DEGREE PROJECT IN COMMUNICATION SYSTEMS, SECOND LEVEL
STOCKHOLM, SWEDEN 2015

Design of IP Multimedia Subsystem for Educational Purposes

MIKAEL RUDHOLM

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

Design of IP Multimedia Subsystem for Educational Purposes

Mikael Rudholm

2015-06-26

Master's Thesis

Examiner and Academic adviser
Gerald Q. Maguire Jr.

Industrial advisers
Henrik Leion (Enea Experts)
Pierre Östlund (Alten)

KTH Royal Institute of Technology
School of Information and Communication Technology (ICT)
Department of Communication Systems
SE-100 44 Stockholm, Sweden

Abstract

Internet Protocol multimedia subsystem (IMS) is an architecture for services such as voice over Internet Protocol (VoIP) in IP based communication systems. IMS is standardized by the 3GPP standardization forum, and was first released in 2002. Since then IMS has not had the wide adoption by operators as first anticipated. As 3G already supported voice and video, the operators could not justify the expense of IMS.

The current emergence of the fourth generation mobile communication system named Long Term Evolution (LTE) has, however, increased the need for knowledge of IMS and of creating services for it. LTE networks are IP only networks that provide low latency. In order to use LTE for making phone calls, VoIP technologies are needed. IMS is the architecture intended to be used for Voice over LTE (VoLTE).

The need for tools for education within IMS was seen in 2006 by Enea Experts in Linköping, Sweden. The author of this thesis designed an IMS for educational purposes, but the project was never fully completed.

This thesis will reexamine the design decisions previously made by the author. The requirements stated by the customer remain: that an IMS with basic signaling and logging should be easy to install, maintain, and evolve at a low cost. A literature study of IMS and VoLTE is presented to contribute with knowledge in these areas. The previous design and implementation made by the author is presented and analyzed. The third-party software that the previous implementation was based on is reexamined. Existing open source components are analyzed in order to identify how they can be used to solve the problem and to identify what remains to be developed in order to fulfill the requirements. New design suggestions, presented in today´s context, are proposed and verified using analytical reasoning and experiments.

The outcome of the final work is new verified design decisions for the customer to use when implementing a new IMS for educational purposes. The thesis should also provide useful insights which instructors and students can use to teach and learn more about IMS.

Keywords: *IMS, Next generation networking, Internet telephony, SIP application, service creation, virtual machines*

Sammanfattning

Internet Protocol multimedia subsystem (IMS) är en arkitektur för tjänster, som IP-telefoni (Voice over Internet Protocol, VoIP), i IP-baserade kommunikationssystem. IMS standardiseras av standardiseringsforumet 3GPP och första utgåvan släpptes år 2002. IMS fick dock inte det breda genomslag bland operatörer som förväntats. Eftersom 3G redan hade stöd för tal och video kunde operatörerna inte se skäl till ytterligare utgifter för IMS.

Den fjärde generationens mobila kommunikationssystem, Long Term Evolution (LTE) är helt IP-baserat och ger lägre fördröjningar i nätet. För att kunna ringa telefonsamtal via LTE krävs VoIP-teknik. IMS är en arkitektur avsedd för att användas för Voice over LTE (VoLTE). Den nuvarande utvecklingen av LTE har därför ökat behovet av kunskap om IMS och av utveckling av IMS-tjänster.

Enea Experts i Linköping insåg behovet av verktyg för utbildning inom IMS år 2006. Författaren av det här examensarbetet designade därför ett IMS för utbildningssyfte. Projektet slutfördes dock aldrig.

Syftet med examensarbetet är att ompröva de tidigare designbesluten. Kundens krav kvarstår: att ett IMS med grundläggande signalering och loggning bör vara enkelt att installera, enkelt att underhålla och möjligt att utveckla till en låg kostnad. Arbetet innehåller en litteraturstudie av IMS och VoLTE för att ge en inblick i dessa områden. Den tidigare designen och implementationen presenteras och analyseras. Tredjeparts mjukvara, som den tidigare implementationen baserades på, omprövas. Befintliga programvaror med öppen källkod analyseras i syfte att kartlägga hur de kan användas för att lösa uppgiften, samt att identifiera vad som återstår att utveckla för att uppfylla kraven. Nya beslut kring design presenteras och besluten verifieras med experiment och analytiskt resonemang.

Resultatet av detta examensarbete innefattar nya verifierade beslut kring design som kunden kan använda vid utveckling av ett nytt IMS för utbildningssyfte. Arbetet erbjuder också värdefulla insikter som instruktörer och elever kan använda för att undervisa samt för att lära sig mer om IMS.

Nyckelord: *IMS, nästa generations nätverk, Internet-telefoni, SIP applikationer, utveckla tjänster, virtuella maskiner*

I have been blessed with a wonderful family.
To my son Joakim, a future user of IMS – and
technologies replacing it, and to my wife Karin,
both with whom I prefer to communicate face to face,
in person.

Acknowledgments

Several persons have given me support, encouragement and advice during the work with this thesis, which I am very grateful for.

First of all I am indebted and greatly thankful to Professor Gerald Q. Maguire Jr. for his great support, quick responses, hard work and good suggestions in both small and big matters. He has taught me many things about academic writing. He never gave up on me, but continued to encourage me all the way through.

I would also like to thank my industrial advisers. Henrik Leion, my first advisor, for the idea of a thesis on the topic of IMS, and for support and feedback during the previous work. Pierre Östlund, my present advisor, for feedback and discussions about how to structure the thesis.

Kenan Avdic, Mats Carlberg, and Tengqingqing Ge have been of great help reviewing, and finding both language and factual errors. Current and previous colleges at Enea Experts and Alten have throughout the years taught me a lot that has been of great use during the work with this thesis. I am also grateful for the encouragement and support my current colleges have given me.

I also give my thanks to my family and friends for their support and encouragement, not only with the thesis, but in my whole life.

My wife Karin Rudholm brings great joy in my life and helps me to focus on other things than work and studies. During the work with this thesis she has given me great encouragement, excellent support with feedback on language and with the work as a whole, even though technology is neither her field of interest nor of knowledge. Thanks for bearing with me when working with this thesis has meant I have too often been absent.

Finally I would like to thank God for the strength he gives and the grace he shows me in all aspects of life.

Linköping, June 2015
Mikael Rudholm

Table of contents

List of Figures

List of Tables

List of acronyms and abbreviations

2G	Second generation of mobile telecommunications technology
3G	Third generation of mobile telecommunications technology
3GPP	The 3rd Generation Partnership Project
3GPP2	The 3rd Generation Partnership Project 2
4G	Fourth generation of mobile telecommunications technology
AAA	authentication, authorization, and accounting
AJAX	Asynchronous JavaScript and XML
AS	Application Server
B2BUA	Back-To-Back User Agent
CD	compact disk
CGI	Common Gateway Interface
CI	Continuous Integration
CPL	Call Processing Language
CRUD	Create, Read, Update and Delete are database actions that can be performed
CS	circuit-switched
CSCF	Call Session Control Function
CSS	Cascading Style Sheets
DNS	Domain Name System
GUI	Graphical User Interface
HSS	Home Subscriber Server
I-CSCF	Interrogating-CSCF
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IP	Internet Protocol
JSLEE	JAIN Service Logic Execution Environment
JSR	Java Specification Requests
LTE	Long Term Evolution
LTS	Long Term Support
MRF	Media Resource Function
MRFC	Media Resource Function Controller
MRFP	Media Resource Function Processor
OS	operating system
OTT	Over the Top
P-CSCF	Proxy-CSCF
PDF	Portable Document Format
PS	packet-switched
PSI	Public Service Identity
PSTN	Public Switched Telephone Network
PUI	Public User Identity
RFC	Request For Comments
S-CSCF	Serving-CSCF
SER	SIP Express Router
SIP	Session Initiation Protocol
SLF	Subscription Locator Function
SSH	Secure Shell
TCP	Transmission Control Protocol
UA	User Agent

UDP	User Datagram Protocol
UE	User Equipment
VMM	virtual machine monitor
VoIP	Voice over IP
VoLTE	Voice over LTE
WebRTC	Web Real-Time Communication
XML	Extensible Markup Language

1 Introduction

This chapter describes the specific problem that this thesis addresses, the context of the problem, the goals of this thesis project, and outlines the structure of the thesis.

1.1 Background

This thesis concerns the design of an Internet Protocol (IP) Multimedia Subsystem (IMS) for educational purposes. IMS is standardized by the Third Generation Partnership Project (3GPP). IMS is both the technologies and the architecture defined as part of the third generation cellular radio networks, which allow the use of multimedia services in packet switched services in a wireless environment. IMS is meant to facilitate the operator's work when creating new services, as compared to previous service frameworks. Camarillo and García-Martín have described IMS as: "It is designed to provide robust multimedia services across roaming boundaries and over diverse access technologies." [1]. IMS technology is further introduced in Chapter 2.

Enea Experts, where this thesis project was conducted, was a computer consultant company working in the telecommunications industry. Enea Experts initiated this thesis project in 2006 for several reasons, the first being that they offered theory courses in the area of IMS and the Session Initiation Protocol (SIP), which is closely related to IMS. Enea Experts desired to add laboratory exercises with IMS in these courses. Another reason for initiating this project, was to gain practical knowledge and experience with IMS to be used in other projects as well as with their customers. Finally, Enea Experts had experience working with technologies similar to IMS that are used to provide services in telephony networks. IMS was of interest as it at the time it was a fairly new technology that was expected to be widely adopted in the mobile world.

The author designed and implemented an IMS for educational purposes during the period of 2006-2008. The project was not fully completed and was never used in education for several reasons. The main reason was that the scope set by the author was not achieved within reasonable time and the author moved on to other tasks before the work was completely finished. Secondly, IMS was never a key area at Enea, and it became even less so when other key personnel with competence in IMS left the company.

As part of this work the author suggested that the product should be released as open source. This was not done for the same reasons as mentioned above.

With the emergence of the fourth generation mobile communication system named Long Term Evolution (LTE) this project once again became (more) relevant. LTE only provides a data network, i.e. LTE is an all IP-based network. In order to make phone calls over an LTE network, technologies such as IMS are required. Compared to the previous mobile technologies LTE provides lower latency [2, 3]. This enables third-parties, so called over the top (OTT) providers to offer even better services than with previous mobile technologies. Skype [4, 5], Facebook [6], and Viber [7] are examples of OTT providers with large user bases that provide video calling. Although IMS deployments were not rolled out at the rate first anticipated. However, with the emergence of LTE networks the interest in IMS has grown. Due to the threat of OTT providers and other competing technologies, the telecommunication industry started an initiative called Voice over LTE (VoLTE) that defines a profile with a minimum set of IMS functionality to facilitate interoperability [8]. The emergence of LTE networks and VoLTE makes IMS in education very interesting today.

Enea Experts was acquired by Alten in the end of 2011. Alten is the company at which this thesis project was completed. From here on Enea Experts and Alten will be referred to as the *customer*.

1.2 Problem definition

The problem addressed in this thesis is to reexamine the design decisions made in the work of "designing an IMS for educational purposes" previously prepared by the author at Enea Experts in 2006-2008. This reexamination will be done by considering today's conditions and with current knowledge of IMS.

The initial problem set by the customer was to define requirements for an IMS to be used in education, create a design, and then create and evaluate a prototype. The intention was to use existing open source software where possible and only develop new software when needed. The customer stated that the design of an IMS for educational purpose should comply with the following high-level requirements (see Chapter 3 for more details):

1. Easy to get started,
2. Easy to maintain,
3. Easy to evolve,
4. Low cost, and
5. The system should implement the following system functionality:
    a. IMS client registration (signaling only),
    b. IMS client to IMS client session (signaling only),
    c. example of usage of an IMS application (signaling only), and
    d. logging of traffic and application activity.

These high-level requirements stated by the customer will be taken into account during this reexamination.

1.3 Purpose

This thesis reexamines the design decisions made by the author when an IMS was previously designed for educational purposes. It also presents the previous design and implementation made by the author. This educational implementation of IMS is referred to as the *product* or the *system* in the rest of this thesis, even though it has never been released and might never be released as an actual commercial product.

This thesis project investigates if and how an IMS can be designed for educational purposes according to the requirements previously set by the customer for the product and with the knowledge of today's conditions. The intention is to give the customer insight in how a prospective new version of such a product could be designed. This reexamination might also benefit others seeking to create a similar tool for IMS education or other related education.

From an ethical point of view the reader of this thesis might benefit by gaining more insight into technologies used in global communications, both now and in the near future. This insight is important to the public, especially as at the current time there is a legal case in Belgium regarding Skype. The question is whether Skype is a telecommunications service and hence must provide access to call contents to law enforcement [9]. Additionally, the U. S. Congress is currently considering where to renew the US Patriot Act – which requires telecommunications operators to provide all of the metadata (i.e. the

signaling information) for all calls in their network. In order to have an informed public, more information about such issues would be useful. The public needs an understanding of just what this signaling information is and whether the service providers need have access to the keys used for encryption of the call contents.

As this work presents reuse of open source software, the reader might find ways of reducing costs by using existing software. The intention of this thesis is to stimulate increase of the use of IMS. A more widespread use of IMS could, for example, increase the use of high quality video calling worldwide. This could result in a reduction of business-related travel and thereby have a positive impact on our environment.

1.4 Goals and expected results

The goal is both to review the current design and find weaknesses, and to present design suggestions improving these weaknesses and therefore better fulfill the requirements set by the customer.

From the above mentioned goals the following results and artifacts are expected:

- a review of weaknesses in the current design, and
- verified design suggestions of how to improve these weaknesses.

1.5 Research methodology

This section introduces the methodology used in this thesis. As this thesis presents both previous and current work by the author, it will also present the methodology and methods used for both parts.

The customer who initiated the thesis project had an idea of the problem to be solved. This is presented in this thesis as the previous work. This problem was addressed using an engineering methodology. The work with the requirements and the implementation was done using an agile software engineering methodology.

The method used contained the following steps:

1. Discuss high-level requirements with the customer to create a more detailed specification of requirements;
2. Conduct a literature study;
3. Find possible third-party software to solve a part of the problem;
4. Evaluate third-party software or possibility to write new software;
5. Implement a prototype based on third-party software and, if necessary, software developed by the author; and
6. Final acceptance test in cooperation with the customer.

In reexamining the design decisions made in the previous work, an engineering methodology is used. The method is as follows:

1. Conduct a new literature study;
2. Critically analyze the requirements and existing design;
3. Evaluate the current state of the third-party software that was used in the previous work, where appropriate, based on analyzed requirements;

4. Propose requirement and design changes; and
5. Verify proposed changes using analytical reasoning and experiments.

Note that this thesis is a rare instance of a thesis that could evaluate the third of the customer's requirements (i.e. easy to evolve) – as time needs to elapse in order to assess how the third-party software would evolve.

1.6 Delimitations

This thesis will reexamine the design decisions made in a previous work by the author. New design suggestions are presented and verified. The new design suggestions are intended to enable the customer to implement a new version of the product. The actual implementation of such a product is outside of the scope of this thesis. The thesis will, however, present the previous design and implementation.

The set of high-level requirements defined in the previous work will be taken into account during the reexamination. Some of the detailed requirements may be modified in discussion with the customer. A motivation for such changes will be presented.

This thesis is an investigation of the design for an IMS to be used *only* for educational purposes. Therefore, the focus will be on the requirements and design suggestions for:

- The most basic essential parts of the core of the IMS,
- The basic functionality as specified by the requirements,
- Investigation of only the signaling in IMS,
- Tools and packaging to ease the system's use in education – but *without* creating a comprehensive or complete educational package, and
- A single example of adding a service to the core of IMS.

The following are out of scope for this thesis:

- The use of media (such as audio or video),
- The use of encryption (such as IPsec and TLS),
- Emergency calls and charging in the IMS,
- Implementing the new design decisions,
- Examine related work in detail*, and
- The release of a potential product, or parts of this thesis, as open source.

1.7 Structure of the thesis

Chapter 2 presents background information relevant to the thesis and related work. The background information includes an introduction to IMS and VoLTE, and related work of other IMS test-beds.

Chapter 3 presents the previous work as regards the requirements set by the customer. The chapter discusses the requirements and makes the high-level requirements more concrete. More detailed requirements for each of the components are created from the customer's requirements. A discussion of which existing components are chosen to build an IMS for educational purposes is also presented.

* Related work will be discussed, but this thesis is primarily a review of the design of the author's own work.

Chapter 4 describes the implementation made in the previous work, which was created to verify the requirements and design presented in the previous chapter. Design decisions to complement the design pre-study are also presented.

Chapter 5 presents a reexamination of the design decisions made in Chapter 4, suggests a new design and provides a motivation for it.

Chapter 6 analyses and verifies the result of the work given by the new design in Chapter 5. It discusses whether and how the goals of thesis have been met.

Chapter 7 aims to answer the problem definition given in the first chapter and what one can conclude from the results. Suggestions for future work are also presented. This chapter also includes some reflections relevant to the economic, ethical, and sustainability aspects of this thesis project.

Appendix A describes some of the customer's requirements in the form of signaling flows between the different IMS functions. A potential implementation is intended to support these signaling flows as they are stated in one of the requirements.

2 Background

This chapter provides basic background information about IMS. Additionally, this chapter describes VoLTE and related work of the design and implementation of IMS test-beds.

2.1 IP Multimedia Subsystem

The section does not aim to give a deep understanding of IMS, but rather to present an overview with complementary references to literature and online resources on the subject.

2.1.1 Why would you need IMS and what is IMS?

IMS can be used to allow communication between parties using different devices and multimedia for communication. Below is a quote of a story presented by Ericsson from an imaginary everyday life example:

> *While in a taxi from the airport, Anna calls her work colleague Andrew on his mobile number to discuss some issues with an important construction project. Anna activates the phones video mode so that she can show Andrew exactly what she is talking about. Andrew views the images on his mobile while they discuss how best to move forward. The two decide that they need a little help from their colleagues back in the office. Anna selects the project work group from her buddy list, sees who is available, and initiates a push to talk group session. John and Jeff answer that they have also been thinking about the problem, and have a few ideas that they would like Anna to look at.*
>
> *When she gets to the hotel, Anna starts her laptop computer, opens her personal buddy list and invites Andrew, John and Jeff to join a videoconference. John opens up a presentation and shares it with his colleagues. At the start of the videoconference, Andrew is still walking back to the office and participates on his mobile phone, but swaps to his PC when he arrives at his desk a few minutes later. [10p. 7]*

While the above story can sound interesting and appealing, this thesis will introduce the advantage of IMS from a technical and business perspective. Looking back on the development within mobile and fixed telecommunications one finds that the second generation (2G) of mobile telecommunications technology provided voice service and basic data services. The third generation (3G) introduced native data communication from the start as well as higher data rates. The voice services in 2G and 3G are circuit-switched (CS) just as in the public switched telephone network (PSTN) where a bidirectional circuit is created from the originating user to the terminating user. The data in 3G is transmitted natively using a packet-switched (PS) domain. The fourth generation (4G), which includes LTE, provides only a PS domain; hence there is no CS domain. The higher data rates coming with 3G allowed users to utilize services on the Internet such as web browsing, streaming movies, and calling using Voice over IP (VoIP) on their mobiles. [1, 8, 11]

The question is, then, why you would need IMS if you can use all these Internet-based services. According to Camarillo and García-Martín the answer is “QoS (Quality of Service), charging and integration of different services.” [1p. 7] With the packet-switching normally used on the Internet, data is transferred in packets with a best-effort approach. This is not desirable for real-time multimedia such as voice and video calls, as there can be long delays and limited bandwidth can lead to a poor user experience. IMS solves this

problem by allocating the required network resource during session establishment, to meet the desired QoS requirements.

The second problem that IMS solves is differential charging for multimedia services. Watching movies using a third-party Internet services can cause a lot of data transfer, which is charged by the operator per byte. With IMS this can be done differently as the service set up with the IMS operator's network knows what kind of multimedia session is used. The session could then be charged in alternative ways, such as per clip or per minute. IMS does not mandate a particular charging or business model, but rather enables different ways of charging and establishing services. This charging can be done online (i.e. credit based) or offline. Charging can be flat-rate, based on QoS, time, or some new metrics.

IMS provides operators the possibility to combine their own services with those developed by third-parties. This makes it easier to increase the number of services that they offer. The IMS standard uses Internet protocols to define the interfaces used when developing these services. This makes it easier for developers who are already used to developing services for the Internet.

Another reason for using IMS is interworking and avoiding user lock-in. Mobile phone users can use their choice of Internet services for communication. In order to use such a service the party that they want to contact needs to use the same service. This encourages closed systems that try to lock the user into their particular service. Interworking between such services are not in their service provider's interest. With the above background Poikselkä and Mayer suggest that IMS is needed and they defines IMS as: "a global, access-independent and standard-based IP connectivity and service control architecture that enables various types of multimedia services to end-users using common Internet-based protocols." [11p. 4] IMS is designed to enable users to choose between different operators and still have interworking and roaming between different networks. However, it should be noted that when operators provide additional services via IMS the same problems for lock-in occur and there is no 3GPP requirement for service portability for these additional IMS services.

2.1.2 History of IMS

Standardization of IMS is done by the 3GPP and was first part of their Release 5, which was frozen in 2002. The 3GPP was organized in 1998 by regional standardization bodies from Europe, Japan, China, South Korea, and the USA. Their mission was to build a third generation mobile phone system after the success of the second, GSM (Global System for Mobile Communication). Since then, the responsibilities of the 3GPP has increased to cover future standards of mobile phone technology. 3GPP maintains an frequently updated website from where their specifications can be downloaded [12]. IMS is only part of the 3GPP releases, as these release also include standards for the core network and the radio access networks.

IMS is built on several internet protocols that were created by another organization called the Internet Engineering Task Force (IETF). Working groups of the IETF have defined the protocols used in the public Internet. IETF and 3GPP are collaborating on the continued standardization of the protocols used by IMS.

There are other organizations, the major ones being TIPSAN, 3GPP2, and CableLabs, which have standardized their own variants of IMS that include other access technologies. Several parts of this standardization were contributed back to the 3GPP. This thesis focuses on the basics of IMS according to 3GPP standardization.

Releases after 3GPP Release 5 have fixed shortcomings as well as added other access technologies, such as wireless local area networks, fixed networks, cable technology, and LTE. Subsequent releases added important functionalities, such as emergency calls. [1, 8, 11]

2.1.3 Overview of the IMS architecture

IMS uses a layered architecture where signaling and media are separated. IMS mainly deals with signaling, while the media can be (and generally is) transferred through a different path than the signaling. 3GPP has standardized functions in IMS, rather than standardizing the actual nodes. Therefore, vendors have the freedom to combine several functions into a single node, or split one function among several nodes. The most important parts of the IMS architecture are shown in Figure 2-1.

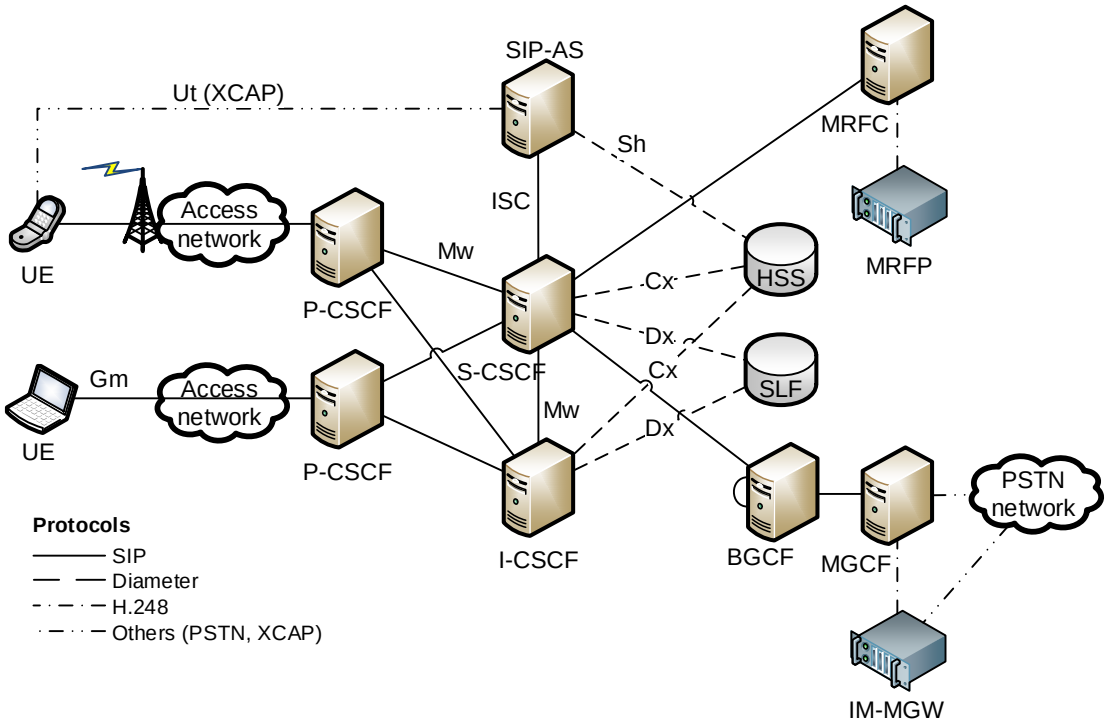

Figure 2-1 3GPP IMS architecture overview

3GPP has standardized the interfaces between the different functions and refers to them using two- or three-letter codes. A few examples of such letter codes are shown in the figure. A complete list of them can be found in specification 3GPP TS 23.002 [13].

Most of IMS is access independent. In Figure 2-1, a mobile phone and a laptop are examples of different types of IMS terminals commonly referred to as User Equipment (UE). As shown in the figure, UEs can use different access networks. The core of the IMS consists of the following categories of logical functions (as shown in Figure 2-1):

- session management and signal routing are handled by the most important function of the IMS, the Call Session Control Functions (CSCFs);
- a database called Home Subscriber Server (HSS) contains user subscription data and requests to this logical database can be load-balanced over multiple instances of this database by using the Subscription Locator Function (SLF);

- services are provided by one or more Application Servers (ASs) (labelled SIP-AS in the figure);
- media functionality, such as conference calls, mixing of streams and playback, is handled by the Media Resource Function Processor (MRFP) which is controlled by the Media Resource Function Controller (MRFC). Together they create the Media Resource Function (MRF) which is controlled by applications in the AS; and
- interworking with the PSTN is taken care of by the Breakout Gateway Control Function (BGCF) that decides where to interwork with the PSTN. The BGCF forwards traffic to the Media Gateway Control Function (MGCF) for signaling conversion where the interworking takes place. The MGCF controls the IMS Media Gateway (IM-MGW) to do media conversion.

There are some major IMS support functions that are not shown in Figure 2-1. These are considered not to be relevant for this thesis, and therefore only mentioned briefly below. The reader can find more relevant information in books by Camarillo & García-Martín and Poikselkä & Mayer [1, 11]. These support functions are used for:

- location retrieval of the UE for use in conjunction with emergency calls;
- border control functions for interworking between IPv4 and IPv6 networks and topology hiding;
- security gateways for confidentiality and integrity protection between different IMS networks;
- policy functions, used among other things in order to enforce the QoS in the access network; and
- charging functionality for online (prepaid) and offline charging.

Since this thesis focuses on the basic signaling in the IMS, the following sections describe the functions that are most important to signaling in more detail. The most important protocol in IMS is SIP, which is used for signaling. Section 2.1.8 briefly describes SIP and other common protocols used in the IMS. [1, 8, 11]

2.1.4 Call Session Control Functions

Signaling in IMS is handled by SIP servers that as a group are named Call Session Control Functions (CSCFs), which are the most important functions of IMS. There are three different kinds of CSCFs. Each user is assigned a serving CSCF (S-CSCF) during the registration of the user. The proxy CSCF (P-CSCF) is the first point of contact for a user when connecting their UE to the IMS network. The interrogating CSCF (I-CSCF) is the entry CSCF for signaling arriving from another IMS network. Each of these three CSCFs are described in more detail below. [8]

2.1.4.1 Proxy CSCF

The P-CSCF is the first point of contact in the IMS for signaling traffic from the UE. The P-CSCF is an inbound/outbound SIP proxy and all signaling to and from UE passes through the P-CSCF. The UE uses the same P-CSCF while registered. The main unique tasks of the P-CSCF are: SIP compression, IPSec security association, interaction with policy and charging function, and detecting emergency calls. [11]

SIP is a text based protocol. However, as the messages can be large, SIP compression must be supported and compression is done if the UE requests it. For example, this

compression can reduce the time needed when sending SIP messages over a mobile radio link.

