NTNU
Innovation and Creativity

Launching an Innovative Mobile Multimedia Communication Application

using different platforms

Runar Gunnerud

Master of Science in Communication Technology

Submission date: June 2007

Supervisor: Peter Herrmann, ITEM
Co-supervisor: Hilde Lovett, Telenor R&I
Bjørn Remseth, Telenor R&I

Norwegian University of Science and Technology
Department of Telematics

Problem Description

At the moment operators around the world are, and will be, investing millions of dollars in the roll out of the IP Multimedia Subsystem (IMS). Although investment has already begun, a major roll out of IMS services is not likely to happen until the middle 2008. Alternative disruptive technologies might evolve in that time. Time to market is a critical factor when developing new services in the telecom sector. Therefore, it would be interesting to investigate, from a developer's perspective, the status quo on how such feature rich communication services can be developed.

The student will develop a communication application, which will use simple services such as message transfer, file transfer and an instant messaging session. This application will then be launched in two ways, one using IMS platform and another using a alternative platform chosen by the student. The thesis will then compare the two experiences when developing and launching the application. This comparison will be evaluated on a number of criteria that need to be addressed in the thesis. Advantages of both platforms will be discovered and compared.

This is a real life experiment that will give hands on experience in developing an innovative mobile communication application that makes use of a number of basic features (e.g. messaging, file transfer and IM), and important knowledge on how to build an application that makes use of IMS service enablers.

If time is available the application can be extended using presence and/or allowing a multi participant session.

Assignment given: 24. January 2007
Supervisor: Peter Herrmann, ITEM

Abstract

The IP Multimedia Subsystem (IMS) is intended by the Telco industry, to make it easy for third-party application developers to create new, innovative services that will help to offset the fall in revenue of regular voice services. However, a slow roll out of the system is increasing the chance of a disruptive technology to fill some of the space that IMS hopes to cover. This thesis presents a hands on example of the implementation of such a new innovative service.

XMPP has been used as an alternative platform to launch the service, and is thoroughly compared to the IMS in this master's thesis. Ironically the service could not be launched on IMS due to technical problems. Results suggest that XMPP could replace IMS as a service platform, thus disrupting the business model of IMS.

Preface

This master's thesis was carried out at Telenors space-ship premises on Fornebu during the spring of 2007. It documents the achievements of the 10th semester of the Master of Science study in Communications Technology at the Norwegian University of Science and Technology (NTNU).

I would like to thank my supervisors Hilde Lovett and Bjørn Remseth at Telenor Research & Innovation for endless inspiration and advice, both *on* subject and completely *off* subject. In addition I want thank the rest of Telenor R&I for giving me opportunity to deepen my knowledge of the Telco industry.

Last I want to thank my parents for bearing with my restless lifestyle, my siblings for being role models in both your extreme ways and all my friends for making life so fantastic. A special thank to the Prince of Morgantown and the Mancunian President for correcting my many errors.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3GPP	Third Generation Partnership Project
ADSL	Asymmetric Digital Subscriber Line
AoR	Address of Records
AS	Application Server
CDMA	Code Division Multiple Access
CS	Circuit Switched
CSCF	Call Session Control Function
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
ETSI	European Telecommunications Standard Institute
GSM	Global System for Mobile communication
GUI	Graphical User Interface
HLR	Home Location Register
HSS	Home Subscriber Server
HTML	Hyper Text Mark-up Language
IDE	Integrated Development Environment
IM	Instant Messaging
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IRC	Internet Relay Chat
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
J2ME	Java 2 Micro Edition
JCP	Java Community Process
JSR	Java Specification Request
MOM	Message-Oriented Middleware
MRF	Media Resource Function
OMA	Open Mobile Alliance
PDA	Personal Digital Assistant
PoC	Push to talk Over Cellular
PSTN	Public Switched Telephone Network
PUA	Presence User Agent
RFC	Request for Comments
RTCP	RTP Control Protocol
SASL	Simple Authentication and Security Layer protocol
SCTP	Stream Control Transmission Protocol
SDK	Software Development Kit
SDP	Session Description Protocol

SIP	Session Initiation Protocol
SLF	Subscription Locator Function
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
TISPAN	The Telecoms and Internet converged Services and Protocols for Advanced Networks
TLS	Transport Layer Security
UA	User Agent
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
URI	Universal Resource Identifier
VoIP	Voice over IP
WCDMA	Wideband Code Division Multiple Access
WLAN	Wireless Local Area Network
WTK	Wireless Tool Kit
XML	eXtensible Mark-up Language
XMPP	eXtensible Message and Presence Protocol

Chapter 1

Introduction

Contents

1.1 Motivation

While mobile data services are still a small part of people's everyday lives, they are slowly starting to emerge and should one day be ubiquitous. Operators around the world are expecting revenues from new rich featured services to compensate for the falling revenues from voice services. The standardization bodies of the next generation mobile networks have chosen to use the Internet Protocol (IP) as a backbone of the architecture for the evolution of the cellular network. So that voice and video, as well as other information, will be transferred solely in packages.

According to Weilenmann (Weilenmann 2003) it is important to define mobility beyond being just related to work. Mobile technology is now a widespread phenomenon that is so well integrated with everyday life that treating the issue right could have a direct impact on people's lives, as could the opposite detrimental.

The IP Multimedia Subsystem (IMS) is the name of this architecture. As Camarillo and García-Marin state in their book (Camarillo & García-Martin 2006):

> Third Generation (3G) networks aim to merge two of the most successful paradigms in communications: cellular networks and the Internet. The IP Multimedia Subsystem (IMS) is the key element in the 3G architecture that makes it possible to provide ubiquitous cellular access to all the services that the Internet provides. Picture yourself accessing your favourite web pages, reading you email, watching a movie or taking part in a videoconference wherever you are by simply pulling a 3G hand-help device out of you pocket. This is the IMS vision.

IMS is not just new services; it is a whole system that is intended to make the next generation mobile network more efficient, thus cutting costs. It will provide the operators with a greater control of the mobile Internet, including charging and quality of service (QoS).

1.1.1 Developers, developers, developers...

One of the Internet's greatest features and possibly an explanation to its wide adoption and richness in services is the use of open standards that make any developer capable of creating a service. In a new way, and in contrast to the Telco Industry, *the users* have a much greater influence on which services that will survive. Recent history has even seen users directly included in the upgrading and evolution of Internet services through invitations to discussion forums and other feedback mechanism.

1.1.2 New game - new rules

So far developing applications and services for the mobile domain has not been a simple task. The few Application Programming Interfaces (APIs) that Telco's have made public have been effectively unavailable even if they are according to international standards (Yates 2004). As Thomas Magedaz, establisher of the FOKUS Open IMS playground (FOKUS June 2007), states in his inquiring article (Magedanz 2006), it would be fatal to believe that with IMS (instead of IN and OSA/Parlay as in the past) the network operators can develop some kind of multimedia services supermarket with dedicated partners in a walled garden approach. The Internet is flourishing with APIs for all kind of services. And while the IMS is intended to ease the development of new innovative services and applications, one should ask if it is possible to develop the same services without integration with the operator. For any operator, like Telenor, this is a critical question, a question that is asked when starting the work of this thesis.

1.2 Objectives

The main objective of this master's thesis is to:

> investigate if there is a faster and better way of delivering the same innovative services to the mobile handset that IMS intends to do by using a different approach and avoiding interaction with the operators platform. To verity this hypothesis, a service will be implemented in this alternative manner.

The thesis will be accomplished though the following steps:

- Finding an alternative platform to launch the service
- Defining criteria for comparison and a model application to be launched
- Develop, implement and launch the application using both technologies
- Analyse the development experience and results from launch

1.3 Scope and limitations

As mentioned above the IMS is an enormous system that provides functionality to many different levels of the telecom domain. This thesis is not meant to test whether IMS should or should not be deployed, because that will be a much larger question, which will include a lot of economical analysis as well as technical.

Choosing what other platform to use will be based on personal experience in the field as well as guidance from the thesis supervisors at Telenor R&I. I will not compare a wide range of different technologies, but rather argue why the chosen one is a good alternative, as a general comparison of many alternatives would be out of the scope of this thesis.

Many of the services IMS is claming to provide, include interaction between users, either with or without noticeable interaction by a central server. These applications could be something as simple as a voice call or an SMS, but they could also be integrated services that make use of a number of features, like messaging and multimedia. Although IMS promises that it should be easy for programmers to develop feature rich applications by using the system's many service enablers, (Gunnerud 2006) shows that by today this is still no easy task. Moreover most applications programmed towards the IMS system today are either stand alone applications or proprietary IMS Clients, none are built on top of a general IMS Client Framework.

Services like video calls and video streaming will most likely be controlled by the operators due to it's high demand of bandwidth, at least with the current per usage pricing. This might change as prices drop, price schemes change and better codec's appear, but for now video is too expensive to pay for on a per KB

basis. Other multimedia material like images and short sound clips are small enough in size to be affordable in a pay per byte price scheme.

So when speaking of innovative services in this thesis, we mean new smart ways of using, sharing and communicating information and multimedia, but thus excluding video due to the fact that cost would not level the comparison. Many of the services we have seen evolve online do not necessarily require use of high bandwidth by the client, as digg.com, twitter.com, facebook.com.

1.3.1 Case study limitations

Hands on experience with developing a mobile application are of great interest when comparing two different ways of deploying a service, one using the operator while the other avoiding it. Since implementation can be an extremely time consuming task and, only a simple service will be possible to deploy because of time constraints. As explained in chapter 3, the IMS contains application servers (AS) that will provide services to the system, one example could be a game server. Due to the time available the implementation will not include using any of these servers. The implementation will only handle a client-to-client experience, although a possible extension will be described in detail.

When defining what is an innovative service, presence would be a natural part of this definition. The IMS used for testing does unfortunately not have a presence server integrated with the system, therefore presence will not be set as a requirement that the case study needs to perform.

1.3.2 Physical accident

The facilities for putting the application into action were to be provided by Telenor R&I, including the IMS platform and the alternative. The IMS is situated at the laboratory of the Program for Advanced Telecom Services (PATS), at R&Is premises in Trondheim. The reader should note that; after implementing the application an launching it on the alternative technology, it was time for launching it using IMS. Just after starting the work with IMS, some renovation work at the PATS lab in Trondheim manage to vibrationally damage the motherboard of the IMS server, putting the IMS server out of function for a month. This affected the thesis greatly by not being able to actually launch the application using IMS. The issue is thoroughly debated in the discussion chapter, chapter 6.

1.4 Related work

The issue of the future mobile Internet is almost a matter of faith, where the two religions are getting closer and closer to a golden mean. Those who believe the Internet world will enter the mobile domain, and the other way around, those who believe the mobile world will embrace the Internet. This can seem like the same thing, but viewing the situation from the Telco and the Internet

standpoint would make it look a lot different. The Telco industry, in Network Equipment Manufactures and other actors, are continuing to research concepts for the evolution beyond 3G networks (Uskela 2003), upholding its steady belief that the telecom world will continue its nice and slow evolution. On the contrary, Internet monsters like Microsoft, Google and Yahoo are entering the mobile world making more and more of their services available for the mobile domain.

The 3GPP is still defining IMS and while the system is starting to be implemented on the network side of many operators, there has not been much research material, concerning development of services on the platform, released. The 2004 Moriana Group report estimated what they call the Telecom Web Services Market to reach a value of 25.000 M EUR in 2007 (Copeland 2006, p.29). However, we are still far from seeing any such numbers coming through this year. As (Gunnerud 2006) states there is still a long way to go on the client side of next generation system.

1.5 Approach

It would be far fetched to try to compare the whole of IMS to another system; in reality this is the job 3GPP is doing. However, when trying a different approach to launching a service, this should be comparable to the approach in IMS. While a comparison framework for software development platforms (SDP) were hard to find, one could consider parts of IMS as a launching platform, or almost a middleware infrastructure. Taking the differences into account, the comparison framework for middleware infrastructures developed by Apostolos Zarras (Zarras 2004) will be used as a base when comparing the two technologies. At least on the higher general level, because deeper down in the framework the differences would be to big.

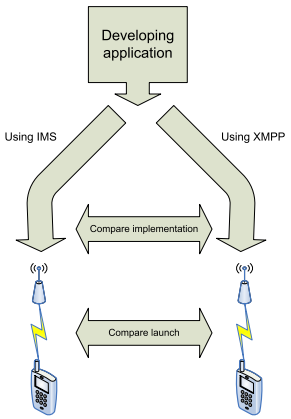


Figure 1.1: Approach process

This thesis will be a combination of a comparison analysis (where comparing parts of IMS and a different approach to solving a specific solution) and a case

study of the development, implementations and launch of the application. The first part of the report will be dedicated to a theoretical approach to the two alternatives, almost a mapping of the alternative technology compared to IMS influenced by the comparison framework of Zarras (Zarras 2004), while the last will be a testing of the technology in practice.

The service will be built with rapid service development process in mind, not with a tight software development process focus. It will be based on the least common denominator of what is believed to be the norms of a simple future communication application.

1.6 Project outline

This section gives a short introduction to what the different chapters include:

Chapter 1	**Introduction** This chapter, introducing the motivation for the thesis. Defining its objective and how it will be reached.
Chapter 2	**Definitions** Defining the criteria for an alternative technology and finding that technology. Also defining what an *an innovative mobile multimedia communication application* is, and introducing a case study application that will fit this definition.
Chapter 3	**IMS Theory** Gives a detailed introduction to the IMS as a whole, with special focus on elements related to service deployment in IMS.
Chapter 4	**XMPP Theory and Comparison** Gives a thorough understanding of the XMPP, how it relates to service deployment and to IMS.
Chapter 5	**ThinkAlike Application** This case study chapter explains how the two technologies are used to launch an innovative service called ThinkAlike. Both design and implementation are shown in detail.
Chapter 6	**Discussion** The results from the case study are in combination with the theory used to discuss the findings in this chapter.
Chapter 7	**Conclusion** Cuncludes the thesis and reflects on the work that has been done. Also suggests future work.

Table 1.1: Project outline

Chapter 2

Definitions

Contents

2.1 Selecting an alternative

When choosing an alternative to IMS for launching a service, it was important to focus on a technology that is new but still has momentum. It needs to provide much of the same possibilities as IMS on service deployment. It also needs to provide enough functionality to build a new innovative application as defined in section 2.2. Key requirements, based on Zarras (Zarras 2004), that will be used for comparison is:

Openness	The technology be able to extending the applications/services using it in various ways, like adding, removing, upgrading, composing services, etc. The openness is split into two parts, one regarding upgrades and addition, and one concerning adding new services.
Scalability	The technology should facilitate the effective operation of the applications/services at many different scales.
Performance	The technology should enable the efficient and predictable, if needed, execution of the applications/services that are using it.

Table 2.1: Platform requirements

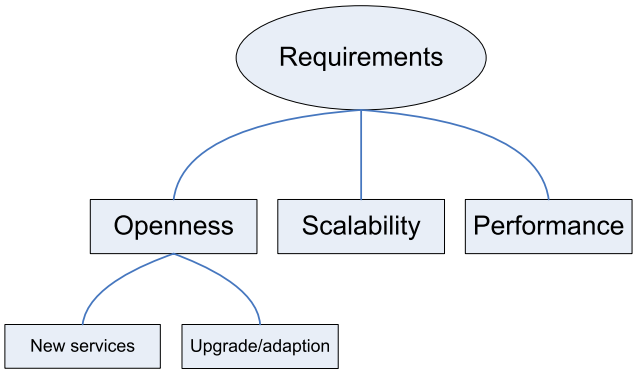


Figure 2.1: Requirement graph

2.1.1 The winner is

We chose to use a technology developed by IETF called the eXtensible Message and Presence Protocol (XMPP). While its architecture and structure are very simple, its XML nature makes it easy to understand for humans, yet powerful in use by computers. Formerly known as Jabber, this technology has been renamed XMPP and put under control of IETF. The technology got a hefty momentum when Google in January 2006 chose to use it for their new instant messaging service, GTalk.

