Institutionen för datavetenskap
Department of Computer and Information Science

Master's Thesis

Open Source SIP Application Servers For IMS Applications: A Survey

Veronica Kumlin

Reg Nr: LITH-IDA-EX--07/066--SE
Linköping 2007

Linköpings universitet
INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science
Linköpings universitet
SE-581 93 Linköping, Sweden

Institutionen för datavetenskap
Department of Computer and Information Science

Master's Thesis

Open Source SIP Application Servers For IMS Applications: A Survey

Veronica Kumlin

Reg Nr: LITH-IDA-EX--07/066--SE
Linköping 2007

Supervisor: **Jimmy Holmgren**
Cybercom Nord AB
Johan Bergqvist
Cybercom Nord AB
David Byers
IDA, Linköpings universitet

Examiner: **Nahid Shahmehri**
IDA, Linköpings universitet

Department of Computer and Information Science
Linköpings universitet
SE-581 93 Linköping, Sweden

Avdelning, Institution
Division, Department

Division for Database and Information Techniques
Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Datum
Date

2007-12-17

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ _____

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ _____

ISBN
__

ISRN
LITH-IDA-EX--07/066--SE

Serietitel och serienummer **ISSN**
Title of series, numbering __

URL för elektronisk version
http://www.ida.liu.se/divisions/adit
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-ZZZZ

Titel
Title Open Source SIP Application Servers For IMS Applications: A Survey

Författare
Author Veronica Kumlin

Sammanfattning
Abstract

In an IMS (IP Multimedia Subsystem) network there are Application Servers (ASs) which host and execute services. These are often SIP (Session Initiation Protocol) servers. The goal for this thesis was to evaluate a selection of open source SIP ASs to see which one was most suited for IMS service development.

Ten different SIP ASs was found in an initial search. They were briefly reviewed and some of them were chosen as candidates for further investigation.

The documentation of the chosen servers were thoroughly explored and evaluated; as was the installation process and how to deploy a service to them.

Finally the servers were installed and a service was implemented and deployed to an IMS network.

Unfortunately, I could not point out one of the ASs to be more suited than another for deploying an IMS service on. This was because they had different pros and cons and none of them were distinguishing in a way that made them superior to the other.

Nyckelord
Keywords SIP application server, IMS

Abstract

In an IMS (IP Multimedia Subsystem) network there are Application Servers (ASs) which host and execute services. These are often SIP (Session Initiation Protocol) servers. The goal for this thesis was to evaluate a selection of open source SIP ASs to see which one was most suited for IMS service development.

Ten different SIP ASs was found in an initial search. They were briefly reviewed and some of them were chosen as candidates for further investigation.

The documentation of the chosen servers were thoroughly explored and evaluated; as was the installation process and how to deploy a service to them.

Finally the servers were installed and a service was implemented and deployed to an IMS network.

Unfortunately, I could not point out one of the ASs to be more suited than another for deploying an IMS service on. This was because they had different pros and cons and none of them were distinguishing in a way that made them superior to the other.

Sammanfattning

I ett IMS-nät (IP Multimedia Subsystem) finns Applikations Servrar (AS) som tillhandahåller tjänster, dessa AS är ofta SIP-servrar (Session Initation Protocol).

Målet för det här examensarbetet var att finna den SIP-server som är mest lämpad att använda när man ska lägga till en tjänst i ett IMS-nät.

Först gjordes en undersökning av tillgängliga SIP AS och ett tiotal hittades. Några av dem valdes ut för att jämföras och utredas mer noggrant. Inledningsvis inspekterades deras dokumentation, sedan analyserades proceduren att installera AS och att lägga till en tjänst på dem. Till sist lades tjänsten till i ett IMS-nät.

Eftersom varken deras fördelar eller nackdelar utmärkte sig på ett sådant sett att någon av dem sammantaget var bättre än den andra, kunde tyvärr inte någon av dem pekas ut som den mest lämpade.

Acknowledgments

I would like to acknowledge my colleagues, Börje Granberg and Theo Öjerteg, for helping and acting as sounding boards.

I would also like to thank my academic tutor, David Byers, especially for help with the report, and my tutors at the company, Johan Bergqvist and Jimmy Holmgren for their assistance and support with this thesis.

Finally I would like to thank my mother, Irèné Kumlin, for refining the language in the report and Susanne Karlsson for making coffee and being a cheerful soul.

Contents

Chapter 1

Introduction

1.1 Purpose

In an IMS (IP Multimedia Subsystem) network (described in ch 2) there are ASs (Application Servers) which host and execute services (see left part of figure 1.1). These ASs may be and often are SIP (Session Initiation Protocol, also described in ch 2) servers.

The purpose of this thesis is to examine available SIP ASs and after that choose which is the best and easiest one to implement a service on (see right part of figure 1.1) according to a number of parameters presented later. It is highly desirable to know the easiest way to do this, because in that case the creators of the service do not have to be highly specialized in the underlying technique; they can be the ones who have the expertise on the particular service.

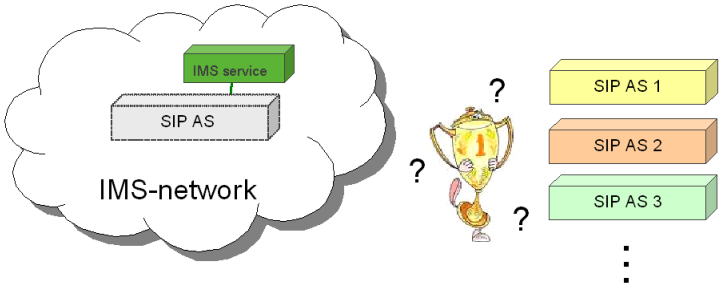

Figure 1.1. Selection of SIP AS

1.2 Question at issue

The question at issue is: Which SIP AS is the best suited one to implement an IMS service on? In order to find the answer the following questions will be investigated:

1. Is the AS's website intuitive and easy to use?
2. Is it easy to find documentation about the AS?
3. Is there a beginner's guide?
4. Will everything work after following the installation guide?
5. Is it dependent on a particular operating system?
6. Are there any dependencies to other programs?
7. Is it easy to deploy the service implemented on the AS to an IMS network?

1.3 Goal

The goal is to find the most suitable SIP AS to use when a service in an IMS network is going to be developed (see figure 1.2). Focus will be on user-friendlieness.

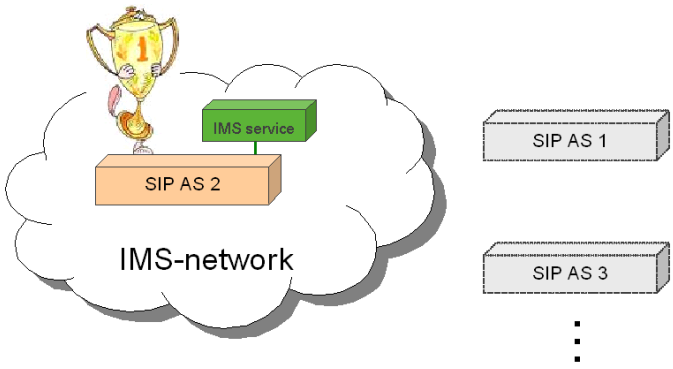

Figure 1.2. Appoint the best SIP AS

1.4 Restrictions

The IMS network that was used in this project is already available in Cybercom's office. The SIP AS should be open source so they are available for this thesis, which has no financial budget.