The P-CSCF is responsible for authenticating the IMS client in the UE as part of the registration process. Once this is done, the P-CSCF guarantees to the rest of the IMS network that the IMS client in the UE is authenticated. The P-CSCF performs SIP protocol verification. Since the rest of the IMS network trusts the P-CSCF, they can also trust all of the signaling that they receive via the P-CSCF. During the registration process the IMS client in the UE and P-CSCF negotiate Security Associations (SAs) used by IPSec in order for the P-CSCF to apply integrity and confidentiality protection to all SIP signaling. More specifically, two unidirectional SAs are set up between the UE and the P-CSCF, then all SIP signaling is sent via these IPSec tunnels.

Interaction with charging and policy functions is also done by the P-CSCF in order to enable QoS and to correctly charge the user for the services and/or traffic they make use of. An IMS most likely contains several P-CSCFs in order to scale to support the subscribers to this IMS, and each P-CSCF serves many users. The P-CSCF can be located either in the user's home network or in the visited network. [1, 11]

2.1.4.2 Interrogating CSCF

The I-CSCF is the point of contact for signaling that is destined for a subscriber or services of the IMS network that the I-CSCF belongs to. The I-CSCF is a SIP proxy listed in the Domain Name System (DNS) of the external network to which this I-CSCF is connected. This DNS entry enables other users (and IMS networks) to know that this I-CSCF is an entry point to this operator's IMS.

It is the responsibility of the I-CSCF to find the next hop for the signaling. In order to do this the I-CSCF contacts the HSS to find out which S-CSCF to route the SIP messages for the designated IMS user or to which AS to route the SIP message to for a service.

During registration of a subscriber, it is the responsibility of the I-CSCF to assign a S-CSCF for that user, based on the capabilities requested by the HSS and the capabilities the different S-CSCFs provide. The I-CSCF can also encrypt parts of the information in outgoing SIP messages in order to hide the inner topology of the operator's network from external parties (such as the user or the currently visited network of this user).

2.1.4.3 Serving CSCF

The S-CSCF is a SIP server and SIP registrar responsible for registration of the user, and for providing routing and service to this user. When a user (via their IMS client) registers with the IMS it is the S-CSCF that contacts the HSS to get authentication data. Using this data it sends a SIP challenge to the UE. When the user is successfully registered the S-CSCF stores the IP address of the UE and maps it to the SIP address of the user. This particular SIP address is called the Public User Identity (PUI). The various identities are further described in Section 2.1.7. The S-CSCF uses the Diameter protocol to communicate with the HSS. During registration, the S-CSCF downloads the user profile and informs the HSS that this specific S-CSCF is now responsible for the particular user.

The user profile contains the service profile which is used to determine when and how a service is to be triggered. This is done by routing the signaling to one or more ASs, based on the initial filter criteria stored in the service profile. The service profile may also contain policy information about which types of sessions the user is allowed to use, e.g., only at best high quality audio and no video. Such policies are enforced by the S-CSCF.

All signaling destined to and coming from an IMS UE passes the assigned S-CSCF. Based on the destination address this signaling may be routed to an AS, e.g., for a group conference call. The S-CSCF can also route the signaling to the PSTN using a BGCF, or route it to another IMS network. Signaling destined to the UE is also sent through the P-CSCF, which handles encryption and SIP compression. The IMS network most likely contains several S-CSCF for scalability and redundancy. The S-CSCF is always located in the home network of the user.

2.1.5 Databases: HSS and SLF

The HSS is a central database for all user*-related data. The data for one user is stored in one HSS. In large IMS networks the load of many users can be shared across several HSSs. A logical function that needs user information, queries the SLF in order to learn which HSS holds the user data for a particular user. The HSS and SLF use the Diameter protocol with a specific IMS application.

The HSS stores subscription information, location information, and which S-CSCF the user is currently allocated to. The HSS also stores the user profile that the S-CSCF retrieves during registration. The user profile contains information about the services that this user is subscribed to. Security information, used for user authentication (as well as ciphering and integrity protection of the communication) is also stored in the HSS.

The I-CSCF queries the HSS to learn which S-CSCF is allocated for the user. The services in the AS can store application specific information related to a user in the HSS. [1, 11]

2.1.6 Application server

An application server can be used to provide services, e.g., group conferencing, user presence, and answering machine. There are three different types of ASs specified in IMS: two of them are for integration with service functions with older technology and the third is a native SIP IMS. In this thesis we will only present the native SIP-AS. An AS can be located in the operator's network or at a third-party service provider's network. If the AS is located in the operator's network, then it can communicate with the HSS to retrieve information about the user and store application specific data. Signaling is routed to the AS via the S-CSCF or the I-CSCF. The AS can be used to:

- process, change, and terminate (answer) incoming SIP sessions,
- originate (start communication), and
- send accounting information to the charging functions.

A single AS might provide one or more services. A given user may be using more than one AS. It is also possible that more than one AS is involved in a single session. The AS can work in different modes:

- Terminating/Originating SIP User Agent (UA): The AS acts as a SIP server/client user agent, while the SIP client/server user agent is running in the UE. The AS could be used as an answering machine or a wake up alarm (for the originating case).

* subscriber

- SIP proxy: The AS can modify the session by adding, removing, or rewriting to the SIP headers or SIP request body, and then routes the session back to the S-CSCF.
- SIP Back-To-Back User Agent (B2BUA): The AS acts as a SIP user agent server and answers the incoming session and then subsequently initiate a completely new session (acting as a SIP user agent client). This new session is sent to the S-CSCF.
- SIP redirect server: The AS tells the originating SIP user agent client that the user can be found elsewhere or that another service could be used.

Each AS can have its own identity, which is called a Public Service Identity (PSI). The user would then be able to initiate a session with the PSI of the AS, i.e. when acting as an answering machine. [1, 11]

2.1.7 User equipment

In general terms the UE is a device that can authenticate and communicate with IMS using application software. When using a mobile terminal as IMS client/server the device will contain a Universal Integrated Circuit Card (UICC), commonly referred to as a SIM-card with several applications. The UICC contains a SIM application used for 2G networks, the USIM application used for later generations of mobile networks, and the IP multimedia Services Identity Module (ISIM) application used for commutating with IMS. Information from the ISIM is used for authentication with IMS. If there is no UICC present, there are other methods available for authentication depending on what the network supports. This section will only describe the concepts behind using a UICC.

The ISIM contains the domain name of the operator's network and one IP Multimedia Private Identity (IMPI). The IMPI is used for authentication and internal bookkeeping by the operator. There is also one or more IP Multimedia Public Identities (IMPU) stored in the ISIM. An IMPU is used to contact the subscriber and could be written on a business card. The IMPU is a uniform resource identifier (URI). The IMPU can be specified either as a SIP URI or as a TEL URI. The SIP URI can for example take the following form "`sip:firstname.lastname@operator.com`" (as defined in [14]). A TEL URI can be used to specify a telephone number, such as "`tel:+46-8-790-6000`" (as defined in [15]).

The USIM can be used by the network to derive an IMPI and the rest of the relevant information if the user has not upgraded to a UICC that contains an ISIM application [1, 8].

2.1.8 Protocols involved

The following subsections briefly describe the most important protocols used in IMS.

2.1.8.1 Session Initiation Protocol

SIP is the most important signaling protocol in IMS. SIP is used for endpoints, called user agents (UAs), in order to locate and agree on properties of a multimedia session. SIP is used between UAs, CSCFs, and ASs. SIP was developed by the IETF. SIP is similar to HTTP and SMTP, in that it is a text based client-server protocol. SIP is specified in RFC 3261 [14] together with several later IETF RFCs that extends it. One important such extension is RFC 7315 [16] which adds features needed by IMS. 3GPP has standardized how SIP should be used in IMS and their most important specification is TS 24.229 [17].

Since SIP is a text based protocol it is easy to read and debug, but the messages can become large. SIP is a client-server protocol, where the client sends a request and expects a response from the server. A single SIP entity can be both client and server. In the path followed by a request there can be several hops between SIP entities before the request reaches its final destination. SIP is designed to be transport protocol independent. It can therefore be used either with Transmission Control Protocol (TCP) or, most commonly, with the unreliable User Datagram Protocol (UDP). SIP has its own handling of acknowledgements and retransmission to provide end-to-end reliability.

SIP clients use requests, such as REGISTER, INVITE, and BYE and the server responds to requests with a numerical response code and a reason phrase. The response code can be:

- Final, such as “200 Ok”;
- Provisional, such as “180 Ringing”;
- redirection responses where the client needs to take further action, such as “301 Moved Permanently”; and
- errors, which are given with 4xx, 5xx and 6xx, such as “486 Busy Here”.

SIP messages consist of a request/response line, headers, and optionally a body. The body of an INVITE message most likely contains a description of the media to be used expressed using the Session Description Protocol (SDP) which is further described in the next section. Figure 2-2 below shows an example of a SIP invite message (quoted from RFC 3261 [14]).

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

Figure 2-2 Example of SIP INVITE request from RFC 3261 [14]

SIP defines several different logical entities. The UA client is the endpoint that initiates a session. In IMS this end point can be located at the UE or an AS. There is a SIP registrar server which stores location information for the UA’s public identity, which indicate where and how this UA can be reached after the UA has registered. In IMS the S-CSCF is the registrar. A SIP proxy server can act as both server and client, routing messages on behalf of UAs. This proxy can modify the session and enforce policy by adding, modifying, and removing headers. A redirect server can respond to a user that the service requested can be found elsewhere. A B2BUA can in response to one dialog initiate a new dialog with another UA. The logical functions in IMS that use SIP are often a combination of SIP proxy servers and B2BUA [8]. Special SIP servers can also provide services, such as presence information, using an extension for SIP specific event notifications [18].

2.1.8.2 Session Description Protocol

SDP is used to describe multimedia sessions, for example, SIP sessions. SDP is important since SIP does not by itself describe these multimedia sessions. SDP is specified by the IETF in RFC 4566 [19]. A SDP body contains media type, CODEC, bitrates, and the IP address and port numbers to be used to send media to an endpoint. The base SDP is extended with an offer/answer model, specified in RFC 3264 [20], that is used in IMS. With the offer/answer model the parties in the session agree upon the parameters to be used, based on their capabilities.

2.1.8.3 Diameter

The Diameter protocol is used for authentication, authorization, and accounting (AAA) in IMS. The Diameter protocol consists of a base protocol defined in RFC 3588 [21] which is extended with so-called Diameter applications. There are several different Diameter applications used in IMS. The CSCFs communicate with the HSS and SLF, using the Cx and Dx interface [22, 23] respectively. The ASs communicate with the HSS using the Sh interface [24, 25]. Other parts of IMS use other Diameter applications.

2.1.8.4 Media protocol

Real-time media, such as audio and video during a session, is transferred using the Real-Time Transport Protocol (RTP). The RTP Control Protocol (RTCP) is used to provide monitoring and feedback on the quality of the RTP media stream. Both RTP and RTCP are defined in RFC 3550 [26]. Typically RTP is transported on top of UDP. Since UDP is an unreliable transport, packets might be lost or arrive out of order. RTP does not expect the delivery to be in order, but can reorder the RTP packets by using the sequence numbers included in the RTP header of the packets. The feedback from RTCP can be used to adjust the bitrate of the media stream and it reflects the status of the playback buffer on the receiving side. In IMS QoS can be used to assure that the required bandwidth is allocated to the session.

2.2 Voice over LTE

The number of LTE subscriptions has increased worldwide and is forecasted by Ericsson to continue to grow. In November 2014 the number of subscriptions had more than doubled during the previous year, from 208 to 424 million. Figure 2-3 shows the number of subscriptions for the most common mobile technologies, with historical data for 2011 – 2014 and forecasts to 2020. [27]

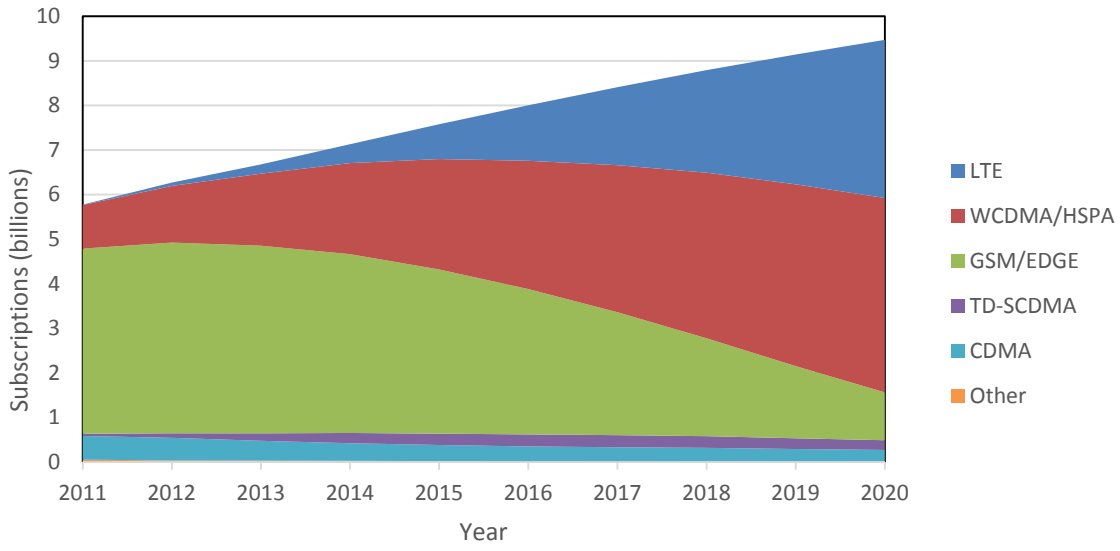

Figure 2-3 **Number of worldwide subscriptions of different mobile communication technologies (data from [27])**

With an increasing number of LTE subscriptions there is a need for services for LTE. VoLTE is considered as the "... likely long-term solution for LTE voice calls, namely the delivery of voice over IP streams that are controlled by the IP multimedia subsystem (IMS)." [8p. 372] LTE is designed without a circuit switched domain and voice calls are intended to be transported over the IP network of LTE. This can be done using IMS, but it is complex and has many options for implementation. When LTE was introduced several operators and equipment vendors feared that the complexity of IMS would delay its acceptance. They also feared that competing technologies and OTT would be chosen by users instead of IMS.

With this fear in mind an industry initiative known as, One Voice, was formed in 2009. The aim of this initiative was to create a minimum profile for voice over LTE using IMS to facilitate interoperability. This initiative gained popularity and the work was moved to the GSM Association, which is the telecommunication industry's main trade organization. The profile became known as Voice over LTE (VoLTE) by the GSM Association. The scope of VoLTE has increased to provide a profile for end-to-end voice and SMS. The idea is that if there are fewer options to choose between when implementing IMS, then IMS will more likely be adopted. Having a common profile will also make it easier to enable roaming between operators. [8, 28]

VoLTE is defined in three documents. There is the profile for voice and SMS [29], and guidelines for roaming and interworking between networks. [30, 31] These documents reduce the number of choices that 3GPP has given in their specifications of IMS. One example is that IPv4 and IPv6 both need to be supported instead of, as 3GPP specifies, the possibility to support both IPv4 and IPv6, or just one of them. A second example is that support for emergency calls are mandatory instead of optional. Several vendors of mobile phones and network operators have adopted VoLTE and are deploying it in all parts of the world. [32, 33]

2.3 Related work

This section presents related research of implementing an IMS core network to be used as educational tools or test-beds.

2.3.1 Open IMS Core

Vingarzan, Weik, and Magedanz from the Fraunhofer Institute FOKUS presented the Open IMS Core project [34] with the aim to implement a prototype of the IMS core network that is conformant to 3GPP Release 6 and is based on open source tools.

The core IMS functions implemented were the three CSCFs and a lightweight HSS. Other network functions were kept out of scope. As a foundation for implementing the CSCFs the SIP Express Router (SER) [35] was used. SER was selected as it “can act as SIP registrar, proxy or redirect server and is capable of handling many thousands of calls per second.” [34] SER is written in the C programming language and runs on Unix-based systems, such as Linux and BSD. SER can be configured with rules for SIP processing. Functionality specific to the CSCFs was implemented as dynamically loadable modules to SER which could be controlled using the configured rules. The challenge faced was to be compliant with the specification while still allowing good performance.

The P-CSCF was implemented to act as a firewall for the core IMS at the application level. By intercepting the registration process and keeping its own registry, the P-CSCF only allows registered users to access IMS. After the registration process succeeds the P-CSCF subscribes to registration notifications from the S-CSCF. The P-CSCF also strips UE originated messages of potentially forged information to reduce the risk of an attack. By being a member of both the access network and core network, the P-CSCF can act as a network address translation router.

The I-CSCF was implemented as a stateless proxy with an interface to the HSS. Based on the information in the requests, it queries the HSS and routes the SIP messages to the correct S-CSCF. The I-CSCF also implements firewalling in order to only allow messages from trusted networks.

The S-CSCF implements the interface to the HSS to be used as described in Section 2.1.4.3. For authentication it implements IMS Digest AKA version 1 and relies on the HSS to generate authentication vectors which are compared with the ones generated by the ISIM in the UE. The S-CSCF implements the registry function and other CSCFs, such as the P-CSCF, can subscribe to it for information about changes in registration states.

The HSS is implemented in C with a multi-threaded architecture. It is mainly glue to the underlying database system. The HSS implements the Diameter protocol and logic. It does not keep any state other than the contents of the database. The database system used is the well-known open source tool MySQL.

Performance goals were set to meet the needs of a fictive network with a hundred thousand subscribers. Real-life values of subscriber activity resulted in a need to support up to 11.57 calls per second. In their measurements the system was able to handle 17 calls per second.

The Open IMS Core project was published as open source [36] and evolved in other research. Vingarzan presents the whole journey in his doctorate dissertation. One of the later major changes in the architecture described above is that the HSS was rewritten in Java and a Web-based graphical user interface was created for it [37].

2.3.2 Other IMS test-beds

Fabini et al. present their work on the CAMPARI testbed where a minimal IMS for “measurement-based performance evaluations” [38] was implemented. In this work, IMS is based on a fork of SER called OpenSER (renamed to Kamailio [39]). In addition to the core IMS network functions some ASs, as well as functions for charging, were implemented. Moreover, different delays in the network between different operators, and between a UE and IMS network, were emulated. In parallel with the CAMPARI IMS testbed, where each IMS function is executed on a separate computer, Fabini et al. present “IMS in a bottle”, which uses a single PC and the VMware Workstation virtualization product.

Galán et al. present the design and implementation of an IMS emulator [40] also based on SER [35] where their focus was on virtualization of the test-bed. In this work, a tool called Virtual Network User Mode Linux [41] was used, which allows networking between different Linux instances that are executed in user mode. They found that using virtualization, the complex network setups are more affordable. Moreover, development can be focused on implementing functionality rather than configuring the network backend.

Peterström in his master’s thesis “IP Multimedia for Municipalities: The supporting architecture” [42] describes Ericsson’s IMS in the Box (IITB) and how it could be implemented in a virtualized environment.

3 Design pre-study during previous work

This chapter presents the design pre-study, as part of the previous work made by the author in April 2008. The following questions will be asked: What is needed in order to use IMS in education? How can one get acquainted with IMS without requiring a great deal of knowledge beforehand? The background chapter shows that IMS consists of several logical functions. What functions are needed to set up a basic IMS for educational purposes?

This chapter presents the high-level requirements set by the customer. Together with the customer, these high-level requirements were rewritten into more concrete requirements. These concrete requirements are used in order to select what to include in the project. The requirements specified in this chapter may result in the use of existing third-party software or the development of custom software. The selection process of what functions are included in the project and what third-party software that is used, is also part of this chapter. Some of the concrete requirements were further refined by the author to formulate detailed requirements. Some of the detailed requirements are specified in Section 3.3, where third-party software is chosen. The requirements set in this chapter are verified by an implementation described in Chapter 4. The design decisions made in this chapter are also complemented by design decisions made in the following chapter.

3.1 High-level requirements

The customer set some high-level requirements to be fulfilled in the resulting product. These high-level requirements are given in Table 3-1. However, these high-level requirements are hard to measure and therefore were seen as guidelines to form more concrete requirements. Together with the customer these high-level requirements were broken down into more concrete requirements, as shown in the following subsections.

Table 3-1 High level requirements

Easy to get started	It should be easy for a beginner to get started learning IMS and to use IMS related software. The user should be able to gradually increase the knowledge.
Easy to maintain	Developers should find it easy to maintain the resulting product. Moreover, the product should be built with modules, with good documentation and be well packaged.
Easy to evolve	The product should be able to continue to evolve beyond the scope of the degree project.
Low cost	There should be a preference for the use of existing open source software in order to decrease development costs.
System functionality	The system should implement the following functionality: a. IMS client registration (signaling only), b. IMS client to IMS client session (signaling only), c. Example of usage of an IMS application (signaling only), and d. Logging of traffic and application activity.

3.1.1 Easy to get started

In order for the product to be easy to get started, it should be:

Easy to install	This calls for a good documentation as well as a limited amount of steps to perform the installation. As the product will contain many different components, it is best delivered with everything preinstalled and configured into one package In summary, to make the product easy to get started with: • it should contain installation guides and documentation with step-by-step instructions of how to install it, and • the product should be delivered as an installation package in order to reduce the number of (manual) steps needed to install it.
Easy to manage	In order for a beginner to manage the product, documentation is essential. To further ease the managing of the product, a management interface should be part of the product, which should also be possible to use in order to control the IMS components and to gradually increase the knowledge about the IMS components. In order for the product to be easy to manage, the following should be included: • documentation describing the contents of the product and how to operate it and • a management interface, giving an overview of the system.

3.1.2 Easy to maintain

In order for the product to be easy to maintain, it should use:

Existing software	This should reduce the time needed, and thereby focus on the value adding parts rather than on developing all the components. Altogether this should make it easier to maintain a large product.
Automated installation	Script parts or all of the installation, making it easier to upgrade the third-party software when new versions are released.
Modular architecture	This makes it possible to separately maintain each module or component. Modules can also be added or replaced as needed.

3.1.3 Easy to evolve

In order for the product to be easy to evolve, the following is considered to be important:

- Use existing software to decrease the amount of development needed, which will help the project evolve faster.

- Good documentation will make it easier for new contributors to work on the project.
- Recommend to release the product as open source, since this will enable a larger number of users to contribute to the project and debugging is a parallelizable activity [43].
- Use open source software components to be able to release the whole package as open source.

3.1.4 Low cost

In order for the product to be easy to evolve, it should:

- Use existing open source software, which will lower development costs.
- Be recommended to be released as open source, in order to facilitate the product's evolution and to get assistance from the community.
- Use a master's thesis student to perform the work as they are significantly cheaper than experienced people.

3.1.5 System functionality

The *System functionality* requirement was described in further detail by the author.

The following are detailed further in Appendix A using signaling flows:

- IMS client registration (signaling only),
- IMS client to IMS client session (signaling only),
- Example of usage of an IMS application (signaling only),

The following are detailed further in Section 3.2:

- Logging of traffic and application activity

3.2 Requirements and design decisions

In this section some of the above mentioned concrete requirements are described in detail, as seen as necessary by the author. It is complemented by the requirements and choices made in Section 3.3.

3.2.1 Application logging and traffic monitoring

Each application or component involved should provide a log of tasks performed by the application. This will simplify the process of problem solving and to provide a better understanding of the system and its dynamic behavior. The applications should therefore send relevant information to a log file, or similar mechanism.

Since this project creates a product to facilitate learning about IMS, it should be possible to view and analyze the traffic between the different IMS functions. It should, for example, be possible to view signaling, such as the SIP signaling described in Section 2.1.8.1.

3.2.2 Application server

The high-level requirement *System functionality* states that an IMS example application should be provided and that only signaling is required. This requirement gives the author freedom to further specify requirements for this example application. The author chose to implement a call screening function to allow users to block certain incoming calls. The remainder of this section discusses the choice of technology used to create the example application.

As this thesis focuses on IMS for educational purposes, it would be a good capability of the system to allow easy creation of IMS applications other than the proposed example. However, this was not a requirement of the customer. Based on this, a soft requirement was set by the author, stating that the technology chosen for the application server should be easily used in order to develop other applications in the future. Since the example application can be considered a new application, only those technologies which can be used in a SIP application server (SIP-AS) will be considered (as was described in Section 2.1.6). The requirements of the SIP-AS will not include the connection to the HSS (that is shown in Figure 2-1).

The relevant software technologies to choose between can be divided into the following categories:

- SIP stack to develop a custom application server,
- SIP proxy with scripting capabilities,
- SIP Servlets,
- JAIN Service Logic Execution Environment (JSLEE),
- SIP Common Gateway Interface (CGI), or
- Call Processing Language (CPL).

All of these technologies are considered potential alternatives to use when developing an example application. They are therefore described below, along with their advantages and disadvantages.

3.2.2.1 SIP stack

A *SIP stack* is a library with functions that handle SIP messages, while providing some level of abstraction. Several SIP stacks exist that could be used for building a custom application server. Although a SIP stack could be used to satisfy the requirements of the example application, this does not seem to be a good option in the long run. If a SIP stack with a high abstraction level is used, the solution is likely to be fairly application-specific. On the other hand, an advantage of this technique is that it allows complete freedom to develop the application since a widely used programming language can be used. If the source code of the SIP stack is available, it could be extended as needed. However, a disadvantage is that depending on the level of abstraction of the stack, a large amount of work might be needed to get basic functionality to work, such as handling the SIP messages in addition to the actual business logic of the application.

3.2.2.2 SIP proxy server

A *SIP proxy* server, such as SER, could be used [35]. For example, SER has its own scripting capabilities in its configuration file. This can be used to make custom calls to a database for information. It should be possible to use SER for this project, just as others already have in their applications. This approach provides a more rapid development

method compared to the SIP stack, as the proxy already has complete functionality for modifying and routing SIP messages. The disadvantage is that it might not be possible to extend the functionality as needed, since the features offered by the scripting language are limited. This method is also limited to using the chosen proxy server for the application, since the scripting language is not a standard used by other SIP proxy servers.

3.2.2.3 SIP Servlet

SIP Servlet is a standard specified by the Java Community Process and described in a standards document called Java Specification Requests (JSR). The current (May 2008) version of the SIP Servlet specification was v1.0 in JSR 116 [44]. JSR 289 [45] will be version 1.1*. SIP Servlet provides a standardized abstraction of the SIP protocol. Applications are written as Servlets that are contained in a SIP Servlet container [46]. Servlets are written in the Java programming language. The container uses a SIP stack beneath it. It should be possible to execute the Servlets on any Java application server that implements the SIP Servlet specification. This technology option can be compared to the SIP stack option. A clear difference is that the SIP Servlet provides a standardized abstraction. The advantage of being able to use other programming libraries still exists. [47]

3.2.2.4 JSLEE

JSLEE is also a standard defined in JSR 22 [48]. JSLEE was designed to fill the industry's need for a high-performance event-driven application server. In a search on the Internet, JSLEE was found to be less common as a technology for application servers, compared to SIP Servlet. The authors of jainslee.org state "JAIN SLEE is a more complex specification than SIP Servlets. JAIN SLEE has standardized the High Availability and Fault Tolerance mechanisms of the programming model and of the lifecycle of the application. SIP Servlets is a programming model, JAIN SLEE is an application environment." [49]

3.2.2.5 SIP CGI

SIP CGI is specified as an informational RFC in RFC 3050 [50]. SIP CGI is analogous to HTTP CGI in how it can be used to create services on a Web server. SIP CGI is not a programming language, but rather an interface between a SIP server and programs written in a programming language. The program is executed when a message is received and the SIP server is configured to use a CGI-program. The SIP server writes to the standard input of the program and expects the program to output a response on standard output. The SIP server provides some abstraction for message handling. An advantage of CGI is that it is language independent. A disadvantage is that although some abstraction is provided, it is still a complex interface. Moreover, few if any, implementations exist. [51]

3.2.2.6 CPL

CPL is defined in RFC 3880 [52]. It was developed to allow non-trusted users to be able to upload their service to a server. For example, this would give a user greater freedom to configure his or her own service profile at an operator's server in a secure way. CPL is based on Extensible Markup Language (XML) and a service is described in a XML document. Operations can be defined for incoming and outgoing calls. Different switches

* However, as of May 2008 this specification was not completed.

and conditions can be defined. For example, the origin of incoming calls can be checked. If the originating user is a friend, the call is forwarded to a voice mailbox with one prompt, while all other users are sent to another voice mailbox. A second example is that outgoing calls to some toll numbers can be blocked. CPL has the advantage of being fairly simple and safe. No external programs can be executed to cause unsafe behaviors. The disadvantage is that operations, actions, and checks are limited to some specific alternatives. A database, for example, cannot be queried by the script. [51]

3.2.2.7 Conclusion

Since the requirement for an IMS application is considered to have lower priority than the basic parts of the product, it must be easy to get an application running. As argued in the beginning of this section, the author set as a requirement that it should be easy to develop applications other than the proposed example. Therefore, the technology chosen should address future needs. In order to make it easier to create a service, it is desirable to have some level of abstraction of the SIP protocol. It is also advantageous to be able to develop applications with a standardized abstraction that allows the use of different implementations of the application server.

