Facilitating the adoption and use of the IP Multimedia System

CHRISTOS PAPAZAFEIROPOULOS

KTH Information and
Communication Technology

Master of Science Thesis
Stockholm, Sweden 2009

COS/CCS 2009-02

Facilitating the adoption and use of the IP Multimedia System

Christos Papazafeiropoulos

<chrpap@kth.se>

2009.03.17

Examiner and academic advisor: prof. G. Q. Maguire Jr., KTH
Industrial advisor: Michael Åström, Ericsson

Abstract

The IP Multimedia Subsystem (IMS) is still under development and not widely adopted in the market. Some companies are reluctant to deploy IMS and some telecommunications vendors believe that IMS will not achieve a desirable market share. The purpose of this thesis work is to give a boost to this technology (i.e., to accelerate its market growth) by providing the community (both developers and operators who might adopt this technology) with an evaluation of the Ericsson Java Application Programming Interfaces (APIs) called Mobile Java Communication Framework (MJCF) APIs. Developers with or even without knowledge of the IMS architecture and signaling should be able to use these interfaces in order to develop applications on top of IMS.

A client-server application is designed and implemented to facilitate this evaluation and to serve as an example for others. The motivation behind this application is the every day needs of the people who search for discounts while they are shopping. Users set up their profile by specifying their preference concerning discounts for specific products; while shop owners publish discounts. When a user is near a store which offers interesting discounts (i.e., discounts that match their profile) new notifications will be sent to his/her mobile device. This application exploits the MJCF APIs and uses several of its basic functions; specifically subscriptions, messages, notifications, and publications are some of the messages that can be utilized through these interfaces.

Throughout the application development, bugs were found in the APIs and corrections were suggested for the documentation. Measurements were made in order to evaluate the memory utilization and delay associated with these APIs. It was observed that the delays added by the APIs are somewhat high and may negatively affect the experience of users. However memory utilization seamed to be low for client applications and quite high for the server side given the resources of today's services and cellular phones.

Sammanfattning

Systemet IP Multimedia Subsystem (IMS) är under utveckling och är inte vida etablerat än. Nogra företag tvekar inför etablering av IMS och nogra telekomföretag anser att IMS inte kommer uppnå önskad marknadsandel. Syftet med detta examensarbete är att ge denna teknologi en skjuts framåt (d.v.s. att öka marknadstillväxten) genom att tillhandahålla den gemenskap av både utvecklare och operatörer som kan tänka sig ta in denna teknologi, med en utvärdering of Ericsson’s Java Applications Programming Interfaces (APIs) kallade MJCF API. Utvecklare med eller t.o.m. utan kunskap om arkitekturen och signalleringen hos IMS ska kunna använda dessa gränssnitt till att utveckla tjänster på IMS.

En klient-server applikation är designade och implementerad för att möjliggöra denna utvärdering och för att agera exempel för andra. Motiveringen bakom denna applikation är det vardagliga behovet hos människor som söker efter rabatter/erbjudanden när de handlar. Användare sätter upp sin profil genom att specificera sina önskemål angående erbjudanden för specifika produkter medan butiksinnehavare publicerar sina erbjudanaden. När en användare är nära en butik som erbjuder någonting interessant (d.v.s. produkter som matchar använadarens profil), så kommer nya notifikationer att anlända till hans/hennes mobil. Denna applikation uttnyttjar MJCF APIet och använder ett flertal av dess basala funktioner; speciellt gällande prenumerationer, meddelanden, notifieringar och publiceringar är några av de meddelanden som möjliggörs genom dessa gränssnitt.

Genom applikationsutvecklingen så blev flera buggar i APIerna upptäckta och förbättringar till dokumentationen föreslogs. Mätningar gjordes för att utvärdera minnesåtgången och fördröjningar associerade med dessa APIer. Det observerades att API fördröjningar är något höga och kan påverka användarupplevelsen negativt. Däremot verkade minnesåtgången vara låg på klientsidan och hög på serversidan, givet de resurser dagens tjänster och mobila telefoner förfogar över.

Acknowledgement

First of all, I would like to express the most respectful gratitude to my beloved parents, who are very important source of spirit of my life, for the selfless love and invaluable virtue given to me.

I would also like to express my sincere gratitude to my thesis supervisor, Professor Gerald Maguire, KTH, for academically guiding me through the whole thesis process. Great thanks are given to my industrial thesis advisors Morgan Lindqvist, senior research engineer in Ericsson, and Michael Åström, research engineer in Ericsson, for the endless patience and precise technical instructions to my thesis project.

Thanks are also given to Peter Håkansson, senior research engineer in Ericsson, as well as to my colleagues Sike Huang and Xiawei Chen who where very helpful throughout my stay at Ericsson.

Contents

List of Figures

Acronyms and Abbreviations

3GPP	Third Generation Partnership Project
ADSL	Asymmetric Digital Subscriber Line
API	Application Programming Interface
CDMA	Code Division Multiple Access
CODEC	CO/DECoder
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications (originally “Groupe Spécial Mobile”)
HSS	Home Subscriber Server
IETF	Internet Engineering Task Force
IMS	IP Multimedia System
IP	Internet Protocol
ITU-T	International Telecommunication Union – Telecommunications
MGCF	Media Gateway Control Function
MGW	Media Gateway
MJCF	Mobile Java Communication Framework
MRF	Media Resource Function
MSC	Mobile Switching Center
PDA	Personal Digital Assistant
PDF	Policy Decision Function
PoC	Push to talk over Cellular
PSTN	Public Switched Telecommunications Network
QoS	Quality of Service
RTCP	RTP Control Protocol
RTP	Round Trip Protocol
SDP	Session description Protocol
SLF	Subscriber Locator Function
SMS	Short Messaging System
TDMA	Time Division Multiple Access
UMTS	Universal Mobile Telecommunication System
XML	Extended Markup Language
WAP	Wireless Application Protocol
W-CDMA	Wideband Code Division Multiple Access
WLAN	Wireless Local Area Network
WWW	World Wide Web

1 Introduction

The internet, a packet-switched network, has rapidly invaded the world communication market. Applications of mass interest, such as the World Wide Web (WWW) and e-mail attracted large numbers of users and have now become essential to almost everyone. In the early 1990s, the GSM standard for mobile communications over *digital* wide area cellular systems was introduced and was extremely successful. GSM is a so called second generation (2G) mobile communication system. The Short Messaging Service (SMS) proved to be the killer-application for this technology, despite the functionality only having been added to the GSM standard to provide *limited* text messaging. Later GPRS was added to provide a packet switched communications system for GSM. Since the introduction of GSM in 1994, the evolution of the wide area cellular telecommunications systems has been both tremendous and rapid. These mobile telecommunication technologies spread around the world and today provide coverage almost at any place on the land surface of the planet where there are significant numbers of people[1]. Different wireless and wired technologies have been introduced – competing for the same customers (users). The need for standardization was apparent, but there were conflicting goals and interests; thus rather than having a single global standard for wide area cellular communications systems, there are several families of standards. Each standard is designed to create a market and to differentiate itself from other standards. The existence of a small number of standards provides users with a better user experience, but can elicit some confusion – especially when users do not know what standard is used where and what standards they should look for, when they buy products and services. Two of the major competing 2G wide area cellular standards were GSM and Qualcomm's Code Division Multiple Access (CDMA).

Less than a decade ago, the 3rd Generation Partnership Project (3GPP) organization was formed in order to continue the development of the GSM platform into a so called third generation (3G) mobile communication system[2]. The major difference between 2G and 3G systems was the introduction of packed switched data communication as a *fundamental* feature of the system, as opposed to being purely circuit-switched communication. Additionally, much higher data rates were introduced for both voice and non-voice data. In the evolution from the initial 2G system there have been numerous steps and new technologies introduced, which have incrementally transformed the older 2G systems into the early 3G systems. This process is ongoing even today, with the latest developments dubbed "Beyond 3G" or alternatively called "forth generation" (4G) [1] .

A major objective of the 3G mobile systems is to bring together three of the most successful human inventions: telephony, Internet, and digital media. A key element in this endeavor is thought to be the IP Multimedia Subsystem (IMS). IMS merges these three technologies and attempts to combine the advantages of the internet with those of more traditional telecommunication networks. IMS defines a framework that is supposed to facilitate the development and deployment of any type of multimedia service. The IMS framework includes or addresses: end-to-end service (that is, it takes into account all the nodes that are needed to realize a service), reachability, mobility, interoperability, convergence, quality of service (QoS), multimedia connections, security, and charging. IMS is seen as an evolutionary step from the separate Internet, Media, and Telcommunication markets to an integrated single market (see Figure 0-1).

One of the major goals of the traditional telecommunication vendors has been to increase the income of the telecommunications industry. To do so, a key element was to enable the traditional telecommunications vendors to *retain control of the network* and to *introduce themselves into all the services* which utilize this network. The telecommunications vendors hoped to maintain or even grow their revenues by convincing the telecommunications operators to invest in new infrastructure. However, there is a problem as without appropriate widely used services there is little reason for a telecommunications network operator to invest in new infrastructure. This problem was made even more complex as the telecommunication vendors (and governments) had just persuaded many (both old and new) telecommunications operators to invest in 3G networks, and many of these operators found that the investments required were very substantial but it was hard to attract customers, leading to an unpleasant end result of many bankrupts and a major world-wide telecommunications market crash. It is against this rather challenging backdrop that telecommunication vendors have tried to introduce IMS.

---

[1] Note there are also maritime and aviation cellular systems, e.g. GSM systems on ships and planes.

[2] A similar organization 3GPP2 was formed to continue the development of the earlier CMDA standard.

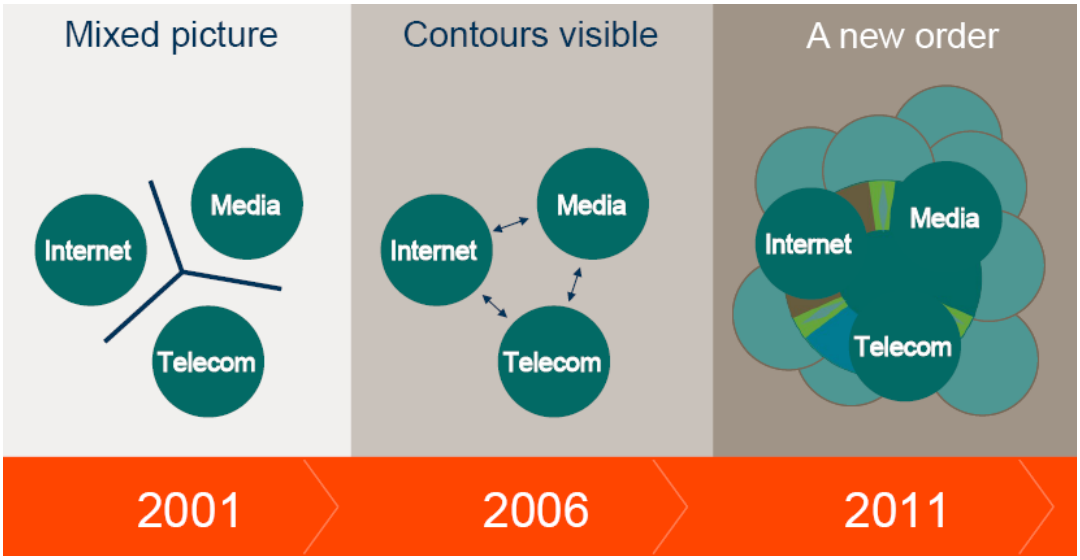

Figure 0-1: Possible evolution of the telecom world (Adapted from [2])

Even more embarrassing for many has been the fact that while the cellular networks have evolved to support higher speed packet data services - the only services besides voice, SMS, and ring tones which have been successful, have been Internet based services. Attempts to create a distinct "Mobile Internet" (using WAP technology) failed. Additionally the mobile operating system wars (between Symbian, BREW, Windows Mobile, Linux, …), the crippled development environments, the poorly documented or hidden interfaces to functions, the incompatibilities between different releases on the same platform and between different platforms have made developing applications, which successfully run on the mobile handset, both difficult and expensive. Even attempts to use Java on handset sets has been of only marginal success, since the underlying platforms are frequently broken or work differently in different releases and on different hardware and OS platforms. Thus users have massively used their cellular handset as a web browser for traditional (or slightly adapted) web content, as a media player for locally stored or streamed content, and for a small handful of applications.

This thesis will describe trends in IMS services, explain some of these services in greater detail, and provide an evaluation of the latest frameworks for developing third party applications. This thesis is intended to pave the way to ease the successful development of IMS applications. Otherwise the failure of IMS is rather certain and with it the failure of many companies.

This thesis is organized as follows:

- **Chapter 2** introduces the reader to the IMS protocols and architecture. It gives an overview of the existing standardized services and the tools for creating 3rd party IMS applications.
- **Chapter 3** describes an IMS Application developed as an example for this thesis.
- In **Chapter 4** the evaluation procedure and the evaluation environment are presented.
- **Chapter 5** presents an evaluation of the client API.
- **Chapter 6** evaluates the server API.
- **Chapter 7** describes the overall approach used by the evaluation procedure and the results.
- **Chapter 8** offers some conclusions and suggests some future work.

2 Background

In this chapter the IMS framework is introduced and described. Initially a brief history of the evolution of wide area cellular telecommunications will be given in order to introduce the IMS architecture and to put it into a historic context. The basic elements, functionality, as well as the protocols of the IMS system will be covered. The IMS architecture was designed to enable a wide variety of services to be provided on top of it. These services can be divided into standardized and non-standardized services. The standardized services will be enumerated and described. Finally, the chapter will conclude with a list of the APIs provided to developers in order to enable them to use these standardized services and to develop new 3rd party applications.

2.1 Brief introduction to Mobile Communications

The evolution of wide area cellular networks from the first and second generation circuit-switched systems to the third generation was due to the increase cost/performance of digital electronics and the addition of a packet-switched domain to the existing circuit-switched domain. IMS is thought by some to be the next step in this evolution, as it builds upon the existence of a basic third generation packet-switched system. In order to understand and familiarize ourselves with IMS, we begin with a brief history of the 3G circuit-switched and packet-switched domains.

2.1.1 From Circuit-switched to Packet-switched

GSM [5] started as a digital circuit-switched network. The architecture was based on that of traditional Packet Switched Telephone (PSTN) [6] Networks but with the addition of mobility for circuit-switched calls. The GSM and PSTN networks are divided in two different planes: the signaling/control plane and the media/user plane.

The signaling plane transports the protocols used to establish a path (the “circuit”) between terminals; as well as the interface for service invocation. This signaling includes the signaling necessary for supporting terminal mobility.

The media plane is used to transfer the user’s data over the path that has established by the signaling plane. The encoding and decoding of the audio (voice) is performed by the use of CODECs. An important aspect of traditional telephony systems was the use of a fixed data rate (typically 56 or 64 kbps). GSM introduced a new CODEC (usually called the GSM CODEC, but formally known as a “GSM 06.10 a Regular Pulse Excitation Long-Term Prediction (RPE-LTP) CODEC”. Subsequently new CODECs have been introduced to enable both greater compression or (for 3G systems) higher quality.

Traditionally the signaling and the media plane followed the same path in the GSM network. 3GPP (release 4 [7]) has split the planes to allow them to follow different paths through the network (similar to the splitting of these planes in the digital PSTN). Initially the Mobile Switching Center (MSC) is the main server for the signaling plane and the MSC media gateway also provides the switching for the media plane. The mobile handsets (terminals) handle both signaling and the media.

3GPP release 99 [8] introduced a GSM packet-switched domain, known as General Packet Radio Service (GPRS) [9]. Using this domain’s mobile terminals can access the Internet and other packet based networks using native packet-switched technologies. New nodes were added to the GSM architecture provide the additional functions needed for GPRS.

UMTS [11] which follows the 3GPP release 5 (2003) [10] specifications was the next step in the evolution of GSM systems. UMTS introduces a new packet-switched technology based on another wireless access technique (W-CDMA) as opposed to the TDMA scheme used for GPRS/GSM.

3GPP release 6 (2005) [10] includes wireless local area network (WLAN) [12] technology along with the existing network to create a new architecture – which supports heterogeneous wireless access networks. 3GPP release 7 [10] adds broadband networks to the set of potential access networks. Today, the 3GPP architecture attempts to include all types of widely used wired and wireless networks. The main obstacle to this evolution was the need for a common standardized way to create applications that can exploit the characteristics of the underlying network while supporting roaming. Roaming is very important to the network operators since it generates a large amount of their revenues and is increasing expected by their subscribers.

This standardized approach is based upon the introduction of the IMS system. The next section gives an overview of this system.

2.2 IP Multimedia Subsystem (IMS)

IMS is a set of several different technologies and exploits many layers of internetworking. IMS utilizes many different protocols to realize its functionality. We present the relevant protocols so that the reader can understand what functions IMS makes available to the application developer.

2.2.1 The main protocols used in IMS

IMS is primarily based on IP protocols, with some legacy protocols used for compatibility with the PSTN and other traditional telecommunications networks. The IP family of protocols has been developed by the Internet Engineering Task Force (IETF). Since IMS utilizes several different access technologies, such as GSM, UMTS, etc. many additional protocols are also necessary. Rather than inventing new protocols, 3GPP decided to reuse protocols that had already been developed in other organizations, such as the IETF and the International Telecommunication Union – Telecommunications (ITU-T). Figure 0-2 shows the main protocols used between the basic elements of IMS. Each of the elements of this figure will be described in conjunction with the relevant protocol(s) on subsequent sections.