1.5 The company

This thesis is commissioned by the company Cybercom Nord AB. Cybercom is a high-tech consulting company that concentrates on selected technologies and offers business-critical solutions, primarily to the telecom sector.

The company was launched in 1995; it was listed on the Stockholm stock exchange in 1999. Cybercom primarily operates and grows in northern Europe.

Today, the Cybercom Group has offices in Denmark, India, Sweden, Singapore, Poland and the UK, with about 1 300 employees.

1.6 Structure of this report

In the first chapter (ch 1) of the report the purpose and the goal are presented. After that comes background information (ch 2). In chapter 3 a description of how the thesis was accomplished comes, and the results are presented in chapter 4. There is a general discussion of the thesis in chapter 5, and the last chapter (ch 6) contains conclusions.

1.7 Other

The web sites reffered to in this report were accurate 2007-11-01

1.8 Acronyms

AAA	Authentication, Authorization and Accounting
AS	Application Server
CSCF	Call/Session Control Function
CVS	Concurrent Versions System
ECS	Enterprise Communications Server
FAQ	Frequently Asked Questions
GCC	GNU C compiler
GUI	Graphical User Interface
GPL	General Public License
I-CSCF	Interrogating CSCF
JDK	Java Development Kit
JRE	Java Runtime Environment
P-CSCF	Proxy CSCF
S-CSCF	Serving CSCF
HSS	Home Subscriber Server
IFC	Initial Filter Criteria
IMS	IP Multimedia Subsystem
IP	Internet Protocol
PBX	Private Branch eXchange
PSTN	Public Switched Telephone Network
RFC	Request for Comments
RTP	Real-time Transport Protocol
SBB	Service Building Blocks
SDP	Session Description Protocol
SP	Service Profile
SPT	Service Point Trigger
SIP	Session Initiation Protocol
sipX	SIP PBX
TP	Trigger Point
UA	User Agent
URL	Uniform Resource Locator
QoS	Quality of Service
VoIP	Voice over IP

Chapter 2

Background

In this chapter an introduction of the SIP and the IMS is presented.

2.1 Session Initiation Protocol - SIP

Here some basic information about the Session Initiation Protocol (SIP) is presented. See Camarillo[1] and the SIP specification[2] for additional details.

2.1.1 Introduction

SIP is a text based protocol that establishes, modifies and terminates multimedia sessions. A session is a lasting connection between entities. SIP can be used to create a new session or to invite new members to an existing session. There are two conditions that have to be fulfilled to create a session. Firstly, the invited user must be willing to participate in the session and secondly, the participants must be able to agree to the media parameters that will be used.

SIP is independent of the type of multimedia session handled (see figure 2.1) and of the mechanism used to describe the session. The most common media transport is RTP (Real-time Transport Protocol) streams carrying video and audio and they are usually described using SDP (Session Description Protocol). When SIP has distributed session descriptions among potential participants it can be used to negotiate and modify the parameters of the session and to terminate the session.

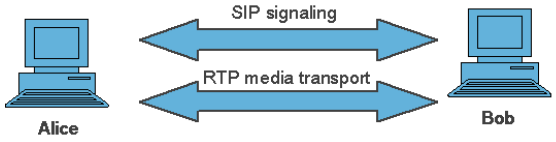

Figure 2.1. SIP is independent of the media transported

2.1.2 Architecture

When a user wants to be found he or she registers his/her current location to a registrar (see figure 2.2). To identify a user in a SIP environment a SIP URL (Uniform Resource Locator) is used. The SIP URL looks like an e-mail address, for example like this: SIP:Bob.Jonsson@company.com. When someone wants to reach Bob he or she uses Bob's public SIP address and the server handles this in one of two ways: either in a proxy mode where the server locates Bob and delivers the description to him, or in a redirect mode where the server returns the SIP address to Bob or an address to another server with more knowledge about Bob's location. To handle this, location servers are used. They store and return locations for users and can get their information from registrars or other databases.

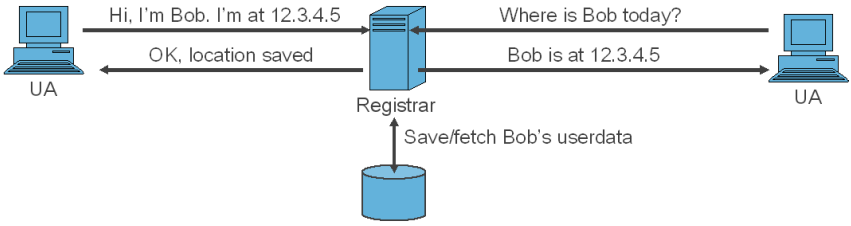

Figure 2.2. SIP registration

The SIP entity that interacts with the user, usually through an interface, is called a UA (User Agent). Some systems automatically establish sessions, and do not have a UA directly connected to the user. The UA mediates all interactions between users and the SIP protocol. They also deliver the media to a proper media tool. The SIP UA can be implemented on many different platforms, for instance in a computer or on a SIP phone.

A simplified example of session initiation and termination is shown in figure 2.3. First UA A sends an INVITE request to UA B, which receives the request and sends an OK response, whereupon UA A returns an ACK to acknowledge that there was a response to the invitation. Now the conversation can start and it lasts until one of the participants, in this case UA B, terminates the session by sending a BYE request. The termination is confirmed by an OK response from UA A.

2.2 IP Multimedia Subsystem - IMS

Here some information of the basic functionality of an IP Multimedia Subsystem (IMS) is presented. See Camarillo & Garcia-Martin[3] and Poikselkä et al.[4] for additional information.

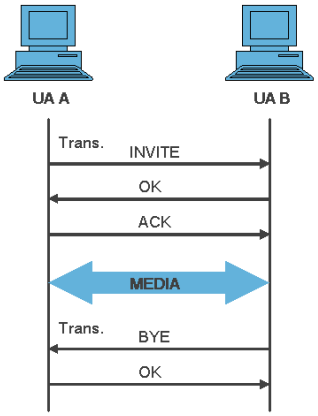

Figure 2.3. SIP transactions

2.2.1 Introduction

The IMS is an architectural framework for delivering IP (Internet Protocol) multimedia services to end users. Typical services can be watching a movie, calling a friend or participating in a video conference.

