subscribers. As described above, the GPRS network should have a SIP outbound proxy, outside the network, which guarantees constant connectivity. At this proxy another location service database, called a Location Service Area (LSA), might be collocated in order to store SIP users' location data for a wide area. The LSA is able to locate users belonging to different networks, potentially each with different technologies (see Fig. 5.7). For our purposes, I will only describe the LSA for WLAN and GPRS, since the Personal Data Assistant used for the project has only two interfaces, specifically GPRS and WLAN interfaces.

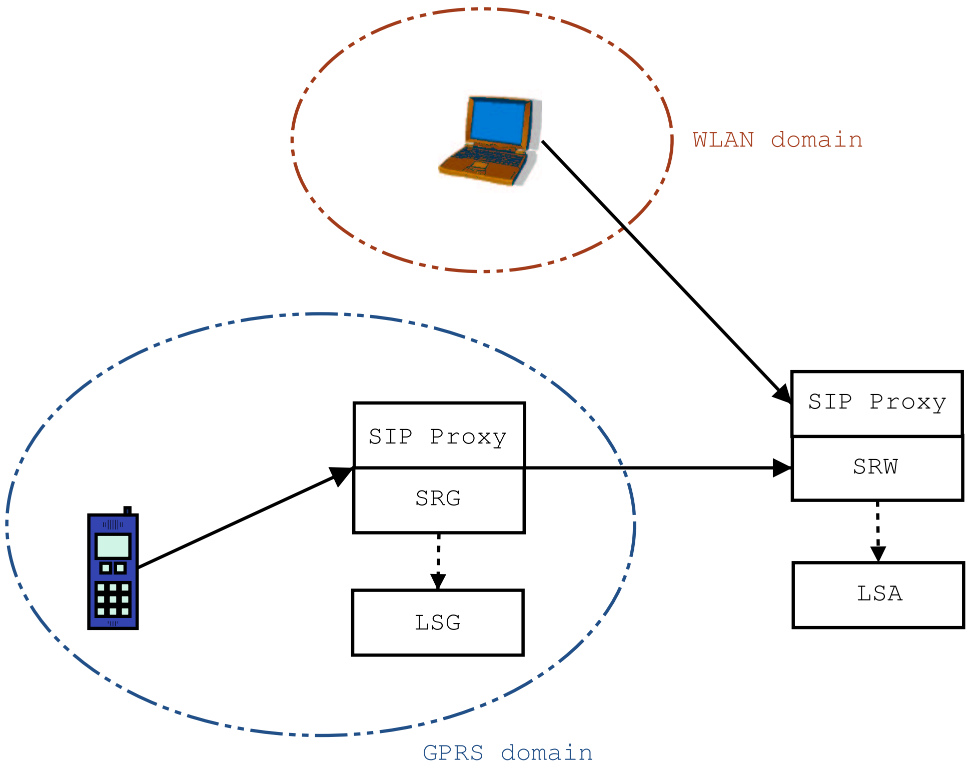

Figure 5.7: Hierarchical structure of SIP Servers for heterogeneous networks

The LSA is located at specific host which is running a SIP Server, responsible for allocating bindings for both the GPRS and the WLAN network domains. I would suggest co-locating these on the same host as a SIP UA Server and Registrar; the domain part of the Request-URI of SIP messages are inspected to see if it can provide registration or updates for users in this domain, when a request is received.

The way to contact this SIP UA Server and Registrar differs depending on the network the subscriber is attached to. A hierarchical structure for two level management mobility uses one SIP Registrar (SRG), for GPRS network, and another SIP Registrar for wider area (SRW) for notifications from the GPRS Registrar and for subscriptions outside the GPRS network. DHCP informs users of the SIP Server proxy, which acts as proxy for registration within the domain or the area.

As described above, in GPRS the subscriber sends a SIP register request to the SIP Registrar for Wide area (SRW), which is contacted via the Service Registrar GPRS (SRG) with a notification of the subscribers' status and their actual location, as was described in section 5.2.4 (Fig. 5.8). Furthermore, the SIP UA does not need to be pre-configured, nor does it have any problem determining the domain which it learns via DNS reverse lookup.

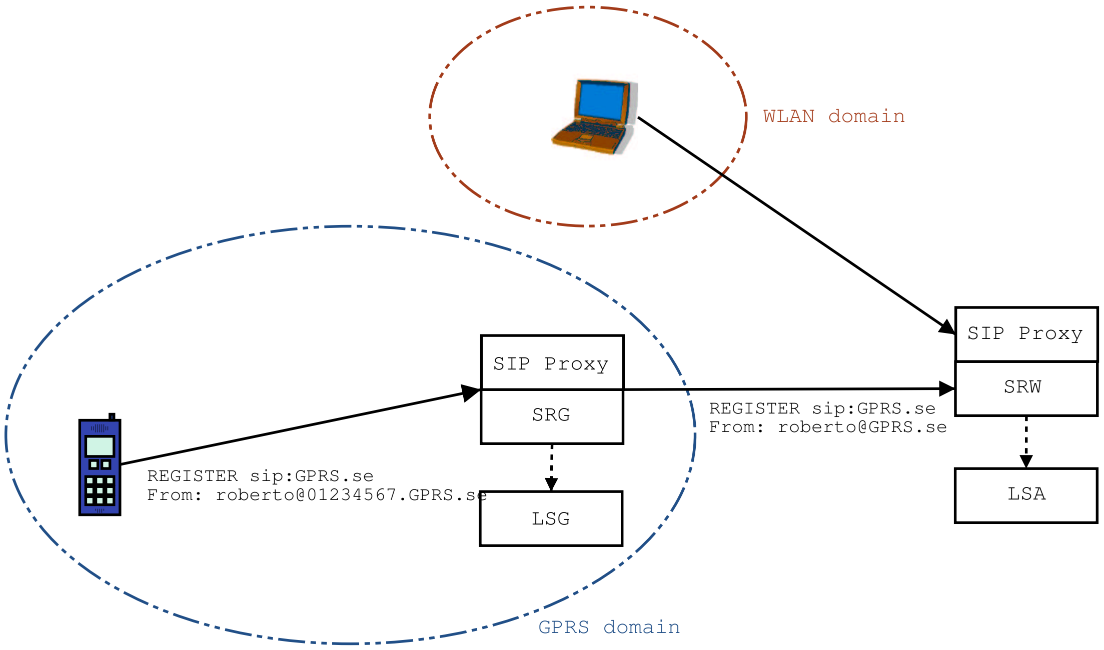

Figure 5.8: Hierarchical Registration for heterogeneous networks

In the WLAN users might also send a multicast packet to the well-known "all the SIP servers" (224.0.1.75) address to discover the SIP Registrar. They will get the acknowledgment of the registration from the SRW, if it is responsible for that WLAN, or from another SIP Registrar.

5.5.2.1 A hierarchical presence structure

The SIP UA sends the SRG its SIP URI and the Contacts indicating where it prefers to receive the call, with related expiration timers. When the registration is received, a new record is added to the database at the LSG; the cell identifier, extracted from the messages of the Base Station Controller (BSC), might also be stored in the location service database to provide information of the physical position of subscribers. The field could also be directly updated by the user, by extending the SIP REGISTER method. The user knows the cell which they are attached to, thanks to broadcast messages sent by the base station that contain the unique cell identifier (C_ID).

The Location Service GPRS (LSG) database looks like:

SIP URI	Contact	Cell ID	Expire
roberto@GPRS.net	roberto@01234567.GPRS.net	12	7200
	roberto@31241341.GPRS.net	3	3600
pietro@GPRS.net	pietro@32412412.GPRS.net	5	3600
marco@GPRS.net	marco@43212342.GPRS.net	1	3600

The SRW database has a similar structure. WLAN subscribers send their SIP URIs and their Contacts, if any, with the expiration timer in the REGISTER method and the BSSID, based on the position of the Access Point . GPRS subscribers are registered with the SRW via the outbound SIP Server located inside the GPRS network. This server acts on behalf of the users for a hierarchical registration. The SRW might also act as a Presence Agent for GPRS users, by accepting SUBSCRIPTION for presentities and sending notifications of status changes [21].

It might be interesting to extend the Extension for Presence package in order to allow a Presence Agent to give the status of a user located in a cell, without knowing their SIP URIs. If one is interested in the users within a particular cell you could learn each of these users' URLs by querying the Presence Agent, using the cell ID.

The SIP Server co-located with a location service might already act as a Presence Server. By extending the Presence package, this Presence Server could accept subscriptions for cell presence or redirect the subscriptions to the GPRS Presence Agent, or might act as a proxy in order to send notifications. These features could also be extended to be used in WLANs; in such a network, the WLAN Base Service Station Identification (BSSID) is stored in the database, to provide the same types of service as described above. The SIP Server co-located with the LSA acts as a Presence Agent, accepting subscriptions, or as a Presence Server if there is a hierarchical structure of Presence Agents serving the WLAN coverage areas.

The reason for using a hierarchical structure for location services is to integrate the WLANs and GPRS networks for service discovery. A user would prefer to discover and use services via the cheapest and fastest connection, hence obtaining them from the closest peer using the interface it prefers.

Furthermore, a geographical identification of the users might be useful while users are interested in context information of a particular area or in peers at a specific location.

Chapter 6

The Service Peer Discovery Protocol

6.1 Introduction

This chapter introduces the proposed service discovery protocol and the decisions made in order to address requirements and issues of such a design. The solution is examined in relation to the existing technologies described above in chapter 2.

The purpose of a service discovery protocol is to allow users to discover services, that are handled by other peers in the network. In this chapter we analyze the issues concerning the design of the proposed Service Peer Discovery Protocol (SPDP), the considerations that we have done to restrict the problem and the designed protocol.

The Service Peer Discovery Protocol (SPDP) needs to be effective in heterogeneous networks and to provide users with a simple mechanism to discover services in the network. It uses SIP to carry messages and to contact other peers to ask for services; it uses other protocols to retrieve services, depending on the their nature. For instance, a user can connect to the peer using the Context Data eXchange Protocol (CDXP) [57, 29] to exchange context data or use FTP to exchange files.

SPDP might use a context server to know which users, among those connected to the network, can handle the required service. Every user in the network must register with the context server, giving their (GPS) position, their SIP URI, the wireless networks to which they are currently attached (WLAN or GPRS), an expire timer for this registration, and the kind of services this node can offer. The (GPS) position is used to determine the location of the user; the SIP URI is used by SIP to locate the user, in order for others to contact the user using SIP; the information about the user's wireless network is used to allow actions based on the available interface(s), for example, preferring a WLAN if available to deliver and retrieve large amounts of data. Finally, the expire timer is used to indicate the duration of the registration. The kind of service is the first discriminator for discovering services.

6.2 Issues and requirements

A user can have network access via several wireless networks, by using different interfaces of their PDA. In this work, we consider a device with two interfaces: a WLAN and a GPRS interface. It can simultaneously be attached to both networks, thus it can have two different IP addresses. A user should register with a context server, known by preconfiguration of the device and located as described in section 6.5.4.2. The context server keeps track of the relevant user information as well as additional environmental and spatial information.