XMPP is an open standard with a large support group in the Jabber Community. It has evolved from an instant messaging and presence protocol but is now a standard way to exchange real time information. It has evolved from the Jabber open source movement and it's openness makes it accessible to a wide range of developers. As described later in this chapter, due to its distributed architecture, not relying on single entities, it scales well. The platform does not provide any QoS beyond what is offered by TCP[1].

2.2 Application requirements

This section will define what is considered a new innovative service. Although it is impossible to foresee what a typical future innovative communication application would be like both camps [2] agrees on that we will see a new type applications evolve. Applications that have a much higher density of communication, both related to high-attention activates between individuals or groups as well as a much higher degree of information exchange between applications or

[1] To XMPPs defence Martin Geddes, a Telco sceptic, states in (Geddes 2006) that "QoS is not needed, wanted nor working".

[2] Meaning the mobile related actors and the Internet actors.

machines. The first part we have seen evolve in multiplayer Internet games and an intense use of IM by the younger generation, but according to The Economist (TheEconomist April 28th, 2006, Special report on a world of connections) the latter part, called Machine To Machine (M2M), is a sector that is growing very rapidly.

The M2M industry is extremely interesting, and that the XMPP would fit it well, but this thesis is focusing on applications and services to be used by customers on mobile phones.

- Message exchange between the applications
- Use of multimedia
- Chat session between users

2.2.1 Do we ThinkAlike?

As the development and implementation of the case study application would be a major task of this thesis, it was important to make it an interesting piece of software, and not solely an application that fits the definition of an innovative service. Inspired by a talk from the Carnegie Mellon University assistant professor Luis von Ahn, about using humans for computation, I came up with an idea about a simple piece of communication software that fill a desire that friends have, to know that they think alike. The application is named ThinkAlike.

The service is intended to be used between friends that have a close relationship and is meant to put a smile on both faces. If a person sees something that she thinks both her and her friend has a reference to, she would take a photo with her mobile phone, adding the word that she associates with the picture and send a challenge to her friend. An example could be a dog looking like one of their common friends. When the friend receives the request he would just see the picture, not the associated word, and would then try to guess what his friend is thinking. After the guesses a chat session starts where the participating parts can have a little chat related to image and result of the guessing.

The application and the implementation is thoroughly explained in Chapter 5.

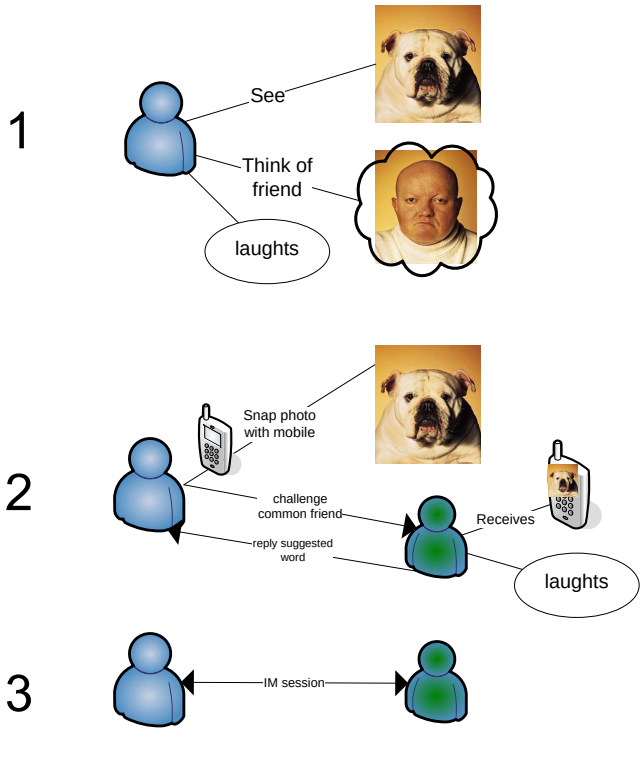


Figure 2.2: ThinkAlike process

Chapter 3

IMS Theory

Contents

3.1 Introduction

The IP Multimedia Subsystem (IMS) is a system first specified by the Third Generation Partnership Project (3GPP) for delivering Voice over IP (VoIP) and

other rich multimedia services. It is mainly based on the specification of the Session Initiation Protocol (SIP) but make use of a wider range of protocols, most of them developed by the IETF. Later IMS was embraced by ETSI/TISPAN as a solution to the fixed/mobile conversion problem, due to it's fundamental function of being network agnostic, able to run on multiple access types - including WCDMA, CDMA2000, GSM and WLAN, in the last case, often making use of a Asymmetric Digital Subscriber Line (ADSL). Another strength of IMS, in addition to being access-independent, is that it ensures continues interworking with legacy systems, both fixed and mobile, e.g. PSTN and GSM.

This chapter will make the reader familiar with the concept of the IMS, it's architecture and services. To understand many of the concepts in IMS it is critical to have a good understanding of the session protocol that 3GPP chose to handle sessions in IMS, namely SIP. Of that reason this chapter will first introduce SIP, before giving an introduction to the technical history of IMS and later a more in depth description of its architecture, then finishing with an explanation of defined services in IMS.

3.2 SIP

The Session Initiation Protocol has been developed by IETF as a application layer protocol for creation and management of multimedia communication session between actors in a network, often the Internet.

3.2.1 History of SIP

It originates from the early work related to distribution of multimedia content, through IETF multicast backbone. SIP was first defined as RFC2543 in 1999, but was later extended to what is the current version of SIP in RFC3261. It has grown to be a popular protocol much due to its simplicity and extensibility, and is in wide spread use in VoIP services on the Internet.

SIP is based on the same request/response scheme as HTTP and SMTP. Designed to be agnostic of the transport protocol used, thus running on both reliable and unreliable connections, respectively TCP and UDP. One of its goals is to promote mobility, meaning that a session indented for you will reach you where you are, in the way that you register your location on a central server. By separating the signalling and the media description it makes it possible to ad new applications or media separately, which is some of its extensible nature.

A initiated session is described by the Session Description Protocol (SPD), which will be introduced later in the chapter. SIP and SDP can be placed in the protocol stack as shown in 3.1.

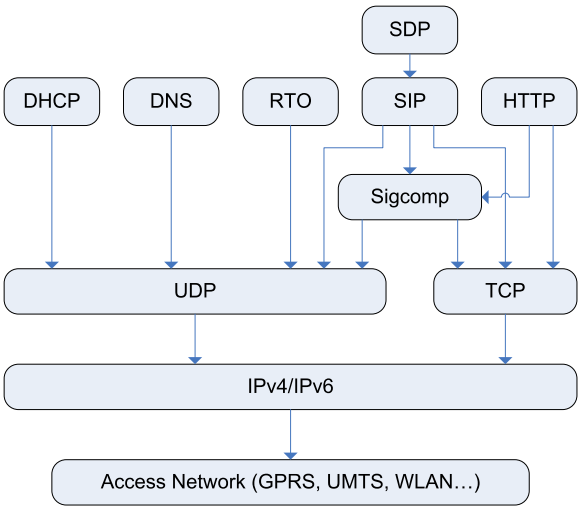


Figure 3.1: SIP in the protocol stack

3.2.2 Architecture

The two main elements of the SIP architecture are the SIP-servers and the User Agents (UA). In many cases it would be ideal if the UAs were able to connect with each other without the need of interference of intermediaries, but in some domains, especially in telecom, it is necessary to keep track of user, both for statistics, administration purposes and potential charging. Figure 3.2 sketches a normal network setup of a SIP system, the details of the figure will be described later in the section.

User Agents

An endpoint in SIP, the *User Agents*, is usually handled by a person, but can also operate on its own as e.g. a voice mail service. Usually a UA initiates a session to communicate with another UA. The implementation of the User Agent is not standardised, just its behaviour. Because of this, one find a range of different types of User Agents, from software applications running on a desktop or a PDA, standard looking landline phones that are in fact a SIP User Agent and lately even built in some high end featured phones. These phones have bare SIP-capabilities, in contrast to the IMS version of SIP explained later. This is enabling a pre-IMS domain for SIP on mobile, a thing that could be a big disruption for IMS. The Nokia N91 used in the case study of this thesis have a SIP User Agent preinstalled

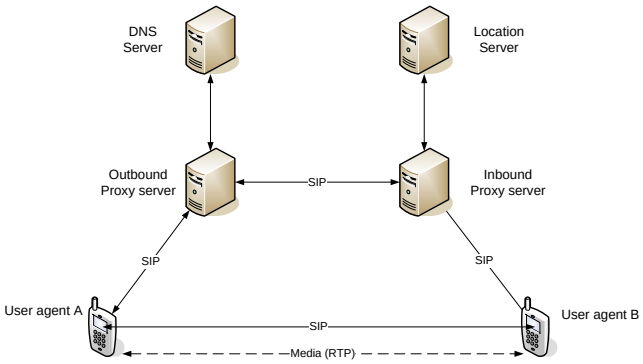


Figure 3.2: SIP Architecture

Identification

In 3.2 User Agent A want to initiate a session with User Agent B. All she needs to know is the Universal Resource Identifier (URI) of the other person, which has the same format as an email address: user@domain. An example of a SIP contact address would be `sip:freddy.baksen@telesyd.no`. If one wishes to use a secure connection over TLS one can use `sips:freddy.baksen@telesyd.no`. A SIP URI may also contain a number of parameters following the domain and a semicolon; this could be what transport protocol to use or additional information passed with the message (usually for routing purposes). In this way a SIP URI could look like this: `sip:freddy.baksen@telesyd.no;transport=tcp`. Another way of identifying a UA is byt using a `telURI`, a SIP URI that uses a regular phone number before the @ in the URI. This is naturally very handy for IMS as it needs to be interworking with legacy CS phone systems.

Registration

One of the important SIP-servers are the *Registrar*, which keeps track of where the users are located, and keeps the binding between a users URI and the address of the host where the User Agent is currently located. For a User Agent to be recognised on the network and be able to send and receives messages it needs to register with the Registrar Server. It is important to realize that SIP allows a user to register more than one User Agent, e.g. both a SIP enabled desk phone and a PDA. They can both be reached simultaneously with the same URI user@domain, but the user have the option of programming where the registrar should route the request based on many parameters, like time, who is calling, the calendar of the user and so on.

Location

The only thing needed when sending a message to a user is the URI of the intended receiver, not the physical location. Personal mobility is one of the key features of SIP and assures that a user can be reached on the same URI regardless of where he is physically located, as long as she is connected to the Internet. As mentioned, a user can register more than one User Agent, each of these UAs have their own private URI that are mapped to the *public URI*, also called Address of Records (AoR). An example would be when Freddy registers his SIP phone located in his cabin in Tjome, this has the URI `sip:freddy.baksen@hytta.tjome.no`. The registrar the map the AoR to Freddys phone at the cabin. Later when someone try to reach Freddy on his publicly known URI, the registrar acts as a Location Server and redirect the request to his cabin, or any other UA he could have registered with the AoR.

Routing

In figure 3.2 all that User A needs to know is the URI of User B, assuming B is registered and online. First A sends the message to its nearest Proxy Server. Which then look up the location of the domain of the receivers URI from a Domain Name Server (DNS). Based on the reply the proxy forwards the message in direction of the domain. When reaching the proxy in the receiving users domain, the messages get routed to an inbound proxy that looks up the current location of the user registered with the URI as an AoR in the registrar or location server. Based on the reply, which might be programmed as mentioned in 3.2.2, the inbound proxy then forwards the message to UA. This receiving User Agent could be located in a different domain than it's home domain, but will still receive the message.

Although the proxy could be considered a simple SIP-router, receiving and forwarding request, it is not stateless like many other routers, but can keep track of the state of session. It can also send messages to more than one receiver, in that cas it is labelled a forking proxy.

There is also another approach to routing done by what is called a *Redirect Server*. Different from joining the signalling path, the redirect server replies to the entity that sent the message, with an instruction of a new address for it to try to reach its desired destination.

3.2.3 Message Format

The SIP message format is simple. It consists of three parts, the starting line, the message headers and the message body.

There are two types of SIP messages, a request and its response. The first line in the response is a status line, giving both a numerical and a textual status of the response, e.g. the good old 200 OK.

In the request the first line is used to define the purpose of the message, called a

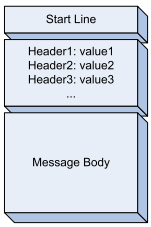


Figure 3.3: SIP Message Format

method, a destination URI and what protocol version that is in use. An example would be:

```
INVITE sip:freddy.baksen@telesyd.no SIP/2.0
```

The RFC 3261 specifying SIP (Rosenberg June 2002*a*) defines six base methods, `INVITE, CANCEL, ACK` and `BYE` which handles session creation, modification and termination. Then there is the `REGISTER` and `OPTION` method, which are used for registering a User Agent and to ask for a servers capabilities. RFC3261 (Rosenberg June 2002*a*) defines additional methods.

The middle part of the message contains header fields. They specify information about the destination and origin of the message in addition to a number of other parameters that that varies from method and what is demanded by the system, e.g. what level of security needed if any. The headers store a lot of information, but the last part of the message usually contains the major part. As described in section 3.2.4, this part usually consist of an SDP.

3.2.4 SDP

Assuming both users are online and registered in its respective domains. User A wants to initiate a session with user B by sending a request in form of a SIP message. This message contains enough information about the intentional session for user B to decide if he wants to participate or not. This description can be in specific formats, one of these specified formats are called Session Description Protocol (SDP) and are defined in RFC2327 (Handley April 1998). SDP is used as the describing format in IMS. The word protocol in SDP might be misleading in this case, as the SDP is simply a standardised way of delivering information, like a schema.

An SDP contains three levels of information, first a session-level description including parameters such as IP addresses, subject of the session and information related to the session. Second is the timing description, which contain information about the start and stop times for the session and one or more media-level parameters. The last level will also contains the media address because this destination might be different from the signalling one.

3.2.5 Offer/Answer model

If a session is to be established between user A and B they need to agree on a number of parameters related to the session, like what media codec to use. This negotiation process has been called the *offer/answer model* and is standardised by IETF in RFC3264 (Rosenberg June 2002*b*). It works in the way that the initiating part of the session makes an offer of possible values for the relevant parameters. When receiving the offer the recipient responds by accepting it or suggesting changes to the session parameters. This exchange continues until an offer is accepted, ensuring the users have a common view on the established session.

3.2.6 Extension

The core SIP provides a basic functionality for session control. But it is also designed to be extendable. These extensions are made to make SIP more suitable for other specific applications. Examples of extensions are the Event Notification Framework defined in RFC3265 (Roach June 2002) or the SIP for Instant Messaging Extension defined in RFC3428 (B. Campbell December 2002). These specifications introduce new SIP headers and responses along with new methods.

It is the inviting User Agent that defines what extensions she supports, requires and does not support. In the response the other part can let know which of the same extensions she supports, and thereby making use of those extensions in that particular session. In the case where a receiver does not support a required extension, the establishment of the session will end.

Extensions are made to make SIP more suitable for specific purposes. When 3GPP chose to use SIP in IMS, a number of requirements was raised that SIP did not meet, like security features or simply the case that the wireless environment is a constraint in it self. IETF took most of these requirements into consideration when releasing the newest version of SIP in RFC3261 (Rosenberg June 2002*a*). Though some of the requirements are documented as extensions to SIP. These extensions will be introduced later in this chapter.

3.3 History of IMS

When 3GPP was founded in 1998, a body was formed to develop a third-generation mobile system based on the second-generation system, GSM, with globally applicable Technical Specifications and Technical Reports.

Just after the 3GPP Release 99, the body started to work on what then was called All-IP. This was the beginning of what later was named the IP Multimedia Subsystem, better known as IMS. Realizing that the development could not be completed for the Release 2000, the release was split into two parts, Release 4 and Release 5. Where IMS was totally excluded from Release 4. After the freeze of Release 5 in March 2002, there were still a lot of features that was postponed due to disagreements. Most of these weaknesses were fixed in Release 6. These

fixes included the interworking with the Circuit Switched (CS) domain and WLANs. Additional security enhancement was also made.