Poikselkä et al.[4] define IMS as:

> *a global, access-independent and standards-based IP connectivity and service control architecture that enables various types of multimedia services to end users using common Internet based protocols.*

The idea of IMS is to offer Internet services everywhere and at any time by using cellular technologies. Today we have the fixed phones, cellular phones, computers and IP TV delivered by PSTN (Public Switched Telephone Network), GSM (Global System for Mobile communications)/UMTS (Universal Mobile Telecommunications System) and Internet respectively. The IMS like to converge these technologies into one network (see figure 2.4).

The IMS uses 3G terminals which in turn use packet-switched technology to perform data communication. This increases the bandwidth which makes it fast and the Internet access increases dramatically compared to circuit-switched technologies. This means that a user can take advantage of all the services that service providers on the Internet offer. The benefits IMS add are QoS (Quality of Service), charging and integration of different services.

QoS means that the network offers guarantees about the amount of bandwidth a user gets and the maximum delay packets experience. This is necessary to provide good quality for the media transferred.

One reason for creating the IMS was to be able to appropriately charge for multimedia sessions. Operators currently charge based on the amount of data transferred without regard for the contents. The IMS provides information about

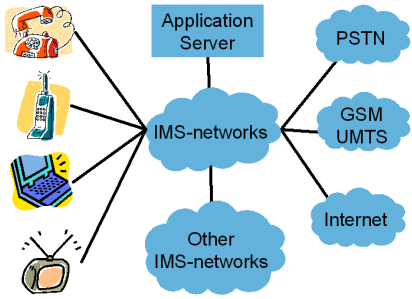

Figure 2.4. IMS network

the services being invoked and the operator can provide an appropriate charging scheme for it.

Providing integrated services to users is the third main reason for the existence of the IMS. The intention is not to standardize applications within the IMS, but rather to define standard interfaces to be used by service developers.

2.2.2 Architecture

SIP is the session control protocol for the IMS and Diameter[1] is the AAA (Authentication, Authorization and Accounting) protocol.

The IMS terminals can attach to a packet network for example through a radio link such as WLAN or through ADSL.

IMS also enables a rational architecture. The network code and communication functions do not have to be re-implemented for each application (see figure 2.5).

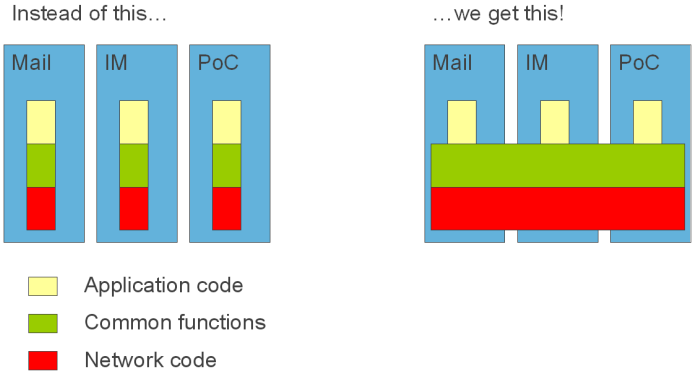

Figure 2.5. Rational architecture with IMS

[1] A computer networking protocol for Authentication, Authorization and Accounting. It is a successor to RADIUS.

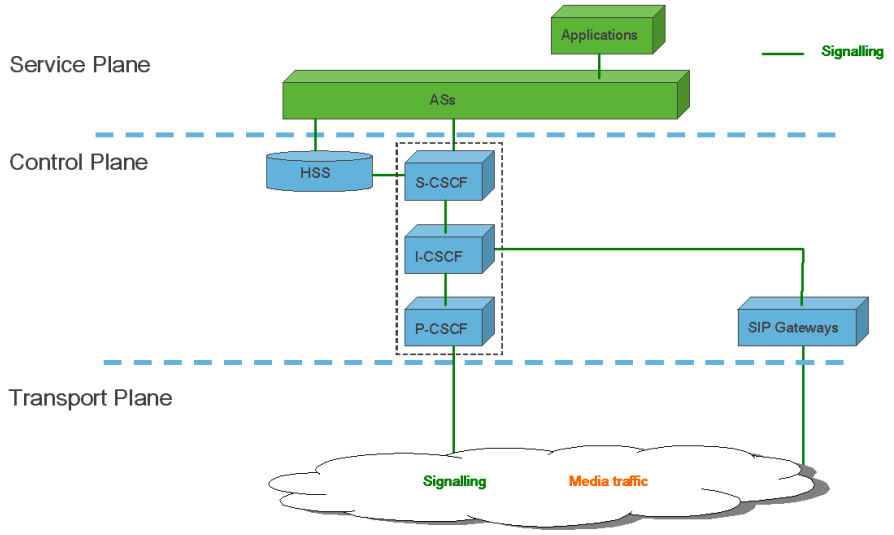

Figure 2.6. Architecture overview of IMS

The most basic functions in the IMS architecture are presented here and a overview is shown in figure 2.6.

One or more user databases called HSSs (Home Subscriber Servers) exist in the network. The HSS contains user related information, like location, security, users profile and the S-CSCF (Serving CSCF) allocated to the user.

There are one or more SIP servers collectively known as CSCFs (Call/Session Control Functions) which process SIP signaling in the IMS, and there are three kinds of CSCFs with different functionalities: proxy, interrogating and serving.

The P-CSCF (Proxy CSCF) is the first point of contact between the IMS terminal and the IMS network. It forwards SIP requests and responses in the appropriate direction. The P-CSCF also includes some functions related to security, such as user authentication and verifying the correctness of SIP messages. As SIP is a text based protocol, which makes the SIP messages large, compression and decompression is needed and it is handled by the P-CSCF.

The I-CSCF (Interrogating CSCF) is a SIP proxy server located at the edge of a domain. When a SIP server sends a message it obtains the address of an I-CSCF of the destination domain. Then the I-CSCF retrieves the user location from a HSS and routes the message to the appropriate destination, usually an S-CSCF.

The S-CSCF (Serving CSCF) is the central node of the signaling plane. The S-CSCF performs the session control and also acts as a registrar. The S-CSCF has an interface to the HSS to get authentication information, to download the user profile and to inform the HSS that it is the S-CSCF that is allocated to the user for the registration. All SIP messages the IMS terminal sends and receives goes through the allocated S-CSCF. It also keeps users from performing unauthorized operations.

There is also one or more ASs (Application Servers). They are entities that host and execute services. The ASs uses SIP to communicate with the S-CSCFs. They behave like SIP application servers toward the IMS network.

The IMS aims to be access independent; as long as the user connects via IP (Internet Protocol) various methods may be used. Native IMS terminals can access the network directly while other phone systems like PSTN and non IMS-compatible VoIP (Voice over IP) systems are supported through SIP gateways.

Chapter 3

Survey

In this chapter, the way this thesis was accomplished is presented. First a research of the SIP ASs available was done. Then the ones that were found were briefly reviewed and some of them were picked out to be investigated further. After that their documentation was explored. Then the procedure of installing the ASs and deploying a service to them were investigated. Finally, the service was deployed to an IMS network.

3.1 Survey of available SIP ASs

First a survey of which SIP ASs that were available was done. About ten different ASs were found, listed in table F. Some of them were selected to be further investigated and compared.