The shared resources, present in the network, are packaged as services and can be offered by devices; the devices and the services can accept actions or perform actions on other services. These resources should be located and discovered using the service discovery protocol. Furthermore, the device might roam among different networks, changing its IP address (device mobility) or users can change their logical identities (SIP URIs) independently of which devices they are using (personal mobility).

For a service discovery protocol it is important that the service information is updated to assure consistency of data. As we are utilizing one or more wireless networks, services may be available for only a small amount of time due to node mobility or personal mobility; furthermore, as the number of providers increases and as the available services increase, the number of resources changes dynamically.

In order to support these requirements, the communication should be peer-to-peer and any negotiation occurs entirely between end-points. Thus, the discovery protocols, introduced in chapter 2, are not well suited for this kind of communication. A central server for locating information or for negotiation may not be able to follow all changes (section 1.3).

UPnP, based on SSDP for service discovery (section 2.2.3.2), specifies that discovery can be done even if there is no central server in the network. SSDP uses multicast messages for service discovery. Although this approach can work well in a small network, it does not scale well in a large wireless network, as multicast messages consumes bandwidth and all users will receive the packet therefore consuming battery power for receiving and processing each of these messages, even if they are not relevant. For instance, SSDP uses multicast messages for announcing the presence of a service, for discovery, and for renewing presence messages; thus, as the number of users in each network increases, that network can be flooded just by signaling messages.

In addition, we need to consider the mobility of devices, users, and services when designing the protocol. While on the move, the protocol should provide a way to locate resource(s) and to adapt the current session(s) to this movement. UPnP uses the DNS system to locate the resources, but this solution does not assure that the protocol scales well when the number of mobile users increases. Furthermore, the protocol should also be effective in a private network where the provider uses a private set of addresses, as in many GPRS networks. Thus, a solution is to use an addressing schema that is also effective in a virtual private network, see [49] for further details. For the protocol designed we have chosen SIP to provide naming and localization of users.

6.2.1 SIP and service discovery

This section provides the reasons for our choice of using SIP to carry discovery packets. SIP provides functions that are advantageous for the design of the discovery protocol for network devices with limited computational power. A similar approach is well illustrated in [46] for instant messaging and presence.

Scalability SIP scales well both in local and wide area networks thanks to the routing capabilities of the intermediate nodes in the network. Proxies can operate both in stateful or stateless mode which offer an high degree of flexibility of the routing strategies. Stateless proxies are faster than stateful, but they don't keep state. SIP uses a routing scheme for traversing the intermediate nodes from the source to the destination, see the Via Header field described in section 4.3.3. It does not use multicast or broadcast for message delivery. Furthermore, SIP provides the capability to specify the route that packets will follow when a dialog is established (Record-route header field), hence, a intermediate node can decide to insert its address in the route set or re-write it according to context information. The intermediate nodes can operate successfully both in stateful or stateless mode.

Simplicity and Flexibility SIP provides attributes that allow extensions to the protocol. It is a really simple text based protocol, and it includes a scheme for extensions. Even if an intermediate node does not understand the extension headers in the packet, it must forward the packet to the next hop. Hence, not all the nodes need to support extensions, therefore extensions can be dynamically added (see section 4.7).

Addressing and Registration SIP has its own addressing and location schemas. Users are identified by their SIP URI, and register current location with the registrar. A user does not need to know the exactly location of another user because a SIP proxy will route the packets to the destination based on the other user's SIP URI.

Personal Mobility Since SIP provides registration of the user's location, the protocol supports personal mobility (see section 5.2.2). Thus, an user is not associated with a device, and their ability to register their location is independent of which terminal they use to access the network (although the device's address is also generally registered).

Security SIP defines authentication and encryption schemas for end-to-end communication. This is an important feature that we need not consider further in our design since it is already addressed by SIP. A strong authentication schema with certificate based authentication can be used to guarantee high security, which is useful since the discovery protocol is used for discovery services and also for giving access to the device's capabilities.

Transport Independence SIP does not specify a transport protocol for sending messages. It can use different protocols, such as TCP or UDP, to encapsulate messages. Hence, messages can traverse heterogeneous networks without any dependence on the underlying network technology.

Event Notification SIP defines in its event framework [42] both subscription and notification schemas. Thus, the proposed discovery protocol needs only to extend the event package for the discovery of services. For further details about the discovery extension see section 6.5.1.

Forking SIP enables request to be forked to different destinations registered with the same SIP URI. Thus, devices, such as a printer or sensor, can be addressed with the same URI if they provide the same service. So the same resource can be queried using only one message since all the destinations should reply to the request even if only one 200 response is reported to the client.

6.2.2 Ad Hoc network and service discovery

This section provides an overview of issues that can arise while designing a service discovery protocol for an ad hoc network. There is a substantial difference between a fixed wired network and ad hoc wireless network, hence the protocol for an ad hoc network should deal with movement and user relocation, resulting in dynamic changes in communication. Devices should be able to communicate directly with other devices within their transmission range. Having little or no networking infrastructure places new limits on the location of a central server for service discovery.

A hybrid approach with some infrastructure, that facilitates communication inside a cell and may even connect to network nodes outside the cell, can be used to provide devices with increased connectivity. One idea is to co-locate the server with the base station as a solution to this problem. However, if the base station is overloaded due to discovery messages, this can cause decreased throughput because all the cell's traffic is routed via the base station, which become the single point of failure and potentially the bottleneck.

Although other approaches can be used, such as broadcasting. Using **only** broadcast messages increases the traffic load in the network, but does not offer good discovery capabilities because the discovery is limited to the cell if the nodes do not forward broadcast messages. Furthermore, due to disconnections, or because device might be out of range, the user might miss updates, and the consistency of data is limited by the frequency of the broadcast messages [25].

For the design of the proposed discovery protocol, multicast messages are used for discovery of resources in a small area. Multicast messages might be defined in such a way that it could be possible to set the number of hops the message will be forwarded by intermediate nodes. The clients can set the time-to-live (TTL) according to their context information. Thus, although the message can cross different subnets, where the network infrastructure is capable of accepting such messages but its scope is limited by the TTL. By using multicast a user can query several users with only one message, this is especially useful when it can exploit the broadcast properties of some wireless networks.

6.3 Representing services

A user should be provided with some basic knowledge in order to communicate with others in the network who require services. A way of identifying services and entities that are available in the network must be defined a priori. Users should be able to point to services in a uniform way while searching, since this would be an obstacle for large scale applications where users identify services in different ways. Thus, the information for representing services and entities should be formatted in a common language that it can be understood by all involved parties. The services and entities should be identified with a unique identifier to facilitate the search.

We propose the use of the eXtensible Markup Language (XML) to describe services, context information, entities, and their capabilities, and a model for their naming and identification. This document provides a description of service information, while context information and its exchange are further discussed in [57].

6.3.1 Taxonomy trees

The model that we propose to identify services, entities, and context information is via taxonomy trees. We have designed three trees, one for each root node in order to differentiate the different information. The exchange and management of the taxonomy trees is not covered by this thesis as we assume that the users already have these functionalities.

All information is stored in the tree, thus the lower levels of the tree are more specialized. The leaves of the tree specify the services, while the intermediate nodes can be considered as a service group. In figures 6.1 and 6.2 we show two examples of taxonomy trees, for services and entities respectively. Each node is identified by the

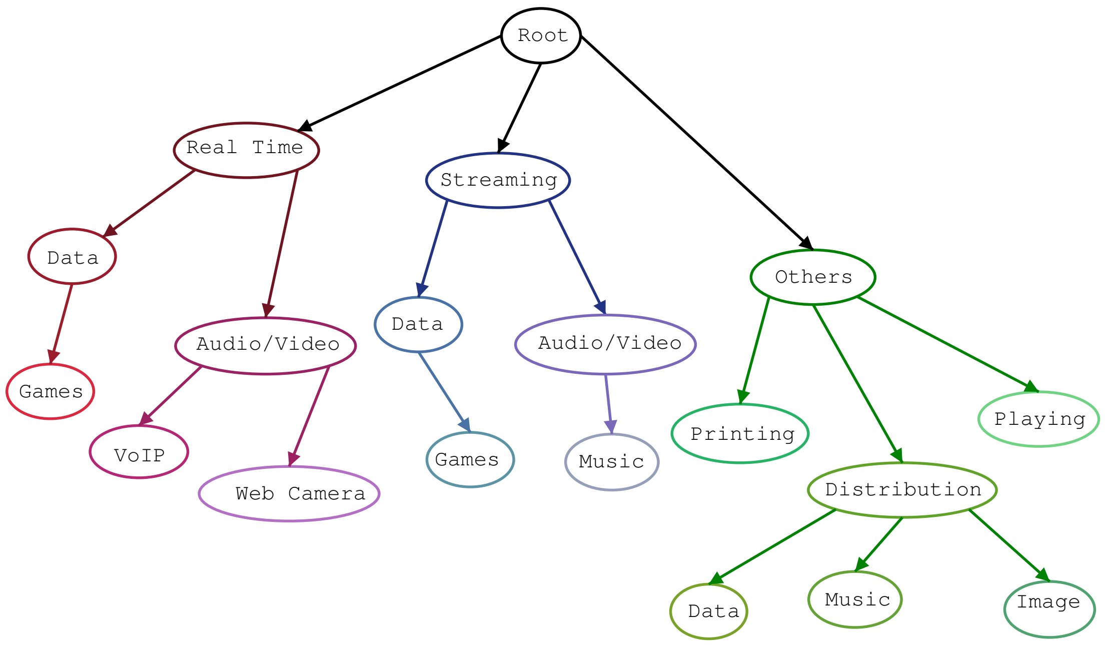

Figure 6.1: Service Taxonomy Tree

path to reach it from the root node. For instance, according to the figure 6.1 the service "Printing" is identified by the path "Root/Others/Printing".

Services are described in documents formatted in XML, which specifies the service parameters and actions that can be performed on the service. Each service can have any number of properties associated with it. This document does not provide a specification of which properties are mandatory for any single service, but we propose to refer to service description in the *Konark Discovery Protocol* [54] or to the *UPnP* [13] specification. Furthermore, using XML for describing the nodes of the tree gives the node easy extensibility of its capabilities.

6.4 Overview of the architecture framework

Assuming the presence of a mechanism for locating users in the network and protocols that offers mobility to users, we need to locate the discovery protocol in the TCP/IP protocol stack. Figure 6.3 shows the underlying protocols.