The SIP stack method is considered to offer too little abstraction and is likely to have a dependence on a specific SIP stack. The same reasoning applies when using a SIP proxy with a scripting language. CPL is good as it is standardized, but it is considered to have too few features and too limited extensibility. SIP CGI is considered to have too little abstraction, although it can be extended and is standardized. A second reason to avoid SIP CGI is that no implementation was found. The remaining options are Java SIP Servlet and JSLEE. JSLEE is considered to be a too complex environment to learn with respect to the scope of this thesis project. SIP Servlet provides a reasonable level of abstraction. Besides, Servlet can be extended and integrated with other technologies. Another advantage is that several commercial implementations exist, along with some free implementations. The fact that several implementations exist gives a greater choice of provider and greater access to information.

3.2.3 Management interface

As stated in the high-level requirement *"Easy to get started"*, the customer together with the author should define a management interface as part of the system. The requirements for this management interface are:

- To give an overview of the system,
- To be possible to use in order to control the IMS components, and
- To help the user to gradually increase their knowledge about these IMS components.

In addition to the above requirements and the other high-level requirements, the author has had the freedom to define more detailed requirements for the management interface. The remainder of this section describes and motivates these detailed requirements.

The requirements set by the author are:

- *A Web-based* interface to access the management interface. This is useful as the end user will not have to install a management interface, hence making it easier to get stated. A Web-based management interface is easy to access from a remote location and may also be possible for several users to use simultaneously.

- *Authentication* should be required before users can access the management interface, as privileged tasks can be performed using this interface. These tasks include controlling the IMS functions and changing the configuration as specified below.
- *A Graphical map* representing the IMS network should be used to give an overview of the functions of the system. This is required by the customer and also suits the *"Easy to get started"* requirement.
- *Distributed location* of the IMS functions should be possible since they can be installed on different computers and even be placed in different networks. This means that the management interface should be able to control an IMS function installed on a remote computer.
- *Secure communication* to IMS functions is desired due to the above requirement for distributed locations. Besides, it would be unwise to risk that a third-party easily intercepts and replays the control requests to the IMS functions.
- *Links to Web-based documentation* make it easier for the user to access relevant documentation by giving it a central place.
- The customer requires that *Control and status information be presented* for each IMS function. For each IMS function it should be possible to:

    - *Control the IMS function:* which includes starting, stopping, restarting, and checking the current state of the function. The user should be able to do this by executing commands on the computer running the IMS function.
    - *Edit and view* configuration and log files,
    - *Create plugins* to extend the functionality of the management interface to support special features of a certain IMS function, and
    - *Link to external Web-based interface of an IMS function.*

An implementation of a management interface will be made to verify these requirements. This implementation is described in Section 4.4. The implementation makes use of third-party software. The choice of third-party software used for the management interface, presented in Section 3.3.4, is based on the requirements presented in this section.

3.2.4 Installation packaging

This section describes additional requirements and general design decisions for the installation packaging as required by the high-level requirement *"Easy to get started"*. Since the product will contain many different components, everything is best delivered as a single package. This can be achieved by using virtual machine software, which is suggested due to previous experience with such software.

Virtual machine software is computer software making it possible to create several virtual machines inside a physical computer. There are different kinds of virtual machine software. Some need to be installed with full control of the computer while others can be installed as an application on top of an existing operating system (OS) called the host OS. To make it easier for a user to get started, it is preferable that one is able to use a virtual machine software that can be installed on top of an existing host OS. Microsoft Windows and Linux based operating systems are considered the most common host OS targets for the use of this system. Inside the virtual machine, a guest OS needs to be installed. The virtual machine needs to support the guest OS required by the other applications of the product. Therefore, the following requirement is stated: Virtual machine software used for the installation packaging should fulfill the following requirements:

- support an installation on top of an existing host OS,
- the host OS can be Microsoft's Windows and/or Linux based, and
- the guest OS required by the other applications of the product should be supported.

The virtual machine software required is of a type called hosted virtual machine monitor (VMM) as described in [53]. This hosted VMM is a thin layer that is installed as an application on an existing OS. This layer provides a virtualized version of the computer hardware. A new OS can be installed on top of the VMM layer. Using this virtual machine software the product can be distributed as virtual machine image. The virtual machine image will contain a preinstalled virtual hard drive with an operating system and with all necessary software pre-configured as well as specifications of the hardware components connected to the virtual machine (e.g., network card and hard drive). The virtual machine image can be started as a new virtual computer on top of an existing operating system.

As stated above, an operating system is required to be installed inside the virtual machine. This OS needs to support the other third-party applications that will create the rest of this product. This requirement is further investigated in Section 3.3.5.

Installation guide and other documentation are also part of the packaging, as required by the high-level requirement "*Easy to get started*". To limit the amount of work needed, the scope of the installation guide will be limited to only describe the installation with Windows as the host OS. However, as mentioned above it should be possible to install the package with both Windows and Linux as the underlying host OS. In addition to an installation guide, other documentation should be authored as needed. For example, to guide the user after they have started the newly installed virtual machine. The other documentation may be both step-by-step guides and documents explaining the details of the product.

To conclude, the installation packaging will be implemented using a virtual machine as described in [53]. It is found that it should be possible to use both Microsoft Windows and Linux based OS as the host OS for the VMM. The OS found to be needed by the other third-party software that will be part of the product, should be supported by the VMM as a guest OS. An installation guide is required with the limitation of only describing installation on Microsoft Windows. Other required documentation will be in the form of step-by-step guides and/or detailed descriptions. The choices of third-party software that complies with these requirements are presented in Sections 3.3.5 and 3.3.6. The requirements of the installation packaging are verified by an implementation of the packaging as will be described in Section 4.5.

3.2.5 Documentation system

The product should be well documented as stated above in the concrete requirements of "*Easy to get started*" and "*Easy to evolve*". This section deals in details with how the documentation of the product should be maintained. Some software is considered necessary in order to maintain the documentation. The software to fulfill this requirement is analyzed in Section 3.3.7.

To make it easier to choose a suitable technology for documenting this product, more detailed requirements have been defined with the use of the high-level and concrete requirements. The detailed requirements for the documentation are:

1. Produce output in different formats: for web (XHTML), for publishing (PDF), and for the first version of this thesis ($\LaTeX$).

2. Easily maintain different versions as the project and documentation are meant to continue to evolve.
3. Collaborate with several authors. The project should be able to continue as an open source project, therefore several authors should be able to collaborate on the documentation.

3.3 Use of existing software

This section describes the use of existing software and the choices made in selecting the software used to realize the product. The use of existing software is promoted by the high-level requirements *"Easy to maintain"* and *"Easy to evolve"*. The high-level requirements *"Easy to evolve"* and *"Low cost"* further encourage the use of open source software. Therefore, the use of open source software will, where possible, be the first choice. Secondly software that is free for commercial use, but not open source, will be chosen.

The sections below describe the decisions made to choose which existing software to use in order to satisfy the requirements and signaling flows. A discussion of the more detailed requirements for each component is also presented. To satisfy the signaling flows as described in Appendix A, some functions of an IMS core network are needed along with some IMS-capable client. The signaling flows also specify that an application server for IMS is needed. The use of existing software for the management interface and packaging, according to the requirements described in the section above, are also discussed in separate sections.

3.3.1 IMS core network functions

This section describes which logical IMS functions are needed and what options that exist in order to satisfy the need. As specified in Appendix A there is a need for a DNS along with the following IMS functions:

- P-CSCF
- I-CSCF
- S-CSCF
- HSS

The other IMS functions described in Section 2.1.3 can safely be excluded as they are not part of these signaling flows. Since each of the IMS functions are separate logical functions they can be added to a later version of the product, if necessary. The main options are to:

- use the Open IMS Core that contains all these components or
- develop the CSCFs and HSS based on some existing SIP proxy or SIP stack.

The Open IMS Core [36] is an implementation of all the three CSCFs and the HSS. As described in Section 2.3.1, Open IMS Core is a project of Fraunhofer FOKUS with support from the Internet community. These four components are based on other open source software, such as the SER and MySQL. The Open IMS Core is released as open source software under GPL version 2 (GPL v2) [54]. SER is the base for the CSCFs and is a SIP proxy that had been used in commercial settings for several years. SER is written in C and runs on Linux, BSD, and Solaris. Its development was sponsored by several companies. SER is built using modules which makes it easier to extend. Each of the CSCFs is written as one or more modules used by SER. Open IMS Core provides a basic HSS written in Java. The

Open IMS Core documentation recommends the use of BIND as a DNS server. BIND is "the most widely used DNS software on the Internet", also open source and is the DNS server shipped with most Linux distributions [55].

The Open IMS Core contains the IMS functions needed to satisfy the signaling flows in Appendix A and it fulfills several of the high-level requirements described in Section 3.1. The fact that SER is used as a base for Open IMS Core is positive since SER has been used commercially for several years. SER is also fully compliant with the SIP standard RFC 3261. At the time of the original decision making, both SER and Open IMS Core had fairly active mailing lists and good documentation. Based on the above, the Open IMS Core was chosen to implement the IMS core functionality in this project. Therefore, the second option of developing the IMS core functions, was not investigated further.

3.3.2 IMS client

The signaling flows constructed from the high-level requirement *"System functionality"* require an IMS client. Third-party open source software was searched for and the following options were considered:

- SIPp (a test tool for the SIP protocol [56]) and
- one of the following IMS clients:
  - IMS Communicator [57],
  - UCT IMS Client [58], or
  - Open IC Lite [59].

3.3.2.1 SIPp

SIPp is a test tool and traffic generator for the SIP protocol. Custom test scenarios are written in XML. For example, these test scenarios can contain the raw SIP message to be transmitted and the type of the expected results. Several macros and other functions simplifies the testing. SIPp can also be used for performance testing since the same scenario can be repeated with a specified frequency and variable data, such as username. SIPp is executed in a console window. It supports the extensions to the SIP protocol needed by IMS, such as different authentication algorithms. SIPp is mainly supported on Linux and UNIX operating systems, but it should also be possible to run under Windows. The project is actively maintained, it was sponsored by HP, and the source code was released under GPL v2.

3.3.2.2 IMS Communicator

IMS Communicator is built by extending an old version of SIP Communicator and uses a graphical user interface. IMS Communicator was created to be used with Open IMS Core. Its features include a setup wizard, voice and video calls, dial history, contact list, instant messaging, and presence support. Both Linux and Windows version are provided. It is released under Apache Software License v1.1 [60] and LGPL v2.1 [61].

3.3.2.3 UCT IMS Client

UCT IMS Client is an IMS client with a graphical user interface. The current version (1.0.8 as of 2008) supports voice and video calls, pager-mode instant messaging, presence, and has IPTV support. UCT IMS Client is intended to be used with Open IMS Core and was authored for that purpose. It runs on Linux and is mainly supported on Debian and Ubuntu Linux distributions. It is released under GPL v3 [62].

3.3.2.4 Open IC Lite

Open IC Lite is a limited version of the Open IC from the authors of the Open IMS Core. It is free, but not open source. It supports audio calls, instant messages, and has an address book. The graphical user interface is fairly clean and it has been built for use with Open IMS Core. Binaries are provided for Linux, Windows, and Windows Mobile. It seems to be fairly actively maintained, but the number of features will always be limited since it competes with the commercial version of Open IC.

3.3.2.5 Conclusion

Since the focus of the project should be on the core and supporting functions, the use of a real IMS client is unnecessary. The high-level requirement *"System functionality"* only requires signaling to be tested, hence media transmission can be excluded. It is also desirable to eliminate potential problems with a graphical user interface and the need for a sound card. The SIPp test tool was chosen since it satisfies the requirement for the signaling flows and other requirements. SIPp is also used by several of the other users of the Open IMS Core mailing list. The other clients should be recommended in the documentation of the project as alternatives, but are not tested and will not be supported.

3.3.3 Application server

In Section 3.2.2.3 SIP Servlet was chosen as the technology to be used for the SIP-AS. As open source software was prioritized, only open source or free software was evaluated. The options found are:

- *Sailfin* is a project that adds SIP Servlets technology extension to GlassFish Application Server. The GlassFish Application Server is an open source application server for Java EE 5. It was sponsored by Sun Microsystems (before acquired by Oracle) who also packaged it in a commercial version. The SIP Servlet technology was contributed by Ericsson. SIP Servlet API 1.0 (in JSR 116) is supported and work is done to support version 1.1 (JSR 289). Sailfin is open source and was released commercially by Sun as Sun Communication Application Server. Sailfin is released under Sun's open source license: Common Development and Distribution License (CDDL) [63]. A risk with Sailfin is that it is a fairly new project in an early stage of development.[64]
- *WeSIP* is a SIP and HTTP Servlet application server that is built on top of OpenSER (renamed to Kamailio [39]). OpenSER is a project that is derived from SER. WeSIP is free for non-commercial use. It connects through a custom made binary interface to the OpenSER server. Outgoing SIP messages are sent through this interface. The OpenSER scripted configuration is used to decide when to forward a message to WeSIP. SIP Servlet API 1.0 (in JSR 116) is supported [65].

Since Sailfin is open source and an active project with a strong commercial sponsor it was chosen as the application server, despite the risk that it was in an early stage of development.

3.3.4 Management interface

The high-level requirement *"Easy to get started"* states that a management interface should be included in the system. This section discusses the choice of existing third-party

software to be used as the management interface. The software should satisfy the requirements for this interface as described in Section 3.2.3.

No alternatives satisfying these requirements were found when searching for products. Webmin is a Web-based interface for system administration for UNIX and Linux based operating systems[66]. Webmin was briefly evaluated but did not give a good overview of the IMS network. For these reason the author decided to develop his own management interface.

Qcodo [67], a PHP development framework, has long been of interest for the author. Qcodo is therefore evaluated for the possibility of being used as a development framework for the management interface. Qcodo is a framework for PHP 5 [68], which is a scripting language for creating dynamic web pages. Because of this, Qcodo can be complemented with PHP functions. To verify if Qcodo and PHP can be used to create the management interface, each requirement will be evaluated, with a discussion on whether or not possible solutions exist. The evaluation of each requirement follows:

Web-based interface	Qcodo and PHP are built for creating Web-based applications. PHP 5 offers an object oriented programming model. Qcodo provides database abstraction and code generation from a data model to object code. This eases the use of a database. The database can be used for storing configurations about the IMS functions in the network. Qcodo also creates basic web pages to create, read, delete, and update objects. A second part of Qcodo provides libraries (QForm and QControl) of functions to create web pages where the logic is separated from the presentation. The intention of Qcodo is to allow rapid development of web applications. Because of these features it was concluded that PHP and Qcodo would support the creation of a Web-based management interface.
Authentication of the user accessing the management interface	The Qcodo manual gives examples of implementing authentication for a web application. The functionality is provided using a database that contains users and encrypted passwords. The QForm and QControl libraries are used to create web pages to allow users to login. This is considered to be a good starting point to implement the authentication need for the management interface. [69]
A graphical map would most likely be built using images	Web pages are good for combining images and text. The map should be able to be modified by the user. This might require the use of different images for different network configurations. Images can be dynamically created using image functions in PHP. An image can be drawn using PHP scripts and the configuration stored in a database.
Link to other web pages	Is an easy task when using a web application development framework. This requirement is considered possible to fulfill with the use of PHP and Qcodo.
Distributed location, secure communication, and control of IMS functions can be	SSH can be used to login to a remote computer and execute commands using an encrypted communication channel. SSH is widely deployed on Linux distributions and since that is what will be used in this project (see Section 3.3.5)

implemented using the Secure Shell (SSH) network protocol	SSH is provided on the computer running the IMS function. The reason for using SSH is that it is widely deployed and a secure way of executing commands. The Linux distribution also contains the SSH server. SSH is defined in RFC 4251 [70] and related standards from the Internet Engineering Task Force (IETF). Support for SSH in PHP is available as described in the PHP-manual [71]. The use of SSH in combination with the SSH support provided by PHP seems to make it possible to satisfy the stated requirements.
Editing and viewing of configuration and log files is to be supported by the management interface.	As described earlier, SSH can be used to securely read and write files on a remote computer. To provide easy editing of files in the browser, a third-party online editor was desirable. A light-weight online editor called CodePress [72] was selected. It provides syntax highlighting, line numbers, and a few other things that can make the editing of configuration files easier and help to fulfill the high-level requirement *"Easy to get started"*. Since it is not a priority to find a very feature rich online editor it was decided that no alternatives other than CodePress needs to be evaluated. The use of SSH to access files remotely and to present them using a web page and CodePress it is considered to fulfill the requirement of being able to edit and view configuration and log files of the IMS functions.
Creating plugins to extend the functionality of the management interface	This could be done by allowing users to author PHP classes that can be installed and registered via the management interface. These plugins can be dynamically loaded and executed as needed. PHP supports calling new dynamically loaded functions using `call_user_func` [73]. This feature of PHP can be used to satisfy the requirement to allow the use of plugins to extend the functionality of the interface. In addition, the code generated by Qcodo can be used by users when they are creating plugins.

As stated above Qcodo can be used to generate object code from a data model. The intention is to store the configuration of the management interface in a database. Therefore, a database is needed. Qcodo supports MySQL, PostgreSQL, and Microsoft SQL Server. Since Open IMS Core requires MySQL and the author has prior experience with it MySQL [74] was the obvious choice.

To conclude, no existing management interface was found to fulfill the requirements specified in Section 3.2.3. Therefore, a new management interface was developed. Because of prior interest in Qcodo, a PHP development framework, Qcodo was evaluated to verify if it could be used to develop the management interface, and thereby satisfy the requirements. The combination of Qcodo and PHP with additional tools has been evaluated for each of the requirements and were found satisfy custom development and the use of the following third-party software:

- *PHP 5* an object oriented scripting language for creating web pages. PHP provides a large library of function that, for example, can dynamically generate images and load functions.

- *Qcodo* is a PHP development framework that provides rapid application development. This is done partly by code generation from an existing data model and with a library of functions to create separation between business logic and presentation.
- *MySQL* is used as the database to store the configuration of the management interface.
- *SSH* can provide a secure way to control IMS functions by executing commands on a remote computer and to remotely access files related to the functions.
- *CodePress is* an online editor that can provide syntax highlighting and other features that makes editing and viewing of configuration and log files easier.

3.3.5 Packaging – Operating system

According to the requirements in Section 3.2.4 an operating system (OS) is needed for the packaging. This section describes the choice of an OS. Since all of the previously chosen software run on Linux based operating systems and some preferably run on Linux, a Linux based OS distribution will be used.

A Linux distribution is a packaging of the Linux kernel and other software (most of which is GNU Tools) used in combination to create a complete server OS, a desktop OS, etc. The Linux distribution required is a server distribution *without* the need for any graphical user interface. Due to previous experience the following Linux distributions were evaluated:

- CentOS 5.0 [75] and
- Ubuntu Server 7.10 [76].

Both of the evaluated distributions support the use of all the other software chosen. It should be possible to use either of them as servers, and they have an advanced packaging system for installing additional required software. CentOS 5.0 is a community repackaging of Red Hat Enterprise Linux 5.0. Ubuntu Server 7.10 is distributed freely by a company called Canonical Ltd. No disadvantages have been found with either of them. However, two advantages have been found with Ubuntu.

The greatest advantage with Ubuntu is that Java from Sun Microsystems is distributed as a package for Ubuntu. This is an advantage as both the HSS of the Open IMS Core and Sailfin (the SIP-AS) require Sun's Java version 1.5. Sun's Java is free to use but is not completely free to redistribute. With Ubuntu, Sun's Java can easily be installed via an Internet connection with a few commands, in contrast to the more complicated manual procedures involved with installing Sun's Java on CentOS. Because of the high-level requirement "*Easy to get started*" the installation provided by Ubuntu is preferred. The second, but minor, advantage is that that Ubuntu and Debian (the distribution from which Ubuntu is derived) seem to be more frequently used by the other users of Open IMS Core. The popularity is measured by a weakly based search on the mailing list for these distributions. According to this, Ubuntu and Debian were mentioned more often than CentOS.

To conclude, the Linux distribution Ubuntu Server 7.10 was chosen as the OS for the packaging. The choice is based mainly on two reasons, first of all, under its current licensing terms Sun's Java is difficult to redistribute. Ubuntu however, offers a simpler installation procedure of Sun's Java, which has to be installed manually by the end user. Secondly, Ubuntu is chosen because of the author's previous experience with Ubuntu.

3.3.6 Packaging – Virtual machine

This section describes the choice of third-party virtual machine software the installation packaging. The decision to use virtual machine software for packaging is discussed in Section 3.2.4. This decision makes it easy for a user to get started using the product, as it can be distributed as a virtual machine image with software preinstalled. The installation of the virtual machine software and the virtual machine image takes fewer steps than installing all of the software one by one. According to the requirements on the virtual machine software, set by Section 3.2.4, it should:

- support installation on top of an existing host OS,
- allow the host OS to be either Microsoft's Windows or a Linux based OS, and
- ensure that the guest OS required by the other applications of the product be supported.

From Section 3.3.5, it is given that the last of the above requirements should be interpreted to support the Linux based distribution Ubuntu Server 7.10 as a guest OS.

Three different virtual machine software products have been evaluated. Several others exist, but have not been evaluated due to the time limit originally planned for this thesis project. The choice of these three is based on previous experience with the software, or the expectation that the software will fulfill the requirements. The following paragraphs discuss each of the following three candidate virtual machine software suites:

- VMware Server [77],
- VirtualBox [78], and
- Xen [79].

3.3.6.1 VMware Server

VMware Server is free for personal and commercial use. It supports Windows and Linux as the hosting OS. It supports several Linux based distributions as guest OS. VMware Server is virtual machine software of the type "hosted VMM" according to the definitions in [80]. This type of VMM can be installed as an application on top an existing OS. Hence VMware Server supports the first requirement stated above. Microsoft Windows Server versions are officially supported as a host OS. Desktop versions such as Windows XP are not officially supported as a host OS by VMware Server. However, tests have shown that it works well on Windows XP. The part of VMware Server that does require the server version of Windows is a Web-based management interface which is not considered useful for the purposes described in this thesis. VMware Server does not officially support Ubuntu Server version 7.10 as guest OS, but it does support earlier versions of Ubuntu. Testing has shown that Ubuntu Server 7.10 works well as a guest OS using VMware Server. VMware server fulfills all of the set requirements. Another advantage is that the author has previous experience with other VMware software.

3.3.6.2 VirtualBox

VirtualBox exists in two different versions: an open source version and one that is free for personal use and product evaluation. The open source version has fewer features. VirtualBox is a hosted VMM and supports Microsoft Windows and Linux based distributions as host OSs. However, Ubuntu Server 7.10 is not supported as a guest OS. Another concern is that no binaries of the open source edition are provided for Windows. VirtualBox is therefore found to comply with some of the requirements, but not all.

3.3.6.3 Xen

Xen exists in several versions. Xen was originally open source, but there are also commercial versions. One of the commercial versions is free, but includes a limited feature set. The version investigated in this thesis was the free commercial version. Xen is a hardware-level VMM according to the definitions in [80] and is not a hosted VMM as required. This means that a user who wants to install Xen needs a computer dedicated to running only Xen. In addition, a second computer running a control application is required [81]. Xen does support Ubuntu Server 7.10 as a guest OS. However, in summary Xen does *not* comply with the requirements.

3.3.6.4 Conclusion

VMware Server was chosen for installation packaging as it is the only of the investigated products that satisfies the requirements.

3.3.7 Documentation system

This section analyzes software to meet the requirements for documentation system stated in Section 3.2.5. Online collaboration utilities, such as a wiki [82], were excluded since such an approach is considered to be beyond the scope of this thesis project. However, online alternatives could be considered further on if the product is released as an open source product. Therefore, only offline alternatives were considered in this thesis. The proposed alternatives are:

- Microsoft Word [83], is a word processor that has been around for many years. It primarily uses its own proprietary format for documents (as of version 2003). Word can be used to export documents as web pages (HTML), but not as XHTML. XHTML is preferred because it is well structured. Word documents can be used for publishing since it is wide-spread. Additionally, by using third-party tools, Word-documents can be exported to Adobe's PDF. No known method exists to easily export a Word-document to LaTeX. From the author's experience it is complicated for several authors to collaborate writing a document with Word. The same applies to version control as the whole document needs to be kept as a separate version. Two different versions of a document are hard to compare for changes.
- OpenOffice.org (OOO) [84] is an open source word processor that has been around for several years. It is not as wide-spread as Microsoft Office. OOO uses a standardized file format that consists of several XML-files compressed into a single archive. OOO can export its document to XHTML, LaTeX, and PDF. For collaboration and version control the same comments applies to OOO as noted above with Word.
- DocBook [85] is a text based document format and not an application in itself. It uses a well-structured format based on XML or SGML. DocBook was initially created for writing technical documentation. DocBook-documents contain only structure information and no information about presentational formatting. Documents can be edited using any text editor or more advanced XML-editors to aid writing correctly formatted documents. The structure of the document can be validated using tools. Publishing of the document is done using tools that translate the DocBook format to a variety of formats such as HTML, XHTML and PDF. LaTeX is also supported if using tools such as dblatex [86]. Since the

DocBook format is text based it can easily be version controlled by using tools such as Subversion [87]. DocBook documents can be split across multiple files (e.g., one for each chapter) that are linked together. This eases the collaboration process.

- XHTML [88] is a successor to HTML and is used to write web pages. XHTML conforms to XML structure, and is therefore more easily processed using computer tools. No known tools exist that can be used to convert XHTML to LaTeX. XHTML-documents can be converted to PDF using the same external tools as for Word. Since XHTML is also text based it can easily be used with version control. XHTML-documents can link to each other, but one document cannot be included as part of another document as with DocBook.

Based on the requirements set for documentation, DocBook was selected as it is the only proposed alternative that complies with all of these requirements. For this reason DocBook was chosen to write the documentation for this thesis project.

4 Implementation of previous work

This chapter presents the previous implementation work done by the author. The design choices presented in the previous chapter were verified by the implementation presented in this chapter. This chapter describes the contributions made by the author and the integrations of third-party software. The decisions of what to implement and what third-party software to use was described in Chapter 3.

Figure 4-1 gives an overview of what was implemented, compared to what is included in the IMS architecture (as was described in Chapter 2). The sections of this chapter describe the implementation of each of the different core IMS functions and support functions. The implemented functions are:

- *IMS core network functions*, implemented as described in Section 4.1 using the existing software package Open IMS Core [36] (as chosen in Section 3.3.1). The Open IMS Core contains the following functions:
  - *Proxy-CSCF (P-CSCF),*
  - *Interrogating-CSCF (I-CSCF),*
  - *Serving-CSCF (S-CSCF), and*
  - *Home Subscriber Server (HSS).*
- *Domain Name System (DNS) support*. Although a DNS server is not an IMS core network function it is required in order for the core network to work. The implementation using the BIND DNS server chosen in Section 3.3.1 is described in Section 4.1.
- An *IMS client*, needed to verify and test the functionality of the IMS system was implemented as described in Section 4.2 using the SIP test tool SIPp, as chosen in Section 3.3.2.
- A *SIP-AS* provides extra functionality to the IMS system. An example application called Blacklist was implemented, as described in Section 4.3 using the SIP-AS Sailfin, chosen in Section 3.3.3.
- A *management interface* was written by the author and it is called IMS Admin. It is a Web-based system and its implementation, described in Section 4.4, is based on third-party software chosen in Section 3.3.4.
- *Packaging* is implemented using a virtual machine, as described in Section 4.5, based on software chosen in Sections 3.3.5 and 3.3.6.