Only open source ASs were chosen since this master's thesis has no finacial budget.

The research was performed on the Internet only, as all open source software to a great extent are available there.

There had to be documentation about the ASs available. This was not the case with several of them. The reason documentation was wanted was because it would probably make them easier to use and to develop new services for. Documented software would also leave a more serious impression.

In addition, it was desirable for them to be fully compliant with the latest version of the SIP specification, RFC 3261[2] (Request for Comments 3261), as SIP is the session control protocol for the IMS.

The ASs also had to be implemented in Java, C or C++ because these are the programming languages known to the investigator.

AS	Website	Comment
Asterisk	http://www.asterisk.org	Was investigated briefly
OpenSER	http://openser.org	Was investigated thoroughly
MjServer	http://www.mjsip.org/mjserver.html	Not investigated since the documentation was very poor
Mobicents	http://labs.jboss.com/mobicents	Was investigated thoroughly
SER	http://www.iptel.org/ser	Not investigated since it had too much in common with OpenSER
SaRP	http://sarp.sourceforge.net	Not investigated since it was written in Perl
Yxa	http://www.stacken.kth.se/project/yxa	Not investigated since it was written in Erlang
Yate	http://yate.null.ro	Was investigated briefly
Partysip	http://www.nongnu.org/partysip	Not investigated since the main place for the documentation was the source code
sipXecs	http://www.sipfoundry.org	Was investigated briefly

Table 3.1. Table with SIP ASs

The requirements are summarized here:

- Open source
- Documentation available
- Compliant with the SIP specification
- Implemented in Java, C++ or C

After a brief review of the ASs found, the ones fulfilling the requirements above were picked out to be further investigated. Due to the limitation of time for the thesis only two were thoroughly investigated: OpenSER and Mobicents. The other servers fulfilling the requirements, Asterisk, Yate and sipXecs, were investigated briefly. A comment on why some of the servers were not chosen can be found in table F.

3.1.1 OpenSER

The first SIP AS chosen was OpenSER - the Open Source SIP Server, which can be found at the website http://openser.org.

OpenSER is a mature and flexible open source SIP server, compliant with RFC 3261[2] and released under the GNU GPL[1]. It can be used on systems with limited resources as well as on carrier grade servers, scaling to up to thousands call setups per second.

It is written in pure C for Unix/Linux-like systems with architecture specific optimizations to offer high performance. It can act as SIP proxy server, registrar server, location server, application server; gateway to SMS/XMPP; or advanced VoIP application server.

OpenSER aims to be a collaborative project for developing a secure and extensible SIP server to provide modern VoIP services.

3.1.2 Mobicents

The second AS was Mobicents, a SIP AS for the Java platform; it can be found at the website http://www.mobicents.org.

Mobicents is the first and only Open Source VoIP Platform certified for JSLEE[2] 1.0 compliance. The default distribution license is GNU GPL and it is compliant with RFC 3261[2].

Mobicents enables the composition of SBB (Service Building Blocks). The JAIN SLEE specification allows popular protocol stacks such as SIP to be plugged in as resource adapters.

Mobicents is applicable to a wide variety of problems demanding high volume and low latency signaling.

3.1.3 Asterisk

Asterisk is a telephony engine and a tool kit. It is released under the GNU GPL and is available to download for free. Asterisk supports SIP and many other different protocols. It is written in C and is primarily developed for Linux but it also runs on BSD, MAC OS and Sun Solaris. Asterisk was developed in 1999 and code has been contributed from coders around the world.

[1] GNU General Public License is a free software license

[2] SLEE (Service Logic Execution Environment) is a high throughput, low latency event processing application environment and JSLEE is the Java standard for SLEE

3.1.4 Yate

Yate - Yet Another Telephony Engine is also a telephony engine. Currently it is focused on VoIP, but it states that it can easily be extended. Yate is written in C++ and it supports scripting in various languages. It is licensed under the GNU GPL, with an exception where it is licensed under MPL[3].

3.1.5 sipXecs

sipX ECS (the SIP PBX Enterprise Communications Server) is a VoIP telephony server. The main component of the system is a SIP switch or a router around which sipX is architected. sipX includes many features of a traditional PBX[4]. sipX is licensed under the GNU LGPL[5] but it has a commercial offering as well. It is developed in C++ for Linux platforms. SIP Foundry is the organization where the development of sipX takes place.

3.2 Documentation review

After the SIP ASs to use were chosen a review of available documentation was done.

At first a review of the websites for the ASs was made. This was followed by an estimation on accessibility and value of the documentation. The quality of the documentation was measured according to the answers to the questions below.

- Was there a beginner's guide and in that case was it easy to find?
- Was the documentation clear?
- Was it easy to find a function that was searched for?

[3]Mozilla Public License is an open source and free software license
[4]Private Branch eXchange
[5]Lesser General Public License

- Was it complete?
- Was it accurate?

An investigation of available Internet forums and mailing lists was made, together with a measurement of their level of activity.

Finally, a search for books on different bookshops on the Internet was done.

OpenSER's and Mobicent's documentation were thoroughly examined, while Asterisk's, Yate's and sipX's were just briefly explored.

3.2.1 OpenSER documentation

Website

The start page of the website for OpenSER, http://openser.org, is shown in figure 3.1. It gave some information about what OpenSER can and can not be. The menu included *Features* and *Downloads* and these pages were clear. It also included *Documentation*, which lead to a documentation repository with several different links that lead to different pages, each with lots of links to documentation, but some of them lead to the same source pages, which could be a bit confusing. One page was a DokuWiki[6] with a site map to use to find things, but it did not represent the structure of the site which also was a bit confusing. In the menu there was also a link, *Install*, which lead to a page with some installation notes and a *Quick start installation guide*.

Actual documentation

There were basically two types of documentation. One was a page for documentation about OpenSER's modules and the other was a DokuWiki page. The documentation for the modules was good and easy to find. There was a link for each module that lead to a page with an overview of the module, a list of dependencies to other modules or external libraries or applications, and finally a description of the exported parameters and functions. Each description was followed by an example of usage.

The DokuWiki had a *Core Cookbook* with good information about the core keywords, values and parameters, and also examples of how to use them. To the right there was a table of contents which gave a good overview of the page. The DokuWiki also had a label, *OpenSER modules*, that could be used to find documentation about a specific parameter or function. The page linked the parameters and functions to the corresponding place in the documentation for the modules. The DokuWiki also had some practical examples and tutorials but the documentation on how to use OpenSER was poor and there was no valuable documentation on how to deploy a new module.

[6] A collaborative website which can be directly edited by anyone with access to it aimed at small companies' documentation needs

Figure 3.1. OpenSER's website

Internet forum

In the menu on the start page one could find *Forum*, which was linked to the actual forum[7]. The layout was good and it was easy to search for topics. To post a new thread you had to register, which was easily done, just picking a user name and fill in your email address. The forum was quite active, there were a few new threads every day and you could get qualified help within 24 hours.