At the link layer we suppose that the device has wireless connectivity, for instance GPRS or WLAN connections; the information gathered at the link layer can be useful

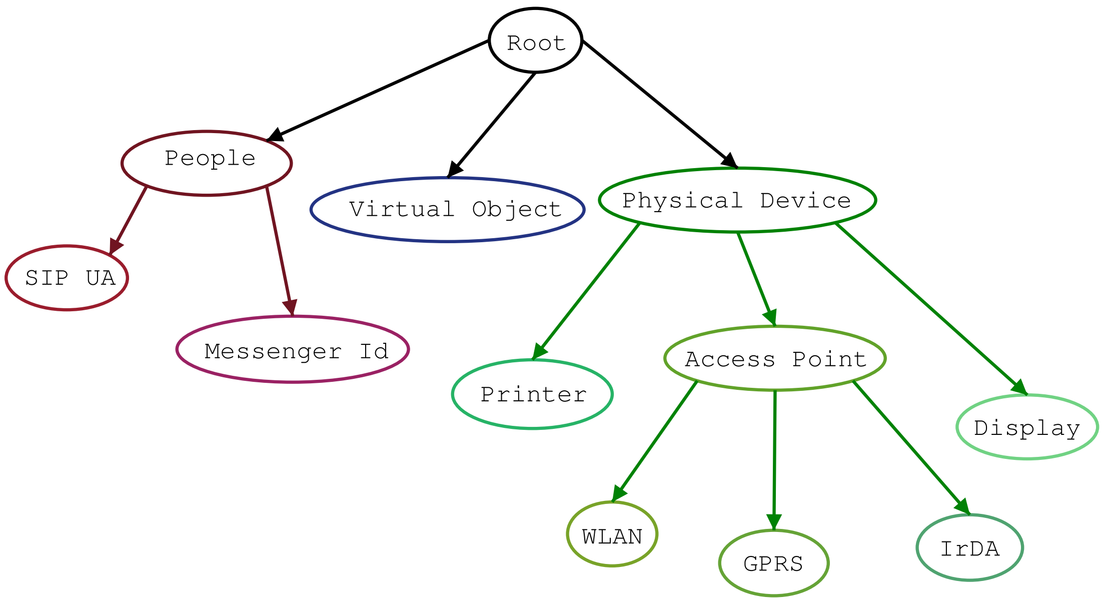

Figure 6.2: Entities Taxonomy Tree

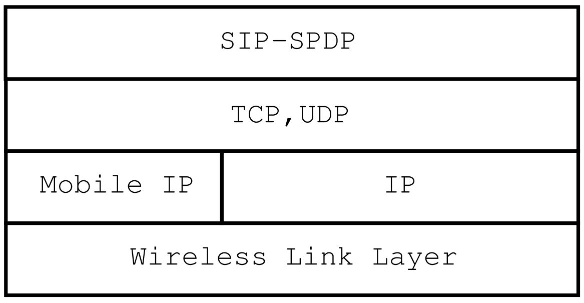

Figure 6.3: Protocol Stack Overview

to understand network availability in order to modify the behaviour of the upper layer protocol(s).

The network layer is characterized by the use of the Internet Protocol (IP), since we suppose that the device is provided with an IP address for communication with other hosts. Since we are working with wireless networks, the mobile device might need to move to different networks while maintaining the ongoing communication, thus, we propose to use Mobile-IP.

At the transport layer, we don't specify any mandatory protocol, since the discovery protocol relies upon SIP for communications.

At the application level, we utilize the Session Initiation Protocol (SIP) to handle sessions between entities. The proposed Service Peer Discovery Protocol (SPDP) is considered as extension to SIP according to the SIP event framework specification [42]. Thus, we co-locate a SPDP User Agent with a SIP UA, capable of handling SIP sessions and redirecting service discovery event to the SPDP UA (Figure 6.4).

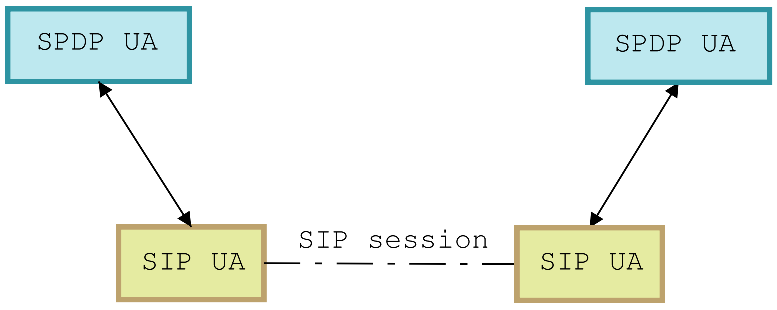

Figure 6.4: User Agent

6.5 Service Peer Discovery Protocol

The Service Peer Discovery Protocol (SPDP) is designed to discover services and to offer a model for negotiating services between end-points without requiring third party negotiation. SPDP is defined as extension of the SIP event framework [42], thus, its messages are carried in SIP packets and it inherits all the request routing and security features of SIP, as well as using SIP for naming and localization of users. As described in sections 4.7 and 6.5.1, the SIP event notification defines two method, SUBSCRIBE and NOTIFY, to handle eventing. All the messages are expressed in XML, and the entities and services are organized in taxonomy trees, as introduced in section 6.3.1, to offer easy and unique identification of the resources.

The SPDP differs from the discovery protocols (UPnP and Jini) described in chapter 2, as it does not require any central server to assist in the negotiation between end points (it is done directly between end-points after the service is located) nor to provide mapping between service and entities. Furthermore, the protocol is designed to perform discovery in wide networks and in heterogeneous networks, such as the combination of GPRS and WLAN, using the services provided by Context Servers. The design of the Context Server and its (network) location are addressed in [29].

6.5.1 SIP extension for service discovery

The Service Peer Discovery Protocol implements an extension to the SIP protocol [20] in order to provide capabilities to SIP User Agents to discover services in the network. The event package used for this purpose is called “Service Discovery”. My proposed package defines that the “Event” header of the SIP message should be set to “sdpEvent” and the “Content Type” should be set to “application/sxdp-xml”.

A user agent interested in a particular service issues a SUBSCRIBE request addressed to a multicast channel or to a specific user. The request is processed by the peer, which replies with a NOTIFY message.

6.5.1.1 SUBSCRIBE

The SUBSCRIBE message is used to establish a dialog between peers. The duration of the subscription is indicated by the value of the “Expires” header. A subscription is always associated with a dialog, that is specified by the dialog ID, and by the event package name. This is confirmed with a 2XX SIP response message which indicates the expires time for the subscription. If the subscribers want to unsubscribe before the timer expires, they send a SUBSCRIBE request with “Expires” value of zero.

The package “Service Discovery” defines a default value of “0” seconds for subscription for service discovery. In this case, each subscription triggers a notification, which closes the dialog. The validity time of a subscription for service discovery is limited to “0” seconds as the discovery message is issued to determine the availability of resources and their presence in the network. A longer expire timer will cause the client to be subscribed for a service with the providers. The SUBSCRIBE message defined in the package is similar to MESSAGE request defined in the Instant Messaging extension specified in [22], however, a SUBSCRIBE message requires the server to send back a notification; the notification guarantees that the server has processed the message since a 200 response only acknowledges the message received.

During the discovery phase the subscription must be set to the default value and the message body formatted in XML must contain the rules for the server. After discovering a service, the client might be interested in changes of user or service status, thus, the “Expires” timer can be negotiated between the client and the server.

6.5.1.2 NOTIFY

When a subscription is received, the notifier sends a NOTIFY response, which contains the event package name and dialog ID, an identification of the subscription (if it is present), and a body. The body contains the information requested in the subscription and, based on this, the client can act (after updating its internal knowledge). If the client subscribes for eventing, the "Service Discovery" defines a lower limit of 5 seconds between two following notifications to avoid network congestion, as proposed for the Presence package (section 4.7.2).

6.5.2 Entities

The SPDP system consists of user agents, that can join or persist in the network and interact each other, and a Context Server, that keeps context information. The user agent client is able to query other peers in the network for services by peer-to-peer messages. The user agent server is responsible for handling requests for service, checking if the service is available, acting as a proxy, and for delivering the service. The proxy accepts requests and performs discovery on behalf of the client.

In this setting a file server has a user agent, which offers a storage service for contents. It is this user agent, whose functions help users to access the file server's contents while dealing with bad connections, limited local memory, and limited battery power.

6.5.2.1 User Agent

User Agents are logical entities that can act as both a user agent client (UAC) and user agent server (UAS). The user agent is identified by a SIP URI and should be able to move around the network(s) as nodes are relocated. It consists of a SIP User Agent which handles the communication with other entities at the application level; it should also be able to send and receive events and to keep track of subscriptions.

The user agent uses an embedded discovery application to provide services via queries and responses which are formatted in XML. This user agent can query other agents to retrieve information about their services and it should be able to respond to such requests from other agents. This enables devices to store information and provides a method to access this information. The XML document for an agent describes the agent's properties, such as network identification, actions, and embedded services.

The agent is located by the SIP URI which it registered with the Context Server for the current domain. As examples of services we will examine both a file server and Context Server in the following sections.

6.5.2.2 File server service

A file server service can be offered by an entity in the network; this section explains advantages and disadvantages of the Service Peer Discovery Protocol in conjunction with such a service. The device acting as a file server needs to support specific functions in order to support requests to store and retrieve content, such as video or music files.

Having a device in the network with such functionality enables more content to be available. A small device can save cost and/or battery power by waiting to fetch a faster and/or cheaper connection than the one used to perform the discovery of the file server. The main purpose of a file server is to extend the effective storage capacity of devices which can access it.

The file server is located by its SIP URI and it can be used during service delivery as intermediate node. The client or the server might choose to forward the content to the file server using the latter as a temporary repository. Figure 6.5 shows a possible scenario: a client "sip:user@domain.se" discovers a service handled by "sip:provider@wireless.se" (**1** in fig. 6.5), the server delivers the content to the file server "sip:file_server@wlan.se" (**2** in fig. 6.5); "sip:user@domain.se" moves to another location (**3** in fig. 6.5) and retrieves the content from the file server (**4** in fig. 6.5). This solution considers the case of disconnection during the communication between two peers and offers an alternative means for content delivery. Furthermore, an intermediate file server can be used to determine policies for routing when there are multiple receivers; a similar solution is already designed for Elvin content based messaging services [8].