3.4 Architecture

An important notice is that 3GPP defines functions of the IMS, not how it should be implemented. In that way it is up to the vendors to build boxes containing one or more of those functions, thought most follow the specification quite strict and stay to the "'one box per function"'-policy.

The figure 3.4 gives a good overview of the architecture. The four elements at top are usually just described as Application Servers (AS), but in this case a number of them are shown. Down in the left corner we find what later is noted as UE, or User Equipment.

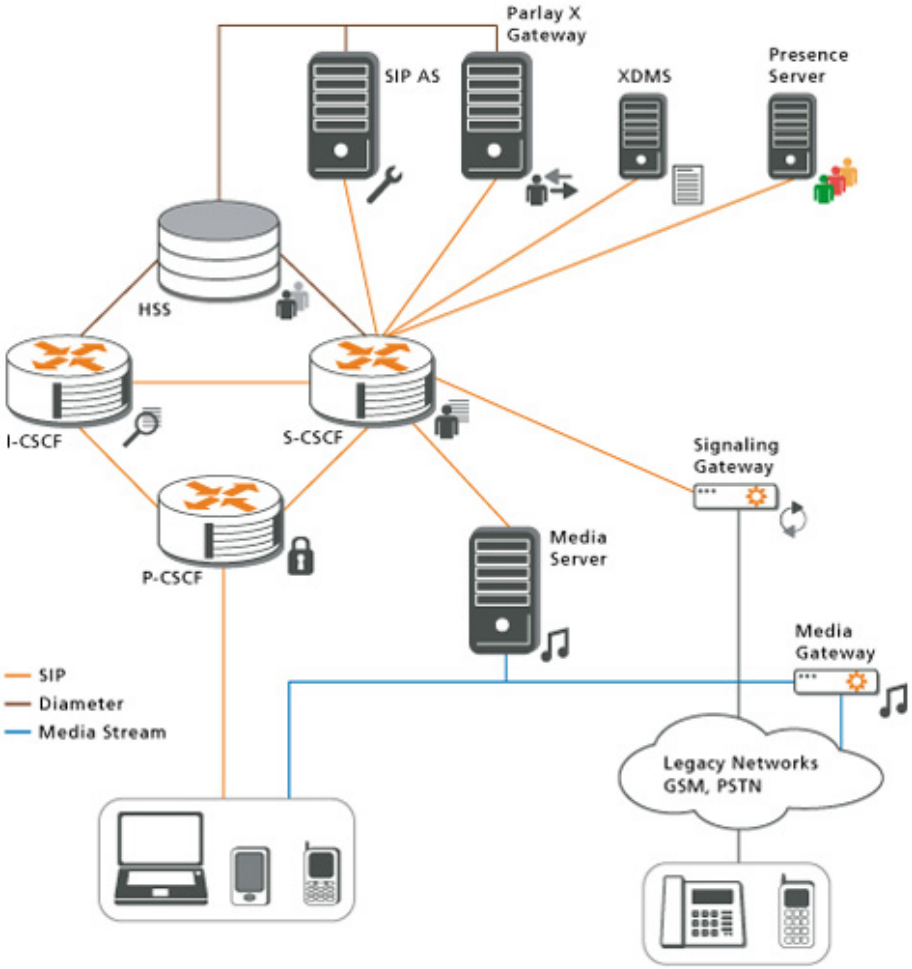


Figure 3.4: IMS Architecture (FOKUS June 2007)

The rest of this section contains an overview of the requirements of IMS given by 3GPP. Followed by a more detailed description of its architecture.

3.4.1 Requirements

Service development

One of 3GPPs most important requirement to the IMS design was the ability to do rapid service development. By this meaning services in it self not being standardized, but rather the service capabilities. This is a turning point for the telecommunication industry, which has a long history, also of success, to very tightly standardize every service provided. Luckily 3GPP is brave enough to look to the rapidly developing world of the Internet and try to incorporate some of that success story in the new mobile network, but as a this thesis will try to explore, the Internet world might be going faster the other way around.

Connectivity

Given by the name of the system, the IMS depends on IP connectivity. From the beginning it was only to support Ipv6, but due to the lack of penetration of IPv6 3GPP have created recommendations about how IP version interworking should be handled.

Quality of Service

An important advantage of telecommunication systems is the ability to guarantee QoS. The Internet is based on best-effort, but when it comes to real time applications on devices with a lot of constraints, like a mobile phone, the QoS is necessary to ensure services that in many cases are critical to society (Oyama September 2005). In IMS the participating parties negotiate its capabilities and express its QoS requirements during the setup of a session, using the Session Initiation Protocol. These capabilities are then reserved in the network. With real time applications the Real-time Transfer Protocol (RTP) is often used to encode and pack media, which is then transferred over the network using a transport layer protocol. The capabilities necessary to conduct a session depend on both the UE and the access network that the participants are connected to. The capabilities can also be reflected by the price of the service, e.g. premium services with higher quality.

Policy Control

The Operators want the ability to control and authorize the usage of bearer traffic. These policy-controls are modelled on a user-to-user basis and give the opportunity of tailor subscriptions for each user, with wanted services. It is necessary to prevent the misuse of bearer resources to be able to provide a guaranteed QoS.

Security

Just as with QoS, security is a fundamental requirement in every telecommunication system. The IMS security is divided into access security and network security, where the latter concerns with security between nodes in the core network and between operators. The first includes authentication and authorization mechanisms between the User Equipment and the IMS core network, in addition to the security applied for IP connectivity, as in e.g. secure WLAN or GPRS. Just like in GSM subscriber information is stored in a smart card in the terminal, but the IMS subscription is stored in a separate application than the UMTS subscriber information.

Roaming

One of the great success factors of GSM was that the customer could expect the same service when reaching other networks, a least where there existed a roaming agreement. This function is obviously inherited in IMS. But with GSM often being the first network deployed in many areas, and the spread of UMTS and IMS are not happening at the same phase, IMS needs to ensure operability with those legacy systems. A roaming IMS subscriber will always connect to his home network to get his services. This is not to hinder roaming of services, because different IMS networks might not provide the same services.

Charging

IMS provides a comprehensive charging architecture. It gives operators and service providers to charge in many different schemes. Charging can be volume based, session based or for a specific service. As a result of every part of an ongoing session, being able to add a new media component, both the calling and the receiving part can be charged, or the charging could be split. Like in GSM, IMS gives the opportunity of charging prepaid, or a more ordinary post paid scheme.

3.4.2 Protocols used

There is a wide range of protocols included/used in the IMS specifications. Most of the protocols used in earlier telecommunication systems have been tailored by ITU-T or ETSI for it's special purpose, but with IMS being deployed fully over IP, it was natural to reuse some of the great work done by IETF, which do most of the standardization related to the Internet. 3GPP have also made adjustments and extensions to make some of the protocols more suitable for use in the mobile domain.

One of the major roles in IMS is controlling the calls and sessions. For this purpose 3GPP chose to use SIP. As mentioned earlier it is a HTTP based protocol, and its ability to make it easy to create new services was of great importance. This protocol is described in further detail in section 3.2. The

Session Description Protocol is also used by IMS, but as with SIP it has already been introduced. Not all protocols used in IMS will be described, only those with greater significance in IMS. The ones are the Real-time Transport Protocol (RTP) for transportation of real-time media over unreliable transport protocols, and the AAA protocol Diameter.

RTP

The Real-time Transport Protocol (RTP) defined by RFC3550 (Schulzrinne July 2003), is a protocol intended to allow users to send real-time media, such as voice or video, over an unreliable connection type, such as UDP. A main part of what RTP does is that it sequences and timestamp media payload. When a media stream is sent on an unreliable IP-connection, there is no assurance that the packages will arrive in the same order they were sent nor that they will arrive with the same timing relationship to each other. The last case is due to variance in delay, or *jitter*, introduced by the IP network.

RTP timestamp the payloads and give them a sequence number, the receiving part of the connection will then have a buffer collecting the payloads and play them in the right order at the right time. If a packet has not arrived by the time it should be played, interpolation techniques will be used by the receiving part often giving minor distortion to the receiving user. There are no Quality of Service (QoS) control provided in RTP, but QoS can be monitored using the RTP Control Protocol (RTCP). This protocol is always used together with RTP and provides the ability to convey QoS statistics and information about the media session participants.

DIAMETER

AAA is a term that refers to Authentication, Authorization and Accounting; the two first are generally linked in IMS. All of these factors are crucial to any telecommunication system. Diameter is a IETF specification where the core protocol is defined in RFC3588 (Calhoun September 2003). DIAMETER has evolved from the RADIUS protocol, which faced some challenges, especially regarding large scale networks, due to RADIUSs lack of congestion control buy running over UDP. DIAMETER runs over a reliable transport that offers congestion control, such as TCP or SCTP.

In short the core DIAMETER protocol defines some functional entities for the purpose of performing AAA functions. A Diameter client, Diameter server or a proxy are such entities. A comprehensive description of these entities is found in RFC3588 (Calhoun September 2003). In IMS the Cx and Dx interfaces are used by the I- and S-CSCF (all to be introduced later) to perform a number of functions, like downloading the authentication vectors of the user from HSS where these are stored (Blanco December 2006). Or any other task related to information regarding a user, also the allocation of a S-CSCF. This protocol is, in simple words, what takes care of the serious stuff, that needs to be tracked, like charging, or security related tasks, knowing who is who, and who are allowed to do what.

The core DIAMETER protocol can be extended with different applications making it suitable for different environments. The IMS takes use of two such applications, namely the DIAMETER SIP Application for is used in the Cx, Dx, Sh and Dh reference points. The second is the DIAMETER Credit Control Application used for the online charging functionality over the Ro reference point.

3.4.3 Entities and functionalities

It was an objective for 3GPP to make the architecture as simple as possible. Yet, in good old Telco tradition, the architecture consists of a variety of entities. Of vital importance are the three types SIP servers handling the exchange of SIP messages, called Call Session Control Functions (CSCF), one proxy, one interrogation and one serving. Another central unit are the databases storing subscriber information, as the Home Subscriber Server (HSS), or the routing database called Subscription Locator Function (SLF), which is required to determinate in what HSS a subscriber is located when a network consist of more than one HSS. Having in mind that IMS is intended to ease the development of new innovative services, the service functions are of great importance in the architecture. A main player in this function is the Application Server (AS), or more correctly, the Application Servers. These entities, in addition to some others, will be described in more detail.

AS

The Application Services provides most of the services available in IMS. They can be divided into three different types. The SIP AS, the Open Service Access-Service Capability Server (OSA-SCS) and the IP Multimedia Service Switching Function (IM-SSF), where the two last which gives an operator the ability to access OSA AS and the CAMEL Service Environment for its IMS subscribers. From the IMS view they are all treated as SIP servers even though two of them are gateways to other applications servers.

The SIP Application Server is the native AS that provides a wider range of value-added multimedia services based on SIP, these could include presence, messaging, Push to talk Over Cellular and conferencing services.. This is where new IMS services will be developed.

The Open Service Access-Service Capability Server (OSA-SCS) provides an interface to the OSA framework AS. It is defined in the 3GPP specification TS 29.198 (Alaoui June 2001). It works on the one side as an application server towards the S-CSCF, and on the other side it provides an API against the OSA AS.

The Customized Applications for Mobile Network Enhanced Logic (CAMEL) provides services in GSM, and the IM-SSF acts, much in the same way as the OSA-SCS as an interface towards the GSM Service Control Function (gsmSFC). The protocol used on that interface is the CAMEL Application Part (CAP) defined in TS 29.278 (Berry April 2005).

P-CSCF

The Proxy-CSCF is the first entry point for any IMS Client connecting to the IMS network. It acts as a SIP proxy server handling all the SIP requests and responses sent or received by an IMS Client. It also handles several other functions, some related to security like the establishment of an IPsec (Thayer November 1998) connection with the terminal. Some related to charging collection, and compression of the SIP messages sent over the air interface. For information this compression is not done to save bytes that are sent over the air interface (compared to the coming multimedia session, this size in neglectable) it is done mainly to make the transmission faster, thus reducing total signalling delay. An IMS network usually consists of a number of P-CSCFs mainly for scaling and redundancy purposes.

I-CSCF

The Interrogating-CSCF is situated on the edge of an administrative domain. It is the entry point for messages arriving from other networks, and for this reason it is listed in the Domain Name Server (DNS) for the same reason as with P-CSCF, an IMS network usually contain more than one I-CSCF.

S-CSCF

The most central node in the SIP signalling path is the Serving-CSCF. It acts both as a SIP proxy server and a SIP registrar, handling most of the communication with the HSS. Every SIP message that is sent from an IMS terminal is handled by a S-CSCF. It reviews every message and decides if and which application servers, if any, the SIP signalling should visit on its path.

HSS

Another central node of IMS is the Home Subscriber Server, which is a central storehouse for information related to the user. For the ones familiar with GSM, the HSS works much in the same way as the HLR. In addition to personal user information and security information the HSS contains information about for what services a user has a subscription. If a network, for scaling reasons, holds more than one HSS, a Subscription Locator Function (SLF) is used to determine in which HSS a user is registered.

Others

Other entities that should be mentioned are the Media Resource Function (MRF) that provides a source of media in the network. It is divided into a controller entity, the MRF Controller, and a media plane entity, the MRF Processor. The IMS architecture also contains a number of entities related to the

interworking with legacy systems, but they are considered out of the scope of this report. Full description in TS 23.002 (Minlinski December 2005).

3.4.4 Reference points

In the telecommunication sector reference points are standardized, not interfaces. A reference point describes all the traffic between two entities, including what protocols to use for the different type of traffic. They do not only specify how to interact with an entity, but which entities that are allowed to interact. Of all the different reference points standardized in the IMS system, some of them are of greater importance when developing new services. These are all reference points related to the interconnection of the User Equipment and the IMS Core Network. They include the Gm reference point that is the main transportation channel of SIP messages between the UE and the IMS CN. But also the Ut reference point, which allow direct communication with the Application Servers, and the Sh reference point. There will not be given a complete description of all the reference points in the IMS system, they are all defined in the 6a.7 section of the specification TS 23.002 (Minlinski December 2005).

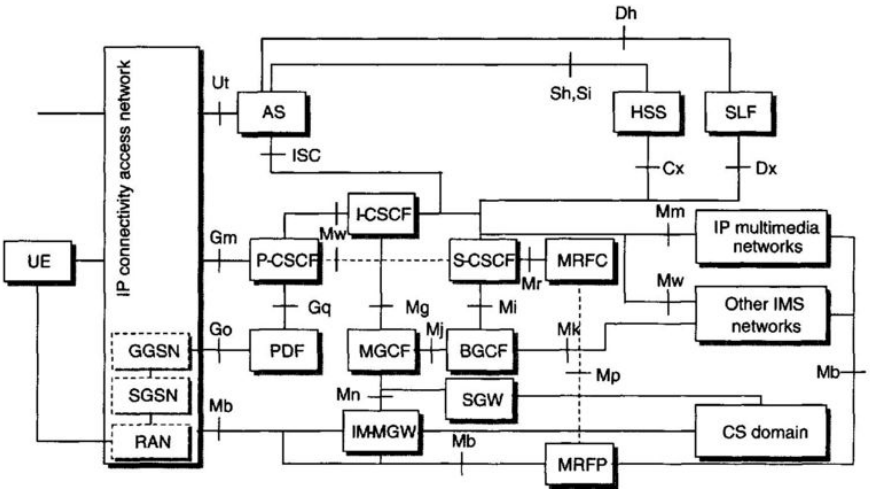


Figure 3.5: IMS Architecture with reference points (Poikselka et al. 2006).

The Gm reference point connects the UE to the IMS and is used for transferring all SIP signalling messages from the UE to the P-SCSF in the IMS core network. The procedures related to the reference point can be categories into three mains: Registration, session control and transactions.

- Registration: Security parameters are exchanged to authenticate both the UE and the network. Used when registering for an AS. Also used with network-initiated de-registration and re-authentication.
- Session control: Both initiated from UE to IMS (and further on to an AS or another Terminal), and initiated somewhere else terminating at the UE, thus sending from the P-CSCF to the UE.

- Transaction procedures: Used to send all standalone messages and replies.