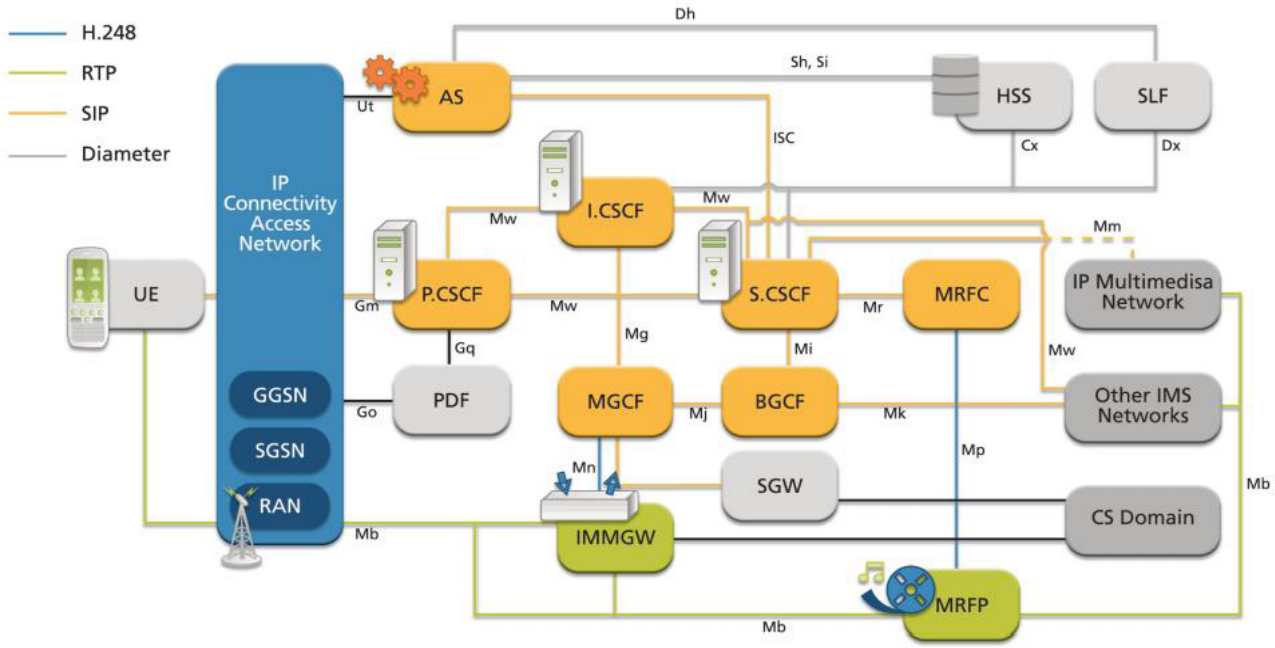

Figure 0-2: IMS protocols (Adapted from [13].)

2.2.1.1 Session Initialization Protocol (SIP)

The main signaling protocol used in IMS is SIP [3]. This protocol was specified by IETF to establish and manage multimedia sessions over IP networks. The design of the protocol is based on HTTP [4]. Just as HTTP, SIP is a request/response protocol and it is text-based which means that it is easy for developers to understand and debug, as well as to quickly create and extend new services. SIP can be transported over UDP, TCP, SCTP, and TLS. Its primary purpose is to establish, modify, and terminate (multimedia) sessions. SIP is independent of the type of multimedia session and independent of the mechanism used to describe the session.

3GPP chose SIP as the basic protocol for IMS mostly because:

- Existing signaling protocols failed to comply with the specified IMS requirements. Some of the existing protocols were not internet protocols and others were difficult to use and extend.
- Internet services should be easily implemented and deployed by using the simple application programming interfaces (APIs) such as those that SIP provides.

Some of the main characteristics of SIP are:

Extensibility This means that the protocol is able to accommodate more features by easily adding additional headers.

Flexibility New extensions can provide new exciting and innovative services; while giving developers freedom to introduce new functionality and design concepts.

Security Since the information that IMS will carry can reveal private details about the subscribers; it is important to secure the information by using authorization and authentication. SIP can be transferred over secure protocols, such as IPSec.

Some of the main extensions of the SIP protocol will be described below.

- **SigComp** [14] is a way to compress SIP messages. This provides a solution to the problem of transferring a large amount of information over narrow-band connections i.e., bandwidth constrained and delay prone. Studies have shown that the battery consumption is also decreased if this extension is used.
- **P(rivate)-headers** [15][16] are headers that can be used in addition to the standard headers to provide information about the access network, the visited network, the caller's identity, etc.
- **Security Agreement** [17] is an extension that is used to negotiate the security capabilities of the different types of the endpoints (mobile phones, wire-line phones, soft phones, etc.). The preferred protocol to provide security for IMS is IPSec. An IPsec security association is established between the endpoint and the Proxy – Call Session control Function (P-CSCF).
- **AKA-MD5** [18] enables the terminals and the network to be mutually authenticated and to specify the cryptographic keys to be used for the IPSec association.
- **Media Authorization** [19] ensures that only authorized media resources are used.
- **Reg-event Package** [20] is used by the terminal and the P-CSCF to know the terminal's registration status on the network.
- **Preconditions** [21] specifies the IMS method for negotiating QoS and other required call behavior between two terminals.
- **IMS Resource Reservation** [22] defines how to make resource reservations for phone calls or other sessions.
- **The Session Description Protocol (SDP)** [23] defines the basic negotiation process for media streams, including the bit rate and CODEC to be used, as well as other media attributes.

2.2.1.2 Media Gateway Control (MEGACO) or H.248

Megaco [25] is the control protocol used between a Media Gateway and a Media Gateway Controller. It is normally used to provide VoIP functionality between PSTN&IP networks or IP networks, but it has been extended to deal with special commands for video processing.

2.2.1.3 Diameter

Diameter [26][27] is the successor to RADIUS authentication, authorization, and accounting (AAA) protocol. It is used for communication between the HSS and the S-CSCF or the HSS and the AS. The information, transferred over this protocol, concerns online/offline charging, locating the relevant S-CSCF and locating a subscriber, security key manipulation, and service filter criteria delivery.

2.2.1.4 Real Transport Protocol (RTP) / Realtime Control Protocol (RTCP)

RTP [28] and RTCP [28] are used for delivering audio and video over the internet. They can also be used with the RTSP [29] protocol which provides personal video recording controls for multimedia applications. RTP can carry any data with real-time characteristics, such as interactive audio and video. RTP does not provide mechanisms to ensure timely delivery nor does it give any Quality of Service (QoS) guarantees. RTCP provides information about reception quality which the application can use to make adjustments to its operation.

2.2.1.4.1 Extensible Markup Language (XML)

XML is a language that allows users to define their own elements. This way the data transferred over the network is structured and easy to manipulate. XML is the main protocol used in combination with SDP, since SDP describes the session attributes and characteristics.

SIP, RTP, and RTCP are the main protocols of the IMS architecture. This architecture will be presented in the following sections.

2.2.2 IMS Architecture and Components

IMS separates the various network elements by means of a layered design. There are separate layers for call control (signaling plane), service control (service plane), and media control (user plane), as well as dedicated servers for security and Operations and Maintenance (O&M).

As illustrated in Figure 0-3, the IMS network's modular architecture uses logically discrete network entities for distinct network functions. For example, every access technology (such as WiFi, UMTS, etc.,) has its own network entity that connects it to the rest of the IMS architecture. These entities are specific to the physical and link layers but internetwork with the core network. Each entity has well-defined interface points and well-defined functionality.

Figure 0-3 also shows several distinct layers, or planes comprising the network:

Access Network	Provides the access technology for communication with the terminal. Access to the users can be provided via any type of access technology, such as cellular radio wireless LAN, fixed LAN, ADSL cable connection, and so on. Since IMS is an access agnostic network, the access type should not affect the services provided.
User Plane (Media/Transport Plane)	Provides support for all the media streams in the network, including voice, video, packet, and data streams.
Control Plane	Comprises the session control layer, call setup, call termination and call routing that determine the termination destination for each call and all services.
Application Plane	Contains all the servers that provide advanced IMS logic and services.

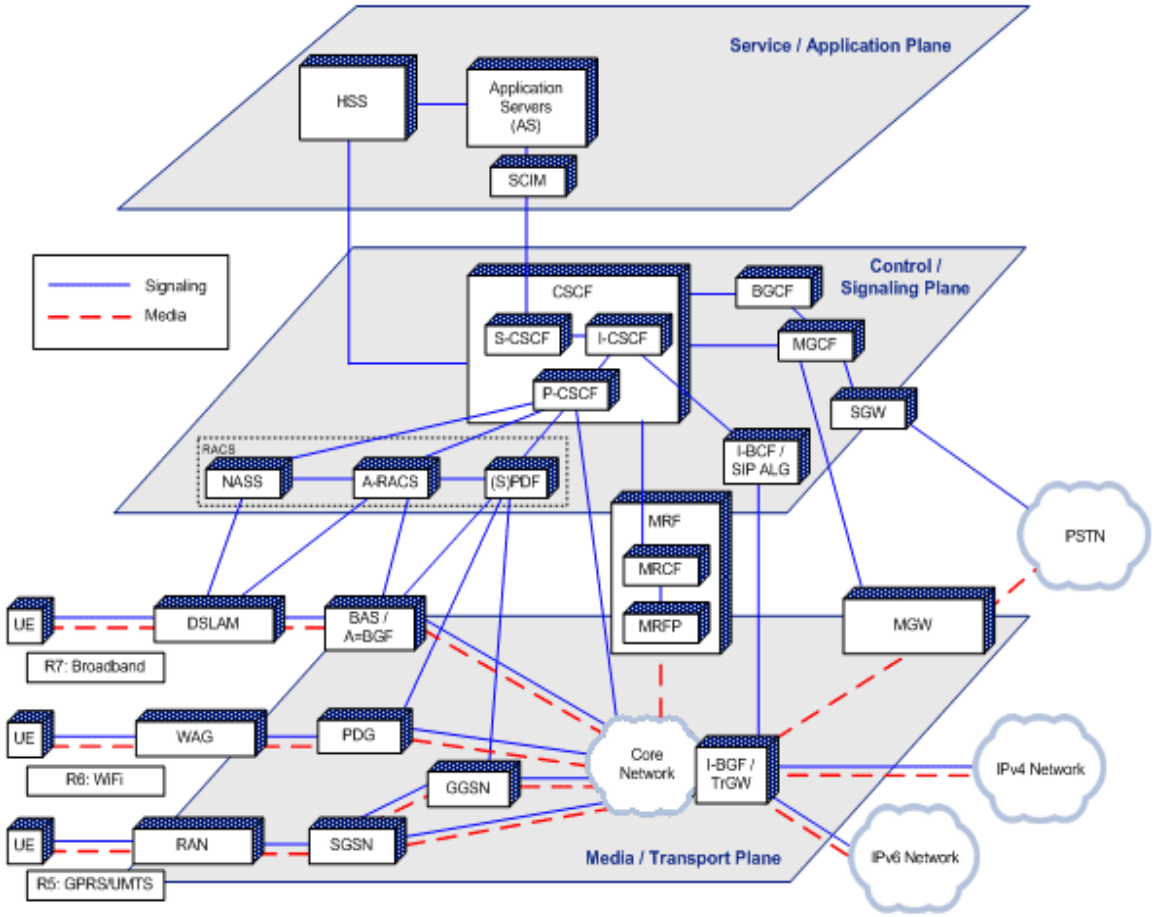

Figure 0-3: The IMS Functional Architecture

Unlike the Internet, IMS networks contain additional embedded control functions in order to provide resource control, security, and QoS in the network. These mechanisms allow the network to adjust its allocation of resources needed for each specific application.

The IMS network consists of a variety of components which are described below.

2.2.2.1 User Equipment (UE)

User equipment denotes a mobile terminal, cellular phone, or PDA with a network access connection. A UE may also be a traditional fixed phone or soft phone. The user equipment normally has one or more interfaces to one or more access networks. It also has the appropriate operating software, installed by the vendors, for communicating with the core network.

2.2.2.2 Home Subscriber Server (HSS) and Subscription Location Function (SLF)

HSS is the main database of the IMS system. It stores information about all the network subscribers. This database contains:

- Subscriber phone numbers, URIs, service profiles, and details about the subscription, such as pre-paid or post-paid (i.e., based upon a contract).
- Security information used to authenticate the subscriber
- Filter criteria that are used to trigger the S-CSCF (see section 2.2.2.5) to route SIP messages to Application Servers (see section 2.2.2.8) or to another destination.

If an IMS network contains more that one HSS, then a SLF is needed. The SLF is a database that maps the subscribers' addresses to HSSs. Thus, when a node needs to contact a HSS, it first queries the SLF to determine which HSS to contact.

2.2.2.3 Proxy-Call Session Control Function (P-CSCF)

A P-CSCF is a SIP proxy server that is the first SIP contact point from and to the UE. All SIP requests initiated from the IMS UE or destined for an IMS UE traverse the P-CSCF. Some of the attributes of the P-CSCF are:

- A P-CSCF is allocated to the UE during IMS registration and does not change for the duration of the registration.
- Communication between the UE and the P-CSCF is encrypted and secured using IPsec.
- The P-CSCF verifies the correctness of all SIP messages sent by the UE.
- The P-CSCF performs compression/decompression of the SIP messages to/from the UE. This is used to support narrowband channels.
- The P-CSCF generates charging information and forwards it to a charging node.
- A P-CSCF may include a Policy Decision Function (PDF) that authorizes use of media plane resources and manages the Quality of Service provided over the media plane.

2.2.2.4 Interrogating-Call Session Control Function (I-CSCF)

The I-CSCF is a SIP proxy server that is the contact point within an operator's network. It is responsible for establishing communication between two different IMS networks, such as a visited network and the subscriber home network. When a SIP server needs to find the next SIP hop for a particular message the SIP server obtains the address of an I-CSCF in the destination domain. The I-CSCF is responsible for the following:

- Providing security associations with other networks using IPsec tunneling.
- Handling charging and management of IMS network resource utilization.
- Providing Topology Hiding Inter-network Gateway (THIG) functionality for hiding the configuration, capacity, and topology of the network from being viewed outside a given IMS network.

2.2.2.5 Serving-Call Session Control Function (S-CSCF)

A S-SCSF is a SIP proxy server that performs session and service control. It maintains a session state and keeps a binding between the subscriber's location and the subscriber's SIP address of record by acting as a SIP registrar. This component is the heart of IMS and is responsible for setting up sessions and making service routing decisions. Thus, when a call is made from a UE, the S-CSCF determines whether to route the call to a standard call destination or to an Application Server (AS).

The S-CSCF communicates with the HSS in order to authenticate the subscriber and download the subsrciber's profile. Based on this profile the server can decide whether to route the SIP message to one or more ASs or to another destination.

The S-CSCF is the responsible node for translating a telephone number in order to route a session to the correct destination. It is also enforces the policies of the network operator. These policies are enforced by means of triggers that control the IMS opperations.

2.2.2.6 Media Resource Function (MRF)

The MRF is responsible for the media crossing the network. It provides the home network with the ability to play announcements, transform media between different codecs, collect statistics, and do any sort of media analysis the network operator wishes.

The MRF is divided into a signaling plane element called the Media Resource Function Controller (MRFC) and a user plane component called the Media Resource Function Processor (MRFP). The MRFC is responsible for:

- Controlling the media stream resources of the MRFP
- Interpreting information coming from an AS or S-CSCF and controlling the MRFP accordingly

- Generating a Call Detailed Record (CDR) and forwarding it to the charging unit.

The MRFP implements all the media-related functions, such as playing and mixing media.

2.2.2.7 Breaking Gateway Control Function (BGCF)

The BGCF is used to connect a call to a regular phone number. It determines if the call is directed to a circuit-switched network, such as the PSTN, and if so routes the call to the appropriate gateway.

If the destination network is the PSTN, then there are three PSTN gateways that the BGCF can contact. Those are:

Signaling Gateway (SGN)	This component interfaces with the signaling plane of the circuit-switched network. Its main function is to convert the different types of low layer protocols.
Media Gateway Control Function (MGCF)	The MGCF controls the media gateway by giving commands that determine what to convert and when. As such, it can take one stream from a Voice over IP (VoIP) network and convert it to a circuit switched call in a TDM network. The MGCF is also responsible for: ▪ Communicating with the S-CSCF ▪ Signaling to the S-CSCF, depending on the routing number for incoming calls from legacy networks ▪ Performing protocol conversion between ISDN User Part (ISUP) and IMS call control protocols.
Media Gateway (MGW)	The GMW is the actual gateway between the IMS network and a circuit-switched network. It provides an interface for transforming RTP to Pulse Code Modulation (PCM) time slots for transmission via a traditional circuit switched network. It also performs transcoding to/from the appropriate CODEC.

2.2.2.8 Application Server (AS)

The AS is a SIP server that hosts and executes services. The AS can play the role of a proxy, a SIP User Agent, or a Back-to-Back User Agent (B2BUA). There are three different types of Application Servers:

SIP AS	This is the server that is intended to be the most used server and will host 3rd party IMS services based on the SIP protocol. Developers have appropriate interfaces to implement and deploy applications.
Open Service Access-Service Capability Server (OSA-SCS)	This server is an intermediate between the OSA Application Server and the S-CSCF. It provides all the OSA functionalities in a secure way.
IP Multimedia Service Switching Function (IM-SSF)	This server is an access point for the CAMEL services that were developed for GSM.