Books

There were no specific books about OpenSER, but it was mentioned in a few books on the market.

[7] http://www.voipuser.org/forum_view_24.html

3.2.2 Mobicents documentation

Website

The start page of the site for Mobicents, http://www.mobicents.org, is shown in figure 3.2. It had several links; *Downloads, Getting Started, Examples, Forum* and more. *Downloads* lead to one of Source Forge[8]'s pages with several download links, but without information about which one to use. The *Getting Started* link lead to *A Quick Start Guide* and *Examples* presented a list of examples. There was also a link to a Wiki[9] where all sorts of information could be found, for example *Essential Mobicents Documents*, *Best Practices* and *Related Reading*. In the big picture it was hard to get a grip of the structure of the website.

Figure 3.2. Mobicents website

[8] https://sourceforge.net
[9] a collaborative website which can be directly edited by anyone with access to it

Actual documentation

The only documentation about the source code was found in the read-me files. There was much documentation on how to use Mobicents, like examples and how-to's, but they were very hard to follow and not always correct.

Internet forum

A link to the forum[10] was found on Mobicents start page. The forum was pretty active; there were often some new threads each day. It had different categories; *Contributors, Users* and *Resource Adaptors*. There was also a search function where you could search for terms in the threads. To post new threads you had to have an account and you could get one by registering a user name and add your email address. A posted thread was often answered within a day.

Books

No books about Mobicents were found.

3.2.3 Asterisk documentation

Website

The start page of Asterisk website, http://asterisk.org/, was clear. At the top of the page there were tabs; *Downloads*, *Support*, *Forum*, *Developers*, *Community* and *About*. On *Downloads* one could download different distributions of Asterisk. In the *Support* tab you could find getting started documentation, tutorials and other documentation. There was also a link to a Wiki. It had lots of information, for example introductions, tutorials and configuration instructions. Under *Developers* there were directions on what to do if you wanted to contribute, and under the last tab, *About*, there was information about what Asterisk is and what it can do.

The website had good structure and was easy to use; it was also easy to find things.

Actual documentation

At the website no documentation about the code was found, but at the Wiki page there was a great amount of documentation. But due to the lack of time the quality of the documentation has not been examined.

Internet forum

Under the *Community* tab one could find a link to the forum[11] and a link to different mailing lists. The forum had different user groups. They were arranged in larger groups, such as *General*, *Asterisk*, *AsteriskNOW* and more. *Asterisk* had the user groups *Support*, *Development*, *General* and *Documentation*. *Asterisk*

[10] http://forums.java.net/jive/category.jspa?categoryID=36
[11] http://forums.digium.com

Figure 3.3. Asterisk's website

Support was very active, with about 20 posts each day and the answers seemed competent.

Books

There are many books available about Asterisk.

3.2.4 Yate documentation

Website

The start page for Yate's website, http://yate.null.ro, had a text describing Yate. To the left there was a large menu. One of the items on the menu was *Download* were you could find different ways to download Yate, for example a tarball[12] or via CVS[13]. The menu had another interesting item, *Compability* which showed

[12] A .tar file that is usually compressed to save disk space

[13] Concurrent Versions System, a revision control system, popular for open source and commercial software development

compatible devises such as phones (mobile, IP and soft) and gateways.

The website was not very pretty, but functional. You could come to all pages via the menu. It made the menu large, but it was easy to see the structure.

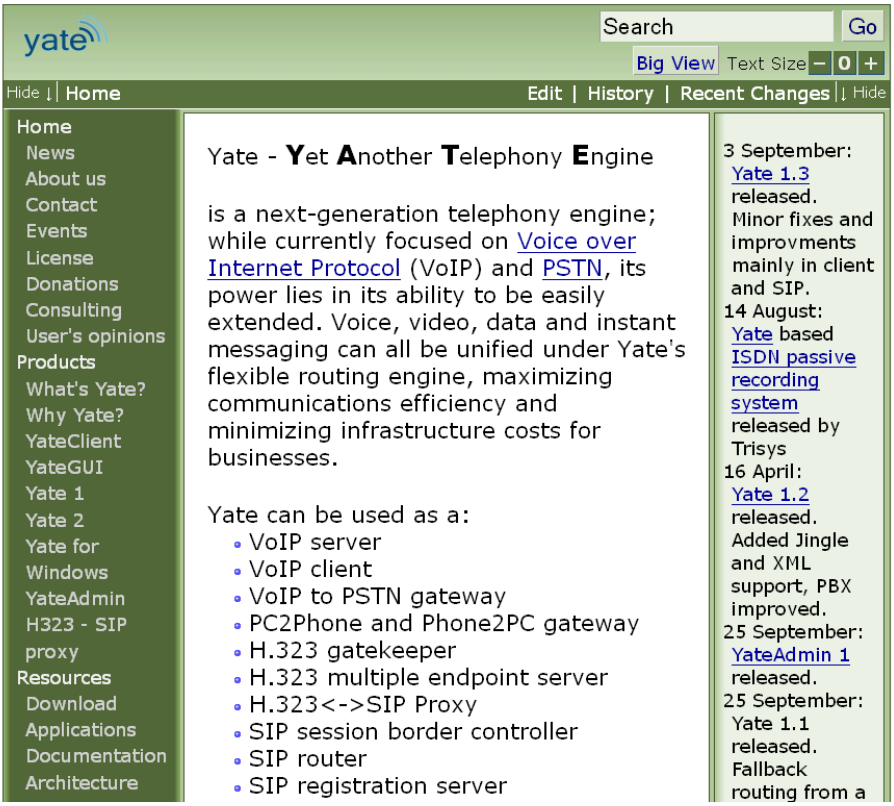

Figure 3.4. Yate's website

Actual documentation

In the menu there was an item, *Documentation*, which contained labels such as *General*, *Application Examples*, *Reference Manual* and *Programmer's guide*. Under *General* there was a FAQ and under *Reference Manual* there were links for compiling and installing, starting and the modules.

Internet forum

Yate did not have any Internet forum, but it had mailing lists, but it was unclear how to use them. To be able to send mails to these you had to subscribe.

Books

There are no books available about Yate.

3.2.5 sipXecs documentation

Website

At the start page for the SIP Foundry, http://www.sipfoundry.org/, there was a users menu to the right which had the item *Start Here*. This is a way of getting to a Wiki with information about installation, setup and troubleshooting. There was also an item, *Mailing Lists*, which led to available mailing lists. The documentation menu was below.

Figure 3.5. sipXecs's website

Actual documentation

In the documentation menu there was a link to a Wiki. The main page for it was structured in sections like: *Overview*, *Different platforms*, *How-to's*, *Configuration*,

sipX at home, *Troubleshooting* and more.

There was also a link to a sipX configuration manual, which led to the corresponding Wiki page; it had two versions, one for users and one for developers. The menu item *HowTo Library* also led to a Wiki page, which had lots of How-to's structured in different categories.