It was possible to use third-party software for several functions. Therefore, the focus of the implementation work was on the management interface and packaging. The functions contributed by the author are described in more detail than other functions. The remainder of this chapter describes the implementation in more detail.

4.1 IMS core network functions

This section describes the implementation of the IMS core network functions and the supporting function (a DNS server) used to verify the design choices described in Chapter 3. The following functions are used (and shown in Figure 4-1):

- Proxy-CSCF (P-CSCF),
- Interrogating-CSCF (I-CSCF),
- Serving-CSCF (S-CSCF),
- Home Subscriber Server (HSS), and
- Domain Name System (DNS).

Note that in Figure 4-1 there are two P-CSCFs. One of them is grayed out to indicate that only one P-CSCF was installed in the package. This P-CSCF has only been tested against SIPp as an IMS client. None of the testing involved a radio access network or mobile phone. In the above list DNS is not a core IMS function, but is required by the core functions and therefore discussed in this chapter.

As noted earlier the IMS core functions are implemented using the Open IMS Core project. This project provided all the core functions listed above. The DNS server used is the BIND DNS server. Both of these third-party software packages provide the necessary functionality and worked without *major* modifications. To fulfill all the requirements as specified in Chapter 3, some modifications had to be made in the source code, and to some of their configurations.

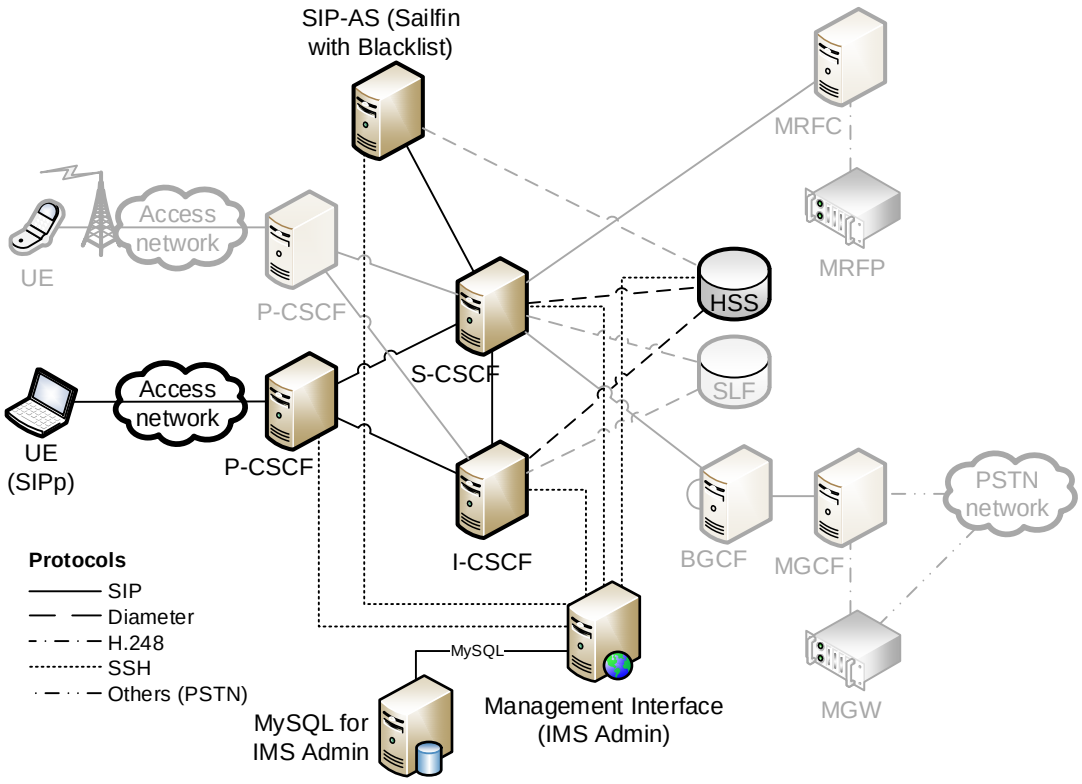

Figure 4-1 IMS architecture implemented (excluding grayed functions)

In the default configuration each of the CSCFs and the HSS from the Open IMS Core project are intended to run in a separate console window. The functions each display text output (log messages) on the console about actions taken and their current state, which is a good feature if debugging and full control are required. However, this conflicts with the *"Easy to get started"* requirement since it raises the barrier for the level of knowledge needed to get the system running. The requirements in Chapter 3 states that a management interface should be used to control the IMS system. Each of the IMS functions then needs to be able to run as a background process (daemon), without writing output to the console. Additionally, logging was also a requirement, hence the textual output that is normally printed to the console should be directed to a log file. This log file can be viewed at a later time, if needed.

In its default configuration the Open IMS Core uses a local IP address available only applications running on the computer where it is installed (i.e. these applications use the *localhost* address). This makes it impossible to connect to the IMS system from a different computer. However, as per the requirements it is desirable to be able to connect via an IMS client from a different computer as well as to connect two separate IMS systems. To enable this, the default configuration had to be changed.

As stated in the *"Easy to maintain"* requirement, installation should be automated. This applies to the Open IMS Core project, as it is an external project that changes. It is desired to include new changes *without* the need to remember all the steps of installing the software. Due to the above mentioned requirements, the same applies to modifications that are made to the Open IMS Core. It should be easy to repeat the modifications on a new release of the Open IMS Core. Therefore the following tasks need to be performed:

- Verify that all the Open IMS Core functions support running as a background process, or modify them if needed,
- Change the configuration of each function so that text output to the console is redirected to a log file,
- Change the configuration to fit an environment where the functions can be accessed from another computer, and
- Script the installation and configuration changes.

The following subsections describe what has been done to realize each of the above tasks.

4.1.1 Background processes

The CSCFs of the Open IMS Core are built on SER, which supports running as a background process (daemon) [35]. When an application runs as a daemon process it cannot be terminated or stopped using a keystroke, but rather it needs to be controlled with process utility applications, such as `kill`. The general practice of Linux operating system is to use shell scripts to start, stop, restart, and check the status of a daemon process. However, the Open IMS Core did not provide any control scripts for the CSCFs. Therefore, suitable scripts were created to control these daemons. The same applies to the HSS. Unfortunately, the HSS did not support being executed as a daemon process. Therefore, the author has modified its source code to introduce this feature. The Open IMS Core team was suggested to include the modification

The BIND DNS server was already included with the Linux distribution used for the packaging. This server runs as a daemon process in its default configuration and shell scripts are already provided to start and stop the server process.

4.1.2 Logging

SER supports logging to the standard logging service (syslog) included in the Linux distribution that is used. The CSCFs configuration files were changed to use the syslog service for log messages. The configuration of syslog was changed to accept log messages from each of the CSCFs and to direct these messages to a separate log file for each of the CSCFs. [89]

The HSS is written in the Java programming language and uses the log4j logging service from the Apache Software Foundation. HSS uses log4j to log to both the console and to a file. The HSS configuration for log4j was changed to turn off logging to the console [90].

The above mentioned changes of the configuration files are scripted to make it easier to apply these same changes to a new release of the Open IMS Core.

4.1.3 Configuration changes

The IP address and domain name of the CSCFs and the HSS were also changed to fit the environment created, as will be described in Section 4.5. Scripts to automate these changes of the IP address and domain name in the configuration files were created.

The DNS server was configured to fit the environment. Two DNS zones were created: one forward zone and one reverse zone. The forward zone is used to translate a host or domain name to an IP address. The reverse zone translates IP addresses to host and domain names. The resulting configuration files can be used as an add-in replacement for default configuration files.

4.1.4 Scripted installation and configuration

Three of the previous sections mention scripts that were created to change the configuration and to control the different functions. In addition, an installation scripts for the Open IMS Core was created. This script downloads the latest version of the software from the Open IMS Core site using the Subversion version control software. The script next compiles the source code and copies the default configurations to the correct locations. Finally, the scripts load the default database tables and data into the MySQL database server. All of these scripts were written by the author.

4.2 IMS client

As noted earlier SIPp was selected for use as IMS client to test the IMS system. While SIPp is test tool and traffic generator for the SIP protocol, it needs scenario scripts written in its custom XML-format to tell it what do [56]. The following four scenarios scripts were used:

- registration for the standard users: Bob and Alice,
- deregistration for the same two users,
- a server scenario* where Alice waits for an incoming call from Bob, and
- a client scenario where Bob registers with the IMS system and calls Alice.

* Server scenario in this case means that SIPp acts as a SIP UA server waiting for a SIP request.

SIPp is a text based console application. The output of SIPp will look similar to the screen shot shown in Figure 4-2 when the register scenario is executed. Together with SIPp the network sniffing tool Wireshark [91] was used to trace network errors and errors in the different protocols and configurations.

The four scenario scripts were contributed by Franz Edler via a private communication. Edler has also published some of these scripts on the SIPp-users mailing list [92]. The scenarios implement the functionality required to verify the system functionality specified in Section 3.1.5. These scenarios reproduce the signaling flows shown in Appendix A.

```
[mickis@imsdev SIPp]$ ./register-alice-netl.sh
Resolving remote host'192.168.10.100'... Done.
---------------------------- Scenario Screen -------- [1-9]: Change Screen --
 Call-rate(length)    Port  Total-time  Total-calls  Remote-host
 10.0(0 ms)/1.000s    3061      0.22 s            1  192.168.10.100:4060(UDP)

 Call limit reached (-m 1), 0.227 s period 1 ms scheduler resolution
 0 calls (limit 30)                    Peak was 1 calls, after 0 s
 0 Running, 1 Paused, 0 Woken up
 0 out-of-call msg (discarded)
 1 open sockets

                                Messages  Retrans   Timeout   Unexpected-Msg
   REGISTER ---------->         1         0         0
        401 <----------  E-RTD1 1         0                   0
   REGISTER ---------->         1         0         0
        200 <---------          1         0                   0
---------------------------- Test Terminated --------------------------------
```

Figure 4-2 SIPp screen shot when testing the registration scenario

4.3 Application server

The application server functionality was given lower priority than the other functions. The requirements state that only SIP signaling is required and that an *example* is to be provided. With this in mind, the author choose to implement call screening for incoming calls as this example. The signaling flow and idea are based on information in *Session Initiation Protocol Service Examples* [93]. The example is seen as a proof of concept of using application servers in IMS.

Sailfin [64] was used as the SIP-AS, the choice of Sailfin was presented in Section 3.3.3. To control Sailfin a shell script was written. This script is similar to those described in Section 4.1. In addition, installation and configuration instructions were written.

The actual call screening application is called *Blacklist* and it was created as a SIP Servlet [46] that runs on top of the SIP Servlet container Sailfin. SIP Servlets are written in the Java programming language. Blacklist was implemented using the *Simple SIP proxy* example from the Sailfin homepage as a starting point [94]. The NetBeans integrated developer environment was used to author the Blacklist application. A plugin to NetBeans was used to ease the development of the SIP Servlet based application [95].

The functionality of the Blacklist application is described below. Blacklist, together with Sailfin, creates a SIP proxy that waits for incoming SIP messages. Sailfin handles the majority of the SIP protocol details. When an INVITE message arrives, Sailfin is instructed

to pass it on to Blacklist. Blacklist checks in its database to see if the callee (the user the INVITE is addressed to) has blocked the caller. If the caller is blocked by the callee, then Blacklist replies to the INVITE with a message telling the caller that the request is forbidden. If the caller is not blocked, then Blacklist causes Sailfin to proxy the INVITE message towards its final destination. Sailfin also takes care of logging and forwarding or dropping of packages related to and not related to each INVITE message. The logic of Blacklist together with Sailfin is shown in Figure 4-3.

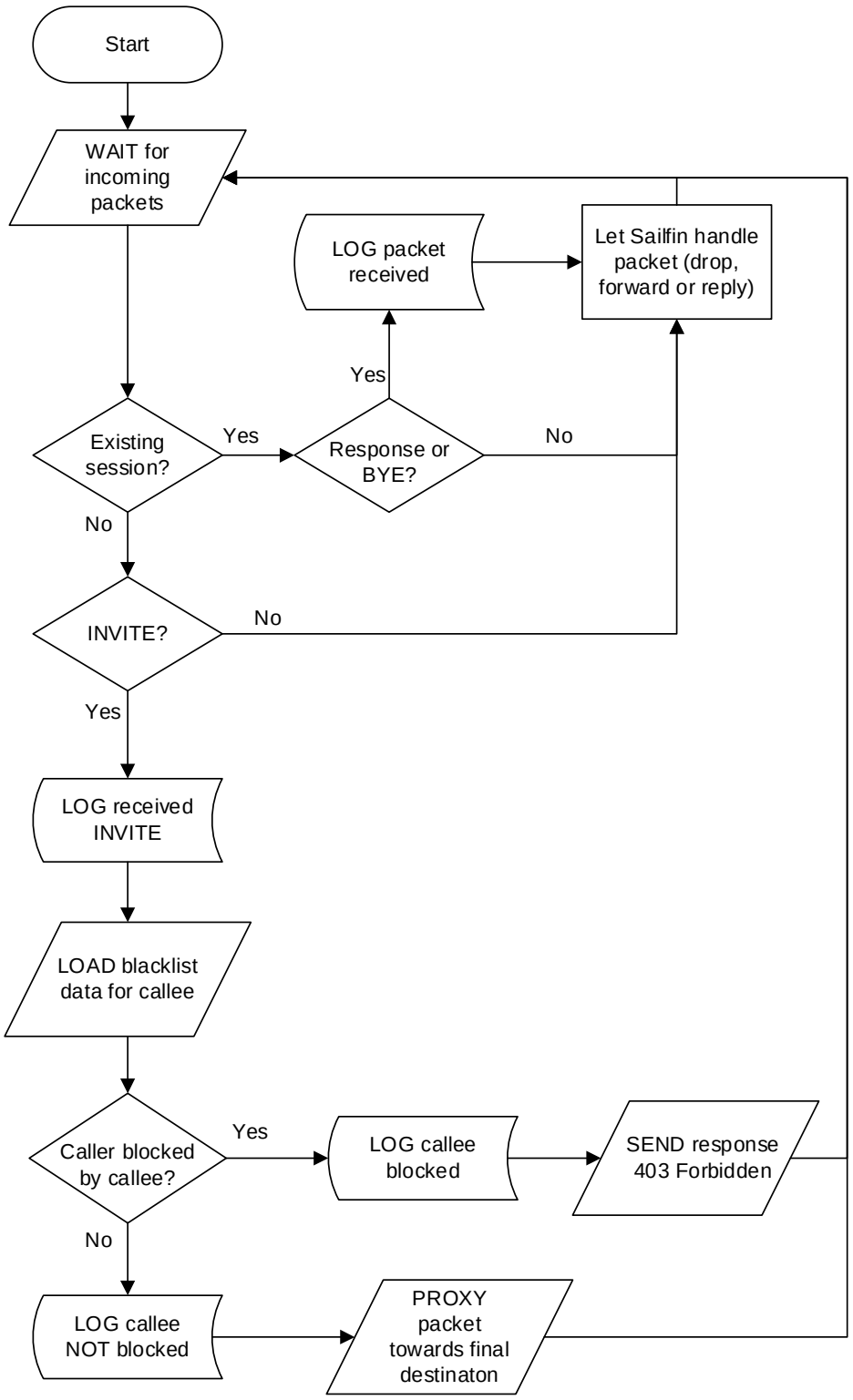

Figure 4-3 Blacklist application flowchart

Blacklist is implemented with a MySQL database as a back end for storing information about which users are blocked by whom. Blacklist is an example of an application where the use of SIP is partly integrated with the use of web pages in the same application. In Sailfin this is called a *converged application*. A converged application uses both a SIP Servlet and a HTTP Servlet. The web pages are displayed using the HTTP Servlet. The

Blacklist application realizes a simple Web-based user interface. With this interface a user can set from which other users it wants to block incoming calls. The web interface updates the data stored in the database.

The Web-based interface proved to have stability problems. This was considered to be due to the fact that Sailfin was in its early stages of development. These stability problems were not investigated further. Because of this, and as a new feature, a second user interface was added as an example plugin to the management interface described in the next section.

The completed Blacklist application contains the following:

- source code in Java as a SIP Servlet,
- binary code in a SAR package (equivalent to Java's compressed .JAR-files, but for SIP Servlets, hence the SAR extension),
- SQL code describing the database structure used to store the call screening data,
- XML-configuration file to load into Sailfin that contains the configuration of the database connection, and
- the documentation for the above.

The Sailfin SIP-AS (and thereby Blacklist) is connected to the IMS using information in the HSS. For each user who wishes to make use of the Blacklist application, a setting is made in the HSS. When such a user registers with the S-CSCF, then the profile of that user is loaded from the HSS. The S-CSCF uses the settings in the profile to process incoming calls for the user. When the S-CSCF receives an incoming call, the INVITE message is passed on to Sailfin and processed through Blacklist. The response from Sailfin is returned to the S-CSCF. As stated earlier the response can be either forbidden or to forward the original message. If the message is to be forwarded, then the application acts as a SIP proxy; while if the message is forbidden then application acts as a terminating SIP UA. More technical and general details about this process can be found in Chapter 2. Details about how the completed Blacklist application is connected to the IMS and is activated, are described in a separate guide titled *Blacklist - an example IMS application*. This documentation was written by the author to fulfill the high-level requirement "*Easy to get started*". This requirement stated that documentation of the product should be included in order to make it "*Easy to manage*".

4.4 Management interface

This section describes the design and implementation of a management interface for IMS that satisfies the high-level requirement "*Easy to get stared*". The requirements for the management interface and choice of third-party software to use were presented in Chapter 3. This section presents more detailed design choices and how they are implemented. As noted previously, the implementation of the management interface is called IMS Admin.

Detailed functionality requirements where created from the requirements in Chapter 3 and are written in a separate guide titled *IMS Admin functionality*. These requirements were specified after and in parallel with the choice of third-party software to be used. They were also modified during development which was needed as the management interface was developed using agile software methods.

IMS Admin is designed to be easy to use and able to customize in the sense that it should be possible to use for different IMS network configurations. To give an overview of the IMS network, a network map is the central part of the management interface (as demanded by the customer). This requires that the contents of the map can be changed for

each IMS network configuration. To accomplish this, four major entities are defined to be used inside the IMS Admin. These entities are as follows:

- The *Network* is used as a logical grouping of nodes. Each network has one associated network map together with zero or more associated nodes.
- The *Network map* is used to organize all nodes associated with a network, with an x and y position. Each network map is associated with just one network. The size of the network map limits the number of nodes that a network can hold.
- *Nodes* belong to a network and can be associated with zero or one host. Nodes can be used as logical representation of a logical function, although several nodes can be associated with the same host.
- *Hosts* are globally defined in IMS Admin and specify on which computer the node is actually running. Each Host can be used by any number of Nodes from any of the Networks defined in IMS Admin.

IMS Admin is built using a database as a backend to store the settings for each of the entities and related settings. Secondly, as required, a Web-based user interface is used to access the data in the database and present it in a form that is easy for humans to read. The following two sections describe the implementation of each of these parts: the database and the Web-based user interface.

The PHP development framework Qcodo [67] is used to build the management interface and to allow a rapid development. To learn Qcodo faster and to get a good starting point, a demo application from the Qcodo developer was used. The demo application called *Dashboard,* described in [96], can be found as a demonstration at [97]. An updated version of the application was provided in a private mail conversation by Mike Ho, the lead developer of Qcodo. That version was provided with unlimited rights. The use of it is described in the following sections.

A user manual with screen shots and detailed descriptions of the functionality of IMS Admin was authored and titled *IMS Admin user guide*. The user manual was written by the author as part of the implementation of the management interface to fulfill the high-level requirement *"Easy to get started"*. This requirement states that documentation of the product should be included to make it *"Easy to manage"*.

4.4.1 Database design and implementation

The foundation of IMS Admin is a database that stores all settings and configurations for the different entities. Figure 4-4 shows the database tables and their relationship to the database used by IMS Admin. In Figure 4-4 the `node` table contains the configuration for each of the node entities. In addition each node can have other settings stored in the tables: `command`, `plugin`, `external_link`, and `file`. These tables provide the node with the following settings:

- `command` contains commands that can be executed to control the node or get the status of the node. The commands that are executed are normally some sort of shell script.
- `plugin` is similar to commands, except that instead of an external command a static PHP function defines the command.
- `external_link` contains web links to external resources used by the node, for example, a separate Web-based interface to control that node.
- `file` is used to store information about configuration and log files related to the node.

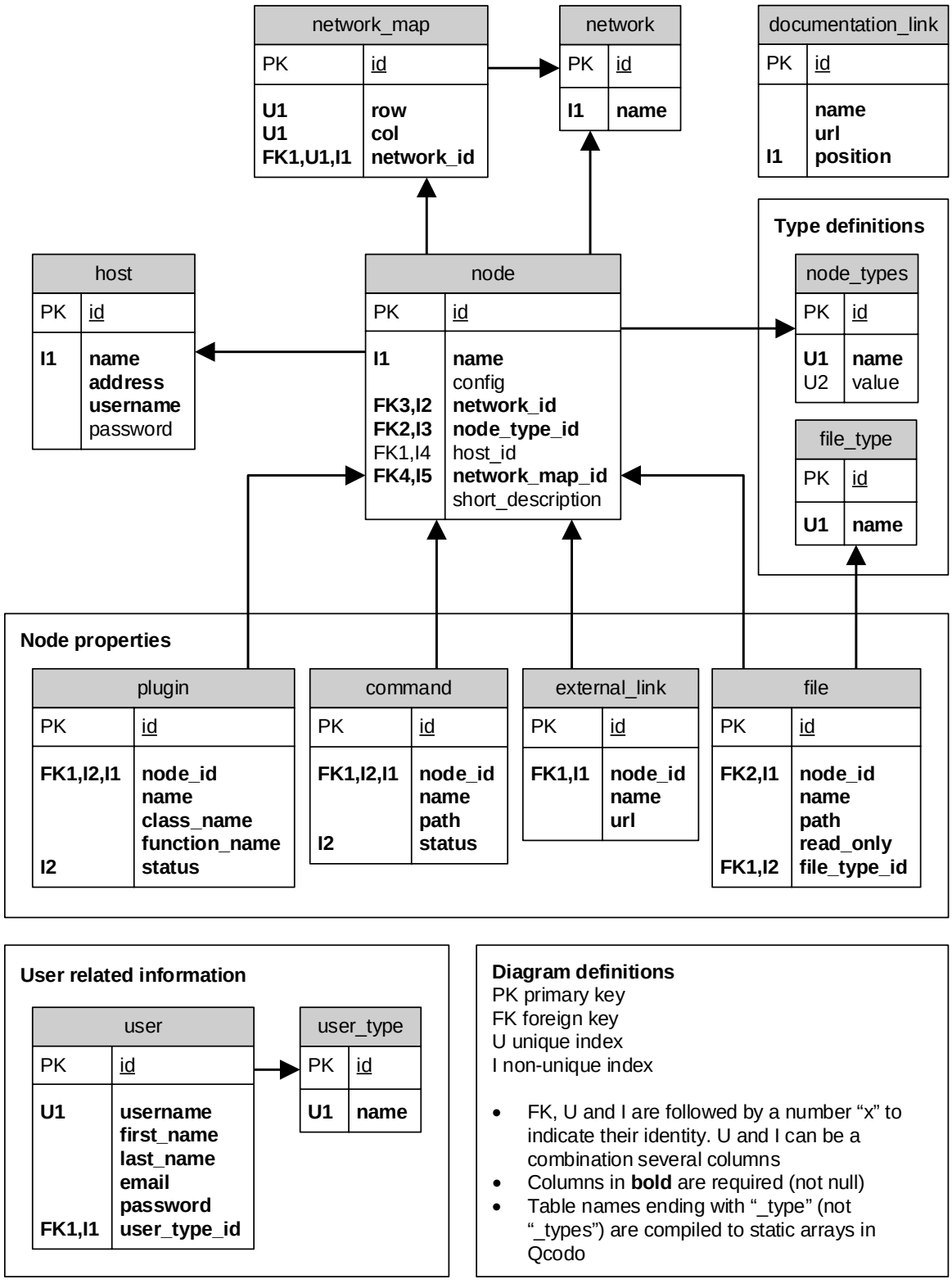

Figure 4-4 IMS Admin database diagram

The node settings are described in detail in the user guide. The `network` table contains the name of each *Network* entity and the `host` table contains settings for each *Host* entity. The `users` table contains user information to allow users to login to the IMS Admin application using a password and different security levels. The `documentation_link` table

contains bookmarks or links to documentation related to IMS. The links are used for the convenience of accessing IMS related documentation from the management interface. The use of these tables are described in more detail in the user guide.

The `network_map` is the representation of the *Network map* entity. This entity is used store the actual network map, which is shown as icons in a rows and columns grid. The `network_map` is used for mapping a node to a row and column position in a logical way and to enforce that only one node at a time is mapped to that position. This design choice allows future versions of IMS Admin to expand the map without the need of changing each node.

The `node_types`, `file_type`, and `user_type` tables are used as static value arrays. They are changed only by the developer and not during the execution of the application. The `node_types` table is used to define what types of nodes that exist. Each node has a node type. For example, the type determines how the icon of the node is rendered by the user interface. The `file_type` table is used to define file types. These types are used in an informative way to allow the user interface to display the files differently (see the next section about the user interface for details). Finally, the `user_type` is used to define what privileges each user has. A description of the different user types are given in the user guide.

As stated earlier, the PHP development framework Qcodo is used to create the database abstraction layer. Qcodo is written in PHP 5 and is object oriented. Qcodo generates code to represent and access the data in the database as objects. Methods to perform database actions such as Create, Read, Update, and Delete (CRUD) are created by Qcodo. The PHP classes generated by Qcodo can be regenerated whenever a database design change is made. This allowed the requirements and product to evolve and change during development. For example, the table `file` a *File* class is generated that is almost an empty skeleton class that extends the *FileGen* class. The *FileGen* class is generated each time Qcodo is asked to generate code. The *File* class allows the developer to write custom functionality that is never overwritten. Therefore, code generation can be performed as often as needed *without* the risk of losing any custom changes. In Figure 4-4 foreign keys use the name convention of the table name they refer with an appended "`_id`". They are also recognized by Qcodo and therefore object references are created. [69]

Qcodo generates special methods for looking up data when indexes are created in the database. For example, index I2 in the table commands allows the interface to fetch all commands for a specific node and its status setting. By using unique indexes the database enforces the rule that no row in the database has the same contents as any other row. For example, this is used for user names when logging in. Qcodo creates a lookup function for the index U1 of the field username in the table user. This lookup function is used to find the user and the corresponding password for a user who tries to login. If the requested username is not found, then it is known that there is no user with that username.

4.4.2 User interface and business logic

The user interface of IMS Admin is a Web-based user interface, as specified by the requirements. IMS Admin is basically an interface to present and change information in the database. Qcodo is used to build the user interface. Qcodo contains the libraries QForm and QControl, in addition to its code generation functionality. The QForm library is used for creating user interfaces in an object oriented way. Each of the QForm objects of the user interface is a PHP object that has some properties. The objects are rendered as XHTML. QControl consists of actions and events. An action can be tied to an event on a

specific form object. The QControl library is built of PHP classes used in QForm and a library of JavaScript functions. A more detailed description of this functionality can be found in the Qcodo Manual [69] and on the Qcodo website [67]. How the Qcodo libraries are used in the user interface is described in more detail later on in this section.

The Eclipse integrated development environment was used to author the application. The Eclipse plugin PDT [98] is used to handle PHP authoring. The application was developed on the Linux distribution Fedora 7. The web browser Firefox was chosen to be the primary supported web browser since it exists in both a Windows and a Linux version.