The Ut reference point enables users to securely manage and configure their network services related information hosted on an AS. They can create Public Service Identifiers, such as a resource list. HTTP is the chosen protocol for the reference point. An example of use of the Ut could be to if you want to add a buddy to your buddy-list, you send that message over the Ut reference point to the Application Server.

The Sh reference point lets an AS (SIP AS or OSA-SCS) communicate with the HSS requesting information about a user, or if it needs to know which S-CSCF is serving a particular user. The HSS maintains a list of which ASs are allowed to obtain and store data.

The Mw reference point connects the different CSCF entities, and its main procedures are related to registration, session control and transaction.

The Cx reference point connects the I-CSCF and the S-CSCF to the HSS, and it typically handles three type of procedures: those related to location management, handling user data and authentication.

The Mm reference point handles the communication between IMS and other multimedia IP networks, allowing the I-CSCF to receive a session request from a SIP User Agent outside of the IMS (this can be both a server and a terminal).

3.5 IMS Concepts

Because many concepts in IMS are so close connected to SIP, some of them described in this section will look a lot like the ones described in the SIP section 3.2. Nonetheless it is important to have a clear idea how identification, registration and session initiation is handled in IMS.

3.5.1 Identification

In every network there is a need of identifying the users, to ensure information reaches its right destination. In IMS there are two types of identities, the Private User Identity and the Public User Identity.

The Public User Identity is the one that is used to reach a subscriber and can be compared to the AoR in SIP, but unlike in SIP the subscriber can have more than one Public User Identity allocated. These identifiers are either a SIP URI or a tel URI, as described in the SIP section 3.2. It's common that a user will have one of both of these, for both SIP terminals to be able to reach it, as well as legacy phone systems. They would usually take the form of either sip:first.last@operateor.com or sip:+4799578455@operator.com, the last being a tel URI.

The Private User Identity is a globally unique address neither specified by a SIP URI nor a telURI, but in the format of a Network Access Identifier (NAI) which

are defined by RFC2486. Originally a subscriber would have one Private User Identity, stored in an ISIM application on the smart card used in 3G phones. But from 3GPP Release 6 an IMS subscriber can allocate more than one private identity. One of these will be stored in the smart card of a 3G phone, but others can be used in other IMS terminals.

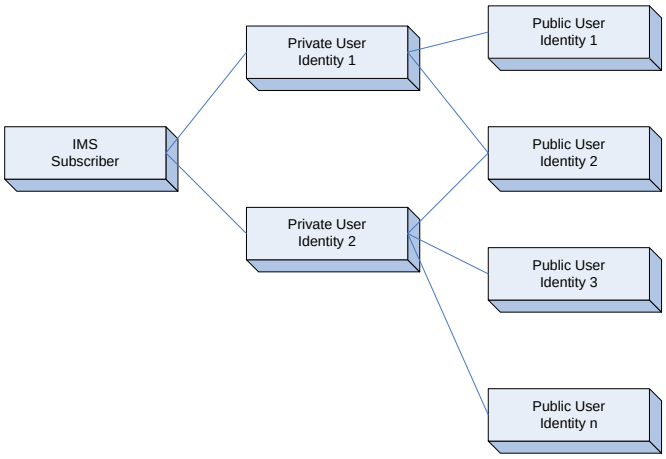


Figure 3.6: Relation of Private and Public User Identities in R6

The identities are related as shown in figure 3.6. Where a user can publish many different public identities, like one for his family and one for work, they all terminate in the same phone, but can be treated differently according to the users needs. Like a Public User Identity identifies a user, a Public Service Identity identifies a service in the same way.

3.5.2 Registration

For an IMS Client to be able to register to the IMS network it needs, first IP connectivity that can be gained through e.g. GPRS or WLAN, and second it needs to discover a P-CSCF to communicate with. The P-CSCF can be discovered either by getting to know where it is as one gain IP connectivity, this is normally done when connection to a GPRS network. The IMS Client can also do a standalone procedure using DHCP and DNS to discover a P-CSCF. The IMS terminal should by now have acquired an IPv6 address and know where to send its IMS requests.

It's now time for the user to do an IMS-level registration to gain access to ones IMS services. The registration procedure done by sending a SIP REGISTER request, which will then bind the users Public User Identities to its IP address and a S-CSCF will be allocated to the user.

The IMS terminal is responsible of keeping the registration active by periodically sending new REGISTER messages to refresh its registration. If this does not happened. The S-CSCF will release itself from the allocation and assume the

client has either lost connection, or been switched off.

3.5.3 Session initiation

The session initiation in IMS is quite similar to the one in pure SIP, as described in the SIP section 3.2. In short: user A wants to start a session with user B. First he generates a SIP INVITE describing the intention of the session and it desired destination to the P-CSCF. The P-CSCF forwards it to the users allocated S-CSCF. The S-CSCF might then interact with some Application Servers before it forwards the message to the I-CSCF of the receiving user Bs home network. The I-CSCF then querys the HSS to find what S-CSCF is serving user B and forward the message to this entity. User Bs serving S-CSCF might also interact with some Application Servers before it forwards the message to the appropriate P-CSCF which then hands it over to its final destination.

User B then generate a response that travel in the reverse direction as just described and after a few more roundtrips the session is established.

3.6 Services

Some services can be provided without any further standardization, like voice-mail, by using the standard concepts described earlier. But IMS has an intention of providing new and exciting services, and some of these services need further standardization to be able to interoperate. It is also a goal that these services, provided by the IMS Core Network, will be available as service enablers in the IMS Client Framework for a third application developer to use when developing innovative rich featured applications. This section will give a brief introduction to some of these services that have been standardized so far: Presence, Instant messaging and Push to talk Over Cellular (PoC) (Camarillo & García-Martin 2006).

3.6.1 Presence

In short the presence service allows a user to indicate a status associated with his identity. He can then allow other users see what status he has indicated. It is simple, but powerful. Two important entities are the Presence User Agent (PUA) which provide status information associated with a user and a Watcher, one which subscribe to the status of another user. Both can be either a IMS Client or another entity such as an Application Server and they both interact with a Presence Agent (PA), often noted in IMS as a Presence Server.

The 3GPP TS 24.141 (Drage December 2006) define an architecture that support the presence service in IMS, this is included from the 3GPP Release 6.

3.6.2 Instant messaging

Next to email and web, instant messaging is one of the definite killer apps of the Internet. It is often separated into two different modes; immediate messaging and session based messaging.

The immediate messaging are single messages sent from one IMS entity to another one, often between two users. It uses the SIP MESSAGE function and generates a SIP message with intended content such as text and other smaller multimedia content such as a picture. There is nothing hindering a dialog of immediate messages although the network considers this as single messages with no relation to each other. If a message is sent to a receiver that is not registered at that moment, it can be routed to an AS for storage until the user connects.

Session based messaging consist of a user inviting another to start a messaging session, thus using the SIP INVITE function, containing a description of the session formatted as an SDP. These sessions can either be peer-to-peer sessions just like the ones we often experience with the instant messaging on the Internet, a textual conversation. Or it can be more like a textual conference, where participants take part in a group chat, much like a channel in the Internet Relay Chat (IRC) online. In the latter case there is an AS handling the creation and administration of groups, and it becomes a router for messages sent to the conference. The session based messaging in IMS is specified in TS 24.247 (Mayer March 2007).

3.6.3 Push to talk Over Cellular

The PoC service can be compared with an instant message, only with voice. When a user push a button it records a message, then when it is released, the message is sent to the desired receiver. It can either be a one to one session, or a group based service where the recorded message is sent to members of a group. There has for a time been some proprietary PoC solutions available, but there has been developed a widely accepted standard that is now handled by the Open Mobile Alliance, which develops it further. The architecture of the PoC service is defined in the (OMA 2006).

Chapter 4

XMPP Theory and Comparison

Contents

This chapter will introduce the XMPP technology, give a detailed introduction to it's architecture and concepts and on as many points as possible try to map the platform to the IMS. Explaining what is its equivalent in IMS, both when it comes to entities and concepts. At the end of every section, there will be a "and what about IMS" part explaining how the issue relates to IMS. Call it comparing on the fly.

When introducing XMPP it is assumed that the reader is familiar with Instant Messaging and Presence concepts. Most of the information in this section is based on the specification of XMPP provided by IETF, the core protocol RFC3920 (Saint-Andre October 2004*a*), and the Shigeoka book putting XMPP into practice (Shigeoka 2002).

4.1 Introduction

The XMPP was born to handle Instant Messaging (IM) and Presence but has grown to be a technology that can handle message transfer in a range of different ways. One of XMPPs most discussed advantages is its open nature. Both it's data formats and protocols are all well documented in the IETF specifications RFC3920 (Saint-Andre October 2004*b*), the core protocol, RFC3921 (Saint-Andre October 2004*b*) for Instant Messaging and Presence. This is indeed in grim contrast to the other large commercial IM systems that are all based on proprietary standards.

XMPP make use of XML-based data formats that employ the popularity and extensibility of the eXtensible Mark-up Language (XML). By being based on a simple distributed client/server architecture, XMPP is simple, scales well and is very well fitted for the dynamic environments that modern real-time messaging systems will form in the future.

Though XMPP first and foremost was developed to handle instant messages between clients and presence information related to these clients, such clients can now be much more than two people chatting away in a typical IM manner. The same system can be used for applications and machines as well, making it a general message and presence handling system. This makes the technology very powerful in the enterprise domain. This is often referred to as message-oriented middleware (MOM).

4.1.1 Message Oriented Middleware

An example could be in a processing plant where a number of machines are critical for operation of the plant. One could then develop a computer system that was integrated with the sensor mechanisms in the machines to support the service of the machines. If a problem occurs a message would be sent to the system and operator, while the machine could change it's presence status to e.g. "Need Service" or "Alert". The operator could then check the status of the service workers and forward a message to one that is available, or the message could be sent directly to the service workers in the first place. The message sent could also have been processed by an expert computer system that has analyzed it and might suggest actions.

4.1.2 Chatbots

The client/server architecture of XMPP makes it likely that the server usually just forwards them to the right destination don't care to much what are messages being sent contain. But some special types of clients, called *Chatbots*, can act as a receiving client performing an action on an inbound message. These chatbots can act as simple clients returning a stock price from a person askin, or they can be much more sophisticated like the expert system in the example above. Say that the error messages from the machines above all get sent to a chatbot, the bot can then react if a number of conditions occur.

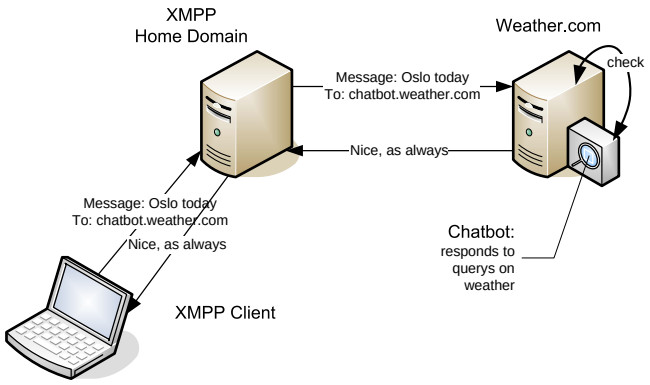


Figure 4.1: Chatbot

4.2 History

As early as in the start 1998, the Jabber project was formed and started by the American born Jeremie Miller. The project was steadily growing but captured a lot of public attention when it was shown on the popular developers news site Slashdot [1] in January 1999. In August 1999 Mr Miller submitted a statement pledging the Jabber community's support for the IETF standards process. Early 2000 the community released version 1.0 of a reference server.

In mid 2000 the Jabber project submitted an Internet draft,Request For Commons (RFC) to the Instant Messaging and Presence Protocol Working Group (IMPP) in IETF, documenting the core Jabber protocol. Unfortunately, the IMPP effort bogged down and the Internet draft was expired without any further contributions. Then again in early 2002, the Jabber community did another try, submitting the protocol once again to the IETF. This time the submission went through and the IETF group, XMPP, was formed. Later in 2002, the

[1] www.slashdot.org

IETF's Internet Engineering Steering Group (IESG) approved the XMPP charter and the hartered work in the XMPP working group was launched.

Without going to deep into the differences between IETF and 3GPP, it's important to notice the open manner of XMPP, also through it's history. It is built and supported by the Jabber community, which continue to work on its extensions. 3GPP might suffer from it's origin and have inherited some of the old "hard to row" sens of ITU, which in it's time was necessary to assure interoperation, but may also hamper rapid adoption to modern times. Nonetheless does IMS use a lot of protocol that are developed in a similar sense by IETF as the XMPP.

4.3 Architecture

The messaging model of XMPP follows the well-know client/server architecture. In general a client only communicates directly with the XMPP servers in it's own domain. However, there has been work on a protocol that makes the client directly communicate with another client for transferring large files or e.g. a VoIP session. These domains divide the XMPP world into separate zones that are controlled and handled by the XMPP servers controlling domains. Other more commercial IM services, like AIM or Microsoft Live Messenger, often use a central server architecture. A message would then be sent by a client to it's server, then from the senders server to the XMPP server controlling the domain of the receiver, this server would then send the message on to the receiving client.

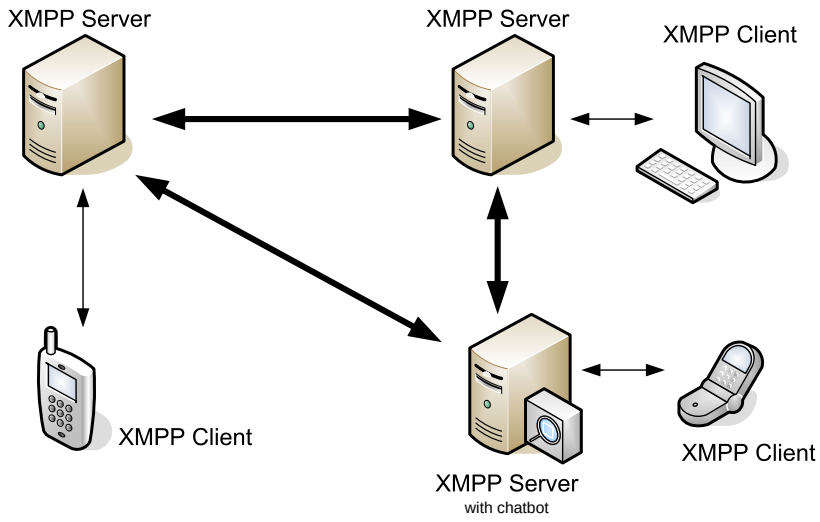


Figure 4.2: XMPP architecture - simple overview

4.3.1 Requirements

When IETF was defining XMPP it was doing so based on the requirements for an Instant Messaging / Presence Protocol defined in RFC2779 (Day February 2000). These requirements were not security related, so those were addressed by the Jabber community and added to the protocol.

4.3.2 Client/server solution

In client/server systems, the client is presenting information to the user and handling request made by the user. Information is passed from the client to a server that offers a defined set of services. The architecture in XMPP has strong focus to make it possible to create simple clients. Most of the logic and processing can be carried out on the server, though the developer is free to implement complicated smart clients mentioned in 4.1.2. The simplicity of creating XMPP clients encourages developers to write clients on a wide range of platforms using different programming languages, helping to spread the good words of XMPP. In addition a simple client architecture makes is more suitable for devices with limited resources and being implemented in embedded devices, like a mobile phone or a mechanical machines control system.

Although a client/server architecture has it's drawbacks, as not serving the direct route of communication, like peer-to-peer technology would provide. This partly centralised control makes it suitable for enterprises for control purposes and enforcing quality of service guarantees. This could be screening messages for sensitive information, then blocking it if an employee was indiscreet.

In relation to IMS this client/server system would be the actual IMS Client on the mobile phone connecting to the P-CSCF. Or in a more general telecom term; the handset connecting to, and authorizing with the mobile network. It is important to have a strongly defined interface between the handset and the network so that any phone would work on any operators network (at least when sticking to the GSM standard), just as any client should work with any domain in the XMPP world. The same is desired concerning the IMS client installed on a handset. Both the interface, or reference point as it is called in the Telco realm, between the IMS client and the P-CSCF and between the IMS client and applications developed to make use of it should be strongly defined for interoperability. Although as (Gunnerud 2006) shows, the latter case is still not true, leading to operator specific applications that might not be interoperable with IMS applications provided by other operators.