There seemed to be a lot of documentation and it was structured in a good way.

Internet forum

No forum for sipX was found, but they had about ten different mailing lists for different purposes like users, developers and announcements.

Books

No books about sipX or SIP Foundry were found.

3.3 Examination of the installation procedure

Because of the limitation in time only OpenSER's and Mobicent's installation procedures were examined in full. For the other ASs, the processes were investigated to the point where problems occurred that took too much time to solve.

The ASs were installed on a virtual machine running the Linux distribution Ubuntu[14].

Firstly, it was necessary to explore the need of other programs and also if there were restrictions on what operating system to use.

Then the installation could start. The steps in the process of installing the ASs were basically these:

1. To find some kind of getting started documentation
2. To install the needed software
3. To download the necessary files
4. To install the AS
5. To run the AS and register a SIP client to it
6. To set up a session between two SIP clients

Finally an estimation of the work to install the ASs was made.

[14] Ubuntu Feisty Fawn (version 7.04)

3.3.1 OpenSER installation

In the menu at OpenSER's website *Install* was found. At the top of the install page they encourage to reading *admin's guide* and *ISSUES*, but none of them were found.

There was also a list of supported architectures and required software, which is presented in Appendix A. If just the most basic features were wanted and the build-essential package[15] in Ubuntu was installed the only things that had to be installed would be Bison[16] and Flex[17].

After that came a *Quick-Start Installation Guide* and *How to Build OpenSER from Source Distribution*. The *Quick-Start Installation Guide* was a bit confusing with lots of abridgments so *How to Build OpenSER from Source Distribution* was chosen.

The installation guide was a bit vague. It did not really tell you what to download or where to download it from. However, it was not hard to figure out, in the menu under *Download* a link leading to the latest released sources was found so the version openser-1.2.2-notls could be downloaded. The installation guide did tell you to build the source code, but it was a bit confusing about whether you should use a certain parameter and how. Neither did it tell you where to run the command to build. But when these details were figured out you could easily just type *openser* to make it run.

Then when trying to register to OpenSER with a SIP client, an error response came back, but under the label *Troubleshooting* in the installation guide they said that it was a common problem which could be solved by adding a row in the configuration file. After doing that it worked fine and a session between two clients could be set up.

3.3.2 Mobicents installation

The start page had a link to a getting started page where *A Quick Start Guide to Mobicents* was found. There you could choose to follow either the guide *How to install and run a Mobicents binary distribution* or *How to build and run Mobicents from source code*. The former one was chosen. To download the binary distribution you were told to follow a link that lead to a page that forwarded to a download page from Source Forge. There were several download links, but no information on which one to use. After a search on the filenames, a thread at Mobicents forum where they explained that one of them was a GUI (Graphical User Interface) installer was found, but there was no information on how to get started, so the other way was chosen consequently and mobicents-all-1_0_03_GA was downloaded.

Mounting a directory and starting Mobicents with a command were the only things left to do. But that did not work. Later MS Windows carriage return tokens were found in a file and had to be removed. But it still did not work, it

[15] build-essential, version 11.3
[16] a general-purpose parser generator, can be found at http://www.gnu.org/software/bison/
[17] a tool for generating scanners, can be found at http://flex.sourceforge.net/

demanded a file that comes with JDK (Java Development Kit), so Java had to be installed. After that Mobicents could start.

To get Mobicents listening for SIP messages the SIP resource adaptor had to be deployed. Instructions on how to do that was found in the read-me file. It was simple, but an example on the Mobicents website gave misleading information.

By default Mobicents was just listening to requests from the local host. How to configure it to listen to a specific ip-address could not be solved, so a thread was posted at the forum and a little later instructions were received.

After that SIP clients could register and it was possible to start a session between them.

There was no place where the required software was listed, but in the FAQ (Frequently Asked Questions) one could find the JDK version needed. Since Java is platform independent and supported architectures never were mentioned, Mobicents is probably also platform independent. This is also noted in Appendix B.

3.3.3 Asterisk installation

Under the *Support* tab one could find the getting started documentation. Firstly one should compare and chose which distribution to use, there were three available; *Asterisk*, *AsteriskNOW* - a Linux distribution, and *Asterisk Business Edition* - a business distribution. There was a document to help with that, which listed the differences. Asterisk was chosen since it was free, but not a whole distribution.

Then it was time to download what was needed, so a compressed file, asterisk-1.4.14.tar.gz, was downloaded and extracted. The directory that showed up contained a read-me file with installation instructions, but these instructions were not the same as in the installation guide on the website, so the installations guides (there were several) on the Wiki page was reviewed and they also showed slightly different instructions. The instructions in the read-me file were chosen as they should be the ones most up to date. At the web page there was a list, shown in Appendix C, with required software, but it was a bit hard to know exactly what should be installed. After installing the required software there was just four steps and then one should be able to start Asterisk. Unfortunately there were problems when doing these steps, and installation was aborted.

3.3.4 Yate installation

Installation instructions were found under the label *Reference Manual* under *Documentation* on the menu. The directives were a bit confusing. It was not clear which architectures that were supported and the needed software just seemed to be an ANSI-C compiler. GNU C compiler (GCC) was recommended and is shown in Appendix D.

The first thing to do was to download Yate, and the latest version, Yate 1.3.0, was chosen.

The installation instructions on the website were not very straightforward. With the downloaded file there was a read-me file with more clear steps for the installation. These steps were followed and everything seemed to work. Next

moment was to run a command to start Yate, which succeeded. But when trying to register with a client, a "401-unauthorized" response was received.

3.3.5 sipXecs installation

Under the item *Start Here* in the users menu there were installation notes. First the hardware requirements were listed, and then there were two links one could follow for installation: *using sipX Single Install CD* and *Manual installation instructions for various platforms*.

The sipX Single CD Installation was created to fully automate the installation process of both the Linux operating system as well as the sipX application, but since the operating system already existed the manual installation was chosen.

If one followed the link with manual instructions there were different sections for the supported distributions. In the section for Ubuntu the instructions said that sipX application should be fully integrated into the Ubuntu package management system, by using the provided archive files sipX could be installed simply by using the package management system. Unfortunately problems occurred, maybe because the Ubuntu version in the instructions was an old one.

3.4 Deploying a new service

When the ASs had been installed it was time to examine the procedure of deploying a new service to them.

It was decided that the service in this thesis should be triggered by a client sending a SIP request of type MESSAGE, with the name of the service in the body, to the AS. The service was called *Getting today's quotation*, see figure 3.6, which should have the word *quotation* in the body of the request. The service would then send a SIP request of type MESSAGE back to the client with the actual quotation in the body.

This service was only deployed to the ASs that were thoroughly investigated, OpenSER and Mobicents.