In the next section of this chapter, we will introduce the IMS services that an IMS network supports. These Services are normally hosted and executed by the AS.

2.3 IMS Services

It should be clear from the previous sections that IMS promises a compact and powerful technology that may provide the infrastructure for the future networks. Although, the main goal of IMS is not the infrastructure but the services it provides. For this reason different organizations were formed to describe and standardize some widely used services and to create open interfaces given to independent developers for creating services over IMS. The process of standardizing new services is still ongoing and it will continue for more time, since hopefully new services of interest will be invented in the future.

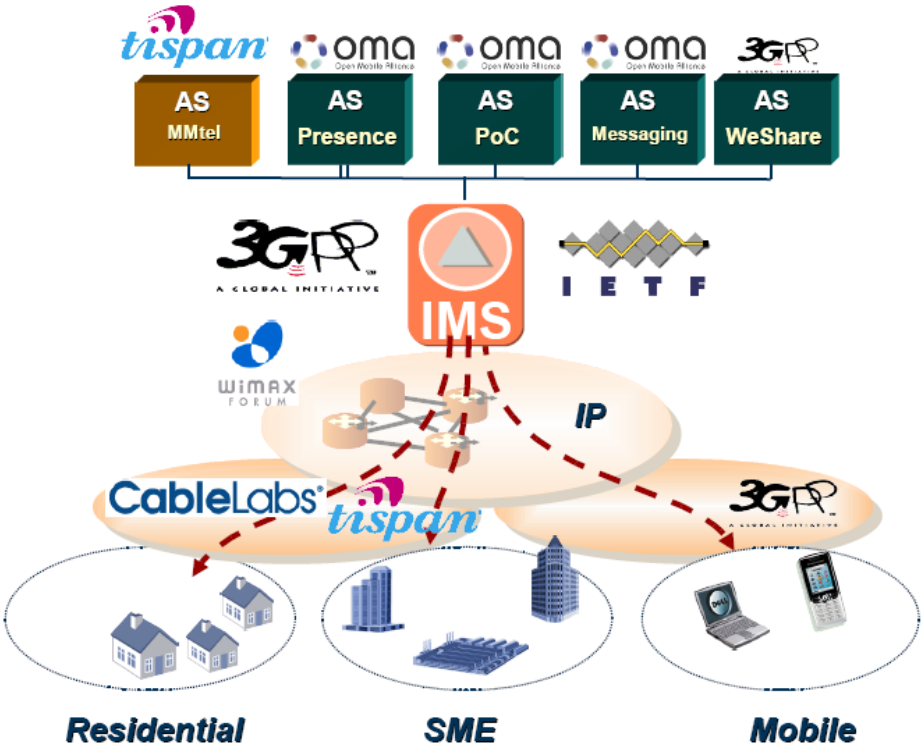

Figure 0-4: Service standardization process (Adapted from [24]).

Figure 0-4 presents the current standardization state for the services that IMS provides. These organizations are developing services that require extra standardization and will not be mentioned further in this study.

The following sections list the most important standardized services provided by IMS, along with a short description.

2.3.1 Standardized Services

2.3.1.1 Presence

Presence [30], [31] and [32] is the service that allows users to publish their status by means of reachability, availability, and willingness of communication to other users. For example, a user can be accessible from three different devices, a laptop at home, a laptop at work, and a handset. When the user is at work, the device at home cannot be used and no messages should be directed to it. Even if the user is logged on from work, he/she may be busy with something (attending a meeting or engaged in another session). In this case the service should not allow call establishment, but should inform the other party about the situation. Additionally, the presence service allows users to give details of their communication means and capabilities e.g., audio and video CODECS supported, location, etc.

A subscriber who publishes his or her presence information is referred as a *presentity;* while the subscriber that access the presence information of the presentity are called *watchers*. The watchers normally subscribe to a presence service and are notified when a change of the watched presence information occurs.

This procedure is normally done by the use of the SIP notification mechanism. The watcher sends a SUBSCRIBE message to the AS to subscribe to the service. Each time an event (presence information change) happens a NOTIFY message is send back to the watcher with all the appropriate information. The presentity sends a PUBLISH message to the service when something changes in the presentity's presence information.

2.3.1.2 IMS Messaging

The requirements for IMS messaging are specified in [33] and the technical realization described in [34]. There is support for two types of message services. The first one is immediate messaging in which the user can both send and receive messages by using the SIP MESSAGE method, whereas the session-based messaging is supported using SIP to first establish a session, then using MSRP to send the message.

2.3.1.3 Push to talk over Cellular (PoC)

The basic idea behind this service is based on the well known “walkie-talkie” application. When a user wants to initiate a talk session with an individual subscriber or with a group of participants, he/she needs to press a button, speak, and then release the button. An AS is needed to implement this service. The OMA-AD-PoC [35] defines the architecture of this service while the 3GPP IMS [35] defines the infrastructure of this architecture.

2.3.1.4 IMS Multimedia Telephony

3GPP has specified IMS Multimedia Telephony in [36] and [37]. The main idea is to allow different media types such as speech, video, etc, to be included in a session between two or more users.

TISPAN specified in [38] a group of supplementary services that can be associated to the IMS Multimedia Telephony. Those services are:

Originating Identification Presentation (OIP)	Originating Identification Restriction (OIR),
Terminating Identification Presentation (TIP)	Terminating Identification Restriction (TIR),
Communication Diversion (CDIV)	Communication Hold (HOLD),
Communication Barring (CB),	Message Waiting Indication (MWI), and
Conference (CONF),	Explicit Communication Transfer (ECT).

These services are hosted on an AS.

2.3.1.5 Global Text Telephony (GTT)

The GTT service is specified in [39]. The main objective of this service is to add the capability to include a real-time text conversation component in a session.

2.3.2 Non-standardized Services

The broad applicability of IMS is a key factor to enable diverse application development. Only our imagination should limit what can be implemented. Developers will have ideas for services that they will need to subsequently implement. They need to have an “open door” to the IMS system. This requires an open programming interface. These interfaces give the freedom to the developers to expand the existing services and to create new ones.

The next chapter shifts our attention from the architecture and design layer to development and implementation of services. We will describe and utilize all the programming interfaces that already have been introduced to the developers’ community. These interfaces can be divided in two different subsets: SIP APIs and Media APIs. SIP APIs give access to the control plane of the system while the Media APIs give access to the user plane of the system.

2.3.3 SIP APIs

The complexity exposed by IMS combined with the SIP protocol represents a serious obstacle for future programmers and service developers. The knowledge needed to create even simple applications is enormous. This approach hinders the use of IMS and makes developers reluctant to use IMS. One way to remedy this situation will be to provide the developers’ community with Application Programming Interfaces (APIs) that encapsulate relevant specific aspects of the IMS functionality, enabling programmers to focus on the application service logic.

A first effort to categorize the interfaces used for SIP programming reveals different trends that each company follows. IMS vendors offer proprietary APIs that can be used only within the vendor’s platform.

Thus, developers inside the company can develop applications. Alternatively, a more organized approach can be followed by establishing standardization bodies that define standard APIs for IMS-application development. The goal is to boost innovation and cost reduction by addressing a much larger community of programmers.

The level of abstraction an API provides will be of great importance for future programmers. High-level API abstraction completely hides the underling SIP functionality and provides to developers an abstract programming model that is not bound to the protocol or network concepts. In contrast, low-level APIs give the programmer the ability to manipulate SIP protocol objects such as: headers, messages, and so on. Of course this categorization exposes some tradeoffs. Low-level APIs can provide great flexibility enabling the development of any SIP application, but programming is time-consuming and requires deep understanding of the protocols and the architecture. Although, high-level APIs can be proven easy to use, fast to deploy, and do not require understanding of the underlying IMS protocols and interfaces, the development of complex applications is nearly impossible.

The APIs that have been created thus far are described in the following paragraphs.

2.3.3.1 Standard APIs

A graphical representation of the standard APIs is shown in Figure 0-5.

<table>
<tr><th>Desktop-Client</th><th>Mobile-Client</th><th>Server</th></tr>
<tr><td>JAIN SIP JSR032</td><td>SIP API for J2ME JSR180</td><td>SIP Servlet API (SSA) JSR116</td></tr>
<tr><td></td><td>IMS API JSR281</td><td>SIP Servlets JSR289</td></tr>
<tr><td colspan="2">JAIN SIMPLE JSR165</td><td>JAIN SLEE JSR22 & JSR240</td></tr>
</table>

Figure 0-5: Standard APIs.

2.3.3.1.1 JAIN SIP & SDP

Java APIs for Integrated Networks (JAIN) SIP is a standard [40] implemented in Java. it exposes a low-level SIP interface. Thus, a developer can handle IP addresses and ports to messages, SIP headers, parameters, etc. JAIN SIP supports RFC 3261 [3] functionality and the following SIP extensions: the INFO method [41], Reliability of provisional responses [42], Event Notification Framework [60], the UPDATE method[62], the Reason Header [63], the Message method [64], and the REFER method [65]. Based on these APIs different SIP entities such as UAs, Back to Back User Agent (B2BUA), and proxies can be built. The disadvantage of this approach is that when developing an application the core SIP entity logic should be implemented first, then the application. This has a cost in time and maintenance.

On the other side, JAIN gives access to the very basic of the SIP protocol and can be used to implement any kind of application without barriers.

When it comes to the server side, JAIN gives the developer the opportunity to follow two different approaches. The first approach is to implement the server from scratch and the second approach is to use a container. The container will take care of the connections and the scalability of the system. The first approach tends to be time and resource consuming and requires different implementations for different applications; while the second approach provides a more specific and well defined way used to implement any application.

JAIN Session Description Protocol (JAIN SDP) [43] provides access to the SDP contents.

2.3.3.1.2 SIP Servlets

The SIP Servlet API is intended to provide an interface between a container and a SIP application. It provides functional interfaces for building SIP applications. The abstraction level it exposes; makes the

creation of server-side SIP applications easy. Furthermore it can be combined with HTTP servlets' interface and offer a broader area of applications. This API is specified in [44] and has evolved to [45].

2.3.3.1.3 SIMPLE Instant Messaging

The SIMPLE IM API defines an interface for accessing presence and Instant Messaging (IM) services based on the SIP protocol. This API is defined in [46]. The SIMPLE Instant Messaging provides a standard portable and secure interface to exchange messages between SIMPLE clients. SIMPLE is an extension of SIP to support presence and instant messaging. The Java Specification Request (JSR) aims to elaborate a JAIN™ API that leverages the Session Initiation Protocol (SIP) to provide instant messaging

2.3.3.1.4 SIP API for J2ME

An API targeting the Java Micro Edition (J2ME) devices is specified in JSR 180 [47]. This API provides an abstraction layer higher than that of JAIN SIP & SDP. The developer does not need to bother with forming simple SIP messages; since the API does this. JSR 180 defines an optional package that provides a general SIP API for J2ME clients, so that SIP applications can operate in devices with limited memory. This interface enables Java-enabled mobile devices to send and receive SIP messages. The API is compact, so it has a small footprint, and is generic, so it can be used with any J2ME profile, not just with Mobile Information Device Profile (MIDP). As a minimum, it requires version 1.0 of the Connected Limited Device Configuration (CLDC), but the APIs can be used with the Connected Device Configuration (CDC) as well.

2.3.3.1.5 JAIN Service Logic Execution Environment (SLEE)

JAIN SLEE is similar to the SIP servlets in the sense it provides an interface to the container as the SIP servlets do. The main disadvantage is the lack of a functional interface for developing applications. This implies the need to be combined with other interfaces that provide access to the SIP protocol functionality, like JAIN SIP. JAIN SLEE is standardized in [48] and [49].

2.3.3.1.6 IMS API

The IMS API is in its final phase of standardization and is scheduled to be released under [50]. This API is targeted at the Java Micro Edition (JME) Platform and provides a very abstract interface to the IMS functionalities. Its main advantage is the access types that provides to the IMS enablers such as presence, Push-to-talk over Cellular (PoC), and XML. The developer does not need any specific knowledge of IMS in order to create simple applications.

2.3.3.1.7 OSA/PARLAY

OSA/PARLAY [51] is a family of standard interfaces that cover several different aspects such as, call control, user interaction, messaging, mobility, presence, and charging. The abstraction they provide gives programmers the ability to directly use these aspects without bothering about the technology used below them. These APIs are all service APIs, rather than protocol APIs. Some of the most important APIs are [53]:

1. Generic Call Control,
2. Multimedia Call Control,
3. Mobility,
4. Data Session Control,
5. Charging,
6. Presence and Availability Management, and
7. Multimedia Messaging Service.

2.3.3.1.8 PARLAY X

PARLAY X provides a very-high-level abstraction compared to the OSA/PARLAY APIs. Their main attribute is that they are based on web services and not on CORBA (as OSA/PARLAY is). Some of the most

Call Notification	Call Handling
Short Messaging	Audio Call
Multimedia Messaging Service	Multimedia Conference, and
Terminal Location	Presence.

important APIs are [53]:

2.3.3.2 Open-Source APIs

Almost every vendor active in this area has created its own proprietary interfaces. There are of course many open source implementations that serve references to the aforementioned APIs. Specifically worth mentioning are:

- JSIP: A NISTreference implementation of the JAIN SIP API [52],
- MOBICENTS: Open-source JAIN-SLEE [54],
- JIPLETS: Servlet-like API that uses JAIN SIP [55], and
- OPENSER: Open source SIP server [56].

2.3.4 Media APIs

Earlier we discussed the SIP APIs which expose the control plane of the IMS architecture. They provide access to the network by establishing sessions, sending messages, manipulating subscriptions, etc. For an application, it is important to specify a way to handle the media. This implies the creation of platform specific APIs that expose the media capabilities of the underlying platform or the development of a cross-platform API that will be a standard.

For this reason two interfaces are specified:

1. The Mobile Media API (MMAPI) [57] specifies an API for Java Micro Edition (JME) Platform. It is intended to allow simple access and control of the basic audio and multimedia resources.
2. The Java Media Framework (JMF) [58] specifies an API for Java Standard Edition (JSE) Platform. It is designed to allow applications to capture, present, store, and process time-based media.

Apart from the APIs focusing on giving access to the media capabilities of the underlying platform; there is also the possibility of accessing existing media in an external platform via a media server. In this case either protocol specific APIs such as SIP and MEGACO (MEdia GAteway COntrol) or protocol-agnostic APIs such as [59] can be used.

The next chapter will introduce the application developed for this thesis project by using the MJCF APIs. A general description will be given and its functionality, architecture, and design will be explained.

3 Super-Discounts: A client – server application over IMS

3.1 General introduction and purpose

Super-Discounts is a complex application built on IMS. It is part of a thesis project conducted at Ericsson research and resulted in a commercial demonstration application. The application consists of two parts. The first part resides on the client side (a Midlet running on a mobile device) and the other on the server side (a Servlet running on an AS). The application provides an easy way for consumers to get informed about shopping discount as fast as possible and without searching for them around the city. This approach incorporates the location of the user device and a user profile. The user sets up his profile by specifying the products he or she wants to get discounts for. After that, the location of the user is determined and the distance between the subscriber and the location of a discount is calculated. If this discount is checked on the subscriber's profile list, then the subscriber receives the discount.

The penetration of the mobile devices has grown tremendously the last years and almost everyone owns a mobile phone. The capabilities of a new generation phone are enough for such an application. In order to describe the application in simple words it is preferable to present a real life example, see Figure 3-1.

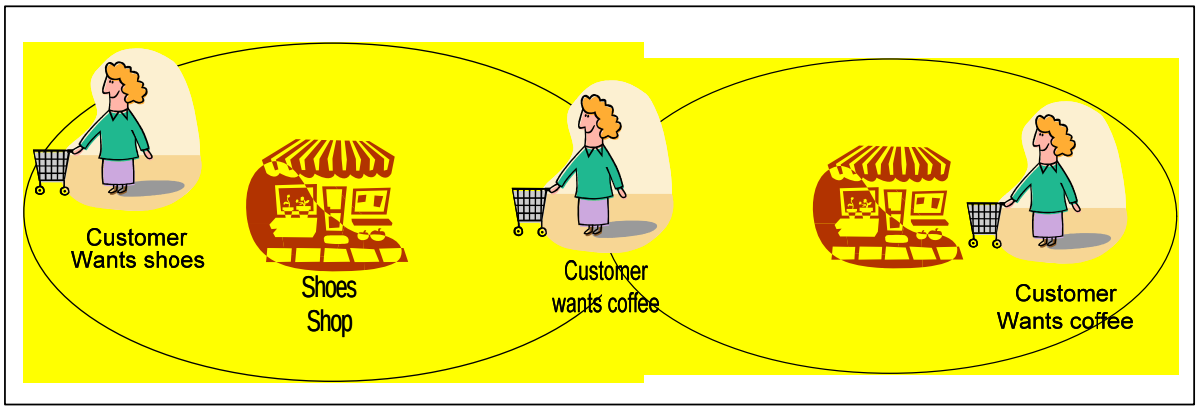

Figure 3-1: Real life scenario.