Here a more detailed description of the Qcodo functionality and how it is used to create the user interface of the IMS Admin is presented. The QForm allows separation of presentation and business logic. Each web page is built up of a template file and a PHP class (form) that extends the QForm class. The class contains the initiation code for all objects of the page, such as text panels, text boxes, buttons, check boxes, etc. The initiation code can load data and tie an action to an event for a certain object. Data is loaded using the database access classes generated using Qcodo. An event can be a click event with the mouse. The form class also contains call back methods for the actions. The call back method to use is specified when an event is tied to an action. When the event occurs for an object that has an action defined for that event, a call is made to the web server that causes PHP to execute the call back method. The web server call and the activation of the call back method are all handled by Qcodo. The call back methods have access to the form object and its local fields and methods.

The template file is a PHP file that mostly contains XHTML code for layout and PHP code that tells where on the page the QForms objects should be rendered. The whole QForm is rendered with the state of the QForm object. Each of the child objects of the instantiated QForm are rendered as XHTML and JavaScript code.

The central part of the IMS Admin user interface is a network map. The network map is used to give an overview of the nodes in the network. These nodes are the same as described in the previous section about the database. Figure 4-5 shows a screen shot of the user interface where the map can be found in the large top right block of the screen. Each node in the map is represented by an icon symbolizing the type of node. The icon also contains the name of the node and connection lines showing logical connections between the nodes. The icons are dynamically generated from the information in the database with a PHP script. The icon script uses an existing image for each node type and adds the name of the node and the connections lines dynamically, as specified in the database. The icon images are cached by the web browser. Figure 4-5 shows what happens when one of the nodes is moved from one position in the map to another. This is an example of a move event tied to the QPanel object that contains the icon. When the event is signaled, an action of type QAjaxAction is triggered. This action type uses the Qcodo AJAX-library to contact the web server. The advantage of using the AJAX action compared to a QServerAction is that the whole page does not need to be reloaded when an event is triggered. The Asynchronous JavaScript and XML (AJAX) functionality uses a library of JavaScript functions that make up the Qcodo AJAX-library. These functions are used to make asynchronous calls to the web server to execute call back methods. When the method is executed Qcodo returns the changes that are supposed to be made to the page as XML. The JavaScript functions perform the changes to the page. All of this is taken care of by Qcodo and abstracted from the developer. For the move event described earlier, the call back method performs the position change of the node in the database and updates the state of the user interface. The other settings of a node stored in the node table are changed using

regular text input boxes, drop down lists, and check boxes. These settings are saved using an AJAX-call.

Draft pages created by Qcodo are used for the extra node settings contained in the other database tables and the general settings. These draft forms are generated by Qcodo in addition to the CRUD-functionality. Qcodo creates drafts of web forms that can be used to do CRUD-operations on each of the tables. These forms have support for foreign keys. For example, this means that when a file type is to be chosen for a file, then a drop down list is used to select the file type. These forms were used as a starting point for some of the user interface. Qcodo generates the forms in two different versions: as classes extending QForm and as classes extending the QPanel. QForms are whole web pages, while QPanels are content grouped inside XHTML -tags. The QPanels objects are used in a QForm. In Figure 4-5 each of the four major blocks are a QPanel that contains other objects. Figure 4-6 shows the user interface for the extra node settings. In this figure, a QPanel is opened as a dialog box covering the rest of the user interface and taking the focus. Within this larger panel are two smaller panels. The left panel shows the list of commands associated with a node. The right panel is used to change a command entry for the command shown in the left panel, or to create a new entry. The left and right panels are based on drafts generated by Qcodo. These drafts are the ones that extend the QPanel class. The general settings described in the previous section about the database have been designed in the same way.

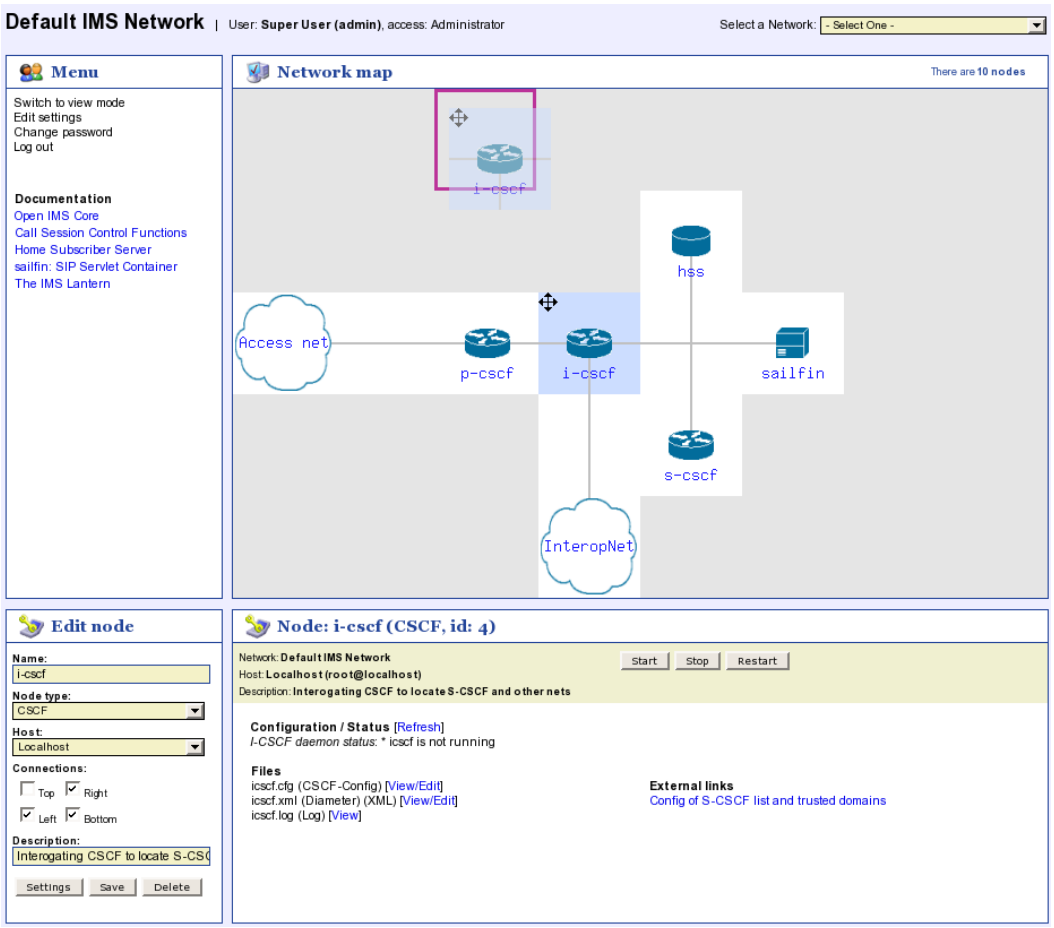

Figure 4-5 IMS Admin user interface: Editing a node and moving a node

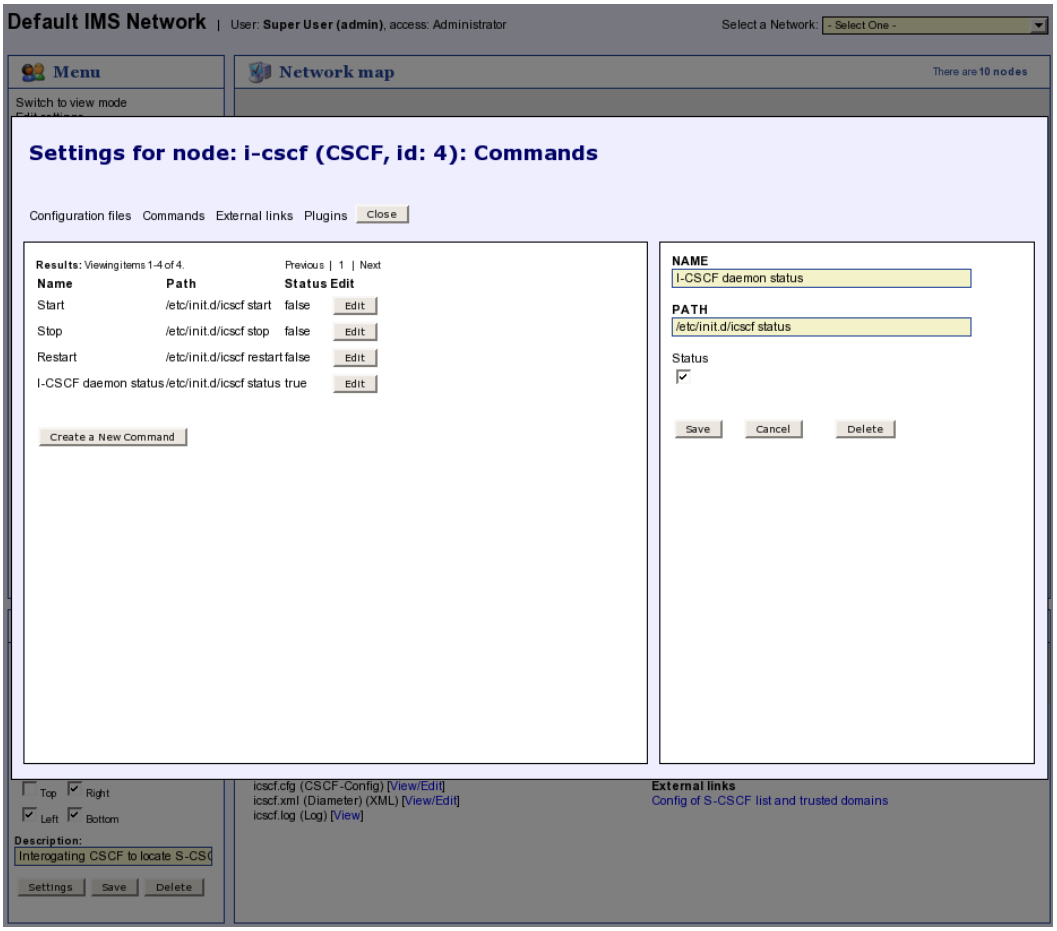

Figure 4-6 **IMS Admin user interface: QPanel draft forms used to change the node settings**

As mentioned earlier each node can contain settings about the commands to control it, as well as configuration and log files related to the node. The commands are divided into status commands and regular (or non-status) commands. The status commands are shown in the status and settings area of a selected node. The status and settings area is shown as the bottom right block in Figure 4-5. This area also contains buttons to execute each of the commands. An example of the result of a status command is shown in Figure 4-5. The status command is titled “I-CSCF daemon status” and after it follows the result of the command. Examples of the command buttons in the figure are: Start, Stop and Restart. The status commands of a node are executed each time the node is selected or when the Refresh link is clicked. All of these commands are executed on the computer running the node using the SSH-protocol. When the button of a regular command is clicked, a new pop-up window is opened that executes the command. The output of a regular command is shown in this pop-up window.

Configuration and log file can be viewed as read-only files or edited using the management interface. These files are opened in an editor by clicking a link in the status and settings area of a selected node. For convenience, the editor is opened in a new separate pop-up window. This allows several files to be open at the same time and the user can continue to perform other tasks with IMS Admin. As described earlier, the editor used is a third-party online editor called CodePress [72]. CodePress is built using JavaScripts. It provides real-time syntax highlighting, line numbers, and a few other features. The file’s content is loaded from the computer running the node using the SSH-protocol. If the file is allowed to be read and written, and this specific user is allowed to write files, then the file

can be changed and saved. When the file is saved the new content of the file is transferred to the web server and the original file is replaced using SSH.

Editing of files can be allowed or disallowed for specific users. This access control applies to other functionalities of IMS Admin as well. IMS Admin implements a simple user access control function. Access control is based on the *user type* of each user. The implemented user types, in order of privileges (starting with the lowest privileges), are: ReadOnly, ReadExec, ReadExecWrite, and Admin. The user types each have an integer value that defines the level of privileges and the relationship to the other user types. The user types form a user level hierarchy where a larger integer value results in more privileges. Each task a user can perform that is access controlled, defines the lowest user level needed to be allowed to perform the task. The task to user level mapping and methods to verify if a task is allowed are gathered in the *ACL* class. This class contains a static method *permit* that can be called with a user and a task name from anywhere in the IMS Admin application where a task is executed. An example of this is when a user of level ReadOnly is logged in, then the buttons to execute commands are disabled and no action is tied to them. The opposite occurs when a user with higher level is logged in.

In this paragraph the layout of the IMS Admin user interface is described. As noted earlier, the Dashboard demo was used as a starting point for the layout. This layout is mainly controlled by Cascading Style Sheets (CSS) [99]. The order in which the objects are rendered is controlled by the layout. This order is set in the templates files, as described above. CSS is used to control the attributes of most objects, such as size, position, color, font family, etc. Some additional layout is specified in the code, such as size of the node icons and their relative positioning within the map. The CSS definitions are stored in a separate file that is included in each of the forms. The layout was generally developed by manual tweaking and iterative testing, using the Firefox web browser on Linux. Firebug [100], as an add-on to Firefox, was used to aid the layout and user interface development and debugging.

A limitation of this version of IMS Admin, is that it does not have complete error handling. Qcodo can raise exceptions and errors when a database error occurs. These errors and exceptions are simply displayed to the user with additional debugging information. This limitation exists because error handling was not prioritized, due to the original time plan of this thesis. A result of this incomplete error handling is that IMS Admin is not a fully multiuser application. Normally a web application is considered a multiuser application, thus many users can access it at the same time. However, the missing error handling may result in unresolved conflicts, for example, when a user is trying to modify or access an object that was recently modified or removed by another user.

4.4.3 Additional contributions and testing

This section describes some additional contributions made to IMS Admin, apart from the database and the user interface. It also describes some of the testing that was performed. IMS Admin is packaged with installation instructions, the user manual *IMS Admin user guide*, and two pre-configured IMS network maps. These pre-configured network maps can optionally be imported after IMS Admin is installed. The first configuration is a *default configuration* that reassembles the IMS nodes described in this chapter as shown in Figure 4-1. This configuration is also part of the complete package described in the next section. It works straightaway to control the IMS nodes, if they are installed and configured. The second configuration is an *example* that shows the different functions of IMS Admin.

To ease the creation of these configurations and future versions of configurations, an *Export* class was implemented. It reads the IMS Admin database and exports a specified network with its configuration. The class creates a file with PHP code that can be executed to import the configuration. The created import file uses the Qcodo generated classes to describe the data that is to be imported.

Installation and configuration instructions for IMS Admin, Qcodo, and other prerequisites for IMS Admin have been written as part of the packaging. Additionally, installation shell scripts where created for Qcodo to ease its installation. Manual testing was conducted during the work.

4.5 Packaging

This section describes the implementation of the packaging made to satisfy the high-level requirement "*Easy to get started*". The requirement states that *installation guides and documentation* and an *installation package* should be created to ease the installation of the IMS system. This should make it easier for a beginner to get started. The implementation of the packaging makes use of the third-party software chosen in Section 3.3, specifically the VMware Server virtual machine software and the Linux distribution Ubuntu.

The following two subsections describe what was done in the area of documentation and the actual installation packaging. The complete result is the final product called "the Open IMS Live distribution".

4.5.1 Documentation

Installation guides and documentation for the IMS Admin management interface and SIP-AS with the Blacklist example application have been produced. This were described in their respective sections earlier in this chapter (i.e. Sections 4.4 and 4.3).

Additional documentation has been written that is specifically part of the packaging. This includes an installation guide to get started with the installation package as described in the following subsection. This guide contains step-by-step instructions of how to install the software on a computer running Windows (as per the requirements in Section 3.2.4). The guide was titled *Getting started using Open IMS Live*. A second guide of how to test and control the installed product has also been written as part of the packaging. This guide was titled *Testing and controlling*. This guide contains step-by-step instructions of how to verify that the installed package works and that the required system functionality (as specified in Section 3.1.5) is present. The guide also contains a summary of commands used to control the IMS nodes without the need for IMS Admin. A summary of the directory structure of the installed package, web links to user interfaces, and a list of the TCP and UDP ports used by the different applications that have been installed is also included.

4.5.2 Installation package

This section describes the implementation of the installation package. During the development of this packaging, the Linux distribution Fedora 7 has been used as the OS for the workstation (as was used for all other development). The result of the installation packaging is a compact disk (CD) with all needed software and documentation. The main artifact produced during the actual packaging process is a detailed guide of how the

packaging can be reproduced. This guide was titled *Creating the Open IMS Live distribution*.

To produce the package, practical testing and installation of VMware Server has been conducted. The chosen operating system, Ubuntu, has been installed within the virtual machine provided by VMware Server. Because of recommendations in the VMware documentation [101], VMware Tools was installed to increase the performance of network communication (among other aspects of performance). Additional tools which the author often uses were installed out of convenience.

Much of the packaging has been described earlier in this chapter in the sections about each implemented part. This includes contributions such as installation scripts. During the implementation of the packaging, all parts have been gathered and documented as to how and when to use them.

Below is a summary of the components that have been installed within the virtual machine, as part of the packaging. The configuration of these is also specified. The installation of each of the component has been documented along with the needed prerequisites for each of them. The components installed and modifications made are as follows:

- an IP address and domain name is chosen for the package. These settings were changed for the Ubuntu system.
- Syslog logging for Open IMS Core is configured as described in Section 4.1.2.
- the DNS server is installed and configured to fit the IP address and domain name chosen. Example configurations from the Open IMS Core were used as a starting point.
- Open IMS Core is installed and configured to use the domain name and IP address along with the newly authored scripts as described in Section 4.1.
- SIPp, including scenario scripts, is installed as described in Section 4.2. When tested the older version of SIPp was found to work the best, hence this version is included in the installation.
- IMS Admin is installed and configured together with the two included configurations as described in Section 4.4.
- Sailfin is installed as the SIP-AS, along with the example application Blacklist and added control scripts.

The resulting package is prepared for distribution. Some performance tuning was done, such as removing unnecessary temporary files as well as other tuning suggested by the VMware Server manual. SUN Java was removed due to the fact that it is not free to redistribute (however, a script was included to make it easy to install once the VM is running). The files created by the VMware Server are called a virtual machine image. These files are compressed to allow easier distribution. The compressed virtual machine image is stored along with:

- all *documentation* that has been produced,
- free *utilities* needed to install and use the completed package on Windows, and
- installation files and tools used during the creation of the packing to allow the packaging to be reproduced are described in the guide *Creating the Open IMS Live distribution*.

4.6 Documentation

In the previous work several guides and other documentation were created as part of the product. The product was intended to be named *Open IMS Live*, hence the name and description of the documents. The documentation created were as follows:

- *Blacklist - an example IMS application* – describes how to activate and verify the functionality of the example IMS application called Blacklist. Blacklist is a basic application that can be used to block calls from certain users. Introduced in Section 4.3.
- *IMS Admin functionality* – defines the functionality of the management interface called IMS Admin. This document was used both for developing the IMS Admin and for testing each functionality manually. Introduced in Section 4.4.
- *IMS Admin user guide* – the user guide for the Web-based management interface called IMS Admin. Introduced in Section 4.4.
- *Getting started using Open IMS Live* – the user guide that is used initially in order to get the final product the Open IMS Live up and running. It includes step-by-step instructions on how to install and start using the Open IMS Live on Windows as host OS. Introduced in Section 4.5.1.
- *Testing and controlling* – describes how the Open IMS Live is tested after installation and how it can be controlled manually. It also gives an overview of the folder structure and port numbers used for different services. Introduced in Section 4.5.1.
- *Creating the Open IMS Live distribution* – describes how the Open IMS Live distribution is composed, and can be used as a guide for developers and contributors of the Open IMS Live. Introduced in Section 4.5.2.

5 Design decisions reexamined

This chapter will reexamine the previous design decisions presented in Chapter 3. The high-level requirements set by the customer, and presented in Table 3-1, were accepted as the continued baseline. The next step presented in Chapter 3, where the high-level requirements are broken down into more concrete requirements, will be discussed below in Section 5.1 and Section 5.2. Furthermore an analysis, when relevant, of software chosen in Chapter 3 as well as new suggestions are presented in Section 5.3. The chapter ends in Section 5.4 with a summary of the above.

5.1 Analysis of high-level requirements

This section presents analytical reasoning about the requirements as they were broken down in Section 3.1, based on experience using the existing product and the author's own experience from other projects.

5.1.1 Easy to get started

As part of this reexamination the documentation and system created as part of the previous work were reexamined. The project's documentation was found to be invaluable and enabled the author to quickly pick up the project again. Regarding the requirement that it should be *easy to install* it was fairly easy to get started using the project since it was packaged as a VMM-image. However, the fact that the VMware Server was discontinued [102] made it slightly more difficult and a substitute was needed. With the installation guides as written it was possible to reproduce constructing the system on a different platform. With this in mind the detailed requirement *easy to install* should remain as specified in Section 3.1.1.

When it comes to *easy to manage* the documentation proved to play a key role when the project is reexamined. However, there appears to be no real need for a management interface when using such as small sized single IMS core network. There is, of course, the need of managing the different functions and having suitable documentation of how the system is constructed. Therefore, this requirement should only demand documentation describing the set-up, how to use it, and an easy way to manage it. The explicit requirement that a management interface should give a good overview should be removed. The detailed requirements of the management interface as specified in Section 3.2.3 will be further analyzed in Section 5.2.3.

5.1.2 Easy to maintain

With earlier implementation a lot of functionality was due to the use of existing software. Therefore this part of requirement should remain. The same applies to utilizing a modular architecture.

Automated installation should be taken one step further to mandate that all parts of the building of the system should be automated. This is enables the use of Continuous Integration (CI) and automated testing. This has proven beneficial in several projects the author has participated in. CI is also recommended by others, including Fowler [103] and Duvall et al. [104]. CI would allow more rapid use of new third-party software since testing and verification of each new release can be automated.

5.1.3 Easy to evolve and Low cost

The requirements *easy to evolve* and *low cost* when broken down are very similar, therefore they are described together in this section. These requirements should both be kept mostly as is, except that the recommendation of releasing the project as open source should be removed. The motivation for these different aspects follow.

The existing system has shown that the use of existing open source components both lowers the cost in terms of reducing implementation time and increasing the amount of functionality. This part of the requirements should be kept as is.

The project itself was never released as open source and the author had no prior experience in releasing projects as open source. This release could be done with a different level of ambition, for example, as independent project (for example, via GitHub) or in cooperation with the existing software that this project builds upon. According to Vingarzan [37Sec. 6.4], an author behind Open IMS Core, releasing code as open source requires a lot of time and effort from the maintainers over a long period of time in order to actually succeed. Whether to release this project as open source has to be further investigated and as stated earlier this consideration is out of the scope of this thesis. Therefore to recommend to release the project as open source is considered to be out of scope, hence this part of the requirement should be removed. However, as mentioned above the use of existing open source software should be done. This facilitates the possibility to release the whole project as open source in the future after a more thorough investigation.

The author can be seen as a new contributor from the perspective of restarting the project after nearly seven years. When doing so the existing documentation proved invaluable as mentioned above. Therefore this documentation should remain as a requirement, independent of whether the project is to be only internal to the customer or later open sourced.

5.1.4 System functionality

The system functionality should be kept at the level specified in Section 3.1.5, since these are the most fundamental IMS scenarios. It should be left to potential future work to extending these scenarios. The more detailed requirements regarding logging are discussed below.

5.2 Analysis of requirements and design decisions

In this section an analysis is presented of the requirements of the corresponding subsections of Section 3.2.

5.2.1 Application logging and traffic monitoring

The requirements for logging in Section 3.2.1 should remain as there is still a need to observe the different applications and view traffic between the different IMS functions. When analyzing the current state of the system some weaknesses have been found with regard to logging:

- All IMS nodes use the same IP address together with different ports. This makes it hard to distinguish the nodes from each other in the traffic.

- The way the project documentation describes how to do logging does **not** allow real-time logging.

To solve these weaknesses the following additional requirements should be added regarding logging:

- It should be required that each logical function sending and receiving traffic data should have its own IP address and use standards based ports. Each IP address should be able to be resolved to a DNS name for the function using that IP address (i.e. use names that are meaningful from a function point of view).
- The ability to do real-time logging should be required. This would facilitate better understanding and debug of the use cases.

5.2.2 Application server

The facts and reasoning stated in Section 3.2.2 still applies and therefore these requirements should remain, that SIP Servlet should be used as technology for the application server.

5.2.3 Management interface

The management interface should be kept to a minimum for the actual use cases needed. The needs are fairly well defined in the first part of Section 3.2.3. However, in the remainder of that section the author clearly overdesigned the management interface without high-level requirements that justify these decisions. The author should not have given himself this freedom as the management interface turned out to be very time consuming to develop. This is one of the reasons the project was not completed within the given time frames. It is also costly to maintain such a specific management interface for slim scenario as management of IMS. The different extra requirements set by the author will be discussed below.

The management interface does not necessarily need to be Web-based. Thou not explicitly stated the intended target group for the system are technically skilled people, since they are expected to be able to analyze traffic flows and eventually develop new IMS applications. From the authors current experience this target group is well aware of how to use a command line interface for management tasks. This is sometimes even preferable as it in some cases are faster to use than web applications. Where appropriate, static web pages should be used for documentation and for linking to references or to applications that provide a Web-based management interface.

There is no actual need for authentication to use the management interface since it is intended to be used in a private environment in education. The same applies to “Distributed location” and “Secure communication”, the whole system is in virtualized environment.

A graphical map that represents the whole system does not need to be editable using the web browser, since this thesis presents a single scenario of a single IMS core network. A simple static map could be provided in a static web page or drawn in text and shown in the Linux console from a script, as done by Rhodes [105] in a similar setup with multiple network nodes.

Instead of controlling the IMS functions from a Web-based management interface, one could use shell scripts from the command line. A script that checks the status of all functions can be authored. Editing configuration files is however easier done in a regular

editor. Command line tools could also be used to view log files. The current design with the web interface does not allow filtering the log files or monitoring the log files in real-time. Using the command line in Linux, these features, which are beneficial when monitoring the IMS functions when testing the system, could be attained.

In summary, the requirement of using a Web-based management system should be removed and replaced with a requirement to use static web page with links to specific functions that provide a web interface and, where appropriate, for documentation. Shell scripts on the command line should be used for management of the IMS functions. The console should be used for viewing log files, and a regular editor for editing configuration files. All this should be documented for the user of the system.

5.2.4 Installation packaging

The requirements specified in Section 3.2.4 should remain. In addition, according to the requirement in Section 5.1.2, the build process of the packaging should be fully automated. The requirement in Section 5.2.1, that each function should have its own IP address also affects the packaging. In order to reduce maintenance, the existing packages of third-party software should, as far as possible, be required to be used without custom modifications.

5.2.5 Documentation system

Some software for documentation is needed as described in Section 3.2.5. The requirement with LaTeX output for the thesis is no longer applicable as this thesis is written in Microsoft Word. The requirement to be able to publish the documentation as PDF-files is not considered to be needed for the project since users will most likely be reading the documentation as reference on the computer. Therefore output as PDF-files should be optional. The detailed requirement that remains for the documentation are:

1. Output formats for web (XHTML), and optionally for publishing (PDF).
2. Easy to maintain different versions as the project and documentation are meant to continue to evolve.
3. Collaborate with several authors. The project should be able to continue as an open source project, therefore several authors should be able to collaborate on the documentation.

5.3 Analysis of chosen software and new suggestions

This section presents analysis of the software chosen as part of the previous work in Section 3.3. Some will be reused, depending on the requirement analysis above. Generally all applications will need to be updated since several years have passed. When the analysis of the requirements resulted in new requirements, this section will present new suggestions of software to meet the new requirements.

5.3.1 Application logging and traffic monitoring

The log files should be handled in the same way using syslog, as described in Section 4.1.2, since this proved to work. Viewing of log files, previously part of the management interface, is described further in Section 5.3.5.

Not enough focus was put on traffic monitoring in the previous work. Monitoring was then handled by using the network protocol analyzer Wireshark [91] and the command line

tool tcpdump [106]. The user documentation stated that tcpdump should be used to capture the network traffic, and that the user should copy the output to the users own system where Wireshark should be used for analysis. That workflow does not allow analysis in real-time. In addition, the user had to install Wireshark in their own system. A new work flow is therefore suggested, which will ease the process. Wireshark should still be used, as it is the de facto standard for network protocol analysis, but should be installed as part of the packaging.