6.5.2.3 Context Server

The use of the Context Server is proposed in order to keep context information in a network. Several Context Server might serve the same network and the need to

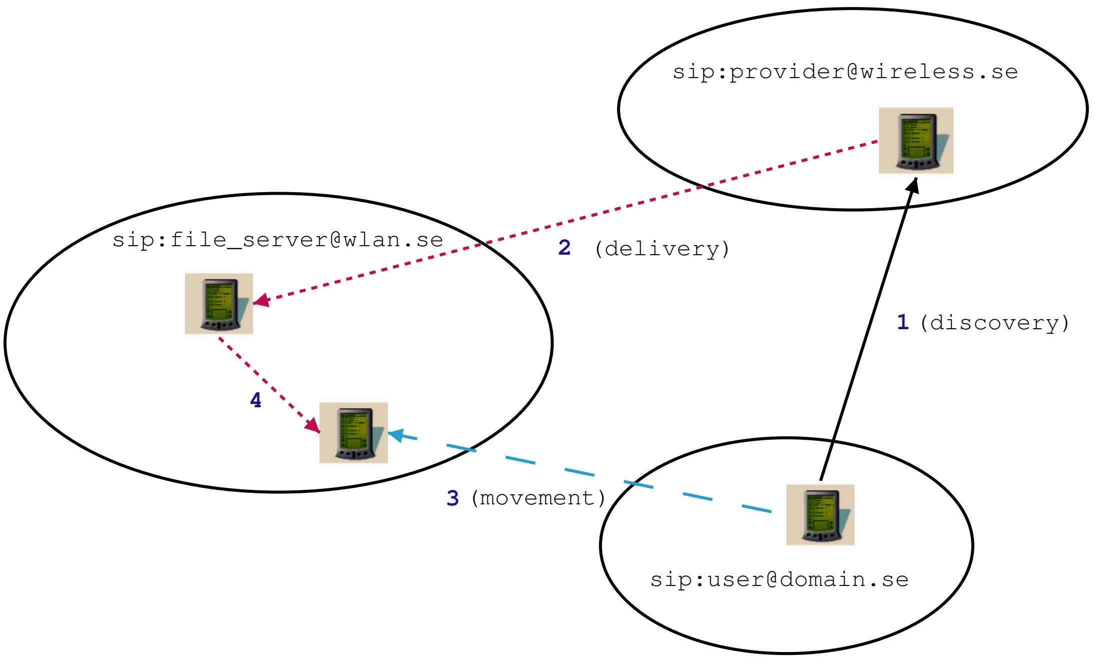

Figure 6.5: File server: delivery scenario

cooperate in order to ensure data consistency. The design of a Context Server and of the services that it offers are discussed in detail in [29].

The Context Server enables context information to be available to users in the network; the Context Server collects and maintain "knowledge" about the network, services, and position of users. It becomes aware of users, since users register their presence where they discover a Context Server in an area.

Using the Context Server with the SPDP protocol enables user to be aware of the location of peers that can provide a required service. Additionally, users can learn about neighbouring networks with good support for a service, thus they may decide to pro-actively perform a handover (this is further described in section 6.6).

6.5.3 Messages

The messages are carried in the body of SIP messages and they are formatted in XML. The Content Type of the SIP message is *"application/sxdp-xml"*. The SPDP protocol defines the messages in the SIP event notification framework, as described in section 6.5.1, and defines three methods to handle service discovery messages. The protocol does not provide message acknowledgment, because this is already done at the SIP level.

6.5.3.1 Methods

The method is carried in the request and specifies the action that the client wants to invoke on the server. The methods, we are concerned with, specify actions that concern the discovery protocol, seen as an application behind the SIP User Agent. Thus, the methods are related to those of the SIP package “service discovery”, i.e., SUBSCRIBE and NOTIFY, and are defined for the application that handles the discovery messages. Three methods were defined: DISCOVERY, ACCEPT, and DENY; these methods initiate and exchange service information and can terminate the discovery process.

DISCOVERY is used to initiate the discovery process and to query a server or an entity for its capabilities or for its data. The method defines messages for requesting information about both services and peers, as specified in the header of the message. The message can be sent on a multicast channel, unicast to another peer in the network, or unicast to the (file or context) server. The client can include a list of conditions that the server checks in order to map the request to the right service. The discovery packet includes a ReplyTo field, this is used if the client performs the discovery on behalf of another entity. DISCOVERY is only sent by clients and is acknowledged with an ACCEPT or DENY packet.

ACCEPT is sent as response to a DISCOVERY message if the server supports the client’s request as specified in the DISCOVERY packet. The query packet includes a list of services or peers, depending on the kind of the request. The ACCEPT message is carried in a notification (i.e., a NOTIFY) and it closes the discovery session.

DENY is issued as the response to a DISCOVERY message if the server can not satisfy the request. A reason can be specified in the packet as a three digit numeric status code, see table 6.1. The DENY is also carried in a NOTIFY.

Table 6.1: DENY code responses

400	The request is malformed, the server is not able to understand it
404	The service is not found or the server does not have any knowledge of peers
406	The request cannot be satisfied. The conditions do not match the properties of the available service
480	The server can not satisfy the request. The service is temporarily unavailable
486	The server does not accept other request. The application is busy

6.5.3.2 Header fields

The SPDP header consists of a field name, the corresponding value, and attributes. This section explains these headers, which are used in the message according to the methods and the discovery path of the taxonomy tree (see section 6.3.1).

Sender indicates the initiator of a discovery request. This must contain a SIP URI and an optional IP address; the protocol supports both IPv4 and IPv6 address. The Sender header contains the IP address to provide some knowledge of the user's location.

Method indicates the type of message. The method also defines which fields are optional or mandatory in a packet.

ServiceID indicates the service path (i.e., the specific service) that the client would like to discover. The service is identified by a path in the service taxonomy tree, see section 6.3.1. The ServiceID and the PeerID are mutual exclusive.

PeerID indicates the entity path, according to the taxonomy entity tree (Figure 6.2), that the client wishes to discover. The path is defined with respect to the entity taxonomy tree, see section 6.3.1.

ExpireTime specifies how long the message content is valid. This header field is optional for ACCEPT messages, but it must be used in DISCOVERY messages. The protocol does not use any timing scheme for messages as this is already provided by the underlying SIP protocol. ExpireTime can be useful in ACCEPT message to indicate how long the peer can offer all the services indicated in the content.

ReplyTo specifies the SIP URI and address of another entity, if this SIP User Agent can not provide, but knows of another User Agent who can provide the service. The field contains a priority attribute.

ConditionList is an optional field for DISCOVERY message, it indicates the parameters that the service or the entity should match. It enables a selection process within the server.

6.5.3.3 Body

The body of the SPDP message is only used in ACCEPT messages and it describes the services or peers found. Below is a description of the body content according to the discovery path of the taxonomy tree:

- Service discovery. The body lists the services found. They are defined by the ServiceID, name, source, and optionally expiresTime. The ServiceID describes the path to the service within the taxonomy tree. The name is the human name for the service, for example the name of a printer. The source field indicates the provider of the service and the expiresTime indicates when the service is available.

- Peer discovery. The peers found in the network are identified by an Identity field, their SIP URIs, and their IP address. The body also lists for each peer the networks to which they are connected and the cellId of the network, if it is available and relevant. The expires field indicates how long the peer is expected to be registered at this location with this particular SIP URI, consistent with the registration procedure. Additionally, the services field consists of a list of services that the peer can provide; the service is identified by the path to it in the service taxonomy tree.

6.5.4 Session Discovery

The Service Peer Discovery Protocol enables agents to establish dialogs in order to ask other devices to perform operations. Its messaging consists of sending a message with the command embedded in the body and of an acknowledgment for each message. Thus, communication is a simple two way exchange i.e., a transaction. For the purpose of SPDP, the operations are specified in the body of SIP messages, while routing and acknowledgment are based on SIP. The flow chart in Fig. 6.6 shows the operations, that the User Agent does, for the discovery protocol.

6.5.4.1 State Diagram

The flow chart shown in Fig. 6.6 describes how the user uses the service discovery protocol and defines the exchange of messages between a client and a server. The User needs to have a SIP UA running to have connectivity at the SIP level, and he/she needs to register with the location server to be reachable. As described in section 6.4, the SIP UA is extended by the SPDP UA, which handles discovery messages. The next step consists of registration with the Context Server (sec. 6.5.2.3), which is located as described in the next section. Thus, the user can be directly contacted by other peers via multicast messages or via the context server.

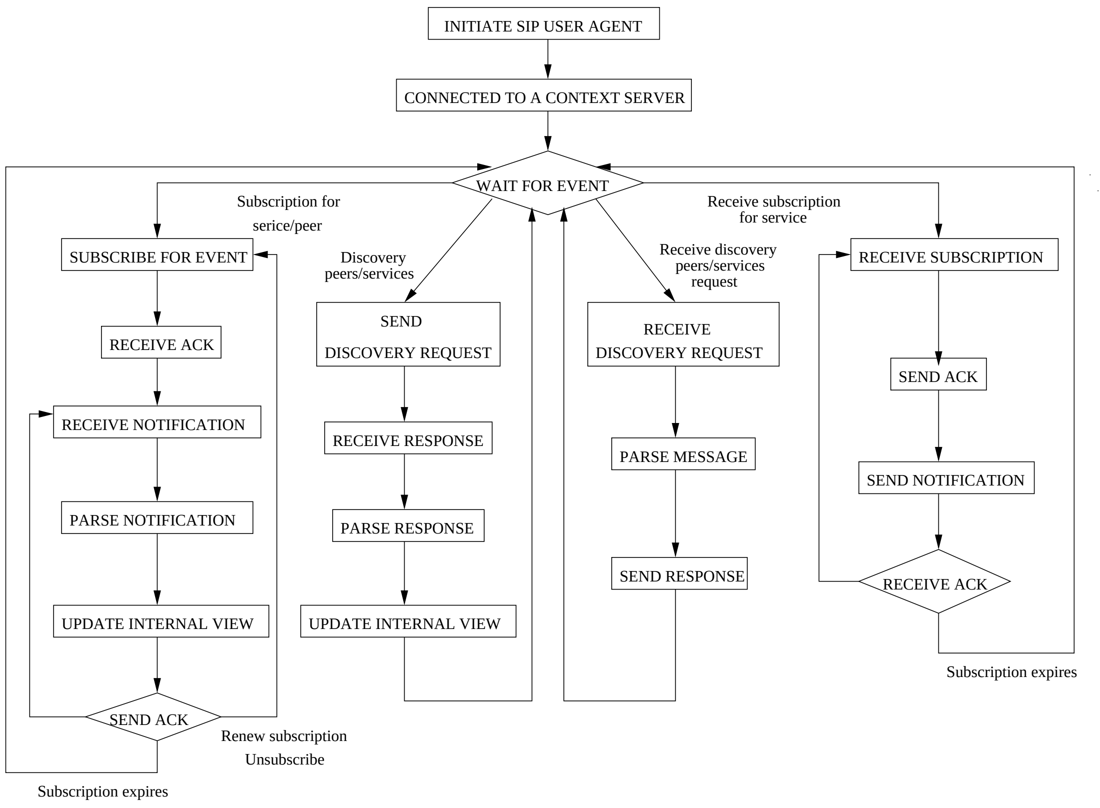

Figure 6.6: SPDP UA Flow chart

6.5.4.2 Locating a Context Server

A user needs to register its services and its location with a context server. More than one context server can be present in the network, thus, the user should locate the context server responsible for their current domain. Furthermore, there is the need for a mechanism to locate the context server that is best able to handle user requests in order to do load balancing of traffic among context servers. The problem can be solved by using SIP naming and localization for SIP server, as earlier described in section 4.6, which is based on the capabilities of the DNS to resolve the request to the right server.

Scalability and high availability are important, this can be provided using clustering techniques. The context servers of a certain domain can be set up in a cluster and using DNS SRV [17] they are ordered based on prioritization and weight to achieve load balancing. The SRV record for a domain can be configured so that there is a preferred server specified with a low-numbered priority value and backup elements with higher priority value and ordered by weight. The appropriate server is selected choosing the one with the lowest priority, and, if there is more than one with the same value, the server with greatest weight is contacted.