4.3.3 Distribution - together they build the world

One of Internet's true killer apps is email, and email system allows separate, distributed email servers that manages it's own email domains and clients. In this same manner each XMPP server manages it's own XMPP domain. These domains are all defined by an Internet domain name, like `telesyd.no`, meaning that all incoming and outgoing messages from a user in that domain will be

handled by the XMPP server controlling the domain. And with a very similar way as email, a user is addressed by a JID (Jabber ID) like `user@telesyd.no`.

This distributed architecture limits the responsibility of each XMPP server to handle it's own users, and it can be scaled thereafter. A small server can handle as few as one user, while bigger domains can handle millions of users and needs to be scaled thereafter. A good example of this is when Google chose to use the XMPP architecture for it's chat service, meaning millions of people would use it, while the test server used in this thesis would only have a dozen registered users.

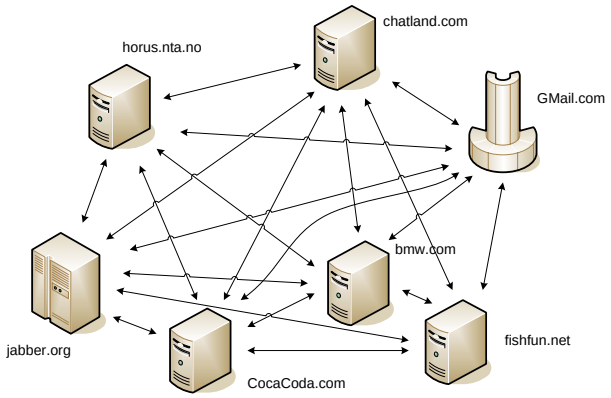


Figure 4.3: Distributed Architecture

All in all does this distributed architecture mean distribution of responsibility, and it means that the total XMPP network can grow without requiring massive resources from any particular actor. Each domain can add as many users as it have resources to handle. It also means distribution of power - you don't need to be Yahoo, Microsoft or AOL to handle a part of a big network, but can have control and autonomy over your own little corner of the big and growing XMPP world, only requiring interoperability with other XMPP servers so that your users can reach out to the network.

This distributed server model is by no means an innovation, but rather old standard technology. Although this can be considered a strength and not a weakness, using a well tested and torn architecture provides predictability. The innovation in IM in general is the addition of presence to communication applications, and XMPPs innovation is the use of an open XML data format for the data being sent in the communication systems.

In relation to IMS one could see the XMPP domains as the operators domains, where each operator handles their own set of users and where it need to assure interoperability for it's users to be able to communicate with costumers of other operators. It can indeed be claimed that the task of interoperability between two operators is a lot greater than between two XMPP servers, needing to handle charging systems and following stronger authentication systems in addition to

handling voice and video streams.

4.3.4 Entities

In reality most of the XMPP servers will be connected to the Internet, making them self-reachable for other connected servers. But in theory one can build a network of XMPP servers in any network, e.g. inside a closed enterprise network. Therefore one can see the XMPP universe broken into a number of logical sets and subsets, where one must contain either one or more of the other:

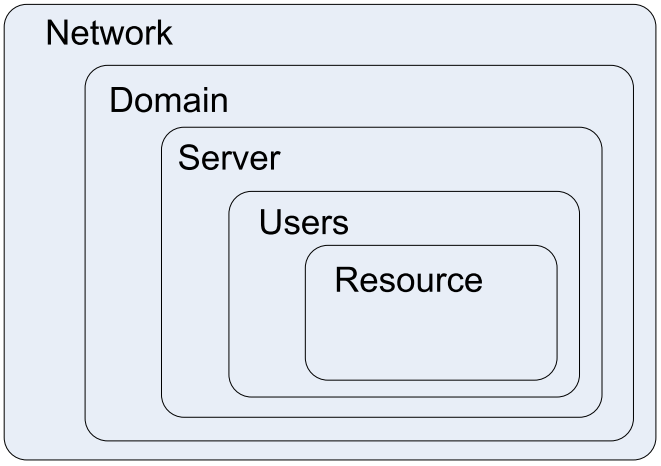


Figure 4.4: The XMPP Universe

Network	Contains at least one domain. All servers in a network can exchange messages.
Domain	Controlling it's own set of users and manages a valid domain name address.
Server	As a logical entity there is one server handling each domain. However, in larger domain there will be more than one server handling the load.
Users	Being the logical entity representing a user. Even though messages are addressed to users they are always delivered to a resource (described in 4.4.3). Users are connects to a server.
Resource	An entity representing the actual delivery endpoint for a user. Like a chat client on a mobile device, or a application using xmpp for communication, like the application developed related to this thesis.

Table 4.1: XMPP entities

In addition to the logical entities described above, XMPP defines something calls

trasports. These are specifically related to IM and make you able to user your XMPP user to chat with other IM systems, like MSN, ICQ or AIM. Because transports are mostly IM specific, they will not be explained much further in this thesis, although one should notice the Jabber communities effort to make this an open system, and not trying to use lock-in strategies on it's users. In addition these transports might be an example of how to build mapping systems for XMPP to other systems that are not related to IM, e.g XMPP to SIP, or to any other proprietary standard handling information exchange.

Looking at the XMPP universe from the level shown in 4.4 it looks very similar to how a the Telco world could have been described. Although most of the world's operators are all connected to makes up the giant telecommunication system of the world, you could se separate networks, like the emergency systems that are separate to the commercial systems due to security and reliability issues. Operators are the one controlling the domains, while servers handle the communication with users. The IMS also introduces the ability to register many contact points to one address, something that was hard in the tight SIM-world of GMS, the last issue looking a lot like how an XMPP user would register many resources.

Of the many IMS entities descried in section 3.4.3, as the CSCF, HSS etc. Most of these does not have a defined counterpart in XMPP. Many of these functions are integrated in the server implementation.

4.4 XMPP concepts

This section will introduce concepts that are important to XMPP as well as some general concepts for an easier comparison with IMS.

4.4.1 XML streams

An essential concept of XMPP is the use of XML streams. The idea is that instead of delivering separate XML documents on a single connection, a persistent connection is used for delivering the XML data elements, or *XML stanzas* as they are called in XMPP. These stanzas are further explained in section 4.4.2, but in general it is XML data elements according to XMPP standards. When an XMPP client wants to connect to a server, it opens a XML stream on top of a TCP connection to the server. The stream is then kept open as long as the communication is intended, providing a channel to transport XML stanzas. In a chat like environment, this means that the stream is kept open as long as a user is logged in. If bidirectional communication is wanted, like it usually is, a stream in the opposite direction would be initiated.

The core specification of XMPP (Saint-Andre October 2004*a*) points out that security mechanism[2] should be used and authentication is needed before any entity accepts stanzas from the connection part. In any case where an error

[2]RFC3920 (Saint-Andre October 2004*a*) specifies the use of TLS and an XMPP-specific profile of the Simple Authentication and Security Layer (SASL) protocol

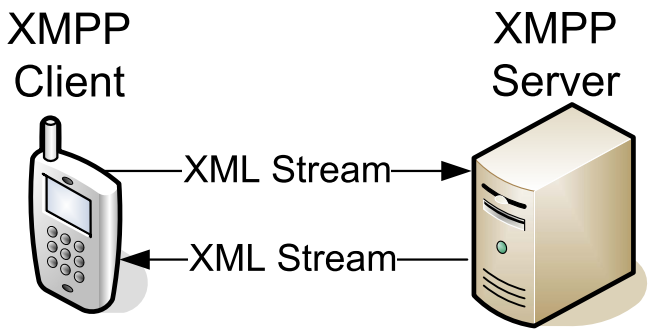


Figure 4.5: XML Streams

occurs in the stream, a standard set of error conditions are set inside an error tag that is sent before closing the stream and connection.

4.4.2 XML stanzas

The XML streams established are used to transfer information between two entities, this is XML information that is formatted in XMPP as XML stanzas, "a series of lines arranged together in a usually recurring pattern"[3]. There are three predefined XML stanzas in XMPP: `massage`, `presence` and `iq`. All these stanzas have five attributes in common:

to	JID of the receiver, either a user or a server
from	specifying the JID of the sender
id	an optional attribute for internal tracking of stanzas
type	specifies a purpose or context of a message
xml:lang	defines the language if the message is intended for humans

Table 4.2: XML stanzas attributes

The three stanza types will be further explained in section 3.6. But a simple `message` stanza could look like this:

```
<message to='jensemann@goverment.no' from='pesant@farm.no>
<body>value the work of this countries peasants!</body>
</message>
```

These stanzas will equal the patterns of SIP messages in IMS. Where the different stanza types in relation with the `type` attribute is handled by the METHOD

[3]The definition of the word "stanza" by the Merriam-Webster Dictionary

types in SIP. In the way that XMPP is defining what is sent between two entities, IMS is defining reference points. However, these reference points are extremely more complex.

4.4.3 Identification

An XMPP domain hosts zero or more users[4]. An XMPP user is a logical endpoint for where to send messages, it's usually represented by a person or user account. The JID (Jabber ID) also contain a resource, both will be described in this section.

Resources

It is very likely that a user would want to register more than one client. One case could be where a user have a chat client logged in on her home computer, but when leaving home she logs into a chat client on her mobile phone. For this situation XMPP introduces a concept called *resources* which extends the users contact address, written like this `user@domain/resource`.

Resources can be a very useful concept when seeing XMPP as a communication platform used to launch different applications. A user can use the same account as basis for many different applications running, where each application is specified in the resource parameter. This technique was used when developing the ThinkAlike application and is described in further depth in chapter 5.

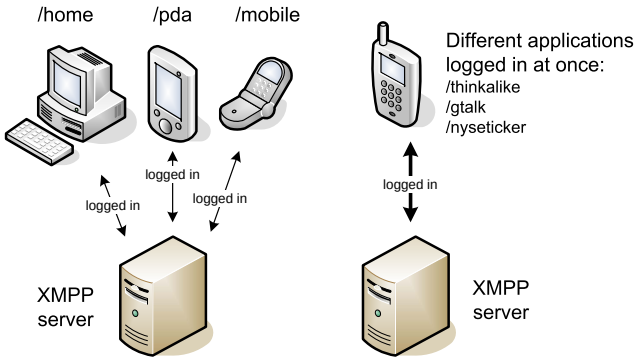


Figure 4.6: The use of resources

In most cases a message would be sent to a user, excluding the resource parameter, the server would then forward the message to all the resources. A server

[4] A XMPP domain handling zero users would in the first place look useless, but a domain could host different services like a chatbot providing weather information.

can also route messages to a specific resource best suited for a message or by preferences set by the user, like if a mobile resource is logged on it is always the preferred recipient.

JID

XMPP embraces a number of features from the success of email. Among them the addressing, containing an XMPP domain and optionally a username and resource. Addressing a user is done by the easy-to-use pattern `user@domain/resource`, though in many cases the resource is omitted. In this way we can see that some domains might just reuse the email addresses that users already have registered, like Google is doing with it's GTalk where a chat service is added to the GMail email service.

When sending a message intended for a server and not a user, the user-part of the address is left out. If one wants to send a message to the server in it's own domain the user can just not specify any receiving address, the server will then implicitly understand that the message is intended for it self.

A strong advantage using this scheme is that it is easy to remember and express in the real world and people are familiar with it's for from email. One could see confusion between email addresses and JIDs, thus the trend that email providers extend their services to include XMPP chat possibilities.

4.4.4 Registration

The term registration is used in XMPP for the action of adding a new user, giving her a new account, defined in the XMPP extension (Saint-Andre January 2006*b*). The concept handled in this section is more to be recognized as `authentication` or logging into a server, but the section is called registration for easier comparison with IMS. For an XMPP client to register with a server it needs a stream already set up connecting it to the server.

In it's simplest form the client could send a info/query message to the server with it's credentials, like username and password. A message like that would look like this:

```
<iq type='set' id='auth_id'>
<query xmlns='jabber:iq:auth'>
<username>test1</username>
<password>passord</password>
<resource>thinkalike</resource>
</query>
</iq>
```

This solution means sending the password in clear text, therefore the XMPP core protocol advices the use security mechanisms like the Simple Authentication and Security Layer (SASL) or Transpor Layer Security (TLS) where TLS

are recommended both for client-to-server communication and server-to-server communication.

4.4.5 Extensions

One of the great features in XMPP is its extensibility. The core protocol handles the basic communication mechanisms while the extensions handle more specific tasks. These extensions are standardized by the XMPP Standards Foundation, where a complete list can be found here: `http://www.xmpp.org/extensions/`. These extensions range from something as basic as the XEP-0053 XMPP Registrar Function (Saint-Andre December 2006*a*) to the complex extension XEP-0166 Jingle (Ludwig June 2007) that handles direct communication between users avoiding the servers when, say, transferring of larger files.

4.5 XMPP services

The name of this section is first and foremost used to lighten the relation this part of XMPP has to IMS and Telco, because the word services are rarely used when talking about XMPP.

In February 2007 the XMPP Council approved an extension called XEP-0030 Service Discovery (Hildebrand February 2007) for discovering information about Jabber entities and the items associated with such entities. Meaning that an entity can provide information about features offered or protocols supported by it, the entity's type or identity and so on. The result is a standards-track protocol for service discovery. In general it could be used for any kind of who-are-you and what-can-you-do queries.

As introduced in section 4.4.2 XMPP are defining three type of stanzas: `message`, `presence` and `Info/Query`. Where the first two serve what their names indicate, while the last is used for everything else, like authentication, registration or more application specific purposes. In IMS these stanza types are partly related to the SIP functions, but still these stanza types reflect what are provided by XMPP, namely easy messaging, extensive presence and a catch-all function.

4.5.1 Message

This stanza is used for sending a message from one entity to another. The application for this stanza is not standardizes in the XMPP Core protocol (Saint-Andre October 2004*a*) so it can be used in many different ways, but in most cases it is used for handling Instant Messages from one client to another. The use of the message stanza in IM is standardized in RFC3921 XMPP Instant Messaging and Presence (Saint-Andre October 2004*b*).

All message stanzas contains the `to` attribute, giving the address of the recipient. A server receiving this stanza would route the message properly based on the value of the `to` attribute. If the server can not complete the routing, like there

is no such user, it will return an error to the sender, thus a `from` attribute is common. If the recipient is logged of and not reachable, the server must store the message and deliver it when the recipient becomes available, often referred to as *store and forward*. A typical message element would consist of the child elements shown in table 4.3.

`<subject>`	Describing the subject that the message is about, or left out if not needed
`<body>`	Containing the initial message, can be translated using the xml:lang attribute
`<thread>`	Indicating if this message belongs in a thread, like an IM conversation
`<error>`	If an error occurred, the standard XMPP error packet isenclosed in the message

Table 4.3: Message child-element

An example of a message stanza would then look like this:

```
<message  from='baksen@televest.no'
to='yunus@teleost.bd'
xml:lang='en'>
<subject>Good morning Yunus</subject>
<subject xml:lang='no'>Våkn opp Yunus!</subject>
<body>We just want to help your people!</body>
<body xml:lang='Vi vil bare hjelpe vårt folk!</body>
<thread>solutiontalk</thread>
</message>
```

4.5.2 Presence

In a general term, the *presence* stanza is the notification part of the basic public-subscribe mechanism, used to deliver information from one entity to multiple recipients. Usually the `to` attribute of the presence stanza is not used, because the message is broadcasted to all the subscribing recipients.

As described in the XMPP Instant Messaging and Presence standard RFC3921(Saint-Andre October 2004*b*), the typical status function of a IM system uses the presence stanza to set status, to subscribe to another users status and to recieve status changes made by users on the subscription list. A simple message for updating ones status is shown below:

```
<presence type='available'>
<status>8 minutes to Berlin!</status>
</presence>
```

3GPP defines the presence service of IMS in TS 24.141 (Drage December 2006). This service is achieved by a specific Presence Server that is an Application Server added to the IMS. In contrast the Jabber community considered presence so important that it was defined as a central part of the XMPP.