In addition to Wireshark, it is suggested that the SIP Capture server Homer [107] should be installed. Homer can be used to capture in the background the whole time without any aid of the user. When the user wants to analyze the SIP traffic, this is done from the Homer web interface which, among other things, shows SIP call flows and statistics. The SIP traffic captured by Homer can also be downloaded for analysis in Wireshark.

5.3.2 IMS core network functions

As concluded in Section 3.3.1, the Open IMS Core [36] satisfies the requirements. The current state of the Open IMS Core project is however not as active as it was before, as can be seen in the source code repository of the project. The source code for the CSCFs has not been changed since 2013-10-23 (as of 2015-06-03). However the development has continued in the Kamailio SIP Server Project [108]. Vingarzan (the leader of Open IMS Core) states that “Important efforts and outcomes of the OpenIMSCore are also the redesigns and incorporation of parts and concepts into the main development trunk of Kamailio (fork of SER), towards offering IMS features in a carrier-grade VoIP deployment” [37p. 95]. According to Carsten Bock, Kamailio is used in commercial IMS [109]. The use of Kamailio for the CSCFs would ease the installation process as installation packages are provided for some Linux distributions. It is, however, still recommended to use the BIND as DNS server, as concluded in 3.3.1.

HSS from the Open IMS Core, called FHoSS, still needs to be used, though, as none is present in the Kamailio project. The FHoSS is still under development; as can be seen in the source code repository some updates were made during 2014 and 2015. The current version of FHoSS supports the open source Java implementation OpenJDK. The advantage is that distribution of Java will be possible, in comparison to the previous version of this project where Sun Java was used as described in Section 3.3.5.

5.3.3 IMS client

SIPp, which was chosen in Section 3.3.2, still satisfies the requirement and is actively maintained.

5.3.4 Application server

SailFin [64], used as application server in the previous project, is not maintained actively; no announcements have been made on the mailing list since 2010. In the previous work SailFin 1.0 was used. Before active development ended, SailFin 2.0 was released.

Preferably some other implementation of SIP Servlet should be used since SailFin is not actively developed. A search for alternatives shows two open source implementations

of SIP Servlet. The first alternative is Cipango [110], with a web site not updated since 2011. The source code of Cipango [111] is found to be updated 2015.

The second alternative is Mobicents SIP Servlets [112], that is clearly backed by the company TeleStax. Mobicents SIP Servlets is considered to have more up to date documentation than Cipango. The documentation states that SIP extensions for IMS are supported [113]. The recommendation is therefore to use Mobicents SIP Servlets as the SIP-AS.

5.3.5 Management interface

In Section 5.2.3 it was decided to not continue to use the Web-based interface, created as part of the previous work. This interface will therefore not be discussed any further. Instead this section will focus on the new suggested design.

Static web pages should be used for documentation and for linking to other Web-based interfaces used, such as the FHoSS. To ease browsing the documentation, the pages should be hosted on a local web server as part of the package. This way the user of the system can access the documentation using a short DNS-name. The choice of web server is considered an implementation detail, but care should be taken to speed and size.

The management of the different functions of the system should be controlled using shell scripts. These scripts should be executable without referring to their exact location. A commonly used scheme of a main script and subcommands could be used to make it easier for the end-user. The scripts should contain help about them self. Introduction to the script should be given in the system documentation. The choice of script language is considered to be an implementation detail.

Editing of configuration files should be done using regular text editors. Some of the most common text editors should be installed to allow most users to find their regular editor. Based on the author's own experience in other projects, the log files should be possible to view in both real-time and afterwards using the regular Linux command-line tool `less` [114] In addition, the command `grep` [115] can be used for filtering logs afterwards. A terminal multiplexer that facilitates running multiple virtual terminals side by side should be installed. This will allow the user to view logs in real-time, side by side with the different functions. Based on the author's own experience, a terminal multiplexer such as `tmux` [116] should be installed. To facilitate for the user to view all logs, a shell script should be authored to start the terminal multiplexer with multiple terminals showing the different log files in real-time. A common location should be used for configuration and log files of the different IMS functions, allowing the user to easily find them. This should all be documented to lower the learning curve for the user.

5.3.6 Packaging – Operating system

Section 5.2.4 states that the requirements of packaging should remain with a few additions. The packaging requires an OS. The selection of OS, as well as other related issues will be described in this section. In the previous work Ubuntu 7.10 was selected, as described in Section 3.3.5, based on the fact that the required Sun Java was easier to install. Moreover, users of the Open IMS Core seemed to use Ubuntu more frequently than other OSs.

As mentioned in Section 5.3.2, Sun Java is not needed any more. Ubuntu is still suggested to be chosen based on the fact that it worked well in the previous work and because there are Long Term Support (LTS) releases. The LTS releases are considered to

be more stable. A new version of Ubuntu is released twice a year and every two years there is a LTS release that is supported for five years [117].

In Section 5.3.1 Wireshark is chosen as network traffic analyzer. Wireshark uses a graphical user interface (GUI). In the previous work the OS was installed as a text based server OS without any GUI. Therefore a GUI needs to be selected for the OS. Based on the author's previous experience, the XFCE desktop environment provided by the Ubuntu variant Xubuntu [118] is a good compromise between ease of use and a lightweight system. The LTS version of Xubuntu is supported for three years, which is considered good enough. The use of a GUI for the OS will make it easier for the project to be self-contained, as a web browser can also be installed to view the documentation. To facilitate the use of Xubuntu, it should be configured with automatic login, automatic start of the services used and of a web browser for the documentation.

According to Section 5.2.4 the packaging should be fully automated. Ubuntu uses a packaging system and provides software packages for the different components. This system can be used for installing other software as needed. Software that is currently not available in packages should be packaged as part of the implementation work.

To facilitate logging, each function in the system should have its own IP address and DNS name, as stated in Section 5.2.4. This can be achieved in several ways. One way is to configure several IP addresses on the same network card. Another solution is to use isolation features of the Linux-kernel to create a container that isolates resources (CPU, memory, network) and processes from the main system [119]. This is called operating-system-level virtualization [120]. In this way an application bundled with its dependencies can be used to create a virtual file system that is then used to execute the application in a container. The container will have its own resources and virtual network interface. Docker is a platform that makes use of these features in the Linux-kernel, and abstract the management of containers, and applications with associated file system [121, 122]. Another advantage of using containers is that the virtual file systems allow the same software packages to be installed for the different CSCFs without the need of changing the default installation location. Docker is scriptable and is meant to be used to package and distribute applications with their dependencies. This facilities the requirement of automatic building. Due to the above mentioned reasons, containers using Docker are selected in order to separate the different functions of the system.

5.3.7 Packaging – Virtual machine

As stated in Section 5.2.4 the requirements for packaging given in Section 3.2.4 remains, which requires a virtual machine monitor, also called hypervisor. As part of the previous work VMware Server was selected as the hypervisor, see Section 3.3.6. The VMWare Server was, however, discontinued in 2010 [102].

A search for new alternatives shows that VirtualBox [78], which was evaluated as part of the previous work, is now open source and binaries for Windows and Linux are provided. Moreover, the author has used VirtualBox the last couple of years. VirtualBox is found to be the only alternative that satisfies the detailed requirements stated in Section 3.2.4.

The requirement that the packaging should be fully automated can be satisfied using either scripts and command line tools for VirtualBox, or a tool called Vagrant [123]. It is recommended that Vagrant should be used, because it abstracts a lot of the scripts otherwise needed. A description and a presentation of the advantage of Vagrant is given below.

Vagrant is a tool that can be used to set-up a development environment within a hypervisor. VirtualBox is one of the hypervisors that are supported. Vagrant uses a configuration, called Vagrantfile, which describes the environment to be set-up. The Vagrantfile includes a reference to a so called box, which is to be used as base for the environment. The box is a hypervisor image that contains an operating system as well as the tools needed for Vagrant in order to install the rest of the required environment. Ubuntu provides official Vagrant boxes which can be used as a base for this project. As Ubuntu provides boxes, updating the base is facilitated as new versions of Ubuntu are released. The base box can be extended using the configuration in the Vagrantfile. A new extended environment can be repackaged and distributed either as a new box or as a Vagrantfile.

The environment for IMS is recommended to be created as an extended Ubuntu box for the general parts, and as a Vagrantfile for the IMS specific parts. The use of an extended Ubuntu box would decrease the time provisioning the IMS environment for end users. Both the box and Vagrantfile could be shared in a directory provided by the creators of Vagrant [124].

End users would use Vagrant and the Vagrantfile created for IMS, for provisioning and maintaining their environment for IMS. If the Vagrantfile and box are shared from a central storage, users would be able to download them and detect new updates. When a user does not want to use the environment, Vagrant can be used to destroy it and start all over again.

In summary, the hypervisor VirtualBox should be used to host the environment and Vagrant should be used to build, maintain and control the environment.

5.3.8 Documentation system

In order to fulfill the requirements in Section 5.2.5, a documentation system is needed. The DocBook-format that was chosen, in Section 3.3.7, as the document system for the previous work. In this section the choice of DocBook-format is reviewed and a new alternative is proposed.

During the previous work, several documents where written using DocBook-format. The experience gained showed that editing was difficult when using a regular text editor due to all XML in the document. There was a need for an editor that understood the DocBook-format and could hide the markup of XML. The amount of markup used in the DocBook-format also made it harder to merge and compare different versions of the documentation. When using an editor for the DocBook-format, the advantage of DocBook was that focus could be on the content and structure of the documentation, and the actual formatting could ignored.

With the above reasoning, the use of a text based, content focused documentation system with minimal markup would be ideal. There are several such lightweight markup languages [125] that can be used for documentation. In the search for such a language, two seems to be more commonly used than others; Markdown [126] and AsciiDoc [127]. Both of these have gained their popularity partly because of their support on GitHub, a large source code hosting site [128]. GitHub hosts repositories of source code using the distributed version control system Git. It is possible to author documents either online at GitHub in an online editor, or offline in a text editor. The documents are version controlled using Git.

A comparison between the two language shows that AsciiDoc is meant to be a possible replacement for the DocBook-format, one that is human readable while still containing powerful features. Markdown, on the other hand, is “best for simple content” [129]. AsciiDoc has been used in several book projects and other project for authoring technical user documentation [127].

From the above reasoning AsciiDoc is the preferred solution. The recommendation is, however, to reevaluate the possible options when a new version of the project is to be implemented. This is important as the choice of project platform, such as GitHub or similar, as well as the choice to release the project as open source or not, would in turn impact the choice of documentation system.

5.4 Summary

This section contains a summary of the new design decisions and the new software that are suggested in this chapter and should be verified further in Chapter 6. The high-level requirements that were broken down in Section 3.1 have been analyzed in Section 5.1. The high-level requirements were analyzed based on experience, using the existing system and the author’s own experience from other projects. The conclusions drawn and recommendations made are summarized as follow:

- Keep the requirements for documentation as this proved valuable when the project was restarted.
- Remove the need for a full-blown dedicated management interface, since it is not needed and costly to maintain.
- Continue to use existing open source software as a lot of functionality was quickly gained and costs where reduced.
- Use automated building of the product using CI to facilitate maintenance of the product.
- Remove recommendation to release as open source as it needs to be investigated further.
- Continue to support the same IMS scenarios as they are the foundation of the project.

The more concrete requirements and design decisions of the previous work, presented in Section 3.2, have been analyzed in Section 5.2, and are summarized as follow:

- Each logical function in the IMS network should have its own IP address and DNS name to facilitate logging.
- Real-time logging should be possible.
- Continue to use SIP Servlet technology for AS.
- Use static web pages hosted on a local web server for documentation and links instead of a management interface
- Use command line tools for managing the IMS functions and for viewing log files.
- Use a regular text editor for editing configuration files.
- Only require a documentation system that produces in Web-based output, has the possibility to version control and admits collaboration with several authors.

An analysis of previously chosen software and new suggestions are presented in Section 5.3. The analysis of the software chosen as part of the previous work, in Section 3.3, have been made where appropriate, based on the analysis of the requirements presented in the two first sections of this chapter. Where the requirements have changed or

the old software no longer fulfill the requirement, new software has been suggested. A summary of the software chosen are presented in Table 5-1.

Table 5-1 Overview of new and previous design

Problem area	New design suggestion	Previous design solution
Traffic monitoring	Wireshark installed inside hypervisor as part of the product to allow real-time monitoring. GUI for Ubuntu is required. HOMER SipCapture should be installed to allow continues traffic monitoring for later analyzes through web interface and Wireshark.	Use tcpdump to capture to file. Analyze capture using Wireshark on users own system.
Application logging	Syslog for storing (unchanged) and viewing using command line tools.	Syslog for storing and management interface above.
IMS core network functions	Kamailio SIP Server with evolved modules from Open IMS Core. HSS from Open IMS Core.	Open IMS Core
IMS client	SIPp (unchanged)	SIPp
Application server	Mobicents SIP Servlets	SIP Servlets with SailFin
Management interface	Command line based interface for controlling IMS functions.	Web-based interface developed by the author for controlling IMS functions, view log files and editing configuration files.
Editing configuration files	Configuration files accessible from central location and edited with regular text editor.	Management interface above.
Packaging - OS	Xubuntu LTS 14.04 New Ubuntu packages authored as needed. Docker for virtualization and packaging of environment for each IMS function and help function.	Ubuntu 7.10 Packaging using archives.
Packaging - Virtual machine	VirtualBox as hypervisor. Vagrant for automation, configuration and distribution.	VMware Server
Documentation system	AsciiDoc	DocBook

6 Analysis

This chapter presents and discusses the results of the work presented in this thesis. The previous work is verified in Section 6.1, using acceptance testing made by the customer. The new design decisions, presented in Chapter 5, are verified in Section 6.2, using analytical reasoning based either on the work of others, the author's own experiments, experience or a combination of these. The chapter concludes with a discussion in Section 6.3 of how the results meet the goals of the thesis.

6.1 Verification of previous work

The previous design decisions presented in Chapter 3 were verified by the implementation presented in Chapter 4. The implementation was manually acceptance tested by the customer to verify that the product satisfies the requirements. The documentation listed in Section 4.6 was used during the verification. This documentation has been reviewed and accepted by the customer.

The testing of the packaging was performed by following the guide *Getting started using Open IMS Live*. The tests described in the guide *Testing and controlling* were used to verify the functionality. The customer used the list of functionality in the *IMS Admin user guide* to manually verify and acceptance test each use case of IMS Admin. The customer found that IMS Admin satisfied the requirements. The SIP-AS, with the example application Blacklist, was also tested as part of the package testing. The testing of Blacklist was conducted by following the guide *Blacklist - an example IMS application*.

The IMS system functionality was confirmed by manually verifying the call flows of the SIPp scripts towards the relevant 3GPP specifications. SIPp, along with the scripts, was used to verify that the different components of the product complied with the system functionality required. At the same time the network communication was recorded using the tool tcpdump and later reviewed using Wireshark. This showed that logging of traffic activity worked as expected. The application logs produced by the different IMS functions while invoking the SIPp scripts was reviewed in order to confirm that the logging of application activity worked as expected.

The entire testing showed that only some minor corrections had to be made and that the packaging as a whole was accepted. The customer representative and industrial advisor Henrik Leion at Enea Experts, in cooperation with the author, performed the acceptance testing. Leion stated that "it is the most well-documented thesis I have seen, and it works." (freely translated from Swedish)

6.2 Verification of new design decisions and software suggestions

This section describes how the new design decisions and detailed decisions of new software, presented in Chapter 5, are verified.

6.2.1 Automated build process

Chapter 5 concludes that "automated building of the product using CI to facilitate maintenance of the product" should be done. This is verified by the author's experience in several other projects where CI in combination with automated testing has been used. In contrast

to step-by-step documentation of how to build the product, the automation process ensures that the instructions given in the scripts are working. This requires tests in the end of the chain verifying that the whole chain works. In this case, the system functionality can be confirmed automatically using the IMS client SIPp, since SIPp can be controlled using scripts. This proves that the whole chain of the functions work. Other similar tests can be authored to test specific parts of the product, such as the command line based management interface. It is also required that each of the components in the building process are possible to automate, which is the case with the suggested software.

6.2.2 Traffic monitoring

Chapter 5 concluded that “each logical function in the IMS network should have its own IP address and DNS name to facilitate logging” and that “real-time logging should be possible”. The use of real-time logging is show from experience by the author, as a teacher in basic network courses and as a systems administrator, to facilitate trouble shooting and especially understanding. If using real-time traffic monitoring, IMS system functionality can be tested while the students observe the signaling of the different functions. In IMS multiple functions are involved in the signaling, as shown in Appendix A.

If each of these functions can be shown in the traffic monitoring, using its functional name, this would ease understanding for the students. In order for the traffic monitoring to show the functional names, each function is required to have its own an IP address which can be translated to only one corresponding DNS name.

The graphical tool Wireshark [91] was in the previous work verified only in order to view traffic after it had been captured. In later experiments conducted by the author, Wireshark has proven successful also when used for capturing traffic in real-time. Additional experiments have been conducted which show that Wireshark performs well running on Xubuntu within VirtualBox.

The traffic monitoring should be extended to perform using the SIP Capture Server HOMER [107] as well, as explained in Section 5.3.1. The decision to use HOMER is verified by live presentations by its authors at the Kamailio World 2015 conference [130, 131].

6.2.3 Application logging

Chapter 5 concludes that command line tools should be used for viewing log files and that real-time logging should be possible. In a project with extensive amounts of text-based logs, the author has gained experience using standard Linux command line tools to view logs in near real-time and filter them. It has shown to be feasible to view log files of several gigabytes in size after they have been recorded.

The tools recommended for working with log files are Less [114] and GNU Grep [115]. The tool Less can be used in order to view log files, to search within the files and to filter them. Less allows the user to continuously read a file that is written to show new log messages. The same applies to Grep, if the correct options are given. In addition tmux [116] can be used. Experiments conducted by Mikael Arvids, a colleague of the author, have shown that tmux can be scripted to start multiple programs side by side. The viewing of application logs using the command line is considered to be verified by the experiments and experiences from other projects.

6.2.4 IMS core network functions

Kamailio, introduced in Section 5.3.2, should be used for the IMS core network functions. The use of Kamailio is verified through an interview with and a presentation made by Carsten Bock, the CEO of ng-voice. In an interview [109] at Telecom Application Developer Hackathon in London, Bock shows a live VoLTE call using Kamailio. Bock also states that Kamailio is used in a commercial mobile network for IMS at Smile Telecoms, South Africa. In a presentation at Kamailio World 2015, Bock shows how Kamailio can be installed and used together with the HSS from Open IMS Core [132]. The last release notes of Kamailio confirm that the IMS features are updated [133]. This indicates that the IMS features are actively developed. Kamailio is considered to be a verified choice based on the presentation, the interview and the fact that it is actively developed.

6.2.5 Application server

The Mobicents SIP Servlets, introduced in Section 5.3.4, is considered to be a verified design decision because of its extensive documentation, a demonstration and the fact that it is backed up by the company TeleStax. A demonstration of Mobicents SIP Servlets shows that it can be used as a Web Real-Time Communication (WebRTC) server [134]. WebRTC is a technology used within web browsers in order to allow browser-to-browser voice and video calling. WebRTC often uses SIP as the signaling protocol, such as Mobicents SIP Servlets [112, 135].

6.2.6 Management interface

Chapter 5 makes the following conclusions regarding the management interface:

- Remove the need for a full-blown dedicated management interface, since it is not needed and costly to maintain.
- Use static web pages hosted on a local web server for documentation and links instead of a management interface.
- Use command line tools for managing the IMS functions.

The reasoning in Section 5.2.3 and in Section 5.3.5 are considered to verify the above conclusions on a conceptual level. A web browser have been tested to work well with the OS and hypervisor verified in Sections 6.2.8 and 6.2.9. In this section the use of a command line based management interface will be discussed and verified.

A command line tool should be authored with base command and subcommands. The base command should be something short, such as `ims`, as it is typed often. Built-in help, using a subcommand such as help, could be used as this is commonly used by other command line tools such as Git and Subversion [87, 136]. Both Git and Subversion are also available with something called command completion, which helps the user with suggestions of how subcommands and arguments should be written. Experience from working with other developers have shown that using command line tools such as Git and Subversion is a common everyday practice. The tool should be packaged using the packaging format of Ubuntu, as described in Section 6.2.8. A command line tool with good documentation and command completion is considered to be a verified design choice based on the reasoning above.

6.2.7 Editing configuration files

The configuration files of the different IMS functions seldom need to be changed, as it is the Web-based configuration of the HSS that will be used most frequently. In case the configuration files should be changed, a regular text editor is to be used rather than a Web-based interface, as concluded in Chapter 5. From experience working with several text editors in Linux, this is expected to work well. To facilitate finding the configuration files and log files, a common location should be used for all IMS functions. Several different text editors, both graphical and text based, should be installed to allow most users to find their regular editor. Experiments have shown that this works well in the environment described in Sections 6.2.8 and 6.2.9. The above reasoning verifies the use of regular text editors for modifying the configuration files.

6.2.8 Packaging – OS

In Section 5.3.6 Xubuntu LTS is chosen as OS together with its package management system. Experiments using both Wireshark and the web browser Firefox with Xubuntu in VirtualBox have proven to work well. The package management, APT, used in Ubuntu is considered to be verified for distributing the software for two reasons: several other projects, such as Kamailio, already have and it is used by both Debian and derived distributions such as Ubuntu [137].

The tool Docker, chosen in Section 5.3.6, should be used to separate the different IMS functions in lightweight containers and provide them with a separate IP address and file system. The choice of Docker is considered to be verified by its documentation, introductory exercises [138] and experiments using “The Network Playground” by Rhodes [105], which uses a network of twelve Docker-based hosts.

6.2.9 Packaging – Virtual machine

In Section 5.3.7 VirtualBox was chosen as the hypervisor for packaging the whole product. Experiments have been conducted with Xubuntu and VirtualBox, as mentioned in Section 6.2.8. The experiments, together with previous experience, verify that VirtualBox is a good design choice as hypervisor. Vagrant, also introduced in Section 5.3.7, should be used to for distribution and management of environment within VirtualBox.

Vagrant is verified to be a valid design decision by experiments conducted using example environments and by reading the documentation. The fact that Canonical Ltd., the company behind Ubuntu, supports Vagrant by providing official Ubuntu boxes also confirms it to be a valid choice. [124, 139]

6.2.10 Documentation system

In Section 5.3.8 AsciiDoc was chosen as documentation system in order to produce Web-based output, to enable version control of the documentation and to collaborate with several authors. AsciiDoc is verified by experiments, by reading the documentation, by the usage in other projects and by recommendation by others.

Experiments have been conducted using AsciiDoc-format [128] at GitHub’s online editor. They proved to be simple for editing without the need for installing any environment. The documentation shows that AsciiDoc is feature rich for technical documentation while still using a simple syntax. The simple syntax and lightweight markup makes it suitable for

merging different versions from collaborators. AsciiDoc [127] is used by several authors of technical books, such as ProGit [136] and user documentation, such the Git user manual. Linus Torvalds, the developer who created Linux and Git, stated “Use ‘asciidoc’ for document markup. Really. It's actually **readable** by humans, and easier to parse and way more flexible than XML.” [140].

6.3 Discussion

This section discusses the results of the new design decisions and if and how they meet the goals set in Section 1.4. The goal of the thesis is to reexamine the current design and find weakness, and to present design suggestions improving these weaknesses and therefore better fulfill the requirements set by the customer. A reexamination of the current design, presented in Chapter 3 and 4, has been conducted in Chapter 5. The reexamination has shown several weaknesses, such as:

- The Web-based management interface is unnecessary and costly to maintain.
- The current design is difficult to maintain since the build process is not fully automated.
- Real-time traffic monitoring is not supported.
- The different IMS functions are difficult to distinguish in the traffic log.

A more complete summary of the conclusions of the review can be found in Section 5.4. All suggestions of new design decisions and new software are verified in this chapter using analytical reasoning based either on the work of others, the author’s own experiments, experience or a combination of these. The new design decisions and suggested software are considered to better fulfill the requirements set by the customer for a design of an IMS for educational purposes. Therefore it can be concluded that the goals of this thesis have been met. Moreover, the reexamination and the new and verified design suggestions are the expected artifacts.

7 Conclusions and Future work

This chapter concludes the thesis by presenting that the goals have been met and what insights that have been gained. Future work are also presented and reflections of ethical, economic and environmental aspects of the contributions of the thesis.

7.1 Conclusions

The author previously designed and implemented an IMS for educational purposes. The goals of this thesis was to reexamine the previous design and find weaknesses, and to present design suggestions improving these weaknesses and therefore better fulfill the requirements set by the customer. As the previous design has been reexamined, several weaknesses have been found, such as: a Web-based management was designed without an explicit need, and it was difficult to observe the signaling traffic of the IMS functions. These weaknesses, and others, have been addressed by providing new design decisions, verified by using analytical reasoning based either on the author's own experiments and/or experience on the work of others, a combination of these. In addition, several of the third-party software used are no longer maintained and new suggestions have therefore been provided. The goals of the thesis have been met as a reexamination has been conducted and new verified design decisions have been provided.

An insight gained from working with this degree project is that there are a lot of open source software available to solve what normally could be considered as only an area for commercial applications. Others working in the area of telecommunications could benefit from this knowledge. More open source is expected to be used in this area in the future. Interested parties are therefore advised to observe the coming work of the following organizations: Fraunhofer Institute FOKUS, Kamailio SIP Server Project, the Telecom Application Developer Hackathon and the company ng-voice. All these organizations have contributed to the use of open source in the area of telecommunications.

When reflecting on the work of this thesis, some things should have been done differently. The previous work was not completed, partly due to the fact that clear project boundaries were not maintained. Strive for perfections in details led to a failure to see the whole picture. This insight is something worth keeping in mind for others working in the area of software development.

If the work would be done over again, a study should be conducted early in the project in order to get clear overview of the project and its requirements. The actual needs of the users of IMS in education should be investigated and all requirements should be derived from these actual needs. In order to avoid that striving for perfection constrains the main efforts to one section, all areas should be developed at basic level before working with details.

7.2 Future work

A continuation of this thesis would be to verify the design decisions further by actually implementing a prototype or full implementation. The application server and an application development environment is then considered to be the weakest part of the new design decisions provided in this thesis. It is therefore recommended that this should be investigated further at an early stage. Another area, related to implementation of the design decisions, which would be worth investigating in case of a future work, is that of

actually publishing the implementation as open source and maintaining an open source project.

Investigating how the design could best be used in education is a natural area of future work, either following an implantation or as input to new design decisions for a future implementation. This would include an investigation of what educational material would be needed. A recommendation is to research what Telecom Application Developer Hackathon could offer in this area.

This thesis only addresses the use of a single IMS core network, i.e. a single operator. This could be extended to support two operators to test other signaling scenarios. A recommendation is then to investigate the networking between the different IMS functions further.

Naturally, using IMS with media in addition to signaling makes education more realistic. To investigate how the design of an IMS for education could be made accessible for IMS clients and VoLTE clients supporting media would be another area of future work.

7.3 Reflections

This thesis provides insight in the technology used for global communication and contributes to a deeper understanding of the signaling in telecommunications. This equips the public with knowledge, and enables them to make more informed decisions in ethical aspects of political discussion in today's society. One example of this concerns the use of law enforcement to monitor commutation among the public.

From an economic point of view this thesis provides insights in how costs can be reduced, giving an overview of several open source software for use in telecommunications, an area where commercial software is normally used. Open source software is actually already used commercially in countries such as South Africa.

This thesis educates about IMS and VoLTE. These technologies could be used for high quality video calling on a global bases. If implemented by more operators, video calling could be used more frequently which would reduce business traveling and thereby improve the environment.

References

[1] G. Camarillo and M. A. García-Martín, *The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds*, 3rd ed. Chichester, West Sussex, U.K: Wiley, 2008.