For instance, a user "sip:roberto@wireless.com" tries to contact the context server responsible for "wireless.com" domain. The user determines their current domain via DHCP. Thus, SPDP tries to open a SIP session with the context server, "sip:context-server.wireless.com". The session will be routed by SIP servers to the context server resolved through DNS lookup. The solution requires global connectivity of the context servers of the area.

6.5.4.3 Discovery

The SPDP protocol provides two discovery mechanisms according to the network the client is attached to and to the reason for the discovery (i.e. service request or peer request). The protocol can be used both in small networks and in wide area networks and depending on the current context the user agent can decide to perform the discovery in a given way. In the introduction, we stated that the device may be connected to multiple and different kinds of networks, thus, the discovery should consider the limitations and the features of each type of network; for instance GPRS networks do not support multicast packets since multicast messages would require the GGSN node to handle the group membership and could cause large amounts of traffic since the GGSN needs to make and sends a point-to-point copy of the multicast message to each mobile terminal. Furthermore, the protocol should be able to scale across a wide area and to traverse multiple hops, thus, the protocol cannot rely upon multicast and broadcast messages to reach the required destination.

Local area. Discovery in a local area is done according to which network the client is attached. We suppose for the local area case that the client is connected to a WLAN network, which may be faster and cheaper than a GPRS network. The client may decide to contact another peer via unicast (if this peer is already known) or to send out a discovery message on a well known multicast address for the relevant scope. Every server who receives the request parses the request and sends a response to the originator of the request. Thus, by multicasting a message the client can query other peers about services without knowing a priori of their existence or address. The advantage of the multicast request is that the client only sends one message, but queries several peers at the same time. However, the client needs to be able to handle all the acknowledgments and responses for this packet.

Wide area. As noted earlier, discovery cannot be done using multicast in a wide area network; both because the packet may cross different networks and due to limitations of the underlaying network technology. In such a scenario, the client needs a way to know the peer's address. To achieve this goal, the user queries a Context Server, located as described in section 6.5.4.2, to get the URIs of peers that **can** offer the service the client is looking for. Although this solution seems to rely upon a central server for discovering services, it does not require the Context Server to subsequently be part of the service (transaction), which is handled directly by the the peers. This is because the client directly contacts the other peer(s) through a unicast message to use services.

Service. The client performs first an internal search to see if already can provide the service or if it already knows another peer(s) who can provide the service required. However, if the service is not present and there is no known peers who can provide it, then the agent will perform a peer discovery based on the client's current connectivity in order to learn of peers that might provide the required service, or the client will multicast a discovery message on the local network. If the client has information of providers of the specific service, it can utilize the service by directly contacting the peer. In the example (shown in fig. 6.7) the client ask for a printing service. Once the service is discovered, the client downloads a printing service description formatted in XML, which defines the service's properties and the actions that clients can perform on it.

```
<sxdp>
    <sender>
        <entityID>sip:roberto@wireless.kth.se</entityID>
        <entityaddress entitytype=”IPv4”>192.168.15.34
    </sender>
    <method>
        <name>DISCOVERY</name>
    </method>
    <serviceID>
        <path>Root/Others/Printing</path>
    </serviceID>
    <expireTime>3600</expireTime>
    <replyTo priority=”3”>
        <entityID>sip:roberto@wireless.kth.se</entityID>
        <entityaddress entitytype="IPv4">192.168.15.34
    </replyTo>
    <conditionList>
        <condition>
            <name>paper</name>
            <value>A4</value>
        </condition>
        <condition>
            <name>quality</name>
            <value>600dpi</value>
        </condition>
    </conditionList>
</sxdp>
```

Figure 6.7: Service Discovery Message

Peer. A peer discovery occurs when the client does not know of other peers in the network that have the service(s) required. This peer discovery is done by querying both the Context Server or other peers based on the parameters that the client specifies in the message, such as network connection, cell Id, or services embedded. The list of peers is inserted in the message body of the SPDP response, and it is formatted in XML. This method can also be used to query a peer about its capabilities if its SIP URI is already known.

6.5.4.4 Eventing

Once a client has discovered a service or peer, it may be interested in changes of the service status or of changes in a peer's properties. The SPDP protocol proposes an eventing schema based on the SIP event framework [42].

Subscription. The client subscribes for a specific service or peer URI by sending a SUBSCRIBE message to the peer that handles the service or to the interested peer. The service identification or the peer properties, identified by a path according to the taxonomy trees (section 6.3.1), are specified in the body of the SUBSCRIBE message. The subscription is time limited and the expires time is expressed in the Expires field of the SIP message header; the time is negotiated between the two entities and it can not be longer than indicated in the request message. Each SUBSCRIBE message is acknowledge by a SIP ACK message.

Notification. When a peer receives a subscription request, it acknowledge the message and it sends a NOTIFY message, that contains in its body the current state of the subscribed services or of its capabilities according to the subscription reason; each message is acknowledged. If the status changes, the notifier issues a NOTIFY message to the subscriber(s) indicating only the changes.

6.6 SPDP for smart handover

This section provides a description of a proposal to use the SPDP protocol to help mobile users learn of the neighboring network using the information kept by the context server, in order to negotiate services with neighbouring network entities **before** changing their point of attachment. This could include locating mobility agents as proposed in Jiang Wu's licentiate thesis [58]. As described above, the SPDP protocol enables SIP entities to communicate and to query other entities for their capabilities. Because the context server provides an overview of the networks in the area it can offer to users specific context information about neighboring services in addition to the services in the **current** cell. Furthermore, users might be attached to several networks with different media access technologies and services thus they may contact context servers in each of these networks.

Following this step, SIP might be used to contact and to establish a session with the access points in a network, since their presence is part of context information kept by the context server (which we assume has complete knowledge of the area). Based on this information that we received from the context server and further information from the APs, the user agent can take decision about roaming and changing its point of attachment. Furthermore, the user may decide to use different interfaces, for example switching to a GPRS connection. Since we suppose that the network area is always covered by GPRS signaling, a user might redirect their traffic to the GPRS interface, while waiting for WLAN connectivity.

Thus, we needed a smart approach to notify the correspondent nodes of the new location of the user, and hence utilize the best interface to which to redirect the session. Some mechanisms to redirect SIP sessions and to handle SIP mobility of the terminal and user have already been introduced in section 5.2; for instance, the session might be redirected to a conferencing address that the users involved in the session join, or a RE INVITE message might be issued. The session is subsequently directed to the new location.

Utilizing more than one interface might be of interest while the user needs to change their point of attachment from one interface to another. Registering multiple addresses with a registrar enables reachability at both locations. Thus, the user might negotiate for their new IP address before performing a handover. This is especially useful if it can predict its movement and therefore could contact with the new access point while on the move.

The SPDP protocol might be used to discover network entities in a local or wide area, since it is possible to associate the access point with a SIP entity which might provide connectivity as a service. This scenario would enable the user to contact the access point and to negotiate its association with the access point before moving, thus exploiting the possibility of anticipating the handover for ongoing sessions and redirecting them to the new location. The SIP handover is performed at the application layer, but, as explained in section 5.2, it might be preferable to have Mobile IP to handle handover rather than requiring TCP connections to be re-established.

6.6.1 Redirection for content delivery

The SPDP protocol enables users to negotiate the delivery of content during the discovery session. The ReplyTo header field indicates the user's preferences for the destination hosts. This feature when used in conjunction with the file server service might offer new possibilities to users in order to better used the network, since the user might discover at need and retrieve the content using the interface that might offer better performances in terms of QoS.

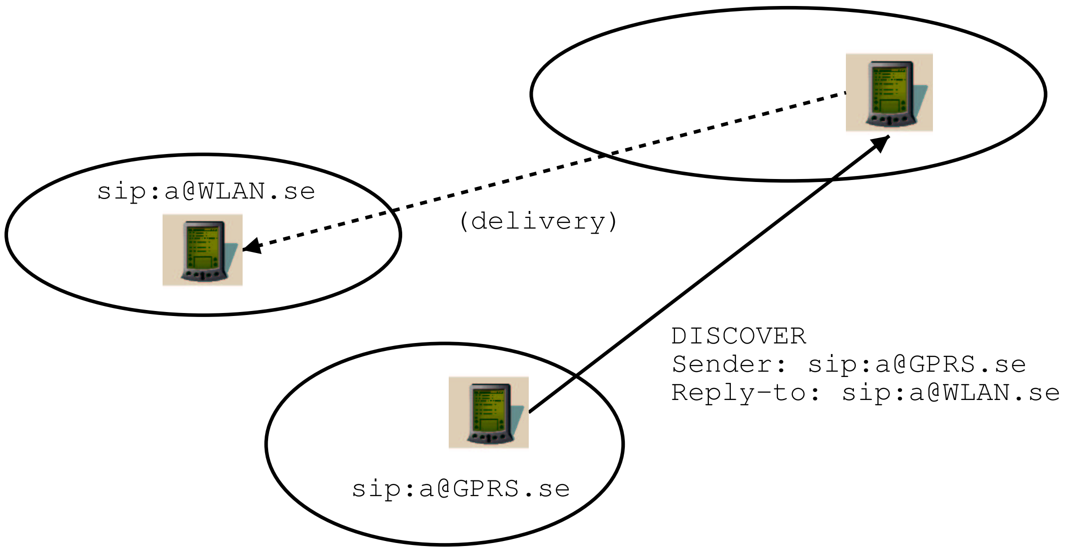

Figure 6.8: Redirection of content delivery to different interface

The user indicates the SIP URI of a file server, that it has already discovered, and its preference for delivering or receiving content in the discovery request. The service provider should accept the request by replying to the host identified in the ReplyTo field. In such a scenario, a user might discover services using one interface, while redirecting the content (delivery) to another of its interfaces while is configured to use a network offering higher throughput (Fig. 6.8). Furthermore, the user might be on the move and, using the redirection feature, it might communicate the new delivery location.

Chapter 7

Analysis of the Service Peer Discovery Protocol

7.1 Introduction

The Service Peer Discovery Protocol (SPDP) introduced in the previous chapter addresses an area with interesting opportunities for mobile users while they are on the move. The SPDP protocol enables users to discover services or available resources in their current area. The behaviour of the protocol depends of the available network interfaces having the ability to use different kinds of discovery mechanisms in order to be effective in a local or wide area. The following sections analyze the use of the SPDP protocol in wireless networks, both cellular wide area networks and WLANs, and examines how the functionalities of the protocol interact with the existing services that the network already provides.

7.2 Cellular networks and services

Cellular networks are an interesting market area for developing applications and services for mobile users because of their extremely large user population. In such networks, new applications need to be aware of the existing traffic in order to avoid negatively impacting it. Traditional cellular networks were designed to be used for voice traffic and the resources were allocated to a user for the entire duration of a call (circuit switched). New applications and services have been proposed to users beyond traditional voice traffic; for instance, the Wireless Application Protocol (WAP) [7] provides an integration of Internet services into the cellular network. However, these Internet services (i.e., web browsing and so on) are under control of the operator, since users can only access Internet services through WAP gateways. Additionally, WAP does not suit well delivery of large quantities of content and was developed primarily to integrate simple data services with the GSM network.