4.5.3 Info/Query

An *Info/Query* (IQ) is commonly used for a type of request-response interaction, where one entity requests information from another entity which then will respond. The `type` attribute plays a major role in the IQ stanza, defining what operation it is trying to do. The attribute can be valued as any of the following: `get`, `set`, `result` or `error`. If an IQ operation is successful then a `get` or `set` operation would be replied with a `result` type stanza, if it fails a `error` type stanza would be replied.

The IQ stanza is a typical catch all solution that will be used for everything not directly related to presence and messaging. It can be used for application specific purposes as shown in the chapter 5. An IQ message needs to be addressed to a server or all the way to a resource. If the server allows it, registering a new account could be done as easy as the following IQ stanza:

```
<iq type='set' id='reg_id'>
<query xmlna='jabber:iq:register'>
<username>freddy</username>
<password>opsjoner123</password>
</query>
</iq>
```

4.6 XMPP on mobile

Some constrains are present when using a mobile phone, limited screen size and input options, limited battery and limited bandwidth. XMPP has not been developed with mobile use in mind, but as this thesis show it can handle many of the tasks related to communication with a mobile handset over an IP connection, like GRPS.

Due to a continuous stream being kept open, the user interface can be quick and responsive, compared to a more common push/pull technology. Even though there are no empirical results that indicates it, the fact that the continuous stream is kept open, can generate battery issues.

Though bandwidth use is becoming less relevant due to higher transfer rates, the XMPP article by Mikko Laukkanen (Laukkanen 2006) claims that the bandwidth requirements between XMPP and the SIP Message and Presence protocol SIMPLE differ greatly, where a simple "Hello" message would be double the size sent using SIMPLE versus XMPP.

Chapter 5

ThinkAlike Application

Contents

A quick outline of the ThinkAlike application was given in chapter 2. This chapter will describe the application both in a concept term and the technical decisions that are made when developing it. First the purpose of the application will be explained, and then it's design while the implementation will be thoroughly explained towards the end of the chapter. The launch of the application will be explained in the last section.

5.1 Concept

ThinkAlike is a simple communication application that employs to good purpose the fact that friends like to think alike, especially younger people. This

application makes a friend challenge another friend to see if they associate the same word related to a picture. In the end of the challenge, the participants will be able to discuss the words associated with the picture using an instant messaging function of the application.

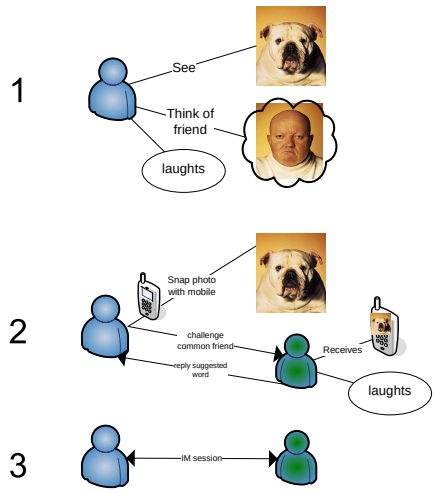


Figure 5.1: ThinkAlike scenario

5.2 Design

Because the application was to be launched using two different technologies it was important to have a clear divide between the operating logic and the communication part of the application. There are no storage requirements beyond simple local storage for each session, thus a natural design solution for this application would be a three layered structure where one is handling the presentation layer, the second is handling the actual logic of the application and then the lower layer handling the communication part of the application as shown in figure 5.2.

It is important that the lower lever is clearly defined so that it is easy to use either IMS or XMPP. One could either develop one application with an easy ”switch” for changing between the technologies, or one could develop two separate applications where the two top layers are mostly the same. The first solution would be better for comparing the results, while it would be a more complicated design that the second. However, the second would demand a higher focus on keeping the rest of the code similar.

Underneath is a description of the three layers, some issues that relate to them and the key tasks that they should be able to do. Then the actual implementation will be introduced.

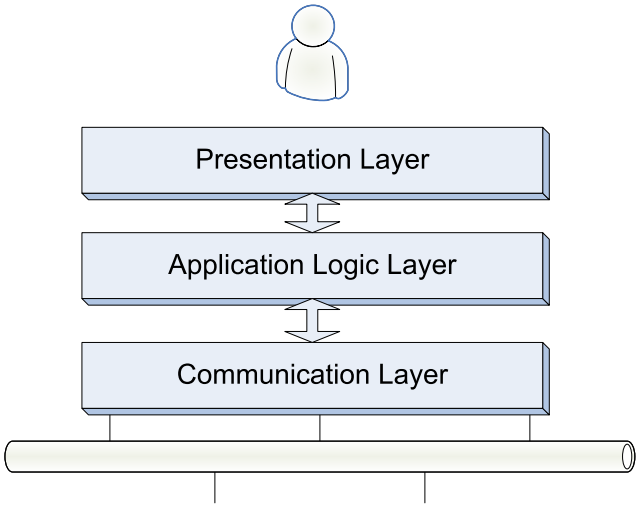


Figure 5.2: Three level design

5.2.1 Presentation layer

When working with presentation and interaction on a mobile phone there are several constraints that come into account, like limited screen size and limited input capabilities. Another important issue related to the presentation layer of a mobile application is that speed and reaction is critical, while people are more tolerant to lagging computers and internet connections they are used to fast responding mobile phones.

Designing a good User Interface (UI) for a mobile application is a difficult task. As this is a case study the important factor is functionality not usability, the most important issue is that a user can perform the task of the application, and most of the users testing the application will be high-end users not so dependent of high usability.

Key tasks handled by the presentation layer:

- Set parameters for register with/logging into server
- Creating a ThinkAlike session/request doing:
  - Capture an image
  - Associating a word to the image
  - Indicate a recipient
- In standby mode being able to initiate a received ThinkAlike request
  - Present the received image
  - Query for word suggestions
- Present result from session at both ends
- Enable chat function with input possibilities

- Enter stand by mode

5.2.2 Application logic layer

This layer will be "pulling the threads" in the application, maybe also doing it literally. This layer will handling input from either the presentation layer or the communication layer, meaning user interaction or interaction with another entity through the communication layer.

Key tasks handled by the application layer:

- Handle input from the user
- Initiate communication
- Handle multimedia conversion if necessary
- Calculate results from guessing
- Control the flow of a session
- Handle communication errors and user misuse

5.2.3 Communication layer

The communication layer will handle all the interaction between the application and the network. It will present the application logic layer with a set of possible functions that makes the application able to perform its purpose. This is the layer that will need to be modified into using IMS or XMPP.

Key tasks handled by the communication layer:

- Login/register with a central server
- Send/receive an application messages to a dedicated recipient
- Send/receive an image
- Initiate and participate in a chat session

5.3 Implementation

Because this thesis is meant to encourage rapid development of new services, it was important to make use of as much open source tools as possible, to make sure that the environment used could be recreated easily and at no or little cost for any developer.

5.3.1 Choosing a platform

When implementing a mobile application today there are a number of possible ways to do this, depending on what type of user you want to reach and how

complex features you want. Some of the available platforms are:

- Symbian OS
- Windows Mobile
- Java 2 Micro Edition
- Linux Mobile

Symbian OS is, among others, used by Nokia, the market leader in handset manufacturing, as the operation system on their phones and are provided in a simple form called Symbian40 on mass-market devices and a version called S60 for high-end phones with more features. Nokia provide a number of tools to develop applications on their Symbian handsets.

Microsoft has definitely entered the business user segment with its Windows Mobile operation system with a common look and feel from their desktop environment. It is also tightly integrated with their Outlook application. Their `.net` technology makes it easy to develop applications for their mobile operation system, but again, like with Symbian, limiting the application to be used on those specific phones.

Linux Mobile is growing at this moment, being adopted by more and more manufacturers, and being embraces by first runner up on the handset-market, and the leader in the US, Motorola. Because of Linux open source foundation it draws a lot of attention from developers. The OpenMoko initiative (OpenMoko June 2007) is a proof of that, building a handset totally open source. But then again, Linux still has not yet any large portion of the market.

On the contra, there is only one option that will reach 2100 million users[1]. Sun Microsystems (Sun), a software firm, took a different approach with it's JAVA programming language, under the slogan *Write once, run everywhere*. This is done by using a so called "'Java Virtual Machine"' that is ported to the specific running environment. A Java application will only run in the Virtual Machine thus meet the same environment everywhere, while the porting of the Virtual Machine only have to be done once for every environment. Sun launched the Java 2 Micro Edition (J2ME), a stripped down version of the Java programming language intended for use on devices with limited capabilities, like mobile phones. Today J2ME can be seen as the only true cross platform alternative for developing mobile applications on mass marked devices. Due to the facts mentioned above, the decision of choosing J2ME as programming platform was simple.

5.3.2 Environment

This section describes the working environment of the thesis, without going to much in detail on the physical location at Telenor R&I on Fornebu, the choice of handsets are explained as well as the programming environment. External

[1] According to the JavaOne conference in May 2007 there is 2100 million Java-enabled handsets

entities provided by Telenor for the thesis, like an IMS server and XMPP server are described in the Launch section 5.4. All the software programming was done on a machine with an Intel dual core processor and 4 GB memory running Microsoft Windows XP.

Figure 5.3: Physical environment

Handset selection

Because the variety of devices running J2ME are so great, there are some common configurations that every device have to meet, while additional features are available through extensions supported by the device. This could include features as image capturing or support for special security mechanism. The Java Community Process (JCP) handles all these extensions as a Java Specification Request (JSR). A JSR defines a specific set of methods available to the programmer when developing an application for a device. The ThinkAlike application will use the extensions given in table 5.1.

JSR-118	MIDP 2.0 meaning a generally well equipped phone
JSR-135	MMAPI 1.1 for use of multimedia, like camera
JSR-180	SIP for Java handling SIP on the handset

Table 5.1: JSRs in use

All but one of the JSRs in table 5.1 are very common extensions available in most came phones on the market. The JSR-180 is only available on the S60 operation system on high-end Nokia phones, meaning that an application using the JSR-180 will only run properly on a high end Nokia phone. However, there might be more handsets supporting the JSR in the future.

While most of the initial testing during development was done with emulators, three phones were used as both *on device debugging*[2] and for real environment testing. Those three phones were a Sony Ericsson W810i (W810), a Sony Ericsson W850i (W850) and a Nokia N91 (N91). The first two are mass-market standard phones, while the last is a high end phone. Most of the Implementation was done using the SE phones, while the Nokia phone was necessary to use the JSR-180.

(a) SonyEricsson W810 (b) SonyEricsson W850 (c) Nokia N91

Figure 5.4: Handsets in use

Developing environment

The experience of developing applications for mobile phones using J2ME can be quite frustrating. At least this is how it has been in the past was. Through the last couple of years a number of great tools have been made available for developing J2ME applications. Of popular Integrated Development Environments (IDE) like *Eclipse* and *NetBeans*, the latter has through it's Mobility Pack module extension made this process even easier. Both the IDEs were tested, but at the moment the NetBeans IDE is very superior to Eclipse when concerning J2ME application development[3].

IDE NetBeans is an open source IDE developed in Java. It has grown into a very good alternative to Eclipse, the long lasting leader among free IDEs. For this work the NetBeans version 5.5 was used. In addition the Mobility Pack Module was added to the IDE. This is a module that lets you easily build the

[2]When connecting the actual phone to the computer and running the application on it through the IDE, makes it possible to debug on the phone.

[3]In should be mentioned that Nokia provides a Eclipse plug-in for developing applications on their handsets, but the plug-in does not work well with other phones.

application for any kind of configuration you want. This module includes the Visual Design Editor, a tool that simplifies a commonly hard task in J2ME, designing the graphical user interface (GUI). It provides a flow view where one can drag and drop different screen elements to create an application flow, and a screen design view where one can add components each screen element. A screenshot of NetBeans showing the flow view can be seen in figure 5.5. The IDE is extremely easy to set up and includes the necessary Java Software Development Kits (SDK) and Wireless Tool Kit(WTK) from Sun.

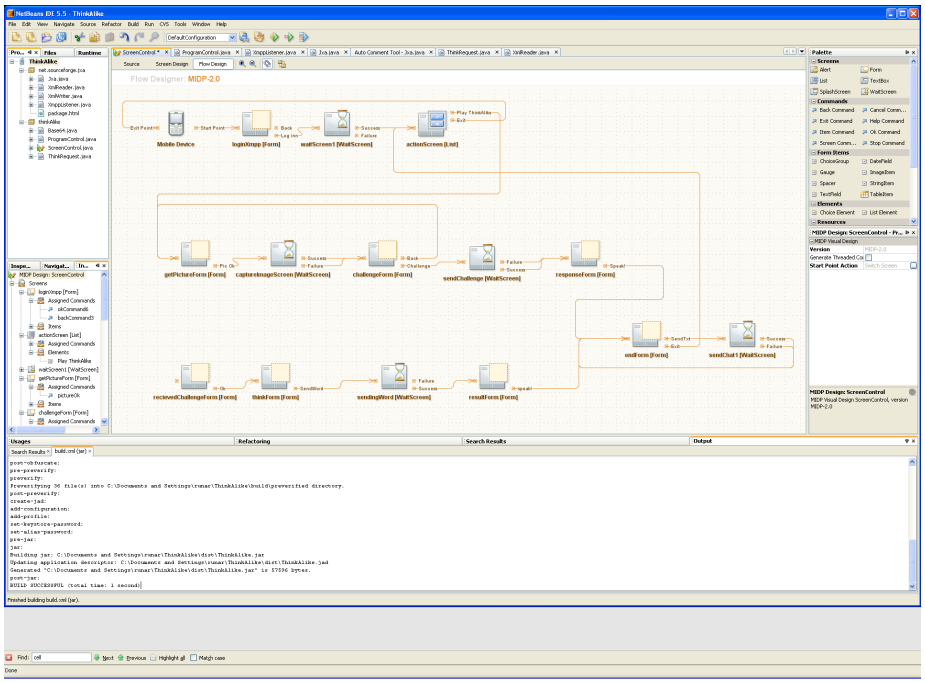


Figure 5.5: NetBeans IDE

Emulators An emulator is a piece of software that, as in its name, emulates the behaviour of, in this case, a mobile phone. A good emulator is important to test the application during development, but as every J2ME developer knows it is extremely important to test the application on devices at regular times to catch errors. Sun provides general emulators for specific types of features, but many handset manufacturers make their own emulators for developers to test their application on a specific phone.

Sony Ericsson (SE) has some very good emulators available, where good means behaving much like the actual phone. The W810 has it's own emulator, while the W850 is using an emulator for all SE phones that support SEs Java Platform 7. While SEs emulators are good at testing a J2ME application, Nokia provides an emulator that operates just like a phone but running on a computer, meaning that you can browse menus etc. on the emulator. For the N91 the S60 emulator is used. It proved to be very comprehensive but sometimes being unnecessary hard to work with, giving unreadable errors and freezing regularly.

(a) SE W810 Emulator (b) SE JP7 Emulator (c) Nokia S60 Emulator

Figure 5.6: Emulators in use

To use either of the emulators above one need to install the SDKs provided by either SE or Nokia, that are tailor-made for their phones. Implying that compiling an application using the SE SDK will not be a good solution if one wants to run the application on a non-SE phone. Using a manufacturer's own SDK will usually give a better-looking result than using the general SDK from Sun.

5.3.3 Presentation layer implementation

One of the first things you do when developing an application, after modelling and designing it, is to draw the screen flow of the application, easily done with the Mobility Pack in NetBeans. A simple communication application like this is mostly built around the interaction with the user. One decide where menus should be and what screen components, like input fields, that would be put on the screen. A close look at the result of this screen flow design is shown in figure 5.7.