Alice (our user) is interested on buying new shoes (see Figure 3-1) for a gala she is invited to, tonight. She wants to find something cheap and nice, but she does not have much time to spend searching through the stores. Super-Discounts offers the solution to her problem. The only thing she needs to do is to check the correct box in her profile. New discounts matching her preference will be directed to her mobile device as she is walking around. These notifications will contain a description of the product as well as images, videos, and audio. Alice does not need to walk far to buy her shoes since the discounts sent based on her location.

Different discounts are published by different stores. The discounts are bound to the location of the store. This will generate a coverage area around the shop. Every customer who walks into this coverage area will be a potential receiver of the discount. Only the customers who are interested in receiving such discounts (based on their profile) will eventually receive the discount.

The main functionalities of this application will be presented in the following section.

3.2 Application functions

In order to develop an application, it is important to specify the requirements and the functions each side (Server - Client) should follow.

Client side functions:

- The user needs to first register with the IMS network by providing a username and password.
- Thereinafter the user needs to subscribe to the service in order to receive or upload discounts. This action informs the server side that a new client is available for receiving discounts or intends to upload a discount.
- The client needs to listen and accept incoming messages or discounts from the server.
- The client needs to have access to his profile and the ability to change it.
- The client needs to have a way to read the incoming discounts
- The client needs to have a way to upload a discount.

Server side functions:

- The server should be able to listen for users that want to subscribe for receiving or uploading discounts.
- The server should be able to learn the location of the user either by asking a location broker or by directly asking the mobile user.
- The server should have access to the profiles of the users in order to make decisions about if a discount should be forwarded or not.

The above functionalities need to be implemented in a similar and IMS compatible way. The new IMS Innovation APIs give us this ability. They provide all the tools to simply develop and test those functionalities. Figure 3-2 gives a graphical presentation of how these functions could be implemented by using the IMS components.

Client user registration:

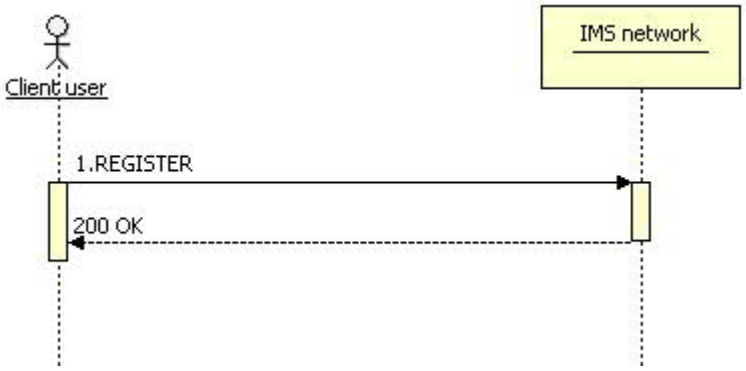

Figure 3-2: Registration phase

In order for the client to register with the IMS network it is needed to send a SIP REGISTER message (see Figure 3-2). This message will contain the public and private user identity and the user's password. When the message is received by the IMS proxy it will be analyzed and the username will be checked against the HSS database which keeps a list of all the users authorized to use the system. If the username and the password match any of the entries in the database, then the user is authorized to access the IMS network. The IMS network stores contact information about the client. This information can be retrieved based upon the "contacts" header in the REGISTER message. A 200 OK SIP message will be the response for an authorized user. This message will contain a record-route which is useful for later communication with the IMS network. The registration process is specified in RFC 3261 chapter 10.2 and 10.3 [3]. If the user is not authorized or any of the credentials is wrong a "401 Unauthorized" SIP message will be sent back.

Client user subscription:

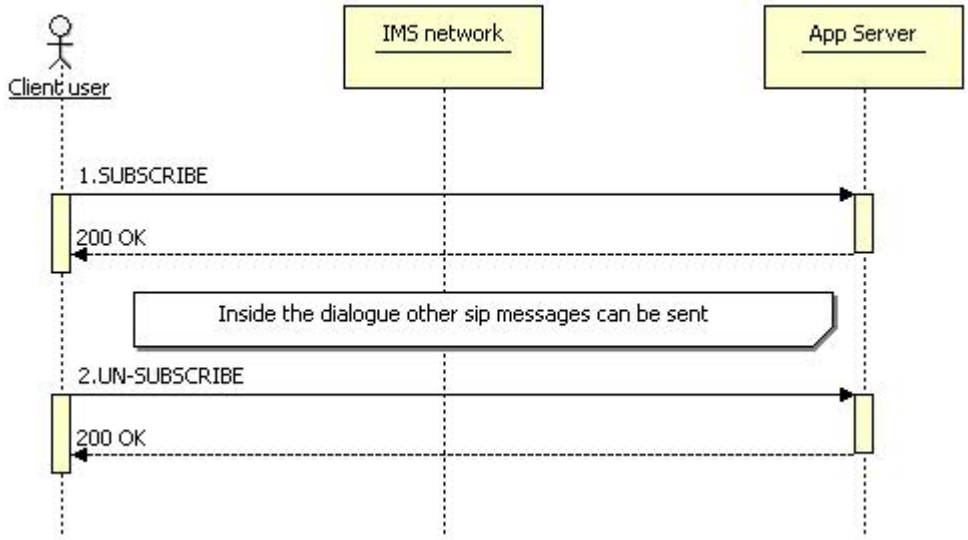

Figure 3-3: Subscription phase

The subscription to the Application Server (AS) can be performed in two different ways. Either a simple SIP MESSAGE is being sent or a SIP SUBSCRIBE message is sent. The difference between these two messages is in the logic that they follow. Based on [60] the SUBSCRIBE method "*is used to request current state and state updates from a remote node".* For this application this is very important as the AS should always know when a user is available to receive discounts or not. The SUBSCRIBE technique provides the server with this information since a new dialog is created. As long as this dialogue is active the user is online and can receive messages and discounts. The SUBSCRIBE method provides two important SIP headers for this reason. The first one is the "Expires" header which informs the AS when the state of the client will expire and when a new SUBSCRIBE message should be expected. A second header called "Event" informs the AS which event package the user wishes to subscribe to [60]. When a client wishes to "un-subscribe" from an event package, then a SUBSCRIBE SIP message is sent with the "Expires" header set to 0. The normal response when everything is okay on the server side is 200 OK. It is important to mention that all the messages originating from the client will pass through the IMS network. The IMS proxies are responsible for forwarding the messages to the AS. The same occurs when the AS wants to send a response or a message to the client.

Client user uploads discount:

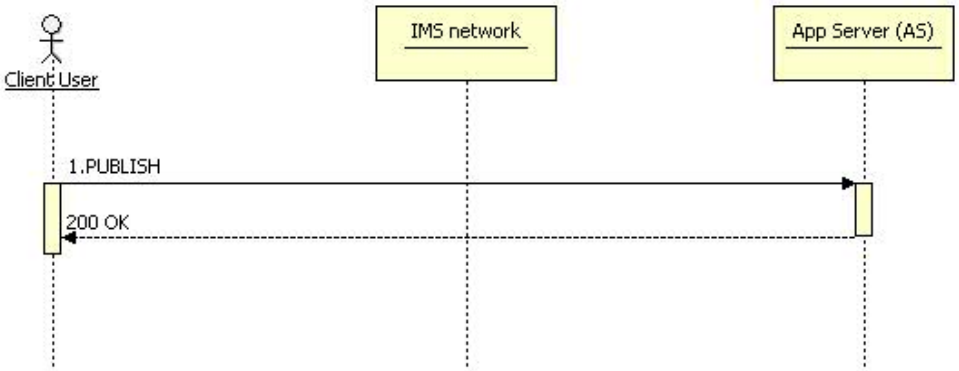

Figure 3-4: Upload discount

The user can upload (see Figure 3-4) a discount either when he is subscribed to the event or not. The user should be registered with the IMS network in order to be able to upload a discount. The SIP message used for this reason is the SIP PUBLISH [61]. The body of the PUBLISH message will carry the discount information. The PUBLISH message is always used while a dialog is active (SUBSCRIBE creates a dialog). This means that the user should also be logged in to the application (subscribed). This is the main reason which a SIP MESSAGE is not used. In general the application will be relieved of checking for any new message if the user is subscribed or not.

Client user receives discount:

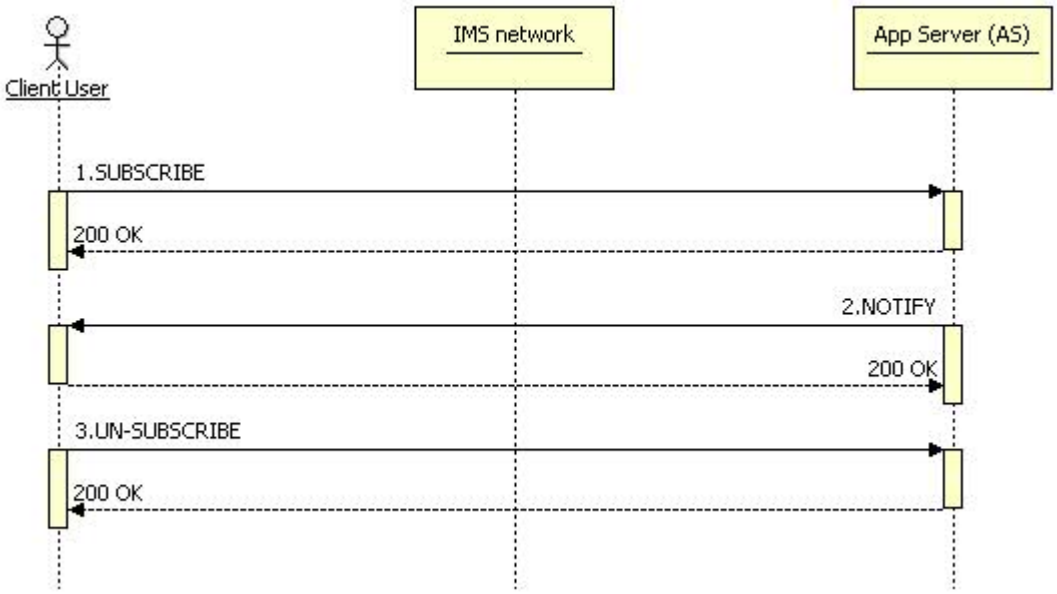

Figure 3-5: Discount received

When a discount is to be sent to a client (see Figure 3-5) a SIP NOTIFY message [60] is used. The body of this message will carry the discount information. Using the notify inside a “subscribe” dialogue assures the server that the user is online. If the discount never arrives, then either a problem with the message body occurs or the client cannot accept the message for some reason.

Client user receives and saves profile:

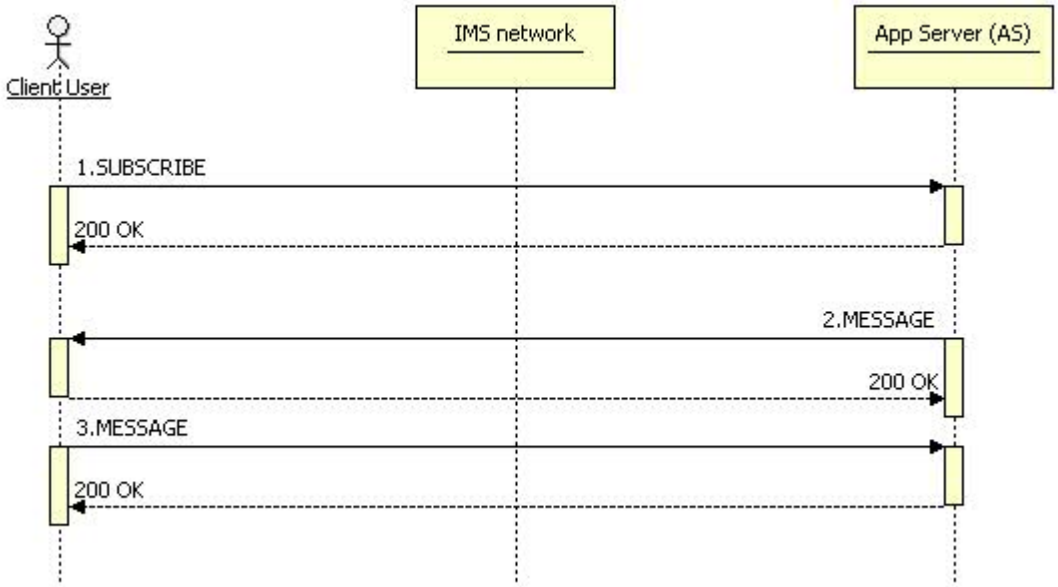

Figure 3-6: Receive and save profile.

The user receives his profile (see Figure 3-6) which was stored on the server in a SIP MESSAGE. The body of the message contains the profile information of the user. The profile is stored on the server in order to make the application more scalable, faster, and more adaptable to future changes. It is better to have the profile on the server if the user wants to access it through a web interface. It is also a good location if the user wants to install the application on different mobile devices.

A complete scenario:

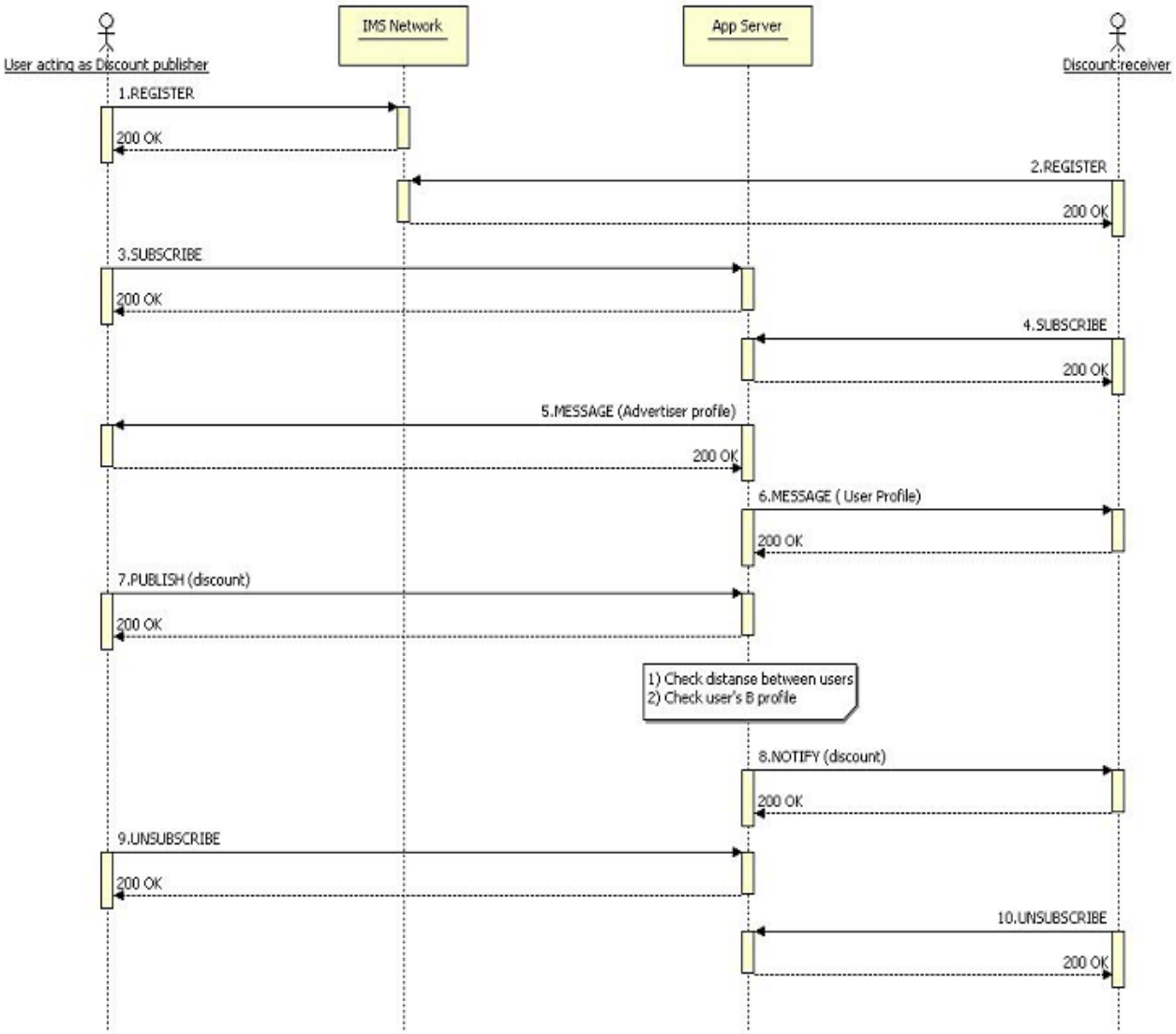

Figure 3-7: Complete scenario

In the normal scenario (see Figure 3-7) users register with the IMS network and subscribe to the AS. When users subscribe their profiles are sent to them. The subscribers can play the role of a shop owner who uploads a discount and of a regular user who wants to receive discounts. When the discount is uploaded it is compared against the profile of the "Discount receiver" user and the location of the "User acting as Discount publisher". If there is a match, then the discount is finally sent to the user.