The development of the GPRS network offers attractive new solutions to users, since the data traffic is packet switched and the user can interact directly with Internet services. This enables new applications to be built in order to meet user and application requirements; resources are used in better way and bandwidth can be allocated as needed, while ensuring that data traffic does not interfere with the circuit-switched voice traffic.

[38, 37] propose to use VoIP in order to deliver voice traffic and to provide interactive multimedia applications over wireless access networks. These papers consider the fact that these services require a lot of bandwidth and special policies to assure a certain level of QoS. Furthermore, the communication should be handled end-to-end in order to provide these services on top of IP over the wireless link access. [37] proposes replacing the current Base Station Controller with a router with a radio interface in order to reduce the latency that is due to the transcoding in the base station controller since the air link propagation and the transmission over the IP network was negligible when the voice and data traffic (web content) is transferred between a phone attached to a cellular network and a user using a VoIP/SIP client.

[38] proposes a model for Internet content and for related applications in order to maximize the number of users considering that multimedia applications have special requirements in terms of latency, robustness, and speech quality. An agent model with some intelligence in the user's device, the network, and the base station has been proposed in order to enable very efficient use of the resources by utilizing smart delivery of content. Agents might behave intelligently on a local level or in concert negotiating their communication and their resource requirements, thus adapting the communications or simply adapting the codec to available network resources, remaining battery capacity, and/or memory storage both in a local context and/or in a non-local context. Furthermore, the use of VoIP increases the voice traffic per cell (sector); using VoIP traffic over wireless link the voice can be delivered via at almost 1.2 kbps [37], giving the possibility to establish upto 26 simultaneous (sessions) calls during the peak hour in a macro cell. This increases the efficiency of the network and as well it in-

creases the interest in enabling services on cellular networks (i.e., GPRS). Since the bandwidth required for VoIP traffic is so low, a user is able to download content while communicating with others at the same time.

7.3 WLANs and services

WLAN networks offer services at high throughput and the available technology assures high bandwidth capable of providing multimedia applications over wireless access networks. In a typical scenario, we can consider WLANs being used in hot spots where users might access services without having any significant limitations on bandwidth; thus mobile users roaming between cellular networks and wireless network can determine the appropriate network link to use for the required traffic, for instance using an agent to make optimal decisions based on context information.

If we consider wireless reconfigurable networks where the mobile users are on the move and should deal with dynamic changes of network resources, an overlay network built using SIP can provide end-points with connectivity (unless the device is out of range or inactive). A negotiation model with SIP entities can be used to adapt to the new environment. For example, a mobile device faced with unpredicted network outage should be able to switch to a different network redirecting the ongoing sessions transparent to other applications that use SIP for signaling. SIP can also handle some variations in link properties by changing communication parameters according to the current link conditions.

7.4 Integration of SPDP with current services

In cellular networks, where the voice traffic takes a large part of the available bandwidth, the SPDP protocol should avoid blocking voice traffic. Thus, the message exchange should not consume so much resources as to cause a degradation of this primary service. However, some studies done in the cellular networks (described in section 7.2) have shown that it is possible to move voice traffic from the traditional circuit switched network to a packet switched solution or IP-network. This solution offers the possibility of smarter use of the resources by using VoIP for carrying voice and SIP for signaling. In this approach resources are not allocated and reserved for phone calls, since the communication is completed carried in voice packets and data may be delivered as background traffic (the key is the low bandwidth required by the voice traffic versus the link throughput). Furthermore, users will be charged only for the packets and not for the time resources are allocated (as they would be in a circuit switched system).

This solution could be combined with the service discovery protocol, since then discovery messages would not influence the voice traffic. There is a potential problem that when a SIP infrastructure is shared between call signaling and service discovery, that the discovery traffic will interfere with call signaling traffic. A solution that differentiates the traffic (i.e., DiffServ) could offer a better performance without causing involuntary blocking of the voice traffic. SPDP requests differ from other sorts of SIP requests in that they carry media, in the form of XML instructions, as payload. Conventional SIP payloads carry signaling information about media, but not media itself. If we consider the SPDP protocol for discovery in local area networks (i.e., WLANs), the messaging between users does not require very much bandwidth as compare to the current WLAN link throughputs. A typical discovery message is around 1150 bytes, while the length of a notification depends on the number of services the request matches. Thus, if the device and the protocol were managed by an intelligent agent, the number of services in the response might be limited based on the local and external context information.

In a cellular network, the SPDP protocol requires that the communication should be handled between end-points or with the context server, since strategies similar to multicast will produce a considerable number of responses messages which is not desirable in a GPRS network. Thus, we only consider unicast traffic for discovering services or a peer's location. Considering a typical GPRS network, the maximum bandwidth is around 40Kbit/s per user, thus, a GGSN node, which support throughputs between 150-250 Mbit/s, enables a peak usage of about 5000 users who might be served by a single SIP proxy server if all the traffic is handled by SIP and even if all traffic is directed outside the GPRS network. Additionally, users subscribe with a GPRS operator, since there are in general few operators in each country, thus most of the traffic is routed within a single operator's GPRS network, unless the user is roaming. Furthermore, since this operator owns the GPRS network, this network can be designed with a hierarchical structure of SIP proxies to provide better performance for the subscribers without creating bottlenecks in any individual SIP server.

To enhance the functionalities provided by the SPDP protocol, we propose to integrate the network and the devices via an intelligent agent, capable of making decisions based on context information. This model, already proposed in [38] to maximize the number of user and to use the resources in an efficient way, can enable SPDP clients to discover services while taking into consideration the local condition of the client and the service provider by adapting the agent's decision-making.

The SPDP protocol uses the Context Server [29] for locating peers, however, the Context Server can also provide information to user for adapting mobile application. This approach might be very effective in a wide area wireless network (for instance, GPRS), since information about wireless channel conditions and resource management might increase the performance for mobile applications [34]. Variation in the available bandwidth, both peak, minimum, and maximum, latency and location of the context information can be used to select the approach used by the SPDP protocol while discovering services; for instance, if the device has multiple-interfaces, "knowledge" of the environmental conditions can influence the choice of the wireless link used for service discovery and content delivery. Most of these issues are also applicable to WLAN or ad hoc networks, thus providing a discovery algorithm that adapts automatically to network conditions. In case of an ad hoc network, a routing algorithm is needed to route the packets between users, which routing algorithm to use could be selected adaptively.

Furthermore, the Context Server (described in section 6.5.2.3) collects information **about** networks, such as neighbouring networks, along with other context information, sensor data, etc.. This information is stored in a database, hence an estimation of the channel conditions can be computed and can be shared with users that subscribe to this context information (using a protocol, such as the Context Data eXchange Protocol proposed in [29, 57]). Using this context information the SPDP protocol might decide to offer enhanced services; hence the handover time might be reduced (i.e., using similar strategies described in section 6.6) or the communication can be adapted by moving the content delivery to a file server service (as described in section 6.5.2.2).

7.4.1 Robustness of the SPDP protocol

The great benefit of the SPDP protocol is that it does not have a single point of failure. As we discussed earlier, SPDP uses different strategies to discover services and adapts the algorithm to the wireless interface in use and to the current discovery area. It is important that the packets arrive at their destination; SPDP messages are reliable, since reliability is provided at the SIP level, even if UDP is used as transport protocol. Additionally, the SPDP protocol can function effectively in an ad hoc network. When devices are close enough to communicate directly, a single SPDP user agent would be able to accept messages for discovering peers in the network and the SPDP protocol might be extended to provide relaying of SPDP messages by adding routing capabilities.

Although the Context Server has complete information about the network, a given device could offer additional local detailed context information that is of interest to users in the same area by sharing services such as sensor information, a device's embedded capabilities or hardware, etc.. For instance, a user might detect the presence of a camera or speakers (for listening to music or news) when entering a new area, thus they might redirect media to such devices or use these device(s) to build other services.

7.4.2 Delay

This section considers the delay in cellular networks, since this delay may be significant. Delay in cellular systems comes from several sources. Competition at the base station, between packet switched data and circuit switched voice traffic when the wireless channel has limited resources causes delay in packet data transmission. Furthermore, the traffic coming from the public Internet is routed by the GGSN to the relevant SGSN node that is managing the mobile device in a specific area, SGSN nodes may be loaded by different amounts of traffic. Thus, the Round Trip Time (RTT) may vary. Delay also occurs due to the error correction done to increase the radio channel performance, this introduces varying delay for the wireless channel (as the number of retransmission will not be constant).

An IP packet is generally fragmented into PDUs for transmission, if a PDU is lost, this requires retransmission of the packet. One must find a good compromise between larger packets which suffer from the need to retransmit all the fragment even when all but one are successfully received and small packets, which introduce extra overhead for their headers. Considering SIP packets, taking into account retransmissions, and the number of messages that are required in some flows, call setup and feature invocation can be adversely affected. To reduce this problem compression of SIP request and responses can be used [11].

7.4.3 Latency

Latency is important when we consider real time application. For a message exchange protocol, such as SPDP, latency is not critical as long as the content is going to be received within a bounded time. The SIP protocol defines the use of timers to estimate when the packet should be retransmitted and the transaction timeouts, if it does not receive response. While using unreliable transport protocols a request is retransmitted at an interval that starts at almost an estimation of the Round Trip Time (SIP defines a default value of 500ms for the RTT) and doubles after every retransmission; a transaction timeouts after 7 retransmission of the packet. However, the timer of the retransmission can be chosen larger if it is known in advance that the RTT is larger (such as on high latency access networks, as cellular networks). Latency may be due to network delays in transmission of the packet and to the application processing time. Thus, a compression mechanism for SIP messages should be used despite a compressed message requiring more processing power it requires less transmission time across the wireless link.

7.5 Different technologies for realizing the SPDP

7.5.1 Using Instant Messaging

An alternative for sending the discovery messages could have been to use the MESSAGE method defined in [22] for carrying discovery request and responses. This solution preserves the end-points role to establish a dialog, thus messages might be sent over different paths from the source to the destination. The MESSAGE extension (described in section 4.7.3) defines rules for processing instant messages; a message might be received by the server, but not processed by the application resulting in lower reliability for the service discovery protocol. Regarding this, the event framework[42] (i.e., the subscription and notification) ensures that the server must respond to the client request in any case, since a subscription triggers at least a notification when it is accepted.

An Instant Messaging model for discovery services has been proposed and designed in [46] for a home network. The authors of the paper prosed a modification to the MESSAGE method, defining a new method called DO, capable of handling discovery messages. When used in small networks the message can send commands to application and devices already discovered in order to trigger actions. This proposal can be used when the user wants to control a device once it is discovered, however, this does not assure complete control of the device and a transaction model can not be established, since instant messaging does not set up a dialog and most of the messages can be exchanged outside an existing dialog. Thus, redirection of the communication or simply the modification of the parameters previously negotiated cannot be done, hence limiting user control.

Using instant messaging for a discovery request in the SPDP protocol could have reduced the burden at the client to respond to a multicast discovery request. However, this model needs modifications to the MESSAGE method to provide a transaction model to negotiate parameters and it needs to work to combine it with the Presence in order to provide enhanced services, for instance the service provider can notify others when the state of a service changes.