3.4.1 Deploying a service to OpenSER

No documentation of the structure of the code for OpenSER was found, but it was figured out that it consisted of a core and a number of modules, each one contributing with its own functionality. So a new module was going to be added. There was no documentation on how to do that, except for a thread on the DokuWiki which was not complete and for an earlier release of OpenSER, but it was something to start with. Together with an examination of the existing modules and some trial and error, a new module with new functions was created. At this stage the functions just added a print to the log. There was no documentation on how to make a new module to integrate with OpenSER, so it was built with the command mentioned in the installation guide for building a separate module and after running the installation command again the module was deployed, and it was possible to load it into the configuration file.

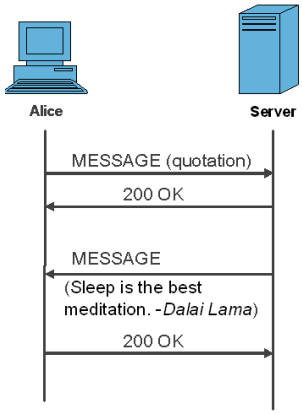

Figure 3.6. The "Getting today's quotation" service

Finally when the function was added and could be loaded it was time to add some functionality, i.e. sending a SIP request back to the client. To create and send the request required some work, but with help from the forum, a function in another module which sent at SIP request was found and gave some guidance. Eventually, with poor documentation and good help from the forum the service was successfully deployed.

3.4.2 Deploying a service to Mobicents

No documentation on how to deploy a new service to Mobicents was found, so a thread was posted at the forum where they suggested taking a look at a website with a simple example to start with. The website first described how to build, deploy and run the example and after that reviewed the design and the source code.

When following the steps to start the example there were some problems. Firstly there was an error in the instructions. Secondly, the SIP client they recommended to use could not be found, because a newer variant of it had been released that never worked. After finding another client the example worked fine.

The review of the design and source code on the website was brief, but gave an overview of the structure of the service in the example. After looking at the website's review a more thorough review of the source code was made to understand how to develop a new service. Then, with lots of guidance from the example and some help from the forum, the *Getting today's quotation* service was ready.

3.5 Deploying the service to an IMS network

After the service had been deployed to the ASs and it was confirmed to be working with a client, the service was deployed to the IMS network in Cybercom's

office. The IMS network was Open IMS Core and could be found at the website http://www.openimscore.org.

To deploy a service to the Open IMS Core you have to add ASs, create TPs (Trigger Points) and IFCs (Initial Filter Criteria), and make a SP (Service Profile).

The ASs, OpenSER and Mobicents, were added by giving them a name and an address.

A TP is a logical expression of SPTs (Service Point Triggers) which are the smallest logical atom in a TP. There are five types of SPTs:

- Request-URI
- SIP Method
- SIP Header
- Session Case (is originating, terminating or terminating to unregistered user)
- SDP Line

To each AS a TP was made, these were:

- SIP Method equal to *MESSAGE* AND Request-URI containing *@openser*
- SIP Method equal to *MESSAGE* AND Request-URI containing *@mobicents*

The IFC is the grouping between a TP and an AS; if the TP is triggered the message will be sent to the corresponding AS. Two IFCs were constructed, one for each AS.

A SP is a group of IFCs, each with an attached priority. The SP created had a default IFC with the lowest priority and the two IFCs created for the ASs.

Chapter 4

Results

In this chapter the results are presented in two tables.

In the first table, table 4.1, a comment on different aspects of the documentation, installation and deployment of a service for OpenSER and Mobicents are found. These ASs were thoroughly examined.

In the second table, table 4.2, only comments for the documentation and installation are presented for Asterisk, Yate and sipXecs. These servers have just gone through a brief examination and the installation process only persisted until a problem occurred.

More descriptive formulations can be found in chapter 3. For more information about required software and supported architectures see Appendix A-E.

In chapter 1.2 there was a list of questions aimed to help answer the question: which AS is the best suited one to implement an IMS service on? The answers to these questions can be found in table 4.1 and table 4.2. Below, a table with the number of the question and where in the result tables the answers can be found is shown.

Question number	**Where in result table to find answer**
1	Documentation -> Website
2	Documentation -> Website
3	Documentation -> Beginner's guide
4	Installation -> Problem
5	Installation -> Architectures
6	Installation -> Needed software
7	Deploying to IMS

	OpenSER	**Mobicents**
Documentation		
Website	Good structure	Lots of information, but no good structure
Actual documentation	Very good documentation about the source code, but little about how to use the server	Almost no documentation about the source code. Many examples and how-to's, but they were hard to follow.
Beginner's guide	No found	A link to *Getting Started* on the start page
Forum	Pretty active, competent replies	Pretty active, competent replies
Books	No books found	No books found
Installation		
Guide	A bit vague, but it worked	Short and clear, but some things were missed
Problems	Just a small one, which had a solution that was presented in the installation guide	Yes a few. Did not find documentation for solutions, but the forum helped.
Architectures	For Linux/Unix-like systems	Platform independent
Needed software	Bison and Flex for the most basic features	JDK 1.4
Deploying a service		
Documentation	A thread at the DokuWiki, not complete, but a start	A brief walkthrough of a simple example
Development	Required some work to get the module working and to make it integrate with the server	Pretty easy, when using the example as guidance
Deploying to IMS		
	Same procedure for all ASs	Same procedure for all ASs

Table 4.1. Table with results

	Asterisk	**Yate**	**sipXecs**
Documentation			
Website	Clear and easy to use	Functional	Ok
Actual documentation	Great amount of information	Gathered under *Documentation* in the menu	All on the Wiki page
Beginner's guide	*Getting Started* under *Support* tab	No found	*Start Here* in the Users Menu on the start page
Forum	Active forum, also had mailing lists	No forum, but mailing lists	No forum, but about 10 mailing lists
Books	Several books!	No books found	No books found
Installation			
Guide	Easy to find, website not equivalent with read-me file	A bit unclear on website, read-me file was better	Easy to find, hard to follow
Problems	Yes	Installed and started without problems!	Yes
Architectures	Linux, Mac, BSD and Solaris	a system that provides pragmatic system calls	Linux distributions
Needed software	Bison, ncurses, openssl and zlib	GCC	Nothing

Table 4.2. Table with results

Chapter 5

Discussion

Since there were so many SIP servers available and there was not time to investigate them all exhaustively, a few of them were picked out. Unfortunately only two of these could be investigated thoroughly and the others were just investigated briefly.

At the first stage, more than two were picked out but one, the MJ server was not further investigated because the documentation was so poor that the research could not continue. SER - SIP Express Router was not further investigated because it had too much in common (almost identical for the purpose of this thesis) with OpenSER, so it was unnecessary to take both of them. They are so alike because OpenSER was forked from SER by two SER core developers and one main contributor that had a different view of the management and development of the SER open source project than the others in the SER project.

The fact that I was new to Linux and the procedure of installing software in this environment had both benefits and drawbacks. The drawback was that when I ran into trouble it took a long time for me to figure out how to solve it, but a benefit was that since I did not know how to install new software I could be more objective when reading and trying to follow the installation guides.