NetBeans will auto generate code for the design made with Visual Design Editor, which means that the code will be a bit bigger that if one had written it from scratch, but the ease of it makes up for that in most cases. The Wait-screen shown in figure 5.7 are NetBeans specific screen elements that are used for tasks that could freeze the application, like communication tasks. This element follows a specific pattern to handle the task by launching it in a new thread. All these are NetBeans specific solutions to problems with J2ME. Figure 5.8 shows a number of screen designs that the presentation layer has implemented.

Java Classes

The presentation layer is implemented with one java class:

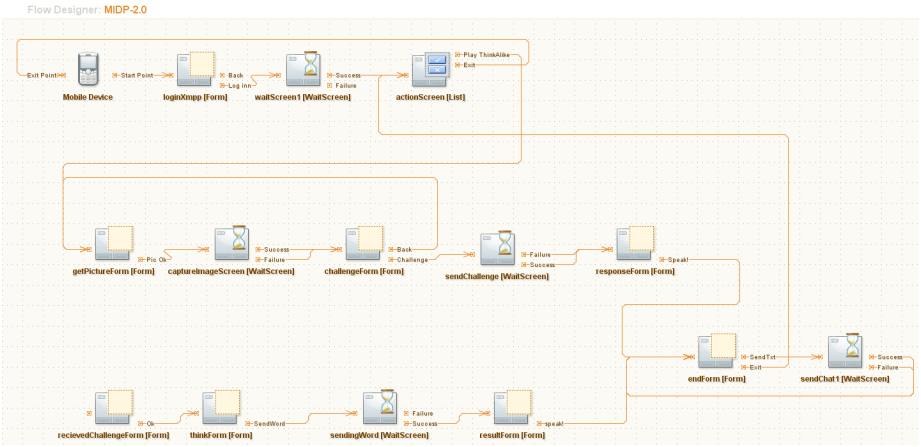


Figure 5.7: Application flow

(a) login (b) challenging (c) waiting (d) chatting

Figure 5.8: Screen designs

`ScreenControl.java`

This class is auto generated by NetBeans and includes numerous methods for modifying every screen element and component that is used. It also handle a lot of the transitions between screens, according to the flow scheme in figure 5.7. The only alteration made to the auto generated code was to indicate what methods was meant to run as separate threads under the wait screens.

5.3.4 Application logic implementation

Most of the screen flow is handled by the `ScreenControl` class, but the centre of the application can still be considered the `ProgramControl` class that handles most of the application logic and coordination between the layers. The user is queried for login in information at the start up of the application, `ProgramControl` then initiates the communication layer and requests login with the parameters provided by the user, the communication layer responds with success or failure.

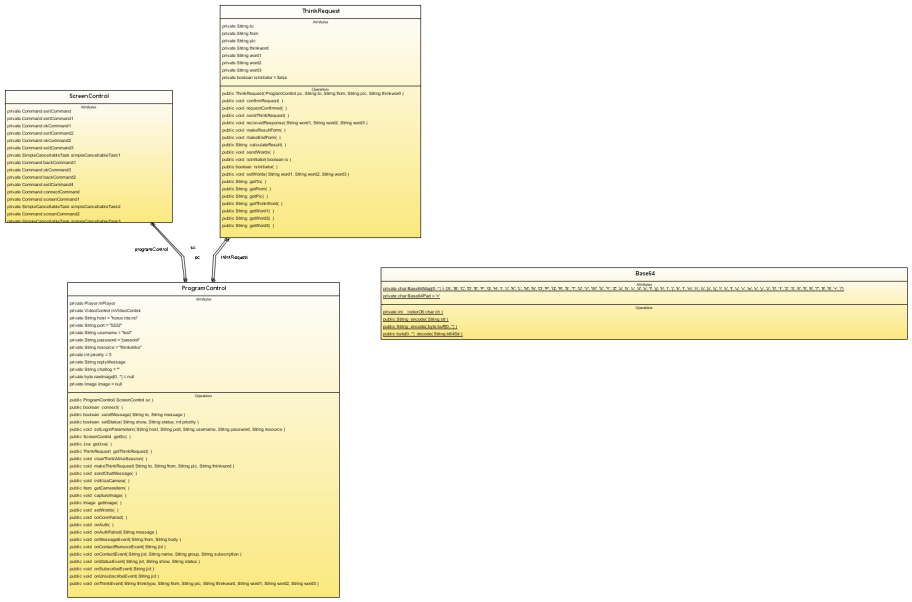


Figure 5.9: Application logic UML

Session class

When a user wants to start a game session with another user, the `ProgramControl` will collect an image and the associated word and then instantiated an object that will handle the session from the class called `ThinkRequest`. On the receiving end of the session a `ThinkRequest` object will be created when the request is received and it will notice the user that a request have arrived.

Image capture

The image is captured and stored in an `Image` object, but for easier transfer the raw bytes are converted into a String using a Base64 encoding. The `Base64` class used to do this encoding and decoding is written by Martin D. Flynn and released under the Apache License, Version 2.0, it was found using Googles Code Search[4].

5.3.5 Communication layer implementation

One object was to keep the application logic and communication layer as separated as possible. The object handling the communication will be instantiated in the `ProgramControl` object. Because of the problems with the IMS server, as mentioned in section 1.3, there has not been implemented any part communicating with the IMS server, but the process will still be explained.

Using XMPP

There has already been a number of different XMPP open source project released online, and it was desirable to make use of some of these in this project. It seems likely that someone already had implemented code that would make communication between an XMPP server and a J2ME client easier by providing a set of methods. I found this in the open source *JXA project* by Swen Kummer, Dustin Hass, Sven Jost, Grzegorz Grasza[5]. This toolkit provided most of the necessary functions, and was simple and easy to understand. An interesting fact is that this project use an XML-parser that was developed for another open source called *mobber*, making this a chain of open source projects linked together. An overview of the classes included in the JXA project is shown in figure 5.10.

Some modifications had to be done to the code to make it fit this project. The XML-reader and XML-writer was not fully compatible with the XML standard, making it necessary to do some alterations. These bugs were posted on the forum page of the project, with possible solutions. In addition the `Jxa` class was extended with two new methods handling the specific actions of sending and receiving a ThinkRequest. The `ProgramControl` implements the `XMPPListener` interface to react on signals received on the communication channel

Login When starting the application the user provides a `host`, `port`, `username`, `password` and `resource`. These are used to establish a connection with the XMPP server. The `Jxa` object will then compose a messages that will use the `XmlWriter` object to actually send the information over the stream to the server. There are no security mechanisms implemented, meaning that the password will be sent in clear text.

[4] http://www.google.com/codesearch
[5] http://jxa.sourceforge.net/

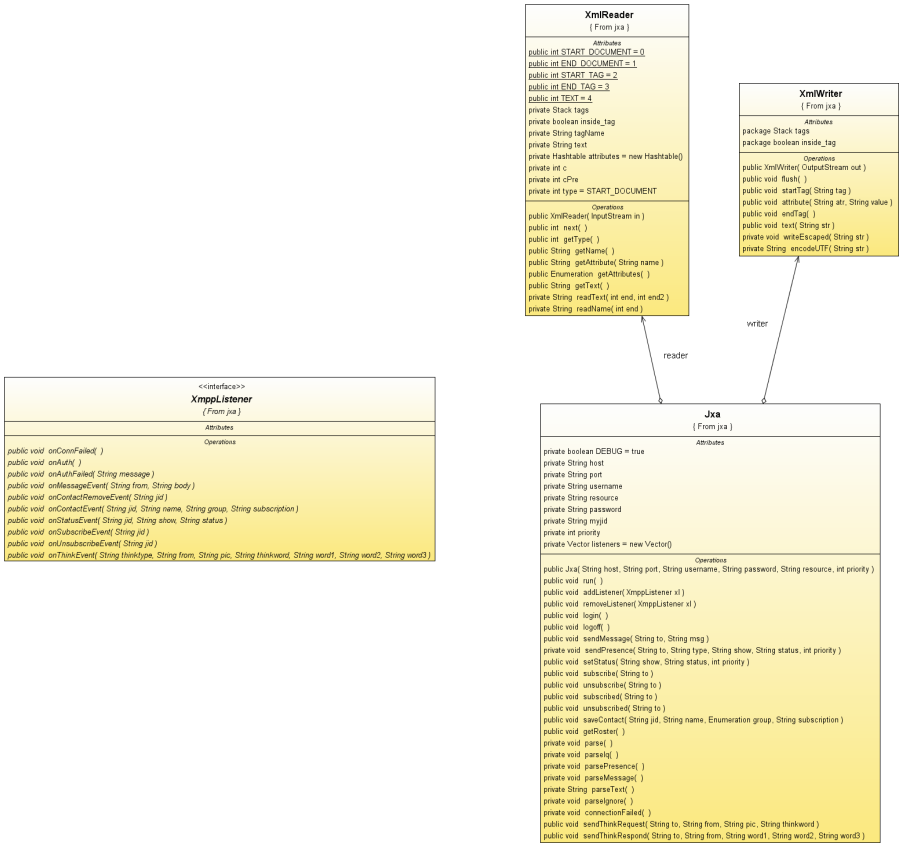


Figure 5.10: JXA classes

Image transfer One option for sending the image would be to upload it on a web storage and send the address of the file to the recipient that would then download the image. This method demands a storage service and is more complicated that the simple solution implemented in this case. As mentioned earlier the image is encoded as a string, this string is then transferred in a ThinkRquest message and then decoded back to an image at the recipient.

ThinkRequest message Although most of the communication between the peers is standard XMPP messaging, two custom made messages were defined. They are both *IQ* stanzas and therefore needs to be addressed to a recipient including it's delivery resource. One are the message sent from one user to another when he wants to initiate a ThinkAlike session, the other is the response from the other recipient. They both contain a `<thinktype>` tag that tells whether it is a request or a response. The two messages are respectively formatted like this:

```
<iq type='get' id='thinkalike' to='test@horus.nta.no/thinkalike'
from='test1' >
<query xmlns='com.runarg.thinkalike'>
<thinktype>thinkrequest</thinktype>
<pic>picturestringoftenveryveryveryloong</pic>
<thinkword>selveordet</thinkword>
</query>
</iq>
```

```
<iq type='get' id='thinkalike' to='test@horus.nta.no/thinkalike'
from='test1' >
<query xmlns='com.runarg.thinkalike'>
<thinktype>thinkresponse</thinktype>
<word1>orden</word1>
<word2>ordto</word2>
<word3>ordtre</word3>
</query>
</iq>
```

Those two messages are the only information exchanged between the participants before the chat session at the end. The chat session uses standard *message* stanzas. There is no presence functionality implemented in the application.

Using IMS

Changing the application to make use of IMS did not seem to be too big a task when starting. After all, when avoiding Application Servers, it was mostly a transition from XMPP to SIP. When programming SIP on mobile phones there are a JSR defined to handle that specific task. This JSR-180 provides a set of methods for handling SIP registration and session control, all of it clearly defined on the JCP webpage (Godeny & Seppanen June 2007). Most of the

testing so far had been done on the SonyEricsson phones, but when using the JSR-180 supported by non of the SE phones, the N91 was up for testing.

Nokia provides a number of sample applications for testing the SIP API provided by JSR-180. Working with these test applications and trying to connect to the P-CSCF showed that is was very hard to configuring the SIP REGISTER message right. Then the accident happened to the IMS server. Until then the server responded `403 Forbidden`, after the accident it responded, the not so good, `480 Temporarily Unavailable`, see figure 5.11.

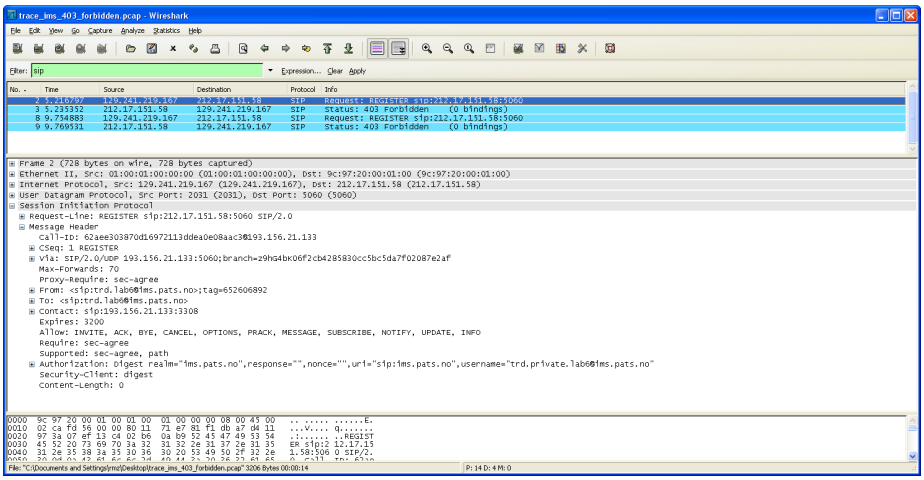


Figure 5.11: SIP REGISTER message

The application would have been a stand alone IMS application, keeping its own connection to the P-CSCF, the opposite of building an application on a IMS Client Framework as described in (Gunnerud 2006).

After the IMS server stopped responding, the development was postponed a couple of weeks in hopes of a working IMS. With still no sign of progress it was more important to finish the application as it was and focus documenting it properly. A week before this thesis was due, the IMS server was up and running, after a month out of service.

Because a test bed for IMS was unavailable, it is not sure that the image could be sent in the same way as XMPP, encoded as a string in a simple SIP message. The IMS server might have limited the length of the message even though the SIP standard does not impose any limitation.

5.4 Launch

This section will describe how the application would be launched on the two technologies, and the environment necessary to handle the service. It will also focus on the launch of the application as an open source project.

5.4.1 Launch with XMPP

The way the application is implemented at the moment, the only thing needed to use it is to have an XMPP user account at any Jabber server that lets you register one, and log in without any security mechanism activated. At this time the application does not make use of any security mechanisms. That means that it will not be able to connect to Googles XMPP server because Google requires the use of SSL login to it's server.

One could also set up ones own XMPP server, running e.g. the, near reference, open source server *Openfire*[6] provided by Jive Software(JiveSoftware June 2007). Telenor R&I has this XMPP server implementation up an running on `horus.nta.no`, and it was the one used for the testing. In the case of this application, all the server knows is that two users are logged in and sending messages between each other.

5.4.2 Launch with IMS

The IMS environment, available through Telenor R&Is PATS lab in Trondheim, is delivered by Nokia-Siemens. Without a full overview of the system features, the only thing needed was the ability to register/login and send simple messages between users, assuming the image could be transferred as an encoded string.

When trying to register with the IMS, problems occurred related to the configuration of the `SIP REGISTER` message. Within a week of trying, the server would not recognise the user that was provided for this testing. The problem was investigated in cooperation with Telenor R&I in Trondheim without finding any solution to why the IMS did not accept the registration even though the SIP message apparently was equal to the ones that was tested in Trondheim. Then the IMS server broke down.

5.4.3 Launching an open source project

Although a bit out of the scope of this thesis, it is in the spirit of the objectives that openness will contribute to rise the overall level of available services and applications. And because the ThinkAlike application already are using GPL licensed open source software, it is licensed GPL itself. As an experiment the application is launched as an open source project on *SourceForge.net*, a centralized location for software developers to control and manage open source software development. Hopefully this will make developers look into the XMPP as a platform for creating communication services in the mobile sphere. The project, with online javadoc, is available at:

`http://sourceforge.net/projects/thinkalike`

A demonstration video of the application is available online at:

[6] This implementation has evolved from the early Jabber reference server.

`http://youtube.com/thinkalikeapp`

5.5 Further development

This section sketches some possible features that could be implemented in the future. First of all *presence* should be implemented to see if ones friends are online or not. Then *higher security* is needed, both in terms of eavesdropping but also on issues like byte control, meaning that the user should control what is sent to her. At the moment the application is vulnerable to someone sending a large file to the application, thus costing money for the user.

The application could also be developed into using a central *game server*, most easily implemented as a chatbot, so that there could be more that one friend in a think request session. This also introduces an interesting possibility: if a system register the outcome of every challenge one could collect information of how well friends think alike, with possibilities to make *network graphs*, just an interesting though. In addition one would gain a large *database with words associated to pictures*, although these words might only be valuable in special social contexts.