7.5.2 Interoperability with other discovery protocols

The SPDP enables communication with devices in both the local and wide area. However, it is important that the SPDP protocol can integrate with existing discovery protocols. For instance, UPnP is widely used in local areas, thus it would be important to have a gateway between the different protocols already in use in order to facilitate the use of the SPDP protocol. Considering a home network, where all devices are UPnP enabled and where a user wishes to control these devices while traveling. A gateway for the entire area might provide the necessary translation between SPDP and UPnP, or since the SPDP protocol is XML based protocol the SIP UA associated with the home device can translate the commands to the specific protocol in use locally.

7.5.3 Using IPv6 instead of IPv4

IPv6 is not yet widely used and it is not clear when the transition between IPv4 and IPv6 will occur. However, thetransition is occuring, therefore it is important to consider the effect of IPv6 on the SPDP protocol and the drawbacks and/or advantages of using IPv6 services. The SPDP protocol is not affected by changing the address scheme, since the header field also supports IPv6 address. However, the issue of the modification of the service, based on a new addressing schema, and how the GPRS network will be integrated with SIP services is already being addressed by the third generation of cellular network. 3GPP defined the use of multiple SIP servers to offer services to the network; voice traffic is completed handled using SIP invitation and session mode.

The improvements by using IPv6 are mainly due to the use of the anycast message, that might simplify the discovery phase. Anycast message introduces an important feature to the discovery protocol since the client issuing an anycast message can query several users and obtain the response from the closest user that can handle the request. This solution reduces the number of message required to discover services in the network; however, the closest user is not necessarily close geographically but rather is close in the sense of network reachability. Thus, strategies used by the SPDP protocol for wide area discovery are still of interest when the client needs to discover a resource in a particular location. Furthermore, the "knowledge" collected by the context server and the decision made in the user agent remain important since context information needs to be considered to address user and network requirements in term of QoS, remaining battery power of the device, and cost of the service.

The wide availability of IPv6 addresses will cause the network architecture to be simpler than proposed in chapter 5. There will be no need to use a Network Address Translator (NAT) to save IP addresses enabling the SIP user to be reachable directly. However, some operators of cellular networks still need to implement some protection from unwanted connections for their subscribers. A firewall could be used at this purpose. Thus, an operator could determine the policies to control both connections and traffic flows. Furthermore, recent firewall implementations use packet filtering and stateful inspection and are capable of handling SIP protocol messages.

Chapter 8

Conclusions and future work

This chapter gives conclusions that I have drawn at the end of this thesis. Furthermore it provides some guidelines for what can be done to further complete and enhance what has been described throughout this report.

8.1 Conclusions

In this thesis we have analyzed the state of the current service discovery technologies and how the Session Initiation Protocol (SIP) works with different wireless networks, and we have designed the Service Peer Discovery Protocol (SPDP) which follows the SIP event framework specification.

The first three chapters of this report provided a technical overview of the different technologies used and considered during the design of the service discovery protocol; the primary emphasis has been on SIP. These chapters also described the existing service architectures, specifically Jini and UPnP, in order to provide sufficient background concerning these protocols so that the reader can understand how we have addressed the limitations of these protocols.

The fourth chapter presented an overview of the issues of having a SIP overlay network upon a wireless infrastructure, and this chapter described a network scenario that enables mobile end-points to communication despite the existence of NAT or firewall which limit the inter-network communication.

From the fifth chapter forward, the thesis presented the proposed Service Peer Discovery Protocol (SPDP) used to determine the location of services in reconfigurable networks, both local and wide area networks. The protocol removes the limitations of UPnP regarding scalability and further enables the discovery of services by considering context information and not requiring a priori "knowledge" of other entities. The SPDP protocol can take advantage of the presence of a Context Server which may support users in the discovery of peers in wide area networks. An analysis of the SPDP protocol was presented in chapter 7; during this analysis, the main emphasis has been on cellular networks since these networks generally have more limited bandwidth than WLAN and the SPDP protocol should avoid blocking voice traffic.

In my thesis work I have designed a protocol which enables entities to communicate and to discover services and resources. This SPDP protocol provides a mean to assist users in service discovery, this can be particularly effective when used for discovery "connectivity services". Knowledge of potentially better connectivity can facilitate mobile users pro-actively changing their point of attachment and pre-negotiating of parameters (including Authentication and Authorization) can decrease the delay in the user receiving service via this new connection.

The SPDP protocol takes advantage of resources located in both local and wide area networks and scales well across these networks. Furthermore, the proposed protocol is built upon a SIP overlay network, thus services need not be coupled with a specific (access) network operator.

8.2 Future Works

The following sections introduce some areas of interest for continuing the work of this thesis, both for improvements in the proposed Service Peer Discovery Protocol (SPDP) and introduction of new network services that are possible to discover via the protocol.

8.2.1 Security

The SPDP protocol has already some security features defined in the SIP standard [20], which enables end-to-end secure communication between entities. However, a stronger authentication mechanism can establish trust between parties, for instance having a Certificate Authority (CA) that issues certificates (following the X.509 directory framework). This enforces and enhances SIP security and enables the authentication of the hosts involved. A CA may be needed, since each mobile device provides personal information and the device may wish to allow others to access to this information. Thus, it is important to determine what information the user shares with whom and when they wish to share it.

Additionally, a certificate framework might limit attacks against the system from inside. It is important to test the security of the SPDP protocol and to implement an appropriate mechanism for protection against malicious intent that could potentially corrupt the internal “knowledge” of the user causing the user to use the wrong strategy while attempting to discover services.

8.2.2 Intelligent Agent

An intelligent agent is needed to analyze the internal “knowledge” and to determine the right action according to context information. Intelligence in the device and in the network might save resources and might determine the best use of the available bandwidth and power battery of the device. Such an intelligent agent could improve the performance of the SPDP protocol during the discovery phase by making decisions as described in section 7.4.

8.2.3 Improvements to SPDP

Enhancements can be made to the design of the SPDP protocol in order to support relaying discovery packets, thus helping other agents to discover services and resources. Relaying these messages enables the protocol to work in ad hoc networks as well, since devices would be able to route SPDP packets, thus services could be discovered by devices who are out of range of direct communication (i.e., that are one or more hops away). Additionally, users of different networks could communicate via an intermediate device with multiple interfaces, but forming an local ad hoc network using one interface while connected to a another network using another interface, this intermediate device can then relay SPDP packets between the two networks.

8.2.4 Testing the SPDP protocol using different scenarios

The implementation of the SPDP protocol has not (yet) been tested, thus no results are available. Therefore, creating a usage scenario is necessary to evaluate the SPDP protocol and to determine how the protocol behaves under different workloads of the network and end-points. For instance, a quantitative measurement of the average discovery time, how the protocol accepts delays in packet delivery, and how the SPDP agent contributes to the latency of the SPDP packets. Furthermore, the test could show the optimal configuration of the resources and the network components, and the necessary dimensioning of these entities, the optimal lease time of a service, and the maximum number of users that can access the same service simultaneously.

Bibliography

[1] *Napster protocol specification*. http://opennap.sourceforge.net/napster.txt, March 2001.

[2] http://www.3gpp.org/, Accessed April 2003.

[3] *Gnutella Protocol Specification v0.4*. http://www9.limewire.com/developer/gnutella_protocol_v0.4.pdf, Accessed February 2003.

[4] http://java.sun.com/, Accessed February 2003.

[5] http://java.sun.com/products/jdk/rmi/, Accessed February 2003.

[6] http://java.sun.com/xml/jaxb, Accessed February 2003.

[7] http://www.wapforum.org/, Accessed May 2003.

[8] P. Sutton, R. Arkins, and B. Segall. Supporting Disconnectedness - Transparent Information Delivery for Mobile and Invisible Computing. In *Proceedings of IEEE International Symposium on Cluster Computing and the Grid CCGrid'01*, pages 277–285, Brisbane, Australia, May 2001. IEEE CS Press, Los Alamitos, Calif.

[9] J. Atkins. Copyright Infringement on the World Wide Web, With the Napster debate as a case study. http://www.eecs.umich.edu/˜aprakash/585/html/copyright.pdf, Accessed February 2003.

[10] M. Moh, G. Berquin, and Y. Chen. Mobile IP Telephony: Mobility Support of SIP. In *8th International Conference on Computer Communications and Networks*, pages 554–561, Boston, Massachusetts, 11-13 October 1999.

[11] G. Camarillo. Compressing the Session Initiation Protocol SIP. RFC 3486, IETF, February 2003.

[12] C. Campo. Service Discovery in Pervasive Multi-Agent Systems. In *Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices*, Bologna, Italy, 16 July 2002.

[13] Microsoft Corporation. *Universal Plug and Play Device Architecture*. http://www.upnp.org, June 2000.

[14] A. Misra, S. Das, and A. McAuley. Hierarchical Mobility Management for VoIP Traffic. In *Proceedings of IEEE MILCOM*, FairFax, USA, October 2001.

[15] A. Friday, N. Davies, and E. Catterall. Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments. In *Proceedings of the Second ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 7–13, Santa Barbara, California, USA, 20 May 2001. ACM.

[16] J. Ervenius and F. Tysk. Dual-mode Capacity in a WLAN-equipped PC for Roaming and Mobility between WLANs and GPRS Networks. Master's thesis, Royal Institute of Technology KTH, Department of Microelectronics and Information Technology, February 2001.

[17] A. Gulbrandsen, et al. A DNS RR for specifying the location of services (DNS SRV). RFC 2782, IETF, February 2000.

[18] D. Box, et al. Simple Object Access Protocol SOAP 1.1. Draft, W3C, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, May 2000.

[19] H.Schulzrinne, A. Dutta, O. Altintas, et al. Multimedia SIP sessions in a Mobile Heterogeneous Access Environment. In *Proceedings of International Conference on Third Generation Wireless and Beyond - 3Gwireless'2002*, San Francisco, USA, 28-31 May 2002.

[20] J. Rosenberg, et al. SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.

[21] J. Rosenberg, et al. A Presence Event Package for the Session Initiation Protocol SIP. Draft draft-ietf-simple-presence-10.txt, IETF, January 2003.

[22] J. Rosenberg, H. Shulzrinne, et al. Session Initiation Protocol SIP Extension for Instant Messaging. RFC 3428, IETF, December 2002.

[23] R. Schollmeier, I. Gruber, and M. Finkenzeller. Routing in Mobile Ad Hoc and Peer-to-Peer Networks. A Comparison. In *Networking 2002, International Workshop on Peer-to-Peer Computing*, Pisa, Italy, 19-24 May 2002.

[24] Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright. Simple Service Discovery Protocol/1.0. Draft draft-cai-ssdp-v1-03.txt, IETF, October 1999.

[25] R. Handorean and G. Roman. Service Provision in Ad Hoc Networks. In *Coordination Models and Languages*, pages 207–219. Springer, 8-11 April 2002.

[26] IEEE 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999 Edition.