Server functionalities:

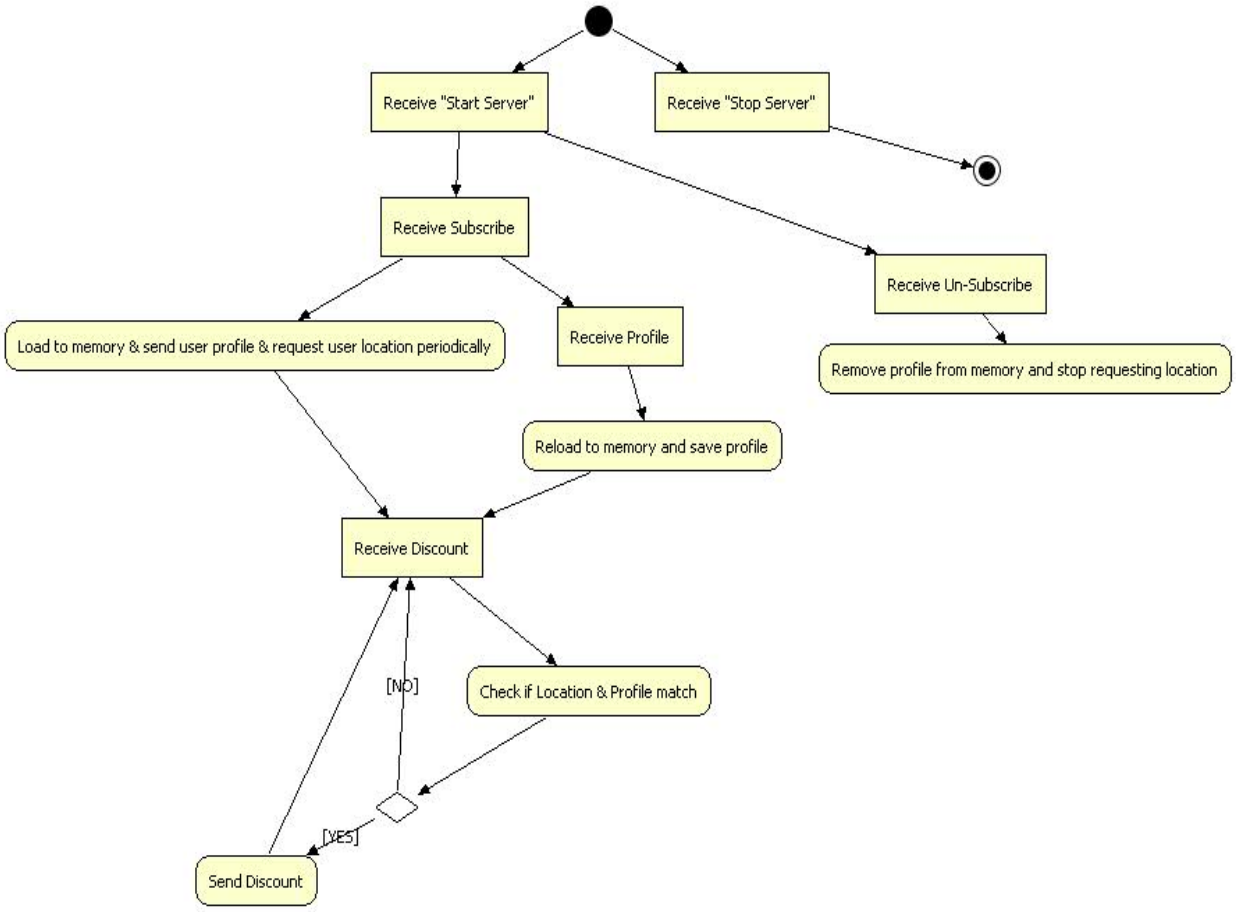

Figure 3-8: Server functionalities diagram

The server can be in two different states (see Figure 3-8). Either the server is in the "running" state or not. When the server is in the "running" state it waits for users to subscribe or to unsubscribe. When a user subscribes to the application his profile is loaded into the application's memory and is sent to the user. At the same time the server starts asking for the location of the user from a location broker. Every a time the server is updated with the location of the user.

If a discount is received by a subscribed user a number of different actions could occur. First of all the server has to match every subscribed user and their location with the location of the discount. This check is done very fast since the location of every user is always updated. If these two locations are sufficiently close then the user is eligible to receive the discount. Next the server checks the profile of the candidate users against the discount. If any user's profile matches, then the discount is sent to these users.

In the next section, the architecture of the application and the programming design will be analyzed.

3.3 Application Architecture & Design

Figure 3-9 presents the overall architecture of the system. Mobile users can be connected to the IMS network through different access architectures. The IMS network is responsible for authenticating the user and forwarding the messages to the appropriate AS. The AS at the top of the picture is responsible for hosting the application and serving the users. For the Super-Discounts application a Location Broker (LB) is needed. This LB is responsible for contacting the operators and learning the location of the users. It is very important to keep this server secure and protected from malicious users.

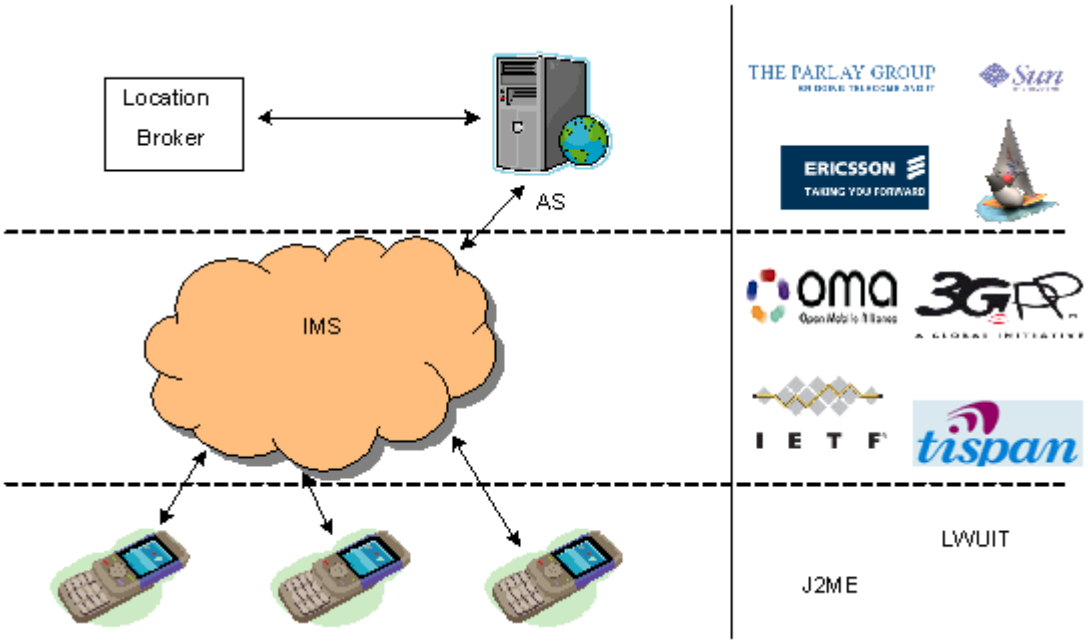

Figure 3-9: Overall application architecture

Considering the AS, there are many options for the implementation to use. The market provides many free opportunities such as:

- SipExchange [66] on top of JBOSS,
- SIP Container [67] Server by Ericsson, and
- Sailfin [68] based on GlassFish.

This application was developed on the Sailfin AS since this was desired by Ericsson. The SailFin project can be considered as a component on top of the GlashFish [68]server. This component adds SIP Servlet functionality based mostly on JSR 116 (SSA 1.0) [44]. Efforts are ongoing to enhance this component to support JSR 289 (SSA 1.1) [45]. The packages provide high availability, clustering features, and an easy way of integrating with the existing services.

Considering the client part, Java MicroEdition was used. On the top of it a new User Interface (UI) library was used. This library is called LWUIT [69] and helps content developers create compelling and consistent Java ME applications. It provides forms, buttons, rendering techniques, and much more. It is easy to use and solves many problems that developers might have been facing if the Mobile Information Device Profile (MIDP) were being used directly.

Based on the architecture above and coming to the programming part, two different package structures were introduced.

Server Package

Starting with the server, the functions described in the previous section can be implemented by using different Java sub-packages. Any of these packages will host these functions. Figure 3-10 presents this approach. On top of the packages an httpServlet exits. This package is responsible for starting and stopping the server. When the administrator starts the server a HTTP message arrives. Directly after that the httpServlet package will initiate the imsTrensactions and the notificationServer.

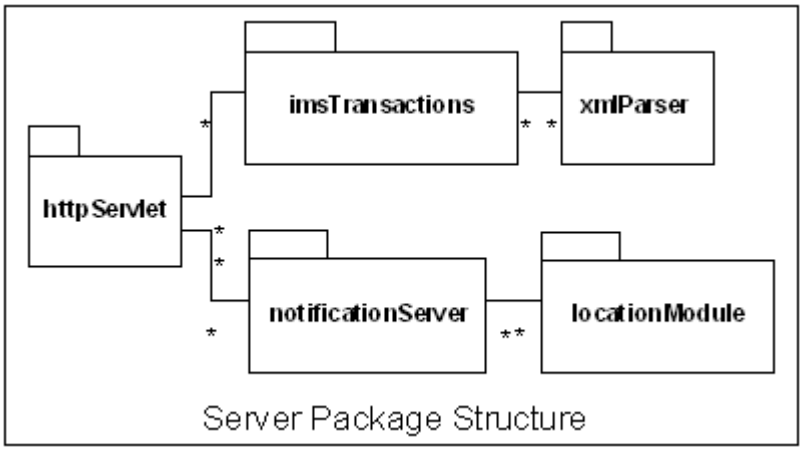

Figure 3-10: The server package structure in Java

The imsTransactions package is responsible for receiving and sending IMS messages. These messages were described in the previous section. By the time a new SIP SUBSCRIBE message arrives, a new user comes online. This user is a candidate for sending and receiving discounts. After that the notificationServer is informed. When a SIP PUBLISH message arrives the server calls the xmlParser package to parse the message body (see Figure 3-11). This message body describes a discount.

```
PUBLISH sip:alice@192.168.10.1 SIP/2.0
From: <sip:alice@192.168.10.1>;tag=1234882888751
Accept-Contact2: *;+g.3gpp.app_ref="urn%3Aurn-xxx%3A3gpp-application.com.imsinnovation.HiFiveApplicationServer";require
Accept-Contact2: *;events="superDiscounts"
Accept-Contact2: *;events="presence.winfo"
Accept-Contact2: *;message;events="presence"
To: <sip:alice@192.168.10.1>
P-Preferred-Identity: sip:alice@192.168.10.1
Content-Type: text/plain
Via: SIP/2.0/UDP 192.168.10.101:4111;received=192.168.10.1;branch=z9hG4bK10c92998c6d190281499ebd5ce2085ca
Max-Forwards: 70
Content-Length: 412
Expires: 3600
Cseq: 413 PUBLISH
Contact: <sip:192.168.10.1:54753;transport=tcp>
Event: superDiscounts
Call-Id: 457c80a3186464661be1708766319a49@192.168.10.1

<?xml version="1.0" encoding="UTF-8"?>
<superDiscounts>
<message-body type="notification">
<title>Discount Title</title>
<description>Discount description</description>
<imageUrl>http://www.themusicbuilders.com/images/songs/twinkle_twinkle_little_star_icon.jpg</imageUrl>
<videoUrl></videoUrl>
<startDate>1234882888673</startDate>
<duration>5</duration>
<category>shoes</category></message-body></superDiscounts>|#]
```

SIP Headers

SIP Body

SIP Message

Figure 3-11: PUBLISH SIP message (discount upload)

In general two different application specific messages can be sent and received from the server. The first one is a discount and the second is the profile of the user. The profile is transferred in the body of a SIP MESSAGE. This message is sent to the client directly after he/she subscribes to the application. On the other hand a profile is sent to the server when the user decides to update his profile. The message delivered by the client to the server for updating his profile is a SIP MESSAGE as well. Figure 3-12 shows a profile update sent by user Alice.

Figure 3-12: MESSAGE SIP message (profile upload)

The two different bodies are following a general Extended Markup Language (XML) format. This format is application specific and describes the different perspectives (discount and profile). It is clear from Figure 3-11 that a discount is constructed of a “title”, a “description”, a “imageUrl”, a “videoUrl”, a “startDate”, a “duration”, and a “category”. These attributes are important for the AS in order to determine in which user the discount should be sent (“category”) and for how long the discount should be valid (“startDate” and “duration”).

From Figure 3-12 we can see the attributes that characterize a profile. In general we can say that a profile consists of elements (“profile-element”) and groups (“profile-group”). Each group can host more elements or more groups. Thus, every possible profile can be created. The administrator of the system can easily extend and update a global profile. All the user profiles can be checked against the global profile and will be updated if any change occurs. This way the business model can be extended and easily adjusted to the needs of the market (new groups of products or services).

The notificationServer is the other important sub-package of the application. It is responsible for checking the locations of the users and their profiles against the new discounts. Since every discount is originated by an end user, a location is attached to it. This location and the location of the user are checked against every prespecified time period (for this demonstration is set to 1 minute). If the distance between any pair of locations is less than a prespecifed distance (for this demo 100 meters) and the profile of the user matches the “category” in the discount then a notificationServer event occurs. This event informs the imsTransactions package and the discount is sent.

Concluding with the server, the package structure is shown in Figure 3-13. The entry point to the application is the ImsTransactionServer. This object implements the SipServletListener which is responsible for initializing the servlet. Despite this fact, the application logic starts from the DiscountsTracker object. This object will be responsible for calling the ImsTransactionServer and the ImsTransactionClient, when it comes to IMS communication as well as the NotificationCoreEngine, when a decision for sending a discount is made. Additional sub-packages are added to help logging and profiling of the application. The fileSystem package is used for storing the profiles on the disk and storing them for the next time a user logs on. The database (dB) package hosts the storage of the application. All the discount objects, the profile objects, the threads, and the user tracking objects are kept in the MemoryDbHandler.

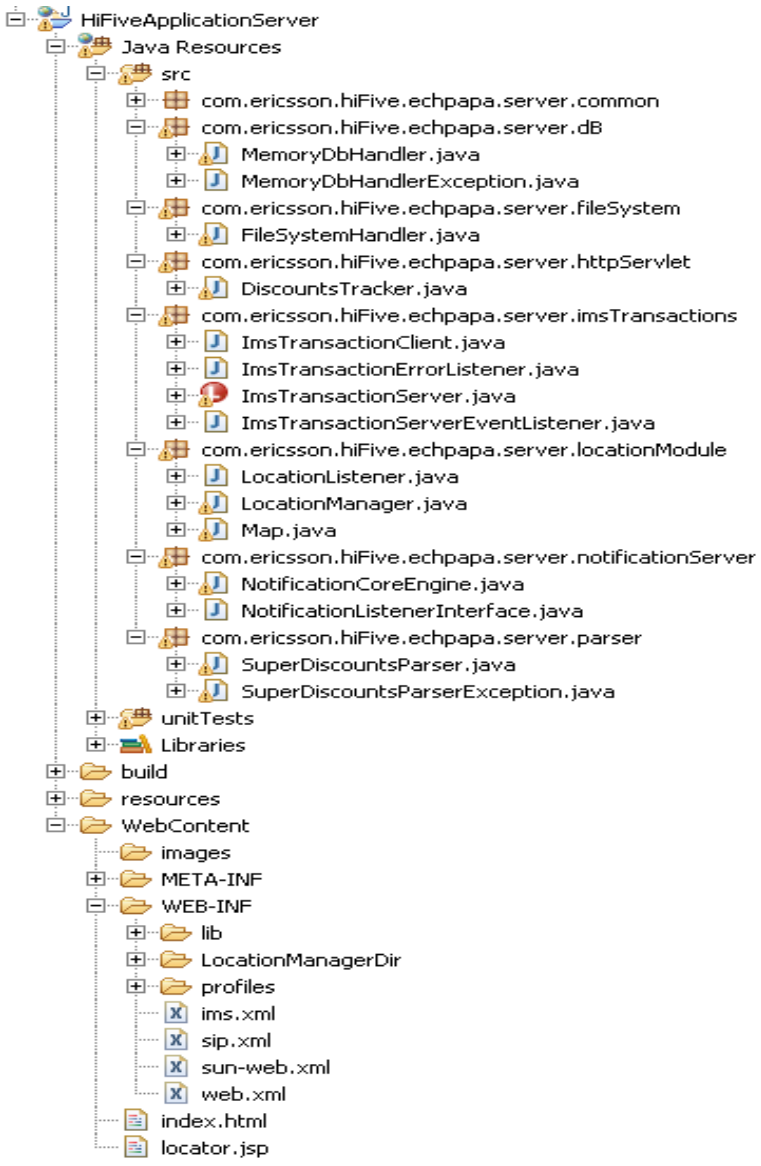

Figure 3-13: Expanded server package structure

Client Package

Based on the structure of the server package a similar approach can be followed for the client package. The client functionality was discussed in the previous section and a diagram of the package can be seen in Figure 3-14. The imsTransactions sub-package is responsible for sending and receiving IMS specific messages. The messages needed for registering with the system, subscribing to the application, uploading a discount, and updating or downloading the profile are handled by this package.