[2] P. Synnergren and T. Dudda, 'LTE Latency Improvement Gains', *Ericsson Research Blog*. [Online]. Available: http://www.ericsson.com/research-blog/lte/lte-latency-improvement-gains/. [Accessed: 08-May-2015]

[3] Nokia Siemens Networks, 'Latency - The impact of latency on application performance'. Nokia Simens Networks, 12-Aug-2009 [Online]. Available: http://networks.nokia.com/sites/default/files/document/LatencyWhitepaper.pdf. [Accessed: 08-May-2015]

[4] E. Steele, 'Skype Celebrates a Decade of Meaningful Conversations!', *Skype Blogs*. 28-Aug-2013 [Online]. Available: http://blogs.skype.com/2013/08/28/skype-celebrates-a-decade-of-meaningful-conversations/. [Accessed: 03-May-2015]

[5] S. H. Andras, 'DataPoint: Microsoft's Skype DAU has grown 82 percent in a year', *SocialTimes*. [Online]. Available: http://www.adweek.com/socialtimes/datapoint-microsofts-skype-dau-gains-82-from-a-year-ago-mau-gains-just-1-25/297800. [Accessed: 03-May-2015]

[6] Facebook, Inc., 'Facebook Reports First Quarter 2015 Results - Facebook'. [Online]. Available: http://investor.fb.com/releasedetail.cfm?ReleaseID=908022. [Accessed: 27-Apr-2015]

[7] S. Millward, 'After Rakuten acquisition, Viber reveals it has 100 million active users', *Tech in Asia*. 14-Feb-2014 [Online]. Available: https://www.techinasia.com/rakuten-acquisition-reveals-viber-has-100-million-active-users/. [Accessed: 27-Apr-2015]

[8] C. Cox, *An introduction to LTE LTE, LTE-advanced, SAE, VoLTE and 4G mobile communications*, 2nd ed. Chichester, West Sussex, United Kingdom ; Hoboken, New Jersey: John Wiley & Sons, Inc., 2014.

[9] R.-J. Bartunek, 'Skype summoned to Belgian court over failure to share call data', *Reuters*, 26-May-2015 [Online]. Available: http://www.reuters.com/article/2015/05/26/us-skype-belgium-court-idUSKBN0OB0M320150526. [Accessed: 15-Jun-2015]

[10] Ericsson AB, 'White Paper IMS - IP Multimedia Subsystem'. 15-Nov-2004 [Online]. Available: http://www.techabulary.com/i/ims/ims_ip_multimedia_subsystem.pdf. [Accessed: 14-May-2015]

[11] M. Poikselkä and G. Mayer, *The IMS: IP multimedia concepts and services*, 3rd ed. Chichester, U.K: Hoboken, N.J. : Wiley, 2009.

[12] The 3rd Generation Partnership Project, 'About 3GPP'. [Online]. Available: http://www.3gpp.org/about-3gpp. [Accessed: 18-May-2015]

[13] 3GPP, 'Network architecture', 3rd Generation Partnership Project (3GPP), TS 23.002, Dec. 2014 [Online]. Available: http://www.3gpp.org/DynaReport/23002.htm

[14] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, 'SIP: Session Initiation Protocol', *Internet Req. Comments*, vol. RFC 3261 (Proposed Standard), Jun. 2002 [Online]. Available: http://www.rfc-editor.org/rfc/rfc3261.txt

[15] H. Schulzrinne, 'The tel URI for Telephone Numbers', *Internet Req. Comments*, vol. RFC 3966 (Proposed Standard), Dec. 2004 [Online]. Available: http://www.rfc-editor.org/rfc/rfc3966.txt

[16] R. Jesske, K. Drage, and C. Holmberg, 'Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3GPP', *Internet Req. Comments*, vol. RFC 7315 (Informational), Jul. 2014 [Online]. Available: http://www.rfc-editor.org/rfc/rfc7315.txt

[17] 3GPP, 'Internet Protocol (IP) multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3', 3rd Generation Partnership Project (3GPP), TS 24.229, Mar. 2015 [Online]. Available: http://www.3gpp.org/DynaReport/24229.htm

[18] A. B. Roach, 'SIP-Specific Event Notification', *Internet Req. Comments*, vol. RFC 6665 (Proposed Standard), Jul. 2012 [Online]. Available: http://www.rfc-editor.org/rfc/rfc6665.txt

[19] M. Handley, V. Jacobson, and C. Perkins, 'SDP: Session Description Protocol', *Internet Req. Comments*, vol. RFC 4566 (Proposed Standard), Jul. 2006 [Online]. Available: http://www.rfc-editor.org/rfc/rfc4566.txt

[20] J. Rosenberg and H. Schulzrinne, 'An Offer/Answer Model with Session Description Protocol (SDP)', *Internet Req. Comments*, vol. RFC 3264 (Proposed Standard), Jun. 2002 [Online]. Available: http://www.rfc-editor.org/rfc/rfc3264.txt

[21] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko, 'Diameter Base Protocol', *Internet Req. Comments*, vol. RFC 3588 (Proposed Standard), Sep. 2003 [Online]. Available: http://www.rfc-editor.org/rfc/rfc3588.txt

[22] 3GPP, 'Cx and Dx interfaces based on the Diameter protocol; Protocol details', 3rd Generation Partnership Project (3GPP), TS 29.229, Mar. 2015 [Online]. Available: http://www.3gpp.org/DynaReport/29229.htm

[23] 3GPP, 'IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents', 3rd Generation Partnership Project (3GPP), TS 29.228, Mar. 2015 [Online]. Available: http://www.3gpp.org/DynaReport/29228.htm

[24] 3GPP, 'Sh interface based on the Diameter protocol; Protocol details', 3rd Generation Partnership Project (3GPP), TS 29.329, Jan. 2015 [Online]. Available: http://www.3gpp.org/DynaReport/29329.htm

[25] 3GPP, 'IP Multimedia Subsystem (IMS) Sh interface; Signalling flows and message contents', 3rd Generation Partnership Project (3GPP), TS 29.328, Mar. 2015 [Online]. Available: http://www.3gpp.org/DynaReport/29328.htm

[26] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, 'RTP: A Transport Protocol for Real-Time Applications', *Internet Req. Comments*, vol. RFC 3550 (INTERNET STANDARD), Jul. 2003 [Online]. Available: http://www.rfc-editor.org/rfc/rfc3550.txt

[27] Ericsson, 'Traffic Exploration', 2014. [Online]. Available: http://www.ericsson.com/TET. [Accessed: 29-Apr-2015]

[28] GSM Association, 'GSMA VoLTE initiative', *VoLTE | Technical Projects*, 2015. [Online]. Available: http://www.gsma.com/technicalprojects/volte. [Accessed: 25-May-2015]

[29] GSM Association, 'IMS Profile for Voice and SMS v9.0', GSM Association, IR.92, Apr. 2015 [Online]. Available: http://www.gsma.com/newsroom/wp-content/uploads/IR.92-v9.0.pdf

[30] GSM Association, 'LTE and EPC Roaming Guidelines v13.0', GSM Association, IR.88, May 2015 [Online]. Available: http://www.gsma.com/newsroom/wp-content/uploads/IR.88-v13.0.pdf

[31] GSM Association, 'IMS Roaming and Interworking Guidelines v15.0', GSM Association, IR.65, Oct. 2015 [Online]. Available: http://www.gsma.com/newsroom/wp-content/uploads/IR.65-v15.0.pdf

[32] GSM Association, 'Voice and Video calls over LTE', *VoLTE | Network 2020*, 20-May-2015. [Online]. Available: http://www.gsma.com/network2020/volte/. [Accessed: 25-May-2015]

[33] 'VoLTE', *Wikipedia, the free encyclopedia*. 15-May-2015 [Online]. Available: http://en.wikipedia.org/w/index.php?title=VoLTE&oldid=662417187. [Accessed: 24-May-2015]

[34] D. Vingarzan, P. Weik, and T. Magedanz, 'Design and Implementation of an Open IMS Core', in *Mobility Aware Technologies and Applications*, T. Magedanz, A. Karmouch, S. Pierre, and I. Venieris, Eds. Springer Berlin Heidelberg, 2005, pp. 284–293 [Online]. Available: http://link.springer.com.focus.lib.kth.se/chapter/10.1007/11569510_27. [Accessed: 21-Apr-2015]

[35] iptel.org, 'About SIP Express Router', 2015. [Online]. Available: http://www.iptel.org/ser. [Accessed: 26-May-2015]

[36] Fraunhofer FOKUS NGNI, 'The Open Source IMS Core Project', *OpenIMSCore.org | The Open Source IMS Core Project*, 09-Feb-2015. [Online]. Available: http://openimscore.org/. [Accessed: 26-May-2015]

[37] D. Vingarzan, *Design and Implementation Aspects of Open Source NGN Test-bed Software Toolkits*. Berlin: epubli GmbH, 2013 [Online]. Available: http://nbn-resolving.de/urn:nbn:de:101:1-201401036295. [Accessed: 26-May-2015]

[38] J. Fabini, P. Reichl, A. Poropatich, R. Huber, and N. Jordan, '"IMS in a Bottle": Initial Experiences from an OpenSER-based Prototype Implementation of the 3GPP IP Multimedia Subsystem', in *International Conference on Mobile Business, 2006. ICMB '06*, 2006, pp. 13–13. DOI: 10.1109/ICMB.2006.2

[39] Kamailio SIP Server Project, 'History | Kamailio (OpenSER) SIP Server', 2015. [Online]. Available: http://www.kamailio.org/w/history/. [Accessed: 26-May-2015]

[40] F. Galán, E. García, C. Chávarri, M. Gómez, and D. Fernández, 'Design and implementation of an IP Multimedia Subsystem (IMS) emulator using virtualization techniques', in *Proceedings of the 13th HP OpenView university association workshop (HP-OVUA 2006), Nice*, 2006, pp. 21–24 [Online]. Available: http://www.researchgate.net/profile/Emilio_Garcia4/publication/228345868_Design_and_implementation_of_an_IP_Multimedia_Subsystem_(IMS)_emulator_using_virtualization_techniques/links/09e41505b3ac986ff0000000.pdf. [Accessed: 26-May-2015]

[41] Universidad Politécnica de Madrid, 'Virtual Network User Mode Linux (VNUML)', 13-Feb-2012. [Online]. Available: http://web.dit.upm.es/vnumlwiki/index.php/Main_Page. [Accessed: 26-May-2015]

[42] D. Peterström, *IP Multimedia for Municipalities: The supporting architecture*. 2009 [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A510473&dswid=9999. [Accessed: 15-Jun-2015]

[43] E. S. Raymond, 'The cathedral and the bazaar', *First Monday*, vol. 3, no. 2, Mar. 1998 [Online]. Available: http://firstmonday.org/ojs/index.php/fm/article/view/578. [Accessed: 16-Jun-2015]

[44] A. Kristensen, 'JSR 116: SIP Servlet API', Dynamicsoft, Inc., Mar. 2003 [Online]. Available: http://jcp.org/en/jsr/detail?id=116. [Accessed: 29-May-2015]

[45] Y. Cosmadopoulos and M. Kulkarni, 'JSR 289: SIP Servlet v1.1', Oracle, Aug. 2008 [Online]. Available: http://jcp.org/en/jsr/detail?id=289. [Accessed: 29-May-2015]

[46] N. Khan, 'The SIP Servlet Programming Model', Oracle, Jan. 2007 [Online]. Available: http://www.oracle.com/au/products/database/sip-programming-model-086334.html. [Accessed: 29-May-2015]

[47] P. O'Doherty, 'JSLEE—SIP Servlet', 04-Nov-2003 [Online]. Available: http://www.oracle.com/technetwork/java/jslee-sipservlet-150033.pdf

[48] S. B. Lim and D. Ferry, 'JSR 22: JAINTM SLEE API Specification', Sun Microsystems, Inc. and Open Cloud Limited, JSR 22, Mar. 2004 [Online]. Available: http://jcp.org/en/jsr/detail?id=22. [Accessed: 29-May-2015]

[49] jainslee.org, 'JAIN SLEE and SIP Servlet', Aug-2006. [Online]. Available: http://jainslee. org/othertechnologies/sleevssipservlet.html. [Accessed: 09-Apr-2008]

[50] J. Lennox, H. Schulzrinne, and J. Rosenberg, 'Common Gateway Interface for SIP', *Internet Req. Comments*, vol. RFC 3050 (Informational), Jan. 2001 [Online]. Available: http://www.rfc-editor.org/rfc/rfc3050.txt

[51] H. Sinnreich, Internet communications using SIP: delivering VoIP and multimedia services with Session Initiation Protocol. New York: Wiley Computer Pub, 2001.

[52] J. Lennox, X. Wu, and H. Schulzrinne, 'Call Processing Language (CPL): A Language for User Control of Internet Telephony Services', *Internet Req. Comments*, vol. RFC 3880 (Proposed Standard), Oct. 2004 [Online]. Available: http://www.rfc-editor.org/rfc/rfc3880.txt

[53] M. Rosenblum, 'The Reincarnation of Virtual Machines', *Queue*, vol. 2, no. 5, p. 34, Jul. 2004. DOI: 10.1145/1016998.1017000

[54] Free Software Foundation, Inc., 'General Public License version 2', Jun-1991. [Online]. Available: http://www.gnu.org/licenses/old-licenses/gpl-2.0.html. [Accessed: 14-Jun-2015]

[55] Internet Systems Consortium, 'ISC BIND', 2015. [Online]. Available: https://www.isc.org/downloads/bind/. [Accessed: 14-Jun-2015]

[56] R. Gayraud and O. Jacques, 'Welcome to SIPp', 20-Apr-2014. [Online]. Available: http://sipp.sourceforge.net/. [Accessed: 14-Jun-2015]

[57] PT Inovação, 'IMS-Communicator', *SourceForge*, 2007. [Online]. Available: http://sourceforge.net/projects/imscommunicator.berlios/. [Accessed: 14-Jun-2015]

[58] D. Waiting and R. Good, 'UCT IMS Client', *SourceForge*, 2012. [Online]. Available: http://sourceforge.net/projects/uctimsclient.berlios/. [Accessed: 14-Jun-2015]

[59] Fraunhofer FOKUS, 'OpenIC Lite', 2008. [Online]. Available: http://www.fokus.fraunhofer.de/bereichsseiten/testbeds/ims_playground/components/open_ic.php?lang=en#openiclite

[60] The Apache Software Foundation, 'The Apache Software License: Version 1.1', 2000. [Online]. Available: http://www.apache.org/licenses/LICENSE-1.1. [Accessed: 14-Jun-2015]

[61] Free Software Foundation, Inc., 'GNU Lesser General Public License, version 2.1', Feb-1999. [Online]. Available: http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html. [Accessed: 14-Jun-2015]

[62] Free Software Foundation, Inc., 'General Public License version 3', Jun-2007. [Online]. Available: http://www.gnu.org/copyleft/gpl.html. [Accessed: 14-Jun-2015]

[63] Sun Microsystems, 'Common Development and Distribution License', Apr-2005. [Online]. Available: http://opensource.org/licenses/CDDL-1.0. [Accessed: 16-Jun-2015]

[64] Oracle, 'SailFin Project', 2015. [Online]. Available: https://sailfin.java.net/. [Accessed: 03-Jun-2015]

[65] VozTelecom, 'WeSIP', 2015. [Online]. Available: http://www.wesip.com/. [Accessed: 16-Jun-2015]
[66] Jamie Cameron and the Webmin community., 'Webmin', 2015. [Online]. Available: http://www.webmin.com/. [Accessed: 16-Jun-2015]
[67] QuasIdea Development, LLC, 'Qcodo Development Framework', *Qcodo Development Framework - Home*, 2015. [Online]. Available: http://www.qcodo.com/. [Accessed: 03-Jun-2015]
[68] The PHP Group, 'PHP: Hypertext Preprocessor', 2015. [Online]. Available: http://www.php.net/. [Accessed: 16-Jun-2015]
[69] K. Meirlaen, 'QCodo Manual', Sep-2007. [Online]. Available: http://qcodo.kri-soft.be/qcodo_manual_v05.pdf. [Accessed: 03-Apr-2008]
[70] T. Ylonen and C. Lonvick, 'The Secure Shell (SSH) Protocol Architecture', *Internet Req. Comments*, vol. RFC 4251 (Proposed Standard), Jan. 2006 [Online]. Available: http://www.rfc-editor.org/rfc/rfc4251.txt
[71] The PHP Group, 'PHP: ssh2 -Manual', 2015. [Online]. Available: http://se2.php.net/ssh2. [Accessed: 16-Jun-2015]
[72] F. M.A.d.S. and M. Hurni, 'CodePress - Real Time Syntax Highlighting Editor written in JavaScript', 2015. [Online]. Available: http://codepress.sourceforge.net/. [Accessed: 16-Jun-2015]
[73] The PHP Group, 'PHP: call_user_func -Manual', 2015. [Online]. Available: http://se2.php.net/call_user_func. [Accessed: 16-Jun-2015]
[74] Oracle, 'MySQL :: The world's most popular open source database', 2015. [Online]. Available: http://www.mysql.com/. [Accessed: 16-Jun-2015]
[75] The CentOS Project, 'CentOS Project', 2015. [Online]. Available: http://centos.org/. [Accessed: 16-Jun-2015]
[76] Canonical Ltd, 'The leading OS for PC, tablet, phone and cloud | Ubuntu', 2015. [Online]. Available: http://www.ubuntu.com/. [Accessed: 16-Jun-2015]
[77] VMware, Inc., 'VMware Server', 2008. [Online]. Available: http://www.vmware.com/products/server/. [Accessed: 04-Apr-2008]
[78] Oracle, 'Oracle VM VirtualBox', 2015. [Online]. Available: https://www.virtualbox.org/. [Accessed: 04-Jun-2015]
[79] Citrix Sustems, Inc., 'Citrix XenServer', Apr-2008. [Online]. Available: http://www.citrixxenserver.com/. [Accessed: 11-Apr-2008]
[80] M. Rosenblum, 'The Reincarnation of Virtual Machines', *Queue*, vol. 2, no. 5, p. 34, Jul. 2004. DOI: 10.1145/1016998.1017000
[81] XenSource, Inc., 'XenServer Installation Guide -Chapter 2. System Requirements', Dec-2007. [Online]. Available: http://docs.xensource.com/XenServer/4.0.1/installation/ch02.html. [Accessed: 11-Apr-2008]
[82] 'Wiki', *Wikipedia, the free encyclopedia*. 27-May-2015 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wiki&oldid=664265890. [Accessed: 16-Jun-2015]
[83] Microsoft, 'Microsoft Word | Document and Word Processing Software', *Microsoft Office*, 2015. [Online]. Available: https://products.office.com/en-us/word. [Accessed: 16-Jun-2015]
[84] The Apache Software Foundation, 'Apache OpenOffice - Official Site - The Free and Open Productivity Suite', 2015. [Online]. Available: https://www.openoffice.org/. [Accessed: 16-Jun-2015]
[85] N. Walsh, *DocBook: the definitive guide*. Beijing ; Sabastopol, CA: O'Reilly & Associates, 1999.
[86] B. Guillon, 'dblatex', 2015. [Online]. Available: http://dblatex.sourceforge.net/. [Accessed: 16-Jun-2015]

[87] C. M. Pilato, B. Collins-Sussman, and B. W. Fitzpatrick, 'Version Control with Subversion', 04-May-2015. [Online]. Available: http://svnbook.red-bean.com/. [Accessed: 14-Jun-2015]

[88] World Wide Web Consortium, 'W3C XHTML2 Working Group Home Page', 2013. [Online]. Available: http://www.w3.org/MarkUp/. [Accessed: 16-Jun-2015]

[89] D.-C. Mierla and E.-R. Modroiu, 'Kamailio (OpenSER) - Debug and syslog messages', 2015. [Online]. Available: http://www.kamailio.org/dokuwiki/doku.php/tutorials:debug-syslog-messages. [Accessed: 16-Jun-2015]

[90] Apache Software Foundation, 'Apache log4j 1.2', 2012. [Online]. Available: http://logging.apache.org/log4j/1.2/index.html. [Accessed: 16-Jun-2015]

[91] Wireshark Foundation, 'Wireshark · Go Deep.', 2015. [Online]. Available: https://www.wireshark.org/. [Accessed: 04-Jun-2015]

[92] F. Edler, 'Mail from Franz Edler on SIPp-users mailing list', 23-Jan-2007 [Online]. Available: http://www.mail-archive.com/sipp-users@lists.sourceforge.net/msg00856.html. [Accessed: 14-Jun-2015]

[93] A. Johnston, R. Sparks, C. Cunningham, S. Donovan, and K. Summers, 'Session Initiation Protocol Service Examples', *Internet Req. Comments*, vol. RFC 5359 (Best Current Practice), Oct. 2008 [Online]. Available: http://www.rfc-editor.org/rfc/rfc5359.txt

[94] SailFin team, 'SipServletProxyExample', *GlassFish Wiki*, 18-Nov-2011. [Online]. Available: https://wikis.oracle.com/display/GlassFish/SipServletProxyExample. [Accessed: 16-Jun-2015]

[95] SailFin team, 'Using NetBeans 5.5.1 to develop SIP and Java EE Applications', Oct-2006. [Online]. Available: https://glassfishplugins.java.net/sip/. [Accessed: 16-Jun-2015]

[96] QuasIdea Development, LLC, 'Qcodo: Presentation and demos', 2015. [Online]. Available: http://www.qcodo.com/wiki/qcodo/demos. [Accessed: 16-Jun-2015]

[97] QuasIdea Development, LLC, 'Qcodo: ACME Enterprises - Issue Tracker Pro - Project Dashboard', 2008. [Online]. Available: http://demo.qcodo.com/dashboard/. [Accessed: 16-Jun-2015]

[98] The Eclipse Foundation, 'PDT: Eclipse PHP Development Tools', 2015. [Online]. Available: http://www.eclipse.org/pdt/. [Accessed: 16-Jun-2015]

[99] World Wide Web Consortium, 'Cascading Style Sheets', 12-Jun-2015. [Online]. Available: http://www.w3.org/Style/CSS/. [Accessed: 16-Jun-2015]

[100] Mozilla, 'Firebug', 2015. [Online]. Available: http://www.getfirebug.com/. [Accessed: 16-Jun-2015]

[101] VMware, Inc., 'VMware Server Documentation', 2015. [Online]. Available: http://www.vmware.com/support/pubs/server_pubs.html. [Accessed: 16-Jun-2015]

[102] VMware, Inc., 'VMware Security Advisories VMSA-2010-0007.1', 12-Apr-2010. [Online]. Available: http://www.vmware.com/security/advisories/VMSA-2010-0007.html. [Accessed: 29-May-2015]

[103] M. Fowler, 'Continuous Integration', *martinfowler.com*, 01-May-2006. [Online]. Available: http://martinfowler.com/articles/continuousIntegration.html. [Accessed: 29-May-2015]

[104] P. M. Duvall, S. Matyas, and A. Glover, *Continuous Integration: Improving Software Quality and Reducing Risk*, 1 edition. Upper Saddle River, NJ: Addison-Wesley Professional, 2007.

[105] B. Rhodes, 'The Network Playground', *GitHub*, 25-Oct-2014. [Online]. Available: https://github.com/brandon-rhodes/fopnp/tree/m/playground. [Accessed: 01-Jun-2015]

[106] The Tcpdump team, 'Tcpdump/Libpcap public repository', 2015. [Online]. Available: http://www.tcpdump.org/. [Accessed: 04-Jun-2015]
[107] QXIP/SIPCAPTURE, 'Homer SipCapture', *Homer SipCapture - 100% Open Source SIP Capture Server & Agents*, 2015. [Online]. Available: http://sipcapture.org/. [Accessed: 04-Jun-2015]
[108] Kamailio SIP Server Project, 'Kamailio SIP Server'. Jun-2015 [Online]. Available: http://www.kamailio.org/w/. [Accessed: 03-Jun-2015]
[109] *Chat with James Body and Carsten Bock - Kamailio Open Source VoLTE*. London, 2015 [Online]. Available: https://www.youtube.com/watch?v=-7RHktpMEyc. [Accessed: 03-Jun-2015]
[110] Nexcom, 'cipango | SIP/HTTP Servlets Application Server', 08-Feb-2011. [Online]. Available: http://www.cipango.org/. [Accessed: 03-Jun-2015]
[111] Nexcom, 'cipango/cipango', *GitHub*, 2015. [Online]. Available: https://github.com/cipango/cipango. [Accessed: 03-Jun-2015]
[112] TeleStax, Inc., 'Mobicents/sip-servlets', *GitHub*, 03-Jun-2015. [Online]. Available: https://github.com/Mobicents/sip-servlets. [Accessed: 03-Jun-2015]
[113] TeleStax, Inc., 'sip servlets', *Telestax Docs Online*, 2015. [Online]. Available: http://docs.telestax.com/sip-servlets-homepage/. [Accessed: 03-Jun-2015]
[114] M. Nudelman, 'less(1) - Linux manual page', 04-Apr-2013. [Online]. Available: http://man7.org/linux/man-pages/man1/less.1.html. [Accessed: 14-Jun-2015]
[115] Free Software Foundation, Inc., 'GNU grep - GNU Grep: Print lines matching a pattern', *GNU Project*, 23-Nov-2014. [Online]. Available: http://www.gnu.org/software/grep/manual/. [Accessed: 14-Jun-2015]
[116] N. Marriott, 'tmux', 2015. [Online]. Available: http://tmux.github.io/. [Accessed: 04-Jun-2015]
[117] Canonical Ltd, 'LTS - Ubuntu Wiki', 23-Oct-2013. [Online]. Available: https://wiki.ubuntu.com/LTS. [Accessed: 04-Jun-2015]
[118] The Xubuntu community, 'Xubuntu', 2015. [Online]. Available: http://xubuntu.org/. [Accessed: 04-Jun-2015]
[119] 'cgroups', *Wikipedia, the free encyclopedia*. 15-May-2015 [Online]. Available: http://en.wikipedia.org/w/index.php?title=Cgroups&oldid=662396135. [Accessed: 04-Jun-2015]
[120] 'Operating-system-level virtualization', *Wikipedia, the free encyclopedia*. 03-Jun-2015 [Online]. Available: http://en.wikipedia.org/w/index.php?title=Operating-system-level_virtualization&oldid=665261742. [Accessed: 04-Jun-2015]
[121] Docker, Inc., 'Docker - Build, Ship, and Run Any App, Anywhere', 2015. [Online]. Available: https://www.docker.com/. [Accessed: 04-Jun-2015]
[122] 'Docker (software)', *Wikipedia, the free encyclopedia*. 31-May-2015 [Online]. Available: http://en.wikipedia.org/w/index.php?title=Docker_(software)&oldid=664920107. [Accessed: 04-Jun-2015]
[123] HashiCorp, Inc., 'Vagrant', 2015. [Online]. Available: https://www.vagrantup.com/. [Accessed: 04-Jun-2015]
[124] HashiCorp, Inc., 'Discover Vagrant Boxes', *Atlas by HashiCorp*, 2015. [Online]. Available: https://atlas.hashicorp.com/boxes/search. [Accessed: 04-Jun-2015]
[125] 'Lightweight markup language', *Wikipedia, the free encyclopedia*. 30-May-2015 [Online]. Available: http://en.wikipedia.org/w/index.php?title=Lightweight_markup_language&oldid=664700376. [Accessed: 10-Jun-2015]
[126] The Daring Fireball Company LLC., 'Daring Fireball: Markdown', 2015. [Online]. Available: http://daringfireball.net/projects/markdown/. [Accessed: 10-Jun-2015]

[127] S. Rackham, ‘AsciiDoc Home Page’, 26-Feb-2015. [Online]. Available: http://asciidoc.org/index.html. [Accessed: 10-Jun-2015]
[128] GitHub, Inc., ‘About GitHub’, *GitHub*, 2015. [Online]. Available: https://github.com. [Accessed: 10-Jun-2015]
[129] O’Reilly Media, Inc., ‘Atlas Documentation: Writing in Markdown’, *Atlas Documentation*, 2013. [Online]. Available: http://docs.atlas.oreilly.com/ch13.html. [Accessed: 10-Jun-2015]
[130] L. Mangani, ‘Troubleshooting SIP Signalling’, Berlin, 27-May-2015 [Online]. Available: https://www.youtube.com/watch?v=LDZYw_sGBzI. [Accessed: 14-Jun-2015]
[131] A. Dubovokov, ‘Introducing Homer and Captagent 5’, Berlin, 28-May-2015 [Online]. Available: https://www.youtube.com/watch?v=Z5IllQMSyKY. [Accessed: 14-Jun-2015]
[132] C. Bock, ‘Kamailio for Building IMS Core Platforms and VoLTE’, Berlin, 27-May-2015 [Online]. Available: https://www.youtube.com/watch?v=rbNP-PN6BOY
[133] Kamailio SIP Server Project, ‘Kamailio SIP Server (SER) - New Features in v4.3.x’, *Kamailio SIP Server Wiki*, 10-Jun-2015. [Online]. Available: http://www.kamailio.org/wiki/features/new-in-4.3.x. [Accessed: 15-Jun-2015]
[134] J. Deruelle, *Mobicents SIP Servlets 2.0.0.Final HTML5 WebRTC Demo*. 2012 [Online]. Available: https://vimeo.com/51744602. [Accessed: 14-Jun-2015]
[135] ‘WebRTC’, *Wikipedia, the free encyclopedia*. 21-May-2015 [Online]. Available: https://en.wikipedia.org/w/index.php?title=WebRTC&oldid=663382827. [Accessed: 15-Jun-2015]
[136] S. Chacon and B. Straub, *Pro Git*, 2nd ed. New York, NY: Apress, 2014 [Online]. Available: http://git-scm.com/book/en/v2. [Accessed: 14-Jun-2015]
[137] ‘Advanced Packaging Tool’, *Wikipedia, the free encyclopedia*. 30-Apr-2015 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Advanced_Packaging_Tool&oldid=660054442. [Accessed: 15-Jun-2015]
[138] Docker, Inc., ‘The Docker User Guide’, The Docker User Guide - Docker Documentation, 05-Jun-2015. [Online]. Available: https://docs.docker.com/userguide/. [Accessed: 14-Jun-2015]
[139] HashiCorp, Inc., ‘Vagrant Documentation’, 2015. [Online]. Available: https://docs.vagrantup.com/v2/. [Accessed: 14-Jun-2015]
[140] L. Torvalds, ‘I’m combining elements from two very different projects I started.’, 06-Mar-2014. [Online]. Available: https://plus.google.com/+LinusTorvalds/posts/X2XVf9Q7MfV. [Accessed: 14-Jun-2015]
[141] 3GPP, ‘Signalling flows for the session setup in the IP Multimedia core network Subsystem (IMS) based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3’, 3rd Generation Partnership Project (3GPP), TR 24.930, Mar. 2015 [Online]. Available: http://www.3gpp.org/DynaReport/24930.htm
[142] 3GPP, ‘Signalling flows for the IP multimedia call control based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3’, 3rd Generation Partnership Project (3GPP), TS 24.228, Oct. 2006 [Online]. Available: http://www.3gpp.org/DynaReport/24228.htm
[143] J. Repiquet, ‘into the 3GPP Evolved Packet System (EPS)’, *in2EPS*, 26-May-2015. [Online]. Available: http://in2eps.com/. [Accessed: 19-Jun-2015]
[144] 3GPP, ‘Service requirements for the Internet Protocol (IP) multimedia core network subsystem (IMS); Stage 1’, 3rd Generation Partnership Project (3GPP), TS 22.228, Dec. 2014 [Online]. Available: http://www.3gpp.org/DynaReport/22228.htm

[145] 3GPP, 'IP Multimedia Subsystem (IMS); Stage 2', 3rd Generation Partnership Project (3GPP), TS 23.228, Mar. 2015 [Online]. Available: http://www.3gpp.org/DynaReport/23228.htm

[146] J. Repiquet, 'IMS Registration signalling flow - User not registered', *in2EPS*, 26-May-2015. [Online]. Available: http://in2eps.com/fo-ims/tk-fo-ims-regflow-1.html. [Accessed: 19-Jun-2015]

[147] J. Repiquet, 'IMS Session Initiation Signalling flow - MO#2 / S-S#2 / MT#2 (1 of 3)', *in2EPS*, 26-May-2015. [Online]. Available: http://in2eps.com/fo-ims/tk-fo-ims-sesflow-2.html. [Accessed: 19-Jun-2015]

Appendix A Signaling flows

Appendix A gives an overview of the signaling flows required by the customer, as stated in the high-level requirements in Table 3-1. The following sections include the signaling flows for:

- Registration of an IMS client.
- A successful session from one IMS client to another.
- An example of blocking a session using the example-application Blacklist.