[27] Sun Microsystems Inc. Jini Technology Architectural Overview. Technical White Paper, http://wwws.sun.com./software/jini/whitepapers/architecture.pdf, 1999.

[28] S. Aggarwal, J. Cohen and Y. Goland. General Event Notification Architecture Base: Client to Arbiter. Draft, http://www.upnp.org/download/draft-cohen-gena-client-01.txt, September 2000.

[29] A. Jarrar. Context Server Support for Opportunistic and Adaptive Mobile Communication. Master's thesis, Royal Institute of Technology KTH, Department of Microelectronics and Information Technology, To appear.

[30] P. Jarske. The GSM System, Principles of Digital Mobile Communication Systems. Technical report, Techical University Tampere, Finland, 2001.

[31] T. Kanter. *Adaptive Personal Mobile Communication, Service Architecture and Protocols*. Doctoral Dissertation, Royal Institute of Technology KTH, Department of Microelectronics and Information Technology, November 2001.

[32] T. Kanter. Going Wireless, Enabling an Adaptive and Extensible Environment. *Journal of Mobile Networks and Applications MONET*, vol. 8(1):37–50, 2003.

[33] P. J. Kühn. Location and Context Based Services. In *IFIP WG6.7 Workshop and EUNICE Summer School on Adaptable Networks and Teleservices*, pages 195–200, Trondheim, Norway, 2-4 September 2002.

[34] Byoung-Jo J Kim. A Network Service Providing Wireless Channel Information for Adaptive Mobile Applications: Part I: Proposal. In *IEEE International Conference on Communications (ICC)*, Helsinki, Finland, June 2001.

[35] G. Mola. Interaction of Vertical Handoffs with 802.11 wireless LANs: Handoff Policy. Master's thesis, Royal Institute of Technology KTH, Department of Microelectronics and Information Technology, To appear.

[36] ALPINE Network. Decentralized Resource Discovery in Large Peer Based Networks. http://cubicmetercrystal.com/alpine/discovery.html, accessed February 2003.

[37] T. Kanter, C. Olrog, and G. Maguire Jr. VoIP over Wireless for Mobile Multimedia Applications. In *Proceedings of the Personal Computing and Communication PCC*, pages 141–144, November 1999.

[38] T. Kanter, P. Lindtorp, C. Olrog, and G. Maguire Jr. Smart Delivery of Multimedia Content for Wireless Applications. In *Proceedings of the 2nd International Workshop on Mobile and Wireless Communication Networks MWCN2000*, pages 70–81, Paris, France, May 2000.

[39] A. Oram. Peer-to-Peer for Academia. www.openp2p.com/pub/a/p2p/2001/10/29/oram_speech.html, 29 October 2001.

[40] C. Perkins. IP Mobility Support for IPv4. RFC 3344, IETF, August 2002.

[41] Anne H. Ren. *A Smart Network and Terminal Framework for Supporting Context-aware Mobile Internet and Personal Communications*. Licentiate Thesis, Royal Institute of Technology KTH, Department of Microelectronics and Information Technology, October 2002.

[42] A. B. Roach. Session Initiation Protocol SIP-Specific Event Notification. RFC 3265, IETF, June 2002.

[43] J. Rosenberg and B. Campbell. A SIP Event Package for List Presence. Draft draft-ietf-simple-presencelist-package-00.txt, IETF, June 2002.

[44] J. Rosenberg and H. Shulzrinne. An Offer/Answer Model with the Session Description Protocol SDP. RFC 3264, IETF, June 2002.

[45] J. Rosenberg and H. Shulzrinne. Session Initiation Protocol SIP: Locating SIP Servers. RFC 3263, IETF, June 2002.

[46] A. Roychowdhury and S. Moyer. Instant Messaging and Presence for Network Appliances using SIP. In *Proceedings of Internet Telephony Workshop 2001*, 2-3 April 2001.

[47] R. Schollmeier. A Definition of Peer-to-Peer Networking towards a Delimitation Against Classical Client-Server Concepts. In *Proceedings of the 7th EUNICE Open European Summer School (EUNICE'01) and the IFIP Workshop on IP and ATM Traffic Management (WATM'01)*, Paris, France, 3-5 September 2001.

[48] H. Schulzrinne and E. Wedlund. Application-Layer Mobility using SIP. *Mobile Computing and Communications Review MC2R*, vol. 4(3):47–57, July 2000.

[49] G. Söderström. Virtual Networks in the Cellular Domain. Master's thesis, Royal Institute of Technology KTH, Department of Microelectronics and Information Technology, February 2003.

[50] H. Shulzrinne. Dynamic Host Configuration Protocol (DHCP-forIPv4) Option for Session Initiation Protocol SIP Servers. RFC 3361, IETF, August 2002.

[51] T. E. Sundsted. The Practice of peer-to-peer computing: Discovery. http://www-106.ibm.com/developerworks/java/library/j-p2pdisc/, November 2001.

[52] F. Thernelius. SIP, NAT and Firewalls. Master's thesis, Royal Institute of Technology KTH, Department of Teleinformatics, May 2000.

[53] R. Troll. Automatically Choosing an IP Address in an Ad-Hoc IPv4 Network. Draft draft-ietf-dhc-ipv4-autoconfig-05.txt, IETF, March 2000.

[54] S. Helal, N. Desai, V. Verma, and C. Lee. Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks. In *Proceedings of the Third IEEE Conference on Wireless Communication Networks WCNC*, New Orleans, March 2003.

[55] E. Wedlund and H. Schulzrinne. Mobility Support Using SIP. In *Proceeding of Second ACM/IEEE International Conference on Wireless and Mobile Multimedia WoWMoM99*, Seattle Washington, USA, August 1999.

[56] J. Rosenberg, J. Weinberger, and H. Schulzrinne. An Extension to the Session Initiation Protocol SIP for Symmetric Response Routing. Draft draft-ietf-sip-nat-02.txt, IETF, July 2002.

[57] A. Wennlund. Context-aware Wearable Device for Reconfigurable Application Network. Master's thesis, Royal Institute of Technology KTH, Department of Microelectronics and Information Technology, March 2003.

[58] Jiang Wu. *A Mobility Support Agent Architecture for Seamless IP Handover*. Licentiate Thesis, Royal Institute of Technology KTH, Department of Teleinformatics, June 2000.

Appendix A

Implementation of the Service Peer Discovery Protocol

A.1 Introduction

The Service Peer Discovery Protocol (SPDP) (defined in section 6.5) has been designed to extend the Session Initiation Protocol (SIP) [20] for supporting service discovery both in the local area and wide area. The implementation of the discovery protocol is based on an application that handles discovery messages. This application implements the functionality discussed earlier in this report, such as discovery of a service and/or of a peer.

Since the SPDP protocol is based on the event framework, defined in [42], the SPDP application is implemented according to this specification; hence SPDP messages are inserted in the body of a SUBSCRIBE or NOTIFY message defined for SPDP events. As introduced in chapter 6, SIP already provides functions to establish communication between entities and to handle the current sessions, thus, my implementation of the service discovery protocol does not need handle other SIP messages or the establishment of SIP session since these functions were already available in the SIP User Agent implementation used to build our application.

A.2 Available software

The implementation and the choice of the programming language was dictated by the available software, provided by Ericsson. Since a SIP User Agent was already available at the start of the project, it has been extended to provide the functionality of the proposed SPDP protocol.

The SIP UA was written in Java and implements the presence package and the instant messaging as defined in the SIP specification [20]. The available UA also provided the possibility to establish sessions (calls) with other entities (identified by SIP URIs) using the Session Description Protocol (SDP). Additionally, the UA implemented the presence package for presentity subscription; this implementation follows the event framework specification, thus the package has been developed based on a subscription and notification model. Finally, the available software enables a SIP user to send and to receive instant messages, using the MESSAGE method and the body of SIP messages of type “text/plain”.

When started the SIP UA initiates a graphical tool, providing a simple interface to the SIP UA and enables interaction by the user. Furthermore, the tool offers the possibility to debug the SIP messages, both those received and sent. Following the SIP specification, the SIP UA listens for SIP messages on port 5060; however, it is possible to change the setup of the SIP UA modifying the listening port. The use of a port other than 5060 enables multiple SIP UAs to run on the same host. When a message is received, it is parsed to check if the message is well structured and the header fields are inspected in order to determine the type of message, the sender of the message, and if it already belongs to an existing session. Each message received triggers an acknowledgment, and the code of the response message specifies if the message has been accepted and if the server could process it.

A benefit of using Ericsson’s SIP UA as the basis for my SPDP implementation was the availability of a subscription and notification framework that could be modified according to the specification of the SPDP protocol. Furthermore, the SIP UA provides client/server functionality and a means for handling call transactions that has been used to keep track of the current subscriptions.

A.3 Description of the implementation

The SPDP protocol has been implemented as an application written in Java [4]. This application is started by the SIP UA and is called upon by the SIP UA when the SIP UA receives SIP messages with “application/sxdp” as value of the “Content Type” header. The SPDP application is also invoked when the user wishes to discover a service, passing to the application the file describing the service.

The body of the SPDP message (formatted in XML) is sent to the application, the application parses the content using JAXB [6], a tool for mapping between XML documents and Java objects. Then, Java objects are produced from the header field and the body of the SPDP protocol.

The SPDP application uses the SIP UA model to keep track of the session established with other entities, in order to send responses and acknowledgments of the received messages. The SIP UA sends an acknowledgment for each incoming message (if the SIP header fields are well structured) but does not check the body. The SPDP message is checked while processing the message in the application that implements the SPDP protocol.

The functions implemented by the application depend on if a service discovery message is received or issued. In the former case, the application determines the availability of the service by matching the requested service with the services that this SIP user can provide. Then, a SPDP response message is formed and is encapsulated in a SIP NOTIFY message. When a user wishes to discover a service, a SPDP DISCOVERY request is created and is encapsulated in a SIP SUBSCRIBE request. Thus, a session (call) is created in order to keep track of the subscription.

The information about services and the known peers are stored locally in XML documents, that can only be accessed by the SPDP application. A singleton pattern is implemented in order to control access to the documents enabling only one thread to access the files.

A.3.1 Testing the implementation

No prototype has been built to test the SPDP protocol. This is due to difficulties to set up a test environment and to define a scenario to verify the behaviour of the protocol. The functionalities of the protocol and of the discovery time depend on the services available in the network and on the users that currently support the protocol extension to SIP.

The prototype for testing the application requires WLAN networks and at least one cellular network (i.e., GPRS). These networks should cover a wide area in order to check the behaviour of the protocol when working in networks with different throughput and load, and to valuate the strategies that are available to the user. Additionally, the network scenario shown in figure 5.3 requires modification of the operator's network since a SIP outbound proxy should be present inside the GPRS network or in proximity of a NAT, thus this entails a control of the test network. A prototype was not built due to the limited possibility to create a real scenario and the limited amount of time for setting up such a scenario.

However, the application was implemented to check if the client and the server were able to process the messages and if the operation within the client or the server are done correctly. This test consists of an exchange of discovery requests for services that are stored locally in the server. SPDP messages, both requests and responses, were handled correctly by the server and the client.