OpenSER was written for Linux/Unix-like systems and Mobicents was platform independent. Both could be installed on the same operating system or they could be installed on different ones. It was decided that both ASs should be installed on the same operating system, a Linux distribution, because this would make it easier to compare the installation process. This question is worth a discussion and in future research the possibility of comparing installations on different operating systems should be considered.

The documentation of the ASs was different from each other. OpenSER had very good documentation on the modules, an overview, descriptions and examples of how to use the parameters and functions in them, but there were few examples on how to use OpenSER. Mobicents, on the other hand, had many examples of its usage, but few about the code and how it was constructed. Later a Javadoc[1] for Mobicents was found on an external website[2], but this information was never

[1] the industry standard for documenting Java classes

[2] www.javadoconline.com

mentioned on Mobicent's website.

OpenSER was built from source code and Mobicents was a binary distribution. Afterward, the comparison might have been more accurate if they both had been built from the source or using precompiled packages since a lot of problem can occur when building the source code.

The service that was deployed to the ASs was very simple. It was made up to investigate the procedure of deploying a service. The result may have been different if another service or a more complicated service had been chosen instead. The procedure of deploying a service could be investigated further, by deploying more services with variations in complexity.

On Mobicents's website, an Eclipse tool for developing JSLEE applications is mentioned. It would be good idea to investigate this tool if further research is to be done on how to develop new services for Mobicents. Also, WeSIP, a SIP and HTTP Converged AS built on top of OpenSER, found at http://www.wesip.com, could be worth to investigate. WeSIP adds a J2EE layer to OpenSER and can be used to develop and deploy new applications in a J2EE environment.

There were a few long lasting problems with Mobicents. The first one occurred when Mobicents were getting started. In the instructions, there were just a few steps in the start-up procedure, but the application did not work after following them. The simple example had the similar flaws. This led to that a lot of time was consumed just for solving these problems.

Creating and sending the responses with OpenSER also required lots of work, just figuring out how to get the right address to send the response to took time.

Much of the estimations done are abstract, but it has been hard to measure the ASs against a concrete scale, so it has been more of a comparison between them, than an independent measurement.

The sections in the background chapter, SIP and IMS, are just a brief, simplified overview of the topics, just enough to make the reader understand this thesis.

Chapter 6

Conclusions

I could not point out one of the ASs to be more suited than another for deployment of an IMS service. This was because they had different pros and cons and none of them was distinguishing in a way that made them superior to the other.

There are many SIP servers available and initially it is hard to know which one is best suited for the task at hand. After working with them for a while you start to see the benefits and drawbacks of them.

Clear and good documentation is very valuable when using something that is new to you. The clearity, is especially important, because if you can not find what you want it does not matter if it is good.

It is important with an easy and straightforward website, it leaves a more serious impression and it makes it easier to use the software.

It is very good if there is an active forum or mailing list, because if you run into trouble they can be very helpful. You can also find solutions by reading about and search for problems that others have had.

After these conclusions I think that Asterisk would be interesting to investigate more properly, as it seems to have good prerequisites.

OpenSER was easier to get started with, but theoretically, Mobicents should have been easier because the installation process had fewer steps. Unfortunately, it did not work after following these steps. It is more problematic to do few steps that do not work, than to do several, more complicated steps that do work.

It would have been valuable to have some documentation on how to add a service. Mobicents lacked this, but it was solved by imitating an example. Doing this on OpenSER would have been hard without the documentation, even though it was incomplete.

It is also very common to run into problems when dealing with software that is under construction.

The steps to deploy a service to an IMS network is not dependent on the AS used in the case with Open IMS Core, because the IMS only forwards the message to the AS, and this is the same for all ASs.

Bibliography

[1] Gonzalo Camarillo. *SIP demystified*. McGraw-Hill Professional, 2002. ISBN 0-07-137340-3.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Request for comments: 3261. Describes Session Initiation Protocol (SIP), 2002.

[3] Gonzalo Camarillo and Miguel A. Garcia-Martin. *The 3G IP Multimedia Subsystem*. John Wiley and Sons, 2004. ISBN 0-470-87156-3.

[4] Miika Poikselkä, Georg Mayer, Hisham Khartabil, and Aki Niemi. *The IMS IP Multimedia Concepts and Services*. John Wiley and Sons, 2 edition, 2006. ISBN 0-470-01906-9.

Appendix A

OpenSER

A.1 Supported architectures

The supported architectures for OpenSER are listed below:

- Linux/i386
- Linux/armv4l
- FreeBSD/i386
- OpenBSD/i386
- Solaris/sparc64
- NetBSD/sparc64

A.2 Required software

The software required for OpenSER are listed below:

- gcc or icc : gcc >= 2.9x; 3.[12] recommended (it will work with older version but it might require some options tweaking for best performance)
- bison or yacc (Berkley yacc)
- flex
- GNU make (on Linux this is the standard "make", on FreeBSD and Solaris is called "gmake") version >= 3.79. - sed and tr (used in the makefiles)
- GNU tar ("gtar" on Solaris) and gzip if you want "make tar" to work
- GNU install or BSD install (on Solaris "ginstall") if you want "make install", "make bin", "make sunpkg" to work

- openssl if you want to compile the TLS support
- libmysqlclient and libz (zlib) -libs and devel headers- if you want mysql DB support (the mysql module)
- libpq / postgresql -libs and devel headers- if you want postgres DB support (the postgres module)
- unixodbc -libs and devel headers- if you want unixodbc DB support (the unixodbc module)
- libexpat if you want the jabber gateway support (the jabber module)
- libxml2 if you want to use the cpl-c (Call Processing Language) and PA (Presence Agent) modules
- libradius-ng -libs and devel headers- if you want to use functionalities with radius support - authentication, accounting, group support, etc

Appendix B

Mobicents

B.1 Supported architectures

Mobicents is platform independent, since it is written in Java.

B.2 Required software

If Mobicents just is going to be run, a JRE (Java Runtime Environment) is the only thing needed. If you also shall build Mobicents you also need a JDK (Java Development Kit). For this release JDK 1.4 or higher is needed.

Appendix C

Asterisk

C.1 Supported architectures

The supported architectures for Asterisk are listed below:

- Linux
- Mac OS X
- FreeBSD
- OpenBSD
- Sun Solaris

C.2 Required software

The software required for Asterisk are listed below:

- ncurses, and associated -devel
- openssl, and associated -devel
- zlib, and associated -devel
- bison, and associated -devel (1.0.X only)

Appendix D

Yate

D.1 Supported architectures

You must have a system that provides pragmatic dlopen()/dlsym() system calls. This is to provide dynamic loading.

D.2 Required software

The only thing that is required for Yate is an ANSI-C compiler, GNU C compiler (GCC) is recommended

Appendix E

sipXecs

E.1 Supported architectures

The supported architectures for sipXecs are listed below:

- Fedora Core
- CentOS
- Debian
- SuSE
- Gentoo
- Solaris
- Mac OS X
- Windows

E.2 Required software

No other software is required.