In 3GPP Release 7 and later releases, 3GPP TR 24.930 [141] is used, for example, signaling flows. The information in this chapter is based on the information in 3GPP TS 24.228 [142] from Release 5, as it clearly shows the I-CSCF and the S-CSCF as separate entities, as compared to 3GPP TR 24.930. Good and accessible information about the 3GPP specifications and detailed example signaling flows are also provided by Repiquet at his website in2EPS [143], which has been used for this appendix.

The interested who wants to get a deeper understanding of IMS from the 3GPP specifications should begin by reading the following specifications:

- 3GPP TS 22.228 [144], presenting the service requirements for IMS; and
- 3GPP TS 23.228 [145], presenting the architecture and main signaling flows of IMS.

A.1 Registration of IMS client

Figure A-1 and Figure A-2 shows the signaling during a normal, successful registration of the UE with the IMS network. The system also needs to be able handle signaling flows where errors occur, but that is not presented in this appendix. The purpose with the registration is to inform S-CSCF that the UE is available and how it can be reached. The text below is inspired by the presentation of Repiquet [146]. Figure A-1 shows that the IMS network challenges the UE to authenticate itself. In the scenario of this project, the UE has the IP address and the address of the P-CSCF preconfigured in the SIPp scripts. In other cases the P-CSCF is discovered using some other procedures, such as DHCP, depending on the access network. The following happens in the different steps of Figure A-1:

a) A SIP REGISTER request is sent from the UE to the P-CSCF. The P-CSCF verifies the SIP message.

b) Based on the user's URI, the P-CSCF uses DNS to find which I-CSCF to contact, which in this case is an I-CSCF in the user's home network.

c) The P-CSCF modifies the SIP message, to assure that all signaling will pass via the P-CSCF, before forwarding it to the I-CSCF.

d) The I-CSCF queries the HSS about the user's registration status. The HSS responds with the capabilities required by the S-CSCF, which the I-CSCF uses to select S-CSCF.

e) The I-CSCF, unlike the P-CSCF, does not modify the message in order to stay in the path of future communication. The S-CSCF stores information from the message about which route to the UE the messages should take in future sessions, i.e. via the P-CSCF.

f) The REGISTER request arrived to the P-CSCF without integrity protection. Therefore the S-CSCF requests at least one authentication vector from the HSS, and informs the HSS that this S-CSCF has been selected to serve the user.

g) The S-CSCF selects an authentication vector to use.

h) The S-CSCF challenges the user by using a “401 Unauthorized”-response. The response contains the challenge, and in the case IPsec is used the response contains keys for using IPsec.

i) The response is forwarded to the P-CSCF.

j) The P-CSCF removes any IPsec keys before forwarding the challenge to the UE.

Figure A-2 shows how the UE responds to the challenge given in Figure A-1 and is successfully registered using the following steps:

a) The new SIP REGISTER request contains the private identity (IMPI) and responds to the challenge. If IPsec is used, this response is protected by IPsec.

b) The P-CSCF queries the DNS in the same way as above.

c) The request is forwarded to the I-CSCF.

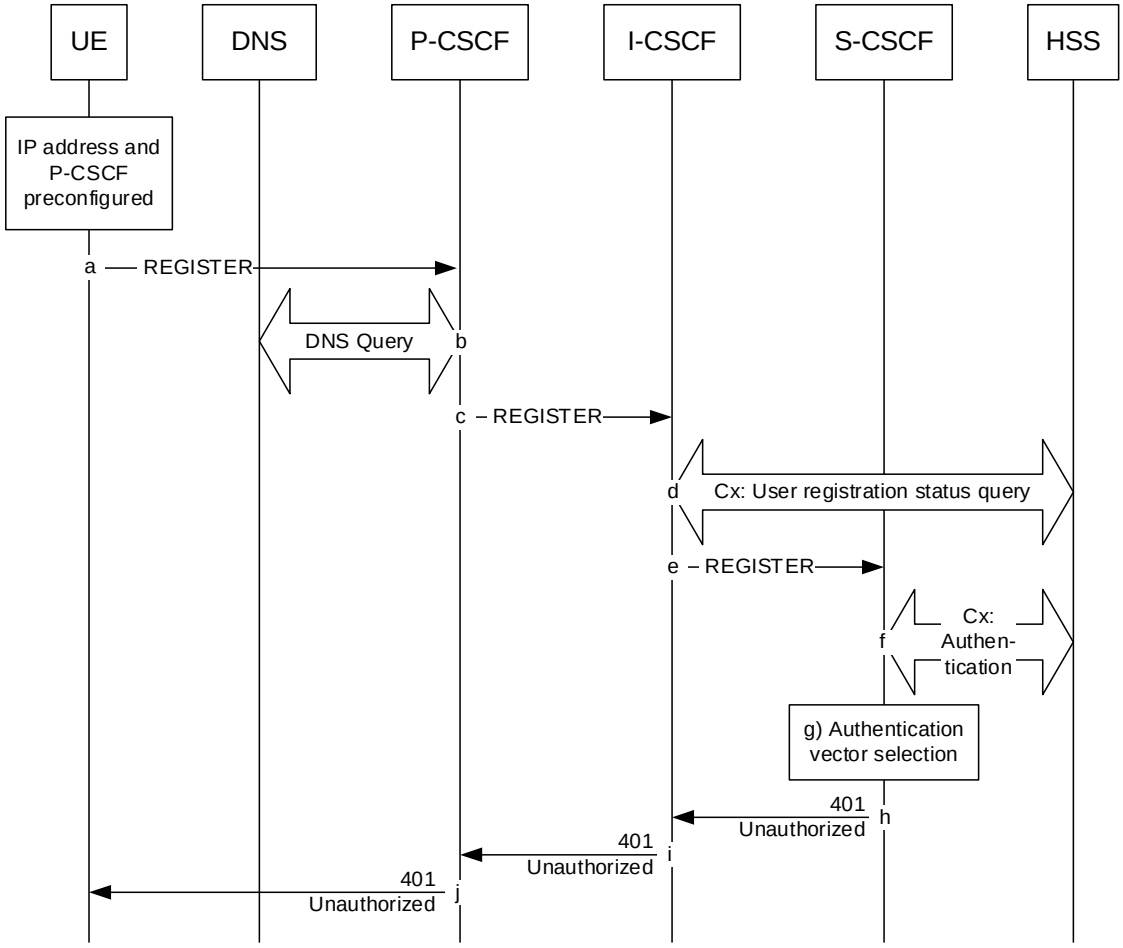

Figure A-1 IMS Registration part 1 - REGISTRATION and 401 Unauthorized (challenge)

d) The I-CSCF queries the HSS and is informed about the S-CSCF that was selected above.

e) The request is forwarded to the S-CSCF.

f) The S-CSCF verifies the response and registers the public user identity.

g) The HSS is informed that the user is registered. If requested by the S-CSCF the HSS includes the user profile in the response.

h) The S-CSCF responds “200 OK”, indicating that the registration was successful. The S-CSCF includes its own URI, which is stored by the UE and P-CSCF for later use.

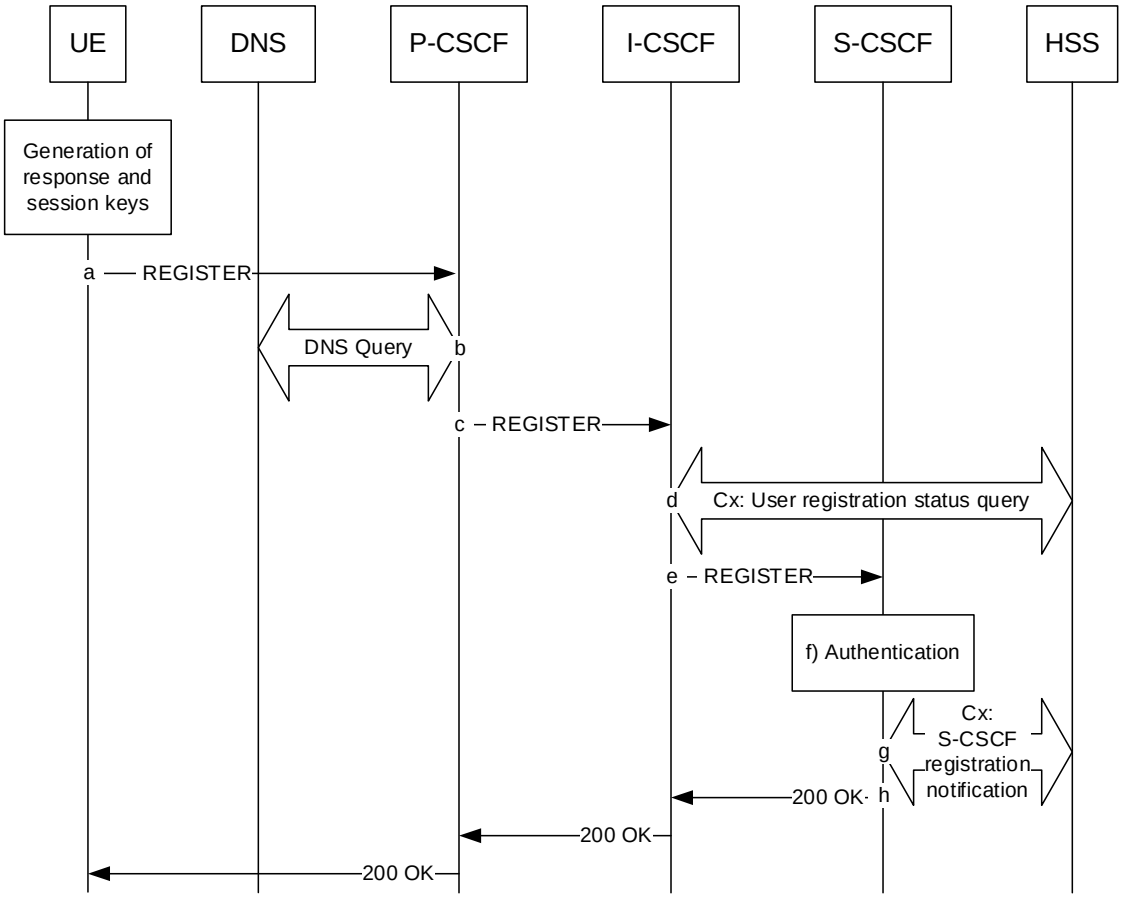

Figure A-2 IMS Registration part 2 - REGISTRATION (response) and 200 OK

A.2 IMS-IMS session

Figure A-3 shows a simplified successful IMS-client to IMS-client session, based on examples from 3GPP TS 24.228 [142] and as explained by Repiquet [147]. In contrast to the specification the example below is simplified and does not include the following:

- The provisional response “183 Session progress”.
- Authorization and approval of QoS resources and associated UPDATE message.
- The use of provisional acknowledgement (PRACK request) to acknowledge the provisional responses 183 and 180.

In Figure A-3, UE1 uses an INVITE request to initiate a session with UE2. The INVITE request contains a body of SDP, describing the media that UE1 is proposing to use. The different parties in the flow indicate that they have received the INVITE request using a “100 Trying”-response. This stops the sender from retransmitting the INVITE. Things worth mentioning in the session are that:

- P-CSCF1 is discovered by the UE, as described in Section A.1.
- P-CSCF1 knows the address of the S-CSCF corresponding to the user, i.e. S-CSCF1, from a previous registration of UE1.
- The P-CSCFs modifies the messages to assure that they remain in the signaling path of the session.
- The I-CSCF2 queries the HSS to find the S-CSCF of UE2.
- In this example the evaluation of the Initial filter criteria, done by S-CSCF1 and S-CSCF2 for UE1 and UE2 respectively, does not result in any further action.
- UE2 indicates for the user that there is an incoming call. This is also signaled to UE1, using the “180 Ringing”-response.
- When the user of UE2 answers the invitation to the session, this is signaled using the “200 OK”-response. The response message contains a body with the SDP media description that UE2 is willing to establish.
- Finally, the “200 OK”-response is acknowledged using an ACK request.

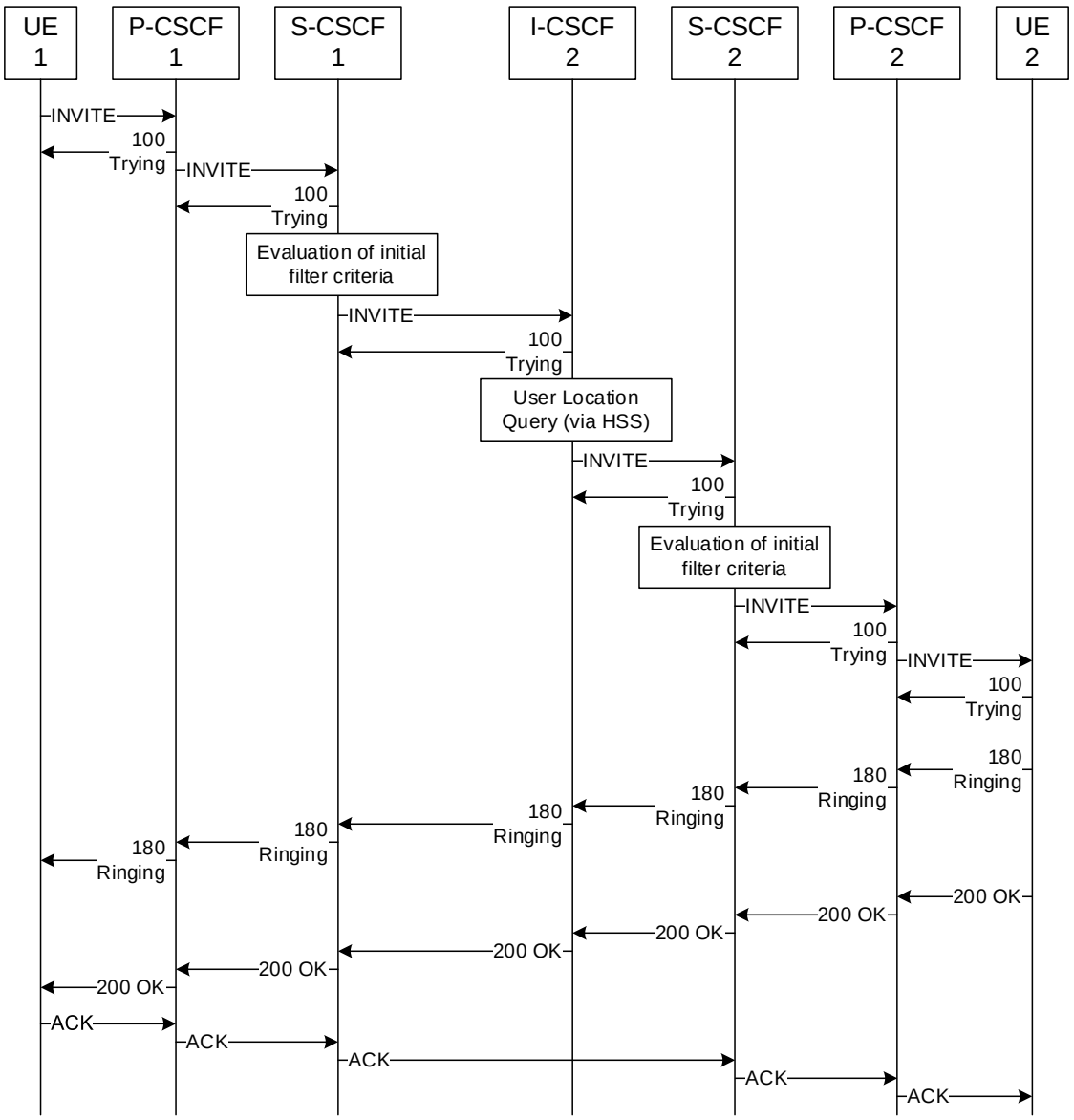

Figure A-3 IMS-IMS Session part 1 - INVITE

A.3 SIP-AS (example of basic functionality)

This section shows examples of the Blacklist application, created as part of the previous work and which implementation is described in Section 4.3. The signaling flows are very similar to the ones in Section 0 except that the initial filter criteria specifies that the AS should be involved in the signaling flow. Figure A-4 shows how the user of UE1 is blocked by the AS since the user of UE2 has enabled the Blacklist application and used it to block the user of UE1. In this scenario the Blacklist application acts as a terminating UA, as described in Section 2.1.6 and shown in Figure 4.3a of 3GPP TS 23.228 [145].

Figure A-5 shows the case where UE2 has the Blacklist application enabled, but has not blocked the user of UE1. In this case the Blacklist application acts as a SIP proxy, as described in Section 2.1.6 and shown in Figure 4.3c of 3GPP TS 23.228 [145].

There are some things worth noting in Figure A-5. The S-CSCF performs an evaluation of the initial filter criteria every time it gets a new initial request. This is done after the AS has handled the INVITE the first time and forwards it back to the S-CSCF. Moreover, the I-CSCF and the AS do not request to stay in the path and therefore the ACK request bypasses them.

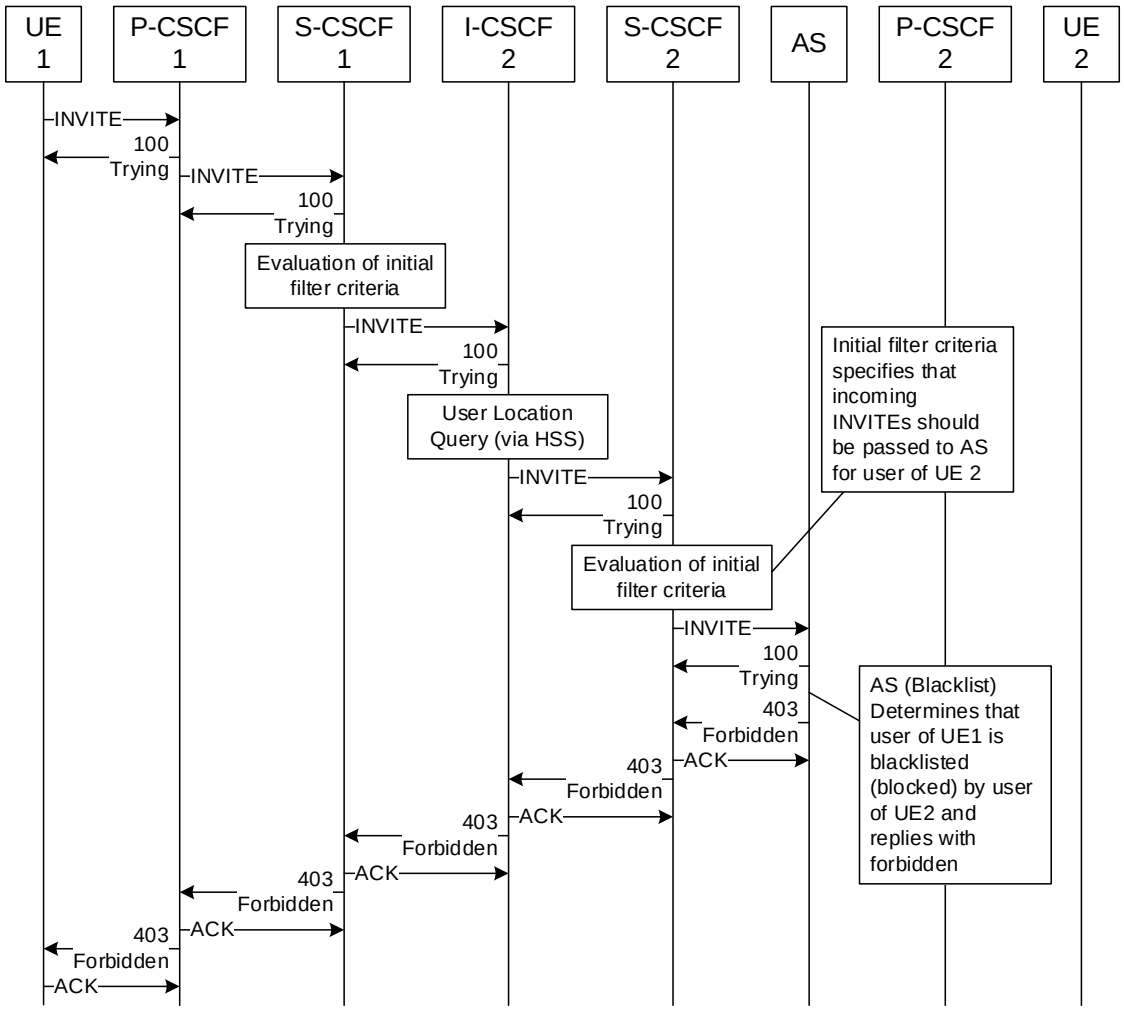

Figure A-4 IMS SIP-AS session, call blocked

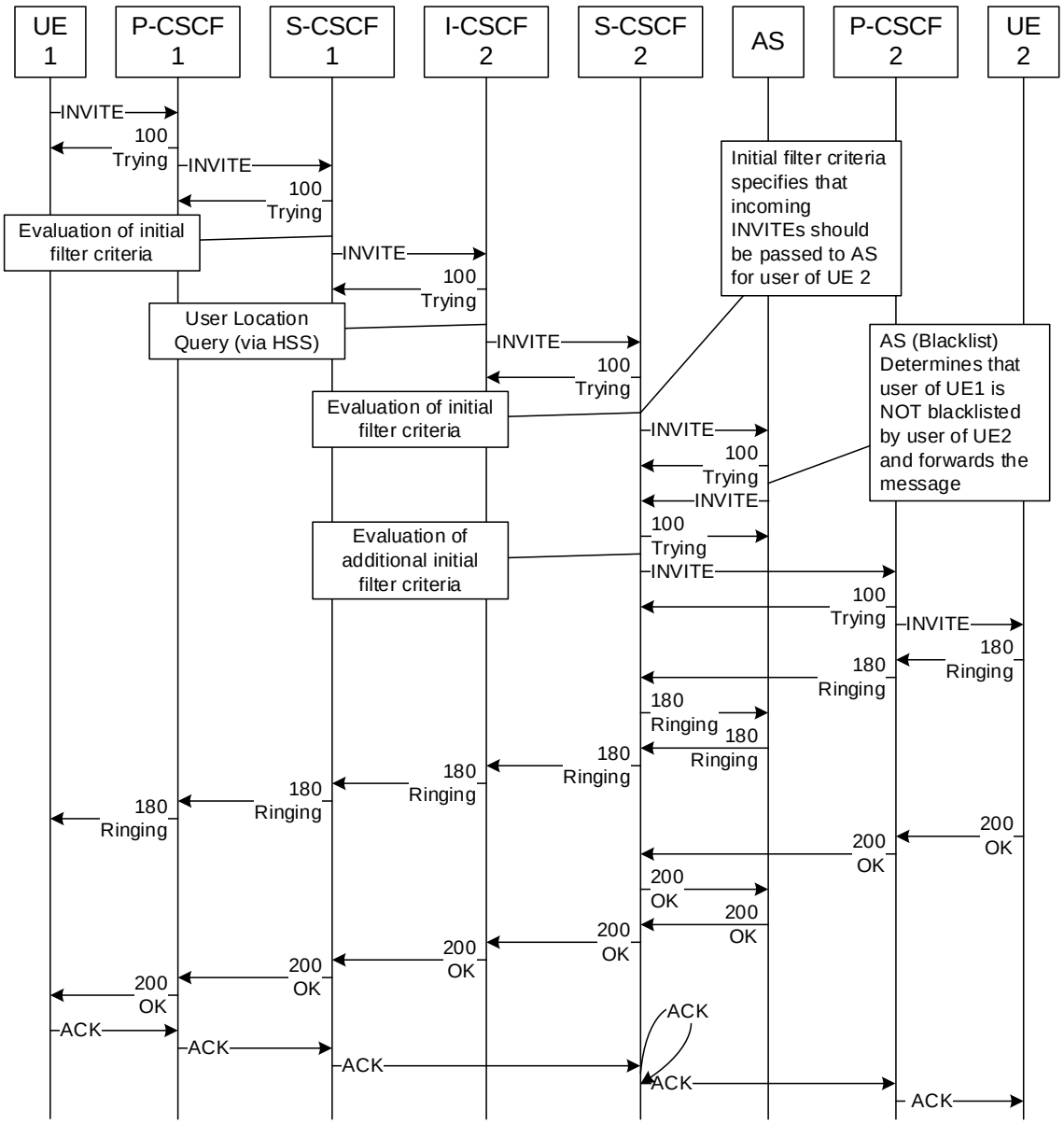

Figure A-5 IMS SIP-AS session, call not blocked

TRITA-ICT-EX-2015:130