Chapter 6

Discussion

Contents

This chapter discusses the solution and findings of the thesis. The first part will discusses the platforms and how they are compared. Next, there is a part debating the choice of application and the chapter finishes owith a critical view of the IMS problem.

6.1 Comparison framework

Because of the incident that put the IMS out of service, the grounds for comparison changed from what was intended at the start of this thesis. The comparison is not as strong as it would have been with the case study running on IMS, but the author argues that in depth knowledge of the platforms and experience from the case study still give valuable results.

When defining the criteria for comparison, the work done by Zarras (Zarras 2004) was put as a base framework. The three key requirements, as shown in table 2.1, were *openness*, *scalability* and *performance*. Based on practical experience of XMPP and a theoretical analysis of IMS, the following summaries and scores are given to the two platforms:

Openness

IMS Through developing this architecture the telecom industry hopes to open up its walled garden, at least just enough for people to get in, but not to let too many "leave the party". But as with other attempts, IMS has been criticised for not gaining the wanted openness that is needed to foster an ecosystem of third party developers(Magedanz 2006). IMS does use a number of open standards defined by, among others, IETF. With regards to service deployment, IMS scores average on term of easy deployment of new services, and low on upgrade and addition.

XMPP Born by an open source community, and still being fed by it, XMPPs architecture is truly open. By autonomous domains each server can easily develop new services, as long as they are compatible to the standards. Because of this autonomy, each domain can be upgraded and modified to follow trends, and easily add new extensions as they appear. Adding a new service does not even need to interfere with any server authority. XMPP scores high on easy deployment of new services, and high on upgrade and addition.

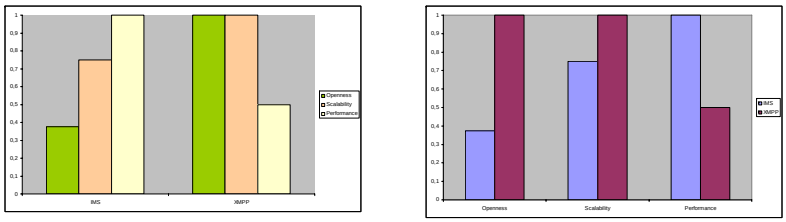
Scalability

IMS Operators have always been good at scalability due to the requirement of reliable networks. IMS is no different and its architecture can span the globe. IMS is a complex system, and could be overkill in smaller implementation, but all operators are in general big enough to handle the size. A popular application/service would run smoothly in IMS, although for deploying a smaller application it could be too much hassle to intervene with the complex system. IMS scores above average on scalability.

XMPP The simple client/server architecture of XMPP is robust and scales well out to many autonomous domains not relying on a centralized entity. Related to deploying new applications this thesis show that it can easily facilitate a very simple service. Even at greater scale, like GTalk(Google June 2007), XMPP executes effective operation of the service. XMPP scores top on scalability.

	Performance
IMS	The legacy of telecom has had high reliability and high performance. The IMS architecture ensures both efficient and predictable execution of the services it provides. By being able to charge, it also needs to ensure reliability. IMS scores top on performance
XMPP	By having a very distributed architecture the XMPP network does impose a high performance network. The fact that a service could be running on a single users computer, like a chatbot, increases the risk for unreliable services because in many cases there are no back up systems to handle breakdowns. Although this is not opposed by XMPP, it is neither encouraged. This makes XMPP fail to reach top score on performance, but average.

By giving each of the requirements above a value of one, where the two factors in *openness* count 0.5 each, we get a result graph as shown in figure 6.1. Although this framework is very simple, it gives a good overview of how the two platforms relate to the issue of getting new services available on the mobile handset fast and cheap. It would have given more information if the application had been used on a larger group of people, although there was not room for this in the timeframe.



(a) By platform (b) By requirement

Figure 6.1: Comparison results

The framework does not take into account one large incentive for development: revenue. But if one wants to foster a great ecosystem of third party application developers there might be other incentives to why developers should contribute. Some of these incentives, as stated in the famous open source article Cathedral and Basar(Raymond 1999), are more related to social recognition and entrepreneurial spirit.

Facebook, a social network site, is a recent example of how an application platform has grown quickly without any economic incentive for the developer and a great example of how openness rapidly fosters a large ecosystem. On the 27th of May 2007 they launched the f8 platform that lets developers create application integrated with the site. Within two weeks, more than 300 applications of different kinds were available, most of them categorised as *just for fun*. According to the Times(Richards June 12, 2007), by early June 2007, Facebook had 24 million users. Their most popular applications so far has been adopted by over

4 million users, with another 10 services having over 1 million users. A total number of adopted application was hard to find. However, this shows a massive take-up due to easy access for the users and easy development for those creating services, through a very open and well documented platform.

6.2 Case study choice

To challenge the platforms and to get a more in depth comparison it would have been necessary to develop a more complex application in the case study. But the ThinkAlike application does test some of the basic features of what we expect to see in new innovative services. A normal extension is described in section 5.5.

The use of NetBeans and its Visual Mobile Designer does increase the size of the code, compared to a tight packed custom made GUI. Some years ago, when defining what was important for an application, a large effort was put into optimising for a specific device and a high focus on bandwidth and so on, like Andersson states in the section *What affects applications and why* in his book from 2001 (Andersson 2001). But as seen, the operators that have deployed 3G networks, now got more bandwidth that they can handle and by the emerge of J2ME and similar technologies the optimisation for each device is no longer critical. This is changing the way one should think about mobile applications.

6.3 The IMS accident

One should always prepare a backup solution to problems that are likely to appear. This accident was not expected and the author argues that it was too unlikely to have a backup plan. If it would have been clear at the time when the incident happened that the system would be down for almost a month, a separate SIP system would have been put up to at least simulate some of the IMS functionality. But it seemed probable that the system would be up and running any time, a back up plan was deemed unnecessary.

Getting a functioning IMS would have made it possible to study response times and other factors related to the application, which now is only assumed to be quick enough to not cause any problem. It should be noticed that this is a laboratory implementation of IMS and does not reflect its vulnerability when implemented in the network. In the latter case there would have been alternative systems taking the place of the P-CSCF that went down. But it does reflect some of the complexity that IMS has, making it hard to find and replace an error.

Chapter 7

Conclusion

Contents

This chapter summarizes the most important findings and concludes this master's thesis. Last it suggests further work.

7.1 Main Results

The main objective of this thesis was, as stated in section 1.2, to:

> investigate if there is a faster and better way of delivering the same innovative services to the mobile handset that IMS intends to do by using a different approach and avoiding interaction with the operators platform. To verity this hypothesis, a service will be implemented in this alternative manner.

This section, through the four steps stated in section 1.2, will show how this thesis approaches and answers the objective.

> **Finding an alternative platform to launch the service.**
>
> Through identifying the needs for a platform to launch new services in section 2.1, it was clear that XMPP would provide the possibilities needed to be a real challenger of IMS as a place where developers wants to deploy their new innovative applications. XMPP is also thoroughly explained in chapter 4.
>
> **Defining criteria for comparison and a model application to be launched.**

It was important to define what were the key requirements for a service delivery platform, because IMS is such a large system only parts of it were to be compared with XMPP. Using parts of Zarras' framework (Zarras 2004) as a fundament when comparing the two platforms. Through the whole chapter 4, XMPP Theory and Comparison, the XMPP platform was mapped towards IMS to clarify the similarities and differences. This proved to be a great supplement to the use of the framework; after all it helps to answer the question in the objective.

By defining what is meant by an innovative service, it was important to investigate both how IMS and other sources predict these services to be. Through the work in section 2.2, Application Requirements, such services were defined and an application to fit the definition was introduced. This application was used as a case study in the thesis.

Develop, implement and launch the application using both technologies.

The ThinkAlike application was designed to test the application requirements, while still being simple enough for feasible implementation. Through the use of good developing tools and a clear goal, the service was up and running after three weeks of intense design and programming, using XMPP as a platform. As this thesis shows, the task of launching on IMS proved to be a little more challenging than expected, although some experience from connection and working with the IMS was gained.

Analyse the development experience and results from launch.

The development experience is extensively documented in the case study chapter 5, ThinkAlike Application, and the results from the launch is discussed in both section 5.4 and in the Discussion chapter 6. Even though it is an unfair comparison, it is almost ironic that the IMS broke down, leaving the only service available out of reach of the operator.

Through hands on experience and study of theory, this thesis has shown that IMS faces real challenges when it comes to launching new innovative applications and services on the mobile handset. It shows that alternative technology is available, and indeed present, that can provide a good platform for new communication applications in a way that avoids interaction with IMS. This thesis has also shown that interaction with the IMS, at the moment, is not an easy task.

This thesis does not predict the fall of IMS, but it does suggest that XMPP could partly replace IMS as a service delivery platform, thus disrupting the business model of IMS.

7.2 Further Work

A natural continuation of this work is to get new innovative services up and running, using IMS. The application developed in this thesis is fairly simple and therefore explores parts of the potential of both IMS and XMPP as a platform for delivering new services. Thus, there should be further investigations on how well XMPP challenges IMS when using application servers and more complex applications, to support the statement made in this conclusion.

Although not unfamiliar with the problem, it will become clearer to the operators that by entering the Internet domain and making use of its advantages, it becomes harder to dictate all the rules. Further work would be to find a solution to this problem. That could be either going backwards and closing off the gardens, and shutting down GPRS access for third party applications like some operators are doing in the USA (T-Mobile June 2007), or one could study new business plans and embrace the opportunity to be a bigger actor in peoples lives.

Bibliography

Alaoui, S. M. (June 2001), *TS 29.198 - Open Service Architecture (OSA) Application Programming Interface (API)*, http://www.3gpp.org/ftp/Specs/html-info/29198.htm.

Andersson, C. (2001), *GPRS and 3G Wireless Applications*, John Wiley & Sons, Inc.

B. Campbell, E. (December 2002), *RFC 3428 - Session Initiation Protocol (SIP) Extension for Instant Messaging*, http://www.ietf.org/rfc/rfc3428.txt.

Berry, N. H. (April 2005), *TS 29.278 - Customized Applications for Mobile network Enhanced Logic (CAMEL); CAMEL Application Part (CAP) specification for IP Multimedia Subsystems (IMS)*, http://www.3gpp.org/ftp/Specs/html-info/29278.htm.

Blanco, G. (December 2006), *TS 29.228 - IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents*, http://www.3gpp.org/ftp/Specs/html-info/29228.htm.

Calhoun, P. (September 2003), *RFC 3588 - Diameter Base Protocol*, http://www.ietf.org/rfc/rfc3588.txt.

Camarillo, G. & García-Martin, M. A. (2006), *The IP Multimedia Subsystem (IMS)*, John Wiley & Sons, Inc.

Copeland, R. (2006), *IMS or VoIP, Who will win?*, Moriana Group, pp. 24–40.

Day, M. (February 2000), *RFC 2779 - Instant Messaging / Presence Protocol Requirements*, http://www.ietf.org/rfc/rfc2779.txt.

Drage, K. (December 2006), *TS 24.141 - Presence service using the IP Multimedia (IM) Core Network (CN) subsystem*, http://www.3gpp.org/ftp/Specs/html-info/24141.htm.

FOKUS (June 2007), *Open IMS Playground @ Fraunhofer Institute for Open Communication System*, http://www.fokus.fraunhofer.de/bereichsseiten/testbeds/ims_playground.

Geddes, M. (2006), 'Don't feed the ims: it bites', *Mobile Communications International* **130**.

Godeny, B. & Seppanen, J. (June 2007), *JSR 180: SIP API for J2ME*, http://jcp.org/en/jsr/detail?id=180.

Google (June 2007), *GTalk, Google Talk*, http://www.google.com/talk.

Gunnerud, R. (2006), A study of the ims client framework. 9th semester study, ITEM, NTNU.

Handley, M. (April 1998), *RFC 2327 - SDP: Session Description Protocol*, http://www.ietf.org/rfc/rfc2327.txt.

Hildebrand, J. (February 2007), *XEP-0030: Service Discovery*, http://www.xmpp.org/extensions/xep-0030.html.

JiveSoftware (June 2007), *Openfire, Real time collaboration server.*, http://www.igniterealtime.org.

Laukkanen, M., ed. (2006), *Extenible Messaging and Presence Protocol*, University of Helsinki, Department of Computer Science.

Ludwig, S. (June 2007), *XEP-0166: Jingle*, http://www.xmpp.org/extensions/xep-0166.html.

Magedanz, T. (2006), 'Can the ims architecture deliver real network programmability', *Mobile Communications International* **130**.

Mayer, G. (March 2007), *TS 24.247 - Messaging service using the IP Multimedia (IM) Core Network (CN) subsystem*, http://www.3gpp.org/ftp/Specs/html-info/24247.htm.

Minlinski, A. (December 2005), *TS 23.002 - Network architecture*, http://www.3gpp.org/ftp/Specs/html-info/23002.htm.

OMA (2006), Oma push to talk over cellular v1.0.1, Technical report, Open Mobile Alliance.

OpenMoko (June 2007), *Worlds first integrated Open Source mobile communications platform*, http://www.openmoko.org.

Oyama, J. (September 2005), *TS 23.207 - End-to-end Quality of Service (QoS) concept and architecture*, http://www.3gpp.org/ftp/Specs/html-info/23207.htm.

Poikselka, M., Niemi, A., Khartabil, H. & Mayer, G. (2006), *The IMS: IP Multimedia Concepts and Services*, second edn, John Wiley & Sons, Inc.

Raymond, E. S. (1999), *The Cathedral & the Bazar*, O'Reilly.

Richards, J. (June 12, 2007), *Times Online: Facebook forced to sell stake after "hardware stuff-up"*, http://technology.timesonline.co.uk/tol/news/tech_and_web/article1898544.ece.

Roach, A. B. (June 2002), *RFC 3265 - Session Initiation Protocol (SIP)-Specific Event Notification*, http://www.ietf.org/rfc/rfc3265.txt.

Rosenberg, J. (June 2002*a*), *RFC 3261 - SIP: Session Initiation Protocol*, http://www.ietf.org/rfc/rfc3261.txt.

Rosenberg, J. (June 2002*b*), *RFC 3264 - An Offer/Answer Model with the Session Description Protocol (SDP)*, http://www.ietf.org/rfc/rfc3264.txt.

Saint-Andre, P. (December 2006*a*), *XEP-0053: XMPP Registrar Function*, http://www.xmpp.org/extensions/xep-0053.html.

Saint-Andre, P. (January 2006*b*), *XEP-0077: In-Band Registration*, http://www.xmpp.org/extensions/xep-0077.html.

Saint-Andre, P. (October 2004*a*), *RFC 3920 - Extensible Messaging and Presence Protocol (XMPP): Core*, http://www.ietf.org/rfc/rfc3920.txt.

Saint-Andre, P. (October 2004*b*), *RFC 3921 - Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*, http://www.ietf.org/rfc/rfc3921.txt.

Schulzrinne, H. (July 2003), *RFC 3550 - RTP: A Transport Protocol for Real-Time Applications*, http://www.ietf.org/rfc/rfc3550.txt.

Shigeoka, I. (2002), *Instand Messaging in JAVA, The Jabber Protocols*, Manning Publications Co.

T-Mobile (June 2007), *T-Mobile no longer supports 3rd party java applets*, http://my.opera.com/community/forums/topic.dml?id=189070&t=1181838462 &page=1#comment2038954.

Thayer, R. (November 1998), *RFC 2411 - IP Security Document Roadmap*, http://www.ietf.org/rfc/rfc2411.txt.

TheEconomist (April 28th, 2006), *A world of connections, a special report on telecoms*, Special Report.

Uskela, S. (2003), 'Key concepts for evolution toward beyond 3g networks', *IEEE Wireless Communication* .

Weilenmann, A. (2003), Doing Mobility, PhD thesis, Götenborg University, Department of Informatics.

Yates, M. J. (2004), 'Enabling applications deployment on mobile networks', *British Telecom Communcations Technology Series 9* pp. 241–252.

Zarras, A. (2004), 'A comparison framework for middleware infrastructures', *Journal of Object Technology* **3**(5), 103–123.