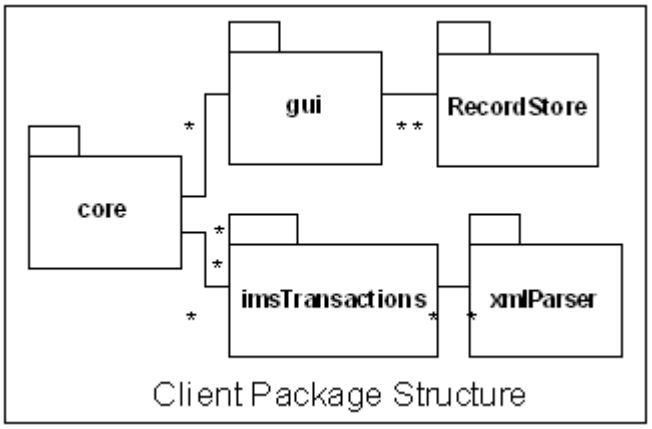

Figure 3-14: The client package structure in Java

The Graphical User Interface (GUI) package on the other hand is responsible for exposing the functions in a user friendly form. This is done by the use of an assistant library called LWUIT. This library provides all the methods for presenting colorful forms, tabs, pictures, and many other graphic components on the screen of a mobile device. It is completely portable and it can perform on any different Java enabled device. The GUI package makes use of this library the user to interact with the IMS network and the application. Several forms see Figure 3-15 are giving the user the ability to browse new discounts, change and update his profile, upload a discount, and much more.

Figure 3-15: Client graphical interface

The package acting somehow as a “proxy” between the “gui” and the imsTransactions sub-packages is the “core” sub-package as presented in Figure 3-14. This package is responsible for getting the messages from the “transactions” layer and interpreting them in a more user friendly way by calling different methods of the “gui” package. When, for example, a discount arrives the imsTransactions will parse it and send it to the core package. The core will have to call the correct method of the “gui” package to represent the discount in a form.

Figure 3-16 shows the package structure. The SuperDiscountsClient object will be invoked when the application starts up (extending the Midlet). This object will start the server (ImsTransactionsServer) and will trigger the registration process. At the same time the graphical interface (GuiController) will be initialized and the transaction with the user will be initiated.

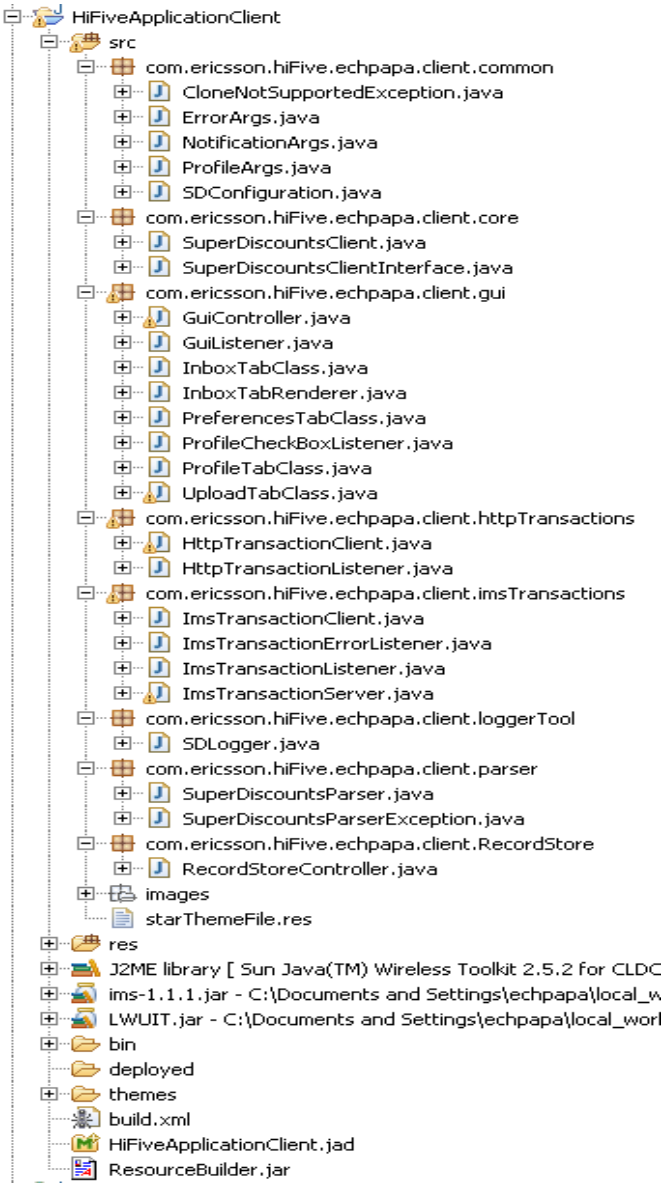

Figure 3-16: Expanded client package structure

The next section explains some extensions that made in the code by using some more packages in order to get some measurements regarding the performance of the MJCF APIs.

3.4 Extensions for measurements

The main objective of this chapter is to introduce some tools and techniques for measuring different characteristics of the application. These measurements can be divided based on memory restrictions and the delay introduced by the use of the APIs while sending different messages.

Since the application is operating over the network, a delay is introduced while the packets are sent from one point to another. By measuring the time needed, for a packet to be sent through the MJCF API and the time needed for the response to come back, we can compute the Round Trip Time (RTT). This RTT includes the delay added by the network. It is very important to measure the impact that MJCF API adds the network delay. In order to do this we need a method to measure the network RTT. This can be done by creating a UDP server and a UDP client. Thus the same message can be sent and an approximation of the RTT can be obtained.

On the other hand, in order to measure the memory impact of the MJCF API when an implementation is done on the server and on the client side, a memory profiler is needed. This profiler should measure specific memory load when different events happen. To do so, the tool is provided by java through the java.lang.Runtime object.

3.5 Future application development

This application, as already explained, resulted in a demo with very specific functions. Of course those functions can be expanded in order to accommodate future needs and to exploit additional business models. Below a list of proposed alternatives regarding the functions and the architecture are given.

- Functionalities (client)
    - The user can receive a map of all the discounts for a specific product and scroll through these offers. Thus, instead of getting only discounts that are very close to the users, he/she can see the relative position among the location of the mobile device and the location of the discounts.
    - If the client wants to have greater accuracy regarding the location of the device could periodically send their GPS location. This of course is subjective to the user as he/she must always have the opportunity to deny such an action.
- Functionalities (server)
    - Since every shop owner has a specific amount of products stored, it will be unsuccessful if more customers come to the shop and there are no products available. In order to solve this problem a shop owner can request from the customers to subscribe to a discount before they pass by the store. For example, the shop owner can specify a number of pieces of a specific product that are for sale and subject to a discount. After this any client that wants this product can subscribe to this discount.
    - Another useful function could be e-payment. Customers can prepay for a product that is on discount immediately after they receive the discount. This can be proved to be very helpful for the customers that do not have time to pass by the shop.
    - Yet another function could be the access to the user profile through a web interface. This interface allows the user to set his/her own preferences and browse all the available discounts. He/she can also subscribe or e-pay for a specific discount.
    - If the server goes (for some reason) down, then all the discounts are lost. In order to avoid this a recovery function can be added.
    - Security can be implemented for the transmission of the profile of the user as well as the transmission of the discount.
- Architecture (client)
    - The client can be implemented by using a state machine. Thus, the coordination between the transaction layer and the presentation layer (graphical interface) can be faster and easily extensible.
- Architecture (server)
    - The server can be implemented by using a state machine. This can be very helpful when it comes to error handling and extensibility. More functions can be added simply by adding a new state.

4 Evaluation Procedure

"Evaluation is the analysis and comparison of actual progress vs. prior plans, oriented toward improving plans for future implementation. It is part of a continuing management process consisting of planning, implementation, and evaluation; ideally with each following the other in a continuous cycle until successful completion of the activity."[70]. Based on this short definition of what an evaluation is, this study will focus on evaluating the Ericsson IMS Innovation framework. The evaluation that follows serves three main purposes:

- To determine if the specified goals and objectives are being met and to which extent [74],
- To find out any troubles and report them,
- To use the knowledge gained, both theoretical and practical, in order to propose future amendments or changes to the API.

These three steps form an iterative procedure in the future until the objectives of the product can be met. This is mainly done because the evaluation is a continuous process that tends to proceed through cycles of evaluation, revision, and reevaluation.

The evaluation procedure will continue by describing the environment and evaluation criteria.

4.1 Evaluation Environment

Ericsson has provided the developers' community with a set of tools and documents for complete development of IMS applications. The developers should, first of all, have access to an actual IMS architecture or to a simulated IMS architecture. At the time this study was conducted, both of the aforementioned alternatives were evaluated.

Ericsson in cooperation with a major telecommunication operator in Sweden have established and deployed an IMS network. The main IMS components have been deployed at the operator and an AS has been set up. This AS is accessible by anyone through a public internet IP address. Every IMS developer can upload an application to the AS and test it. A tutorial [74] is available with guidelines on how to upload and deploy an IMS servlet.

Additionally Ericsson has released the SDS 4.1 simulator [71] which supports almost all the functions of an IMS network. The simulator is implemented in C++ and Java. Developers can access the interfaces through multiple Eclipse plug-ins. The developer has access to all the CSCF proxies as well as to the provisioning view where he can configure the HSS and the service triggers. SDS does not consider with charging and the developer cannot handle these aspects of IMS. SDS includes the appropriate tutorials and developer guides for the functions it provides.

When an application is developed and tested on SDS.A portable Personal Computer (PC) was used. This PC comes with an Intel Core 2 Duo CPU at 2.1GHz and 2Gb memory. The SDS 4.1 simulator is installed on it and the Wireless Toolkit [72] is used.

The developer community also has access to multiple APIs that represent the different functions of the IMS system. Those APIs are:

- IMS-Innovation Client API
- IMS-Innovation Server API
- IMS-Innovation Presence API
- IMS-Innovation Media API

For the ease of design and deployment of an IMS application, Ericsson provides several tutorials. For example the developer can find a tutorial for creating Server side applications and Client side applications. Additionally, these are guidelines for using presence and media services.

An evaluation of these APIs was conducted by the author of this document. The author is a student with deep knowledge in internetworking and programming. He has worked with the configuration and maintenance

of large network infrastructures; as well as with the development of mobile applications. He has extensive knowledge of Object Oriented (OO) programming in C++ and Java.

4.2 Evaluation Criteria

The main purpose of the evaluation was to evaluate this framework with regard to exposing IMS capabilities to independent developers. This framework is not strictly bound to specific programming APIs. Instead, the evaluation contains the whole concept including tutorials and programmer guides:

- To create a client application using the MJCF client API,
- To create and deploy a server-side application using the MJCF server API, and
- To install all the appropriate software and simulators in order to develop and test these applications.

In addition to these, a number of additional APIs provide the developer with the opportunity to deliver media, obtain the device's position and publish this location, as well as to use the presence service.

The basic audience for this evaluation is independent developers that have little or no knowledge of the IMS infrastructure and concepts. Therefore, the evaluation should focus on how to provide the developer with the easiest and most useful way to create IMS applications. These APIs should be:

- Well structured, which means that they should follow an object oriented structure with well named interfaces and detailed java documentation
- Easy to learn, so as not to confuse the functionalities because of bad method naming or complex dependencies with other functions,
- Easy to use, leading to rapid code development, this code should be clear to others in order to facilitate future development and maintenance,
- Efficient, so it can be easily expanded and reused,
- Stable and bug free while minimizing memory and CPU utilization.

The compatibility or inheritability of each version of the API is another important factor, since this will help Ericsson to facilitate new APIs and new functionality.

The documentation consists of tutorials and developer guides. These documents shall:

- Be well formed, by means of a clear structure and the content it contains. Thus the document has to give all the appropriate information in a suitable order (by incorporating appropriate figures and diagrams).
- Be easy to read and to understand, in a way that the reader can quickly find and easily understand the relevant material.
- Be informative, in order to present all the relevant knowledge needed for the developers to understand any perspective of the services.
- Have lots of references to help the reader find additional information and to expand their knowledge.
- Contain code snippets and examples to facilitate easier and faster application development.

In order to start evaluating these APIs we have split our evaluation into two different parts. The first has to do with the API used on the client side and the second one with the API used on the server side. Figure 4-1 shows a schematic presentation of a generic client-server architecture.

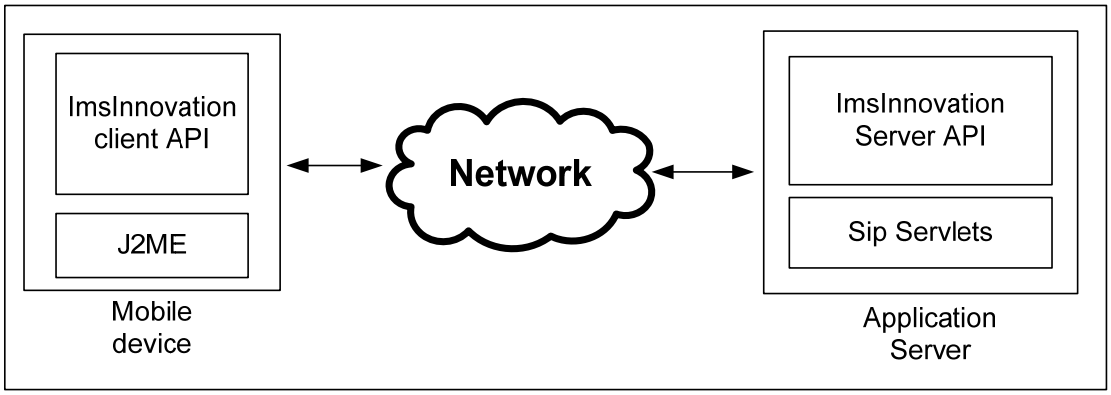

Figure 4-1: The overall system architecture.

The next chapter will focus on the MJCF Client API. The chapter after that will focus on the corresponding server architecture.

5 Evaluation of the client API and documentation

5.1 Inside the Client API

The MJCF client API is used on the top of a J2ME MIDlet. The interfaces are invoked only after a MIDlet is started. From a network perspective the mobile acts as a server by listening on a specific port for incoming messages, and also acts as a client when it comes the time to send messages. For this reason the API is structured as a client-server architecture. A basic introduction is shown schematically in Figure 4-2.

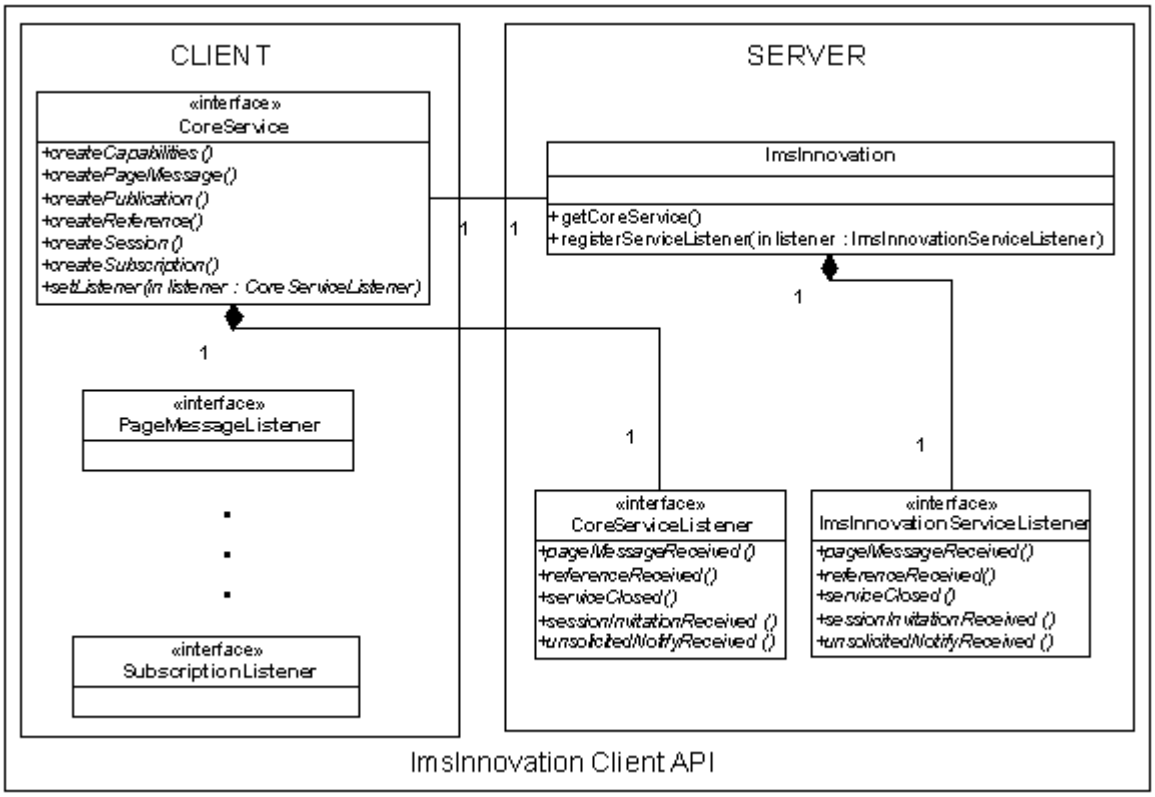

Figure 4-2: Inside MJCF client API.

The server part of the API when it is started simply registers with the IMS by sending a SIP REGISTER message. Subsequently it waits for incoming messages. When a message arrives it is parsed and the correct interface method is invoked. The developer should handle the incoming message. In addition the client part of the API is used through the server.

The ImsInnovation object instantiates a CoreService object when it is constructed. The CoreService object acts as “the factory” for SIP messages that the developer wants to send. If, for example, a message (SIP MESSAGE) needs to be sent, then the code used is depicted in Figure 4-3.

```
try {
        String iari = "3gpp-application.+++";
        ImsInnovation imsInnovation = new
                ImsInnovation(this, this.getClass().getName(), iari, this);
        CoreService coreClient = imsInnovation.getCoreService();
        PageMessage mMessageClient =
                mCoreClient.createPageMessage(null,"sip:toName@something");
        mMessageClient.send("some text", "text/plain");
} catch(IllegalArgumentException e) {
        e.printStackTrace();
} catch (ServiceClosedException e) {
        e.printStackTrace();
} catch (ImsException e) {
        e.printStackTrace();
}
```

Figure 4-3: How to send a message with the MJCF client API

Those two different parts (server and client) are followed by the appropriate error listeners which are of great importance when an error occurs. The evaluation conducted in this chapter focuses basically on:

- For the server part:
    - Check if the SIP MESSAGE and NOTIFY are received correctly; and
    - Check if the user is registered and if the correct error messages are received when something goes wrong during the registration phase.
- For the client part:
    - Check if the SIP SUBSCRIBE, MESSAGE, and PUBLISH are sent correctly; and
    - Check if the 200 OK or an appropriate SIP error message is received when an error occurs.

5.2 Evaluation overview

Generally speaking, the client APIs and documentation are adequate and very helpful for the developers. By using them a developer can easily send simple messages and develop simple IMS applications. The messages described above were all sent correctly (Some problems did occur and are explained in 5.2.2). The developer's portal gives all the necessary basic instructions for installing the tools and starting a simple application on a mobile device.

The documentation prepared by Ericsson for the client is called "Client MIDlet Tutorial" [73] and is intended to help the developers create a MIDlet for a mobile client. The APIs can be downloaded from a SVN server [74]. The API evaluated in this report is stored under "Deliverables/trunk/ImsInnovationFramework/Client/binaries/" and the java documentation for this API is stored under "Deliverables/trunk/ImsInnovationFramework/Client/documentation/"

The detailed advantages and disadvantages regarding the APIs used in this thesis and the documentation will be covered next.

5.2.1 Advantages

The evaluation of the client API is divided in two different parts. The first concerns the documentation and the second the APIs. Themselves based on the evaluation criteria (see 4.2) the advantages observed are enumerated in sections 5.2.1.15.2.1.2.

5.2.1.1 Documentation

The "Client MIDlet Tutorial" [74] is a well organized, easy to read, and quite informative. It describes in details the steps that are needed to create a client application using the MJCF API. It starts with the registration phase which is common for any application, then describes some alternatives for sending messages and using the presence functionality.

The Java documentation follows the standard approach and provides all the useful features of such

documentation. The comments and the description are quite good and describe the main idea of what each method does.

5.2.1.2 API

The advantages of the API can be summarized as:

- **Well-designed structure**. The structure of the API is based on JSR 281 [50] and follows its architecture. This is very important, because it follows a general way of building APIs which is very helpful for both experienced and inexperienced programmers. The functions are divided into different packages that are easily understandable by the programmers and easily accessible through interfaces.
- **High abstraction level**. It maintains access to the lower layers, but provides abstractions only for sending simple messages and using media communication. It also supports the presence functionality. This is important when it comes to creating simple applications. The abstract interfaces can be used to quickly and easily access difficult and advanced aspects of the IMS functionalities by using core objects.

5.2.2 Disadvantages

In a similar way, as we did concerning the advantages, we split between the documentation and the API.

5.2.2.1 Documentation

While the "Client MIDlet Tutorial" follows a correct way of explaining the different functions it does not give enough examples or snippets of code to enable a developer to develop a simple MIDlet application. One of the misunderstandings faced while implementing the Super-Discounts application was how to send a simple message. In Figure 4-3 such a way was described. The tutorial gives another way which is very confusing but still correct. Both the ways should be mentioned.

The layout of the document does not follow a professional template. It does not have a table of contents. Graphical representations (pictures) of a simple client application developed on Eclipse by using SDS and the wireless toolkit should have been included.

5.2.2.2 API

The disadvantages of the API can be summarized as:

- **Subscribe event support is incomplete**: when a SIP SUBSCRIBE is sent with the "Event" header containing an event that is created by the user an IllegalArgumentException is thrown. This problem occurs because the API cannot accommodate more than one "Event" header or multiple event packages in the same "Event" header.
- **Error in listener for registration**: when a client tries to register with the IMS network it cannot be notified if an error occurs. A new listener should be created to solve this problem.
- **New headers cannot be added**: when the user needs for some reason to extend an application by using additional SIP headers other than the default header, errors occur.

The next section presents some measurements to help the developers' community and the developers of the APIs to understand the relative delay and memory consumption added when using the MJCF APIs.

5.3 Delay & Memory measurements

These measurements are conducted in order to reveal the real burden the MJCF APIs might face when an application is developed. The clients connect to the server by sending messages over the IMS network. Those messages are initiated by the MJCF API and sent over the network. When a SIP message is sent, a response containing a "response code" [75] is received back. This "round trip" of messages reveals the response time to process of the SIP message.

Figure 4-4 compares the response time needed for the same SIP messages when sent over UDP using a custom java implementation with sockets and when using the MJCF APIs. In general the MJCF APIs use a UDP socket filled with the SIP message in the end. Thus the difference is due to the API.

The time added by the API occurs because it processes the message, first at the client side and then at the server side before it sends the message through the socket. It should be mentioned here that a SIP SUBSCRIBE message creates a dialog [3]. The dialog needs to keeps its state via a state machine and several timers need to be updated when they expire.

Figure 4-4 reveals that less delay is added when a SIP MESASGE is sent. Those messages do not create any dialog and do not carry any headers other than the usual ones. This delay it is still quite big (~150ms) for applications that send hundreds of messages per second. The difference is so large that the end user will experience this difference and this may be too great to be acceptable.

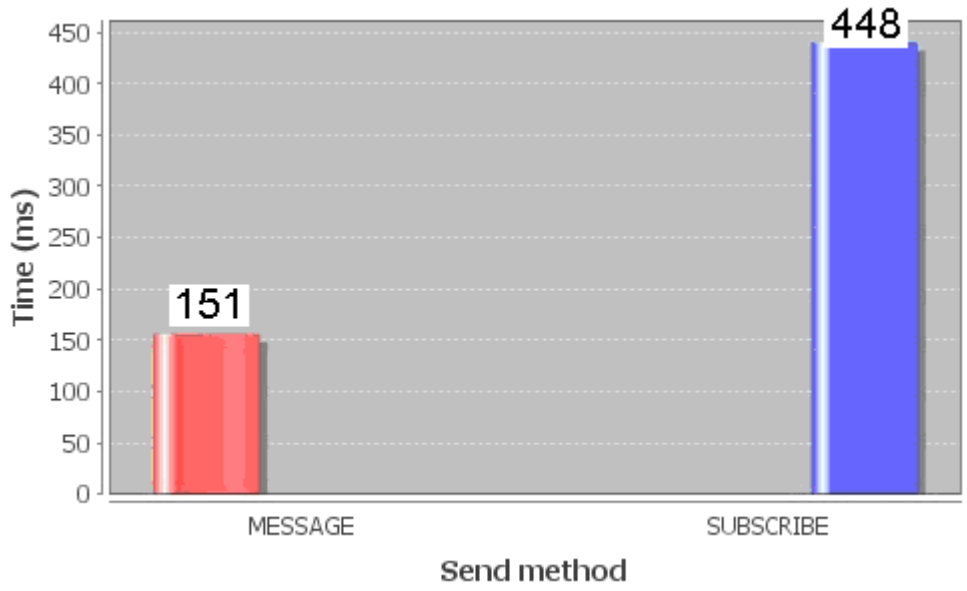

Figure 4-4: Time delay added by the MJCF API while sending a SIP MESSAGE or a SIP SUBSCRIBE

Another measurement made for the client side concerns the memory utilized by the API using. Memory is used when new messages are sent or received. Figure 4-5 illustrated the different phases of the application while messages are sent and received.

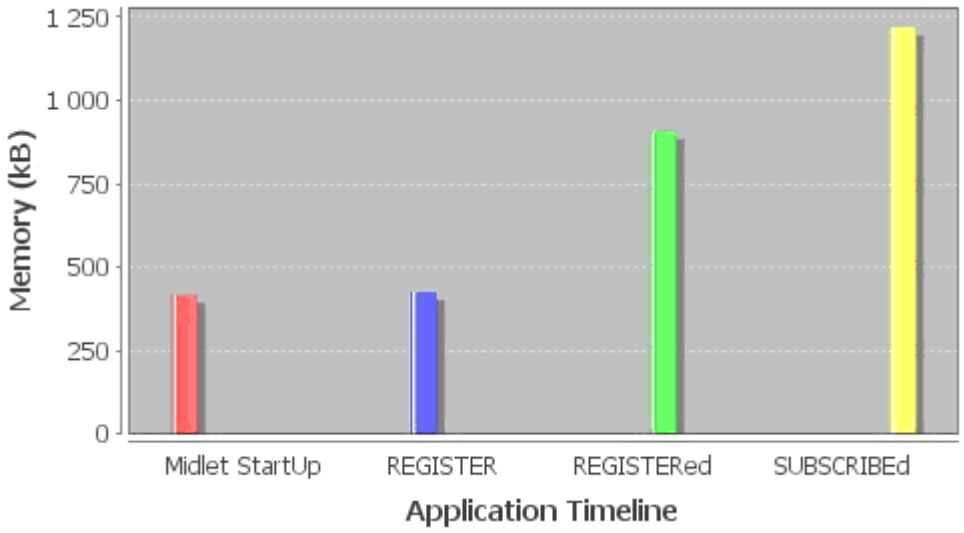

Figure 4-5: Memory consumption

At the very beginning the MIDlet starts. This action requires an initial amount of memory. Directly after that a SIP REGISTER message is sent. At that point we see that the MJCF API requires a minor amount of additional memory. When the response for the SIP REGISTER is received the memory utilization increased since in addition to the MJCF API, the LWUIT library is used. However, the LWUIT library has not yet

been tuned to save memory; hence these values might be a little bit misleading.

We conclude from these measurements that the MJCF API adds a significant delay when messages are sent, but it does not consume much additional memory. This is very important since mobile devices have limited memory. In addition to these delays the access technologies add delay that will negatively affect the user's experience.

6 Evaluation of the Server API and documentation

6.1 Inside the Server API

The MJCF server API is used on the top of a SIP Servlet. These interfaces are invoked only after the servlet is deployed. From a network perspective the server listens on a specific port (the default 5060) for incoming messages. It can also send messages to the clients. For this reason the API is structured as a client-server architecture. A basic introduction is given schematically in Figure 4-6.

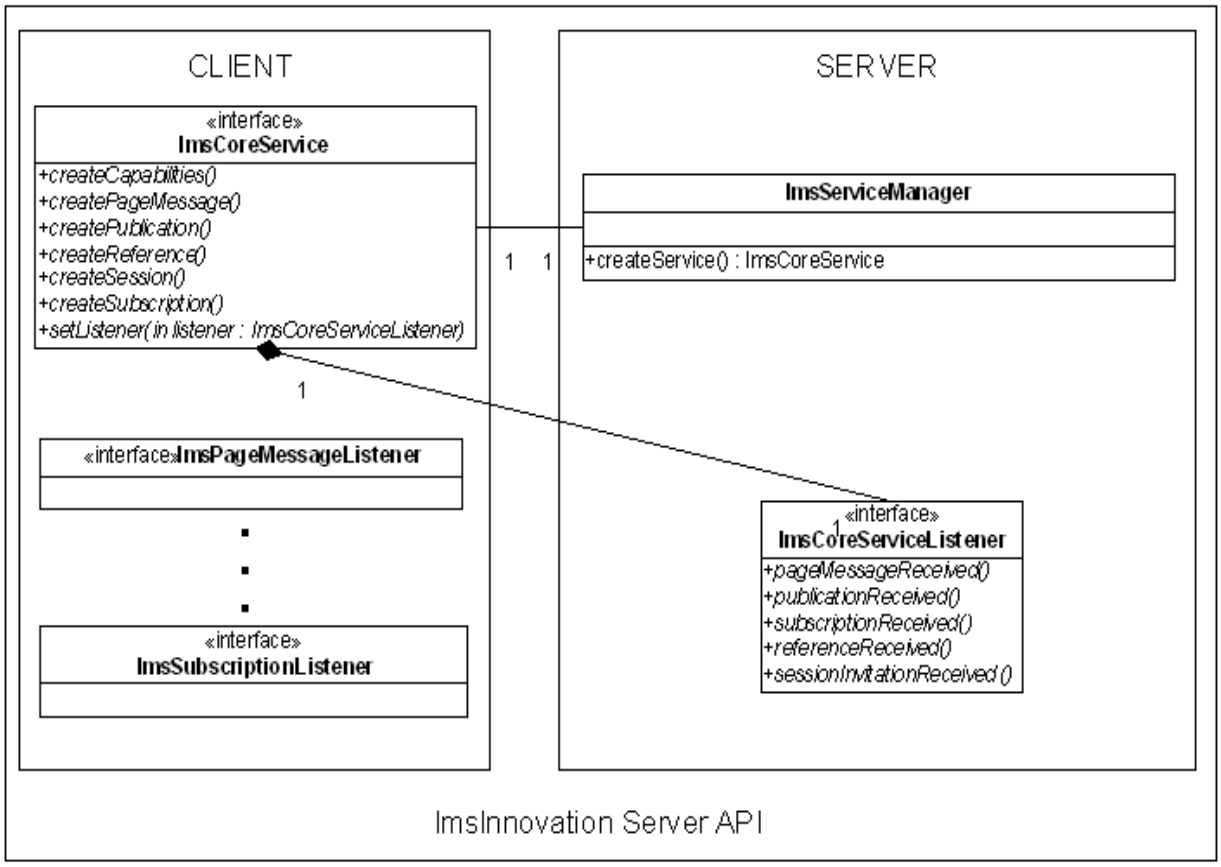

Figure 4-6: Inside the MJCF server API

The server part of the API starts by listening for IMS specific messages. After that it waits for incoming messages. When a message arrives it is parsed and the correct interface method is invoked. The developer must write code to handle potential incoming messages. The client part of the API is used through the server. The ImsServiceManager object instantiates an ImsCoreService object when it is constructed. The ImsCoreService object acts as “the factory” of the SIP messages that the developer wants to send.

Those two different parts are followed by the appropriate error listeners which are of great importance when an error occurs. The evaluation conducted in this chapter focuses basically on:

- For the Server part:
    - Check if the SIP MESSAGE, PUBLISH, SUBSCRIBE, and the responses are received correctly.
- For the client part:
    - Check if the SIP NOTIFY and MESSAGE are send correctly
    - Check if the 200 OK response is received or an appropriate SIP error message is received when an error occurs.

6.2 Evaluation overview

Generally speaking, the Server APIs and documentation are sufficient and very helpful for developers. By using them a developer can easily send simple messages and develop simple IMS applications. The messages needed to the demonstration application are all successfully sent. The Ericsson IMS developer [74] portal gives all the basic instructions for installing the tools and starting a simple application on a mobile device.

The documentation prepared by Ericsson are called “Servlet Creation Tutorial” [73] and “Deployment Tutorial” [73]. The first explains how a servlet can be created and the second how this servlet can run on a servlet container (the AS). The APIs can be downloaded from a SVN server [74]. The API evaluated in this report is stored under “Deliverables/trunk/ImsInnovationFramework/Server/binaries/” and the java documentation for this API is stored under
“Deliverables/trunk/ImsInnovationFramework/Server/documentation/”.

The detailed advantages and disadvantages regarding the APIs used in this thesis and the documentation will be covered in the next chapters.

6.2.1 Advantages

The evaluation is divided in two different parts. The first concerns the documentation and the second the APIs. Based on the evaluation criteria (see section 4.2) the advantages observed are given in section 6.2.1.16.2.1.2.

6.2.1.1 Documentation

The “Servlet Creation Tutorial” is a well organized, easy to read, and quite informative. It describes in details the steps that are needed to create a servlet by using the IMS API on the Eclipse platform. It starts by showing the simple steps to create a project, create a servlet, and set up the deployment descriptors.

The “Deployment Tutorial” documents how a developer can upload his/her application to the Ericsson Application Server. This is a common way of deploying applications on the Sailfin AS. This document is short and very informative and does not confuse the reader.

The Java documentation follows the standard approach. The comments and the description are quite good since they give the main idea of what each method does.

6.2.1.2 API

The advantages of the API can be summarized as:

- **Well-designed structure**. The structure of the API is based on JSR 289 [45] and follows its architecture. This is very important because it follows a general way of building APIs that is very helpful for the experienced and inexperienced programmers. The functionalities are divided into different packages that are easily understandable by the programmers and easily accessible through interfaces.
- **High abstraction level**. It keeps the access to the lower layers but it provides abstractions only for sending simple messages and using media communication. It supports also the presence functionality. This is important when it comes to creating simple applications. The abstract interfaces can be used fast and easy while a way of accessing difficult and advanced aspects of the IMS functionalities is available, by using core objects.

6.2.2 Disadvantages

As before we explain the disadvantages split between the documentation and the API.

6.2.2.1 Documentation

While the “Servlet Creation Tutorial” is following a professional approach in explaining the different functions, however it does not give the developer the most important guidance. The MJCF server API extends the Java SIP Servlet (javax.servlet.sip.SipServlet). This is very confusing, if the developer does not understand

this from the beginning. When these APIs are used, a new SIP-servlet should not be used. If a second SIP servlet is used then inter-servlet communication needs to be established.

6.2.2.2 API

The disadvantage of the API is primarily the absence of an:

- **ImsNotificationListener.** This listener would be useful when a SIP NOTIFY message is sent, but a 200 OK is never received in confirmation (or an error happens).

6.3 Delay & Memory measurements

The server is responsible for receiving requests from different clients and handling them based on the logic of the application. Additionally the server can send messages to the users. These functions make use of the MJCF API since the messages are sent over the IMS network. Measurements have been made to determine the impact of the APIs on the memory consumption of a service.

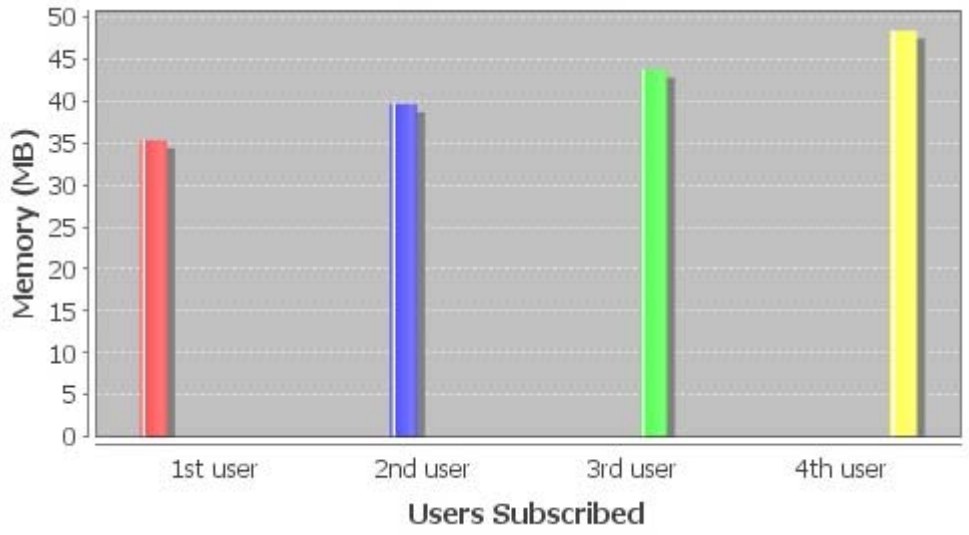

Figure 4-7: Memory consumption while new users connect to the application

Figure 4-7 presents the memory occupied when users connected to the server. In the Super-Discounts application this is done by receiving SIP SUBSCRIBE messages from the clients. We notice that when a new client connects to the server, approximately 5MB of additional memory are needed.

Considering that this application is a demonstration of a grater one, we conclude that the memory consumption is extended and may cause the need of many additional memory cards while new users are added to the system.

We have to take under consideration that an AS is designed to host more than one application at a time. Based on this fact we can easily conclude that the amount of memory consumption on a real AS will be large and will probably make the system cumbersome and slow.

7 Conclusions and future work

This thesis project used the new MJCF APIs delivered publicly by Ericsson in order to develop an innovative application which operates over an IMS network. The application is based on a client-server logic. The client would need to be installed on every Java enabled mobile device which can connect to an IMS network infrastructure. The server is located on the operator side; while the operator's IMS system hosts the implementation and the logic of the application. The server can utilize databases and interaction with other systems such as location brokers, charging servers, media servers, etc.

The application developed based on these APIs, exploits the idea of discounts by moving them from normal everyday life to our mobile phones. Messages are sent over the IMS network transparently to the user. The user experiences a normal internet application without caring about the access technology that is used.

Based on this application an evaluation of the APIs was conducted while development. Bugs were found at each step during and corrections were made to the APIs when needed. The APIs also have related by tutorials and Java API documentation. These documents were evaluated and changes were proposed in order to improve them.

The evaluation showed that the APIs were well designed and very abstract for the developers, making them easy to use and understand. With a few lines of code any kind of SIP message can be sent and any media can be delivered. The documentation was well formed and easy to read and follow. Of course continuous changes and amendments are needed. During the evaluation, bugs were found and many will still exist. New applications that use other functionality will be developed and will expose new short comings of the code and documentation.

The measurements showed that simple applications can be easily developed using the MJCF API. The memory consumption on the server side can be very large for applications hosting many clients, but most importantly the memory consumption on the client side is low as the client memory program is restricted and not extensible. The delay on the other hand seemed to be too high for such a simple application.

This project adds just a small brick in the wall of the IMS application development. Future work should be done and more measurements based on these new APIs must be conducted. This future work should include:

- A complete study with comparisons of delay and memory consumptions between the different phases of the APIs' abstraction. A better understanding of how those APIs can affect the development of larger and more complex applications.
- This demo application can be extended by adding functionality that uses presence information and enables session establishments. This will reveal more bugs and the need for more abstract functionalities.
- Testing with a big number of real devices should be made in order to mitigate the delays and the memory consumption on the server side.
- Since the server needs to manage many clients a more abstract API can be designed to produce a more general solution.

References

[1] "4G", http://en.wikipedia.org/wiki/4G, (23/02/09)

[2] Jagdeep Walia, "IMS, Enabling new, attractive convenient user services", power point, Ericsson, (23/02/09)

[3] J. Rosenberg et al, "SIP: Session Initiation Protocol", IETF, RFC 3261, June 2002, http://www.ietf.org/rfc/rfc3261.txt, (25/02/09)

[4] R. Fielding et al, "Hyertext Transfer Protocol – HTTP/1.1", IETF, RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt, (25/02/09)

[5] "Global System for Mobile communications GSM", http://en.wikipedia.org/wiki/GSM, (25/02/09)

[6] "Public Switched Telephone Network PSTN", http://en.wikipedia.org/wiki/Public_switched_telephone_network, wikipedia, (25/02/09)

[7] Overview of 3GPP Release 4, Summary of all Release 4 Features, v.1.1.0 (draft) ETSI Mobile Competence Centre 2004

[8] Summary of all Release 5 Features, ETSI Mobile Competence Centre, Version 9th September 2003

[9] 3GPP TS 23.060, "General Packet Radio Service GPRS", http://www.3gpp.org/ftp/Specs/html-info/23060.htm, (25/02/09)

[10] 3GPP releases overview, http://en.wikipedia.org/wiki/3GPP , (23/02/09)

[11] "3GPP specification series 25 - Radio aspects of 3G, including UMTS", http://www.3gpp.org/ftp/Specs/html-info/25-series.htm , (23/02/09)

[12] "IEEE 802.11 Wireless Local Area Networks", http://www.ieee802.org/11/, (23/02/09)

[13] Radvision, "IMS SIP and Signaling, The Radvision perspective – A technology overview", Radvision, Technical Report, 2006.

[14] R. Price et al, "Signaling Compression (SigComp)", IETF, RFC 3320, January 2003, http://www.ietf.org/rfc/rfc3320.txt, (25/02/09)

[15] M. Garcia-Martin, E. Henrikson, and D. Mills, "Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd Generation Partnership Project (3GPP)", IETF RFC 3455, January 2003, http://www.ietf.org/rfc/rfc3455.txt, (25/02/09)

[16] C. Jennings, J. Peterson, and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks ", IETF, RFC 3325, November 2002, http://www.ietf.org/rfc/rfc3325.txt (25/02/09)

[17] J. Arkko et al, "Security Mechanism Agreement for the Session Initiation Protocol (SIP)", IETF, RFC 3329, January 2003, http://www.ietf.org/rfc/rfc3329.txt, (25/02/09)

[18] A. Niemi, J. Arkko, and V. Torvinen, "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)", IETF, RFC 3310, September 2002, http://www.ietf.org/rfc/rfc3310.txt, (25/02/09)

[19] W. Marshall, "Private Session Initiation Protocol (SIP) Extensions for Media Authorization", IETF, RFC 3313, January 2003, http://www.apps.ietf.org/rfc/rfc3313.html, (25/02/09)

[20] J. Rosenberg, "A Session Initiation protocol (SIP) Event Package for Registrations", IETF, RFC 3680, March 2004, http://www.ietf.org/rfc/rfc3680.txt, (25/02/09)

[21] G. Camarillo and P. Kyzivat, "Update to the Session Initiation Protocol (SIP) Preconditions Framework", IETF, RFC 4032, March 2005, http://www.ietf.org/rfc/rfc4032.txt, (25/02/09)

[22] G. Camarillo, W. Marshall, and J. Rosenberg, “Integration of Resource Management and Session Initiation protocol (SIP)”, IETF, RFC 3312, October 2002, http://www.ietf.org/rfc/rfc3312.txt,

[23] M. Handley and V. Jaconson, “SDP: Session description Protocol”, IETF, RFC 2327, April 1998, http://www.ietf.org/rfc/rfc2327.txt, (25/02/09)

[24] Sorin Georgescu, “CSNC07 IMS: An architecture for convergent Next Generation Multimedia Services. Research and standardization challenges”, 2007, *www.iaria.org/conferences2007/filesICSNC07/IMSConvergentMultimediaServices.ppt* ,Ericsson presentation, (23/02/09)

[25] “Media Gateway Control Protocol (Megaco)”, http://en.wikipedia.org/wiki/Megaco, (25/02/09)

[26] P .Calhoun et al, “Diameter Base Protocol”, IETF, RFC 3588, September 2003, http://tools.ietf.org/html/rfc3588, (25/02/09)

[27] 3GPP TS 29.229, “Cx and Dx interfaces based on the Diameter protocol; Protocol Details”, http://www.3gpp.org/ftp/Specs/html-info/29229.htm, (25/02/09)

[28] H. Schulzrinne et al, RFC 3550, “RTP: A Transport Protocol for Real-Time Applications”, July 2003, http://www.ietf.org/rfc/rfc3550.txt, (25/02/09)

[29] H. Sghulzrinne, A. Rao, and R. Lanphier, “Real Time Streaming Protocol (RTSP)”, IETF, RFC 2326, April 1998, http://www.ietf.org/rfc/rfc2326.txt, (25/02/09)

[30] 3GPP TS 22.141, “Presence Service Stage 1”, http://www.3gpp.org/ftp/Specs/html-info/22141.htm, (25/02/09)

[31] 3GPP TS 23.141, “Presence Service Architecture and Functional Description Stage 2”, http://www.3gpp.org/ftp/Specs/html-info/23141.htm, (25/02/09)

[32] 3GPP TS 24.141, “Presence Service using the IP Multimedia (IM) Core Network (CN) subsystem Stage 3”, http://www.3gpp.org/ftp/Specs/html-info/24141.htm, (25/02/09)

[33] 3GPP TS 22.340, “IP Multimedia Subsystem (IMS) messaging; Stage 1”, http://www.3gpp.org/ftp/Specs/html-info/22340.htm, (25/02/09)

[34] 3GPP TS 24.247, “Messaging service using the IP Multimedia Core Network subsystem; Stage 3”, http://www.3gpp.org/ftp/Specs/html-info/24247.htm, (25/02/09)

[35] 3GPP TS 23.979, “3GPP enablers for Open Mobile Alliance (OMA) Push-to-talk over Cellular (PoC) services; Stage 2”, http://www.3gpp.org/ftp/Specs/html-info/23979.htm, (25/02/09)

[36] 3GPP TS 22.173, “IP Multimedia Core Network subsystem Multimedia Technology Service and supplementary services; Stage 1”, http://www.3gpp.org/ftp/Specs/html-info/22173.htm, (25/02/09)

[37] 3GPP TS 24.173, “IP Multimedia telephony service and supplementary services; Stage 3”, http://www.3gpp.org/ftp/Specs/html-info/24173.htm, (25/02/09)

[38] ETSI TS 181 009, “Telecommunications and Internet Converged services and Protocols for Advanced Networking (TISPAN); Service and Capability Requirements”,

[39] 3GPP TS 22.226, “Global Text Telephony (GTT); Stage 1: Service Description”, http://www.3gpp.org/ftp/Specs/html-info/22226.htm, (25/02/09)

[40] JSR 032, “JAIN SIP API”, http://jcp.org/aboutJava/communityprocess/mrel/jsr032/index.html, (25/02/09)

[41] T. Bates, R. Chandra, and E. Chen, “BGP Route Reflection – An Alternative to Full Mesh IBGP”, IETF, RFC 2796, April 2000, http://www.ietf.org/rfc/rfc2796.txt, (25/02/09)

[42] J. Rosenberg and H. Schulzrinne, “Reliability of Provisional Responses in the Session Initiation Protocol (SIP)”, IETF, RFC 3262, June 2002, http://www.ietf.org/rfc/rfc3262.txt, (25/02/09)

[43] JSR 141, “SDP API”, http://jcp.org/aboutJava/communityprocess/pfd/jsr141/, (25/02/09)

[44] JSR 116, “SIP Servlet API”, http://jcp.org/en/jsr/detail?id=116, (25/02/09)

[45] JSR 289, “SIP Servlet v1.1”, http://jcp.org/en/jsr/detail?id=289 , (25/02/09)

[46] JSR 165, “SIMPLE Instant Messaging”, http://jcp.org/en/jsr/detail?id=165, (25/02/09)

[47] JSR 180, ”SIP API for J2ME”, http://jcp.org/en/jsr/detail?id=180, (25/02/09)

[48] JSR 22, ”JAIN SLEE API Specification”, http://jcp.org/en/jsr/detail?id=22, (25/02/09)

[49] JSR 240, ”JAIN SLEE (JSLEE) v1.1”, http://jcp.org/en/jsr/detail?id=240, (25/02/09)

[50] JSR 281, ”IMS Services API”, http://jcp.org/en/jsr/detail?id=281, (25/02/09)

[51] “The Parlay Group”, http://www.parlay.org, (25/02/09)

[52] “JSIP”, http://jsip.sourceforge.net/ , (23/02/09)

[53] Rogelio Martinez Perea, ”Internet Multimedia Communications Using SIP”, Morgan Kaufmann, ISBN 978-0-12-374300-8

[54] Mobicents, ”The Open Source SLEE and SIP Server”, http://www.mobicents.org/index.html, (23/02/09)

[55] “Jiplet Container – a container for SIP applciations”, http://www.cafesip.org/projects/jiplet/index.html, (23/02/09)

[56] “Opensips”, http://www.opensips.org/, (23/02/09)

[57] JSR 135, “Mobile Media API”, http://jcp.org/en/jsr/detail?id=135, (25/02/09)

[58] JSR 908, “Java Media Framework, version 2.0”, http://jcp.org/en/jsr/detail?id=908, (25/02/09)

[59] JSR 309, “Media Server Control API”, http://jcp.org/en/jsr/detail?id=309, (25/02/09)

[60] A.B. Roach, “Session Initiation Protocol (SIP) – Specific Event Notification”, IETF, RFC 3265, June 2002, http://www.ietf.org/rfc/rfc3265.txt, (25/02/09)

[61] A. Niemi, “Session Initiation Protocol (SIP) Extension for Event State Publication”, IETF, RFC 3903, October 2004, http://www.ietf.org/rfc/rfc3903.txt, (25/02/09)

[62] J. Rosenberg, “The Session Initiation Protocol (SIP) UPDATE Method”, IETF, RFC 3311, September 2002, http://www.ietf.org/rfc/rfc3311.txt, (25/02/09)

[63] H. Schulzrinne, D. Oran, and G. Camarillo, “The Reason Header Field for the Session Initiation Protocol (SIP)”, IETF, RFC 3326, December 2002, http://www.ietf.org/rfc/rfc3326.txt, (25/02/09)

[64] B. Campbell, et al., “Session Initiation Protocol (SIP) Extension for Instant Messaging”, IETF, RFC 3428, December 2002, http://www.ietf.org/rfc/rfc3428.txt, (25/02/09)

[65] R. Sparks, “The Session Initiation Protocol (SIP) REFER method”, IETF, RFC 3515, April 2003, http://www.ietf.org/rfc/rfc3515.txt, (25/02/09)

[66] “SipExchange Server”, http://www.cafesip.org/projects/sipexchange/architecture.html, (25/02/09)

[67] “Ericsson Mobility World Developer Program”, http://www.ericsson.com/mobilityworld/sub/home.html, (25/02/09)

[68] “SailFin”, https://sailfin.dev.java.net/, (25/02/09)

[69] “Lightweight UI Toolkit - LWUIT”, https://lwuit.dev.java.net/, (25/02/09)

[70] *SIL International* , “What is Evaluation”, http://www.silinternational.com/lingualinks/literacy/ReferenceMaterials/GlossaryOfLiteracyTerms/WhatIsEvaluation.htm, (25/02/09)

[71] “Ericsson Service Development Studio (SDS) – 4.1” , http://www.ericsson.com/developer/sub/open/technologies/ims_poc/tools/sds_40 , (23/02/09)

[72] “Sun Java Wireless Toolkit for CLDC”, http://java.sun.com/products/sjwtoolkit/, (23/02/09)

[73] "Tutorials and Guides", http://developer.labs.ericsson.net/apis/mjcf/tutorials-and-guides, pdf, Ericsson, (24/02/09)

[74] "MJCF Ericsson APIs", http://developer.labs.ericsson.net/apis/mjcf/downloads, Ericsson, (24/02/09)

[75] "List of SIP response codes", http://en.wikipedia.org/wiki/SIP_Responses, wikipedia, (25/02/09)

COS/CCS 2009-02

www.kth.se