IMS platform prototype

Leonard Broman

Luleå University of Technology
MSc Programmes in Engineering
Computer Science and Engineering
Department of Computer Science and Electrical Engineering
Division of Computer Engineering
Abstract

The future in telecommunication systems goes towards a split between the network provider and the service provider, and a convergence of media distribution and communication networks. This has caused the 3GPP to develop an open specification for the future communication system, IP Multimedia Subsystem, providing real-time multimedia services. The system is designed as a modular and flexible communication platform running on top of any transport medium providing IP transport capabilities. It is also designed to be to interface towards other networks such as the PSTN.

This report covers research and analysis of deploying a demonstration prototype of the IMS platform. Using present open source solution and developing missing components to enable services and solve network-topological problems such as NATs.
2.2 Single client services ........................................... 16
  2.2.1 Client configuration data ................................. 16
  2.2.2 Presence service ........................................... 17
2.3 Text services ...................................................... 17
  2.3.1 Pager-mode instant messaging ............................. 17
  2.3.2 Session-mode one-to-one instant messaging .......... 18
  2.3.3 Session-mode instant message conferencing .......... 18
  2.3.4 Single-client text services ............................... 19
2.4 Audio and video services ..................................... 19
  2.4.1 Single client media service ............................... 19
  2.4.2 Multimedia conversation session ......................... 19
  2.4.3 Multimedia conference ................................... 20
2.5 System topology ................................................. 20
  2.5.1 Call/Session Control Function ......................... 20
  2.5.2 Home Subscriber Server ................................. 22
  2.5.3 Media Resource Function ................................. 22
  2.5.4 Application Server ...................................... 22
  2.5.5 Media Gateway Function ................................. 23
3 Technical aspects and solutions ............................... 25
  3.1 Network restrictions ......................................... 25
  3.2 Network Address Translation ............................... 26
    3.2.1 NAT and UDP ........................................... 26
    3.2.2 NAT and TCP ........................................... 27
  3.3 General requirements in restricted networks ............ 27
  3.4 Generic solution to IMS on restricted networks ......... 28
    3.4.1 Session border controller system topology ........... 28
3.4.2 SBC as NAT/Proxy ........................................ 28
3.4.3 SBC for QoS and network resource monitoring ....... 29

3.5 Volatile networks ............................................. 29
3.5.1 Signaling connection instability ......................... 30
3.5.2 Media connection instability ............................. 30

3.6 Conference sessions in SIP ................................... 31
3.6.1 Creating the focus/session ............................... 31
3.6.2 Subscribing to session changes ........................... 32
3.6.3 Inviting new participants ................................. 32

3.7 MSRP conferencing ........................................... 32
3.7.1 Connecting to the session ............................... 32
3.7.2 Authentication mechanisms .............................. 33

3.8 Audio and Video conferencing ............................... 34
3.8.1 RTP media and conferences ............................. 35

3.9 Conclusions .................................................. 35

4 System design .................................................. 36
4.1 Topological solution, DMZ to Private network border controllers ........................................ 36
4.1.1 Multi-host systems ........................................ 36
4.1.2 TCP initiation from private network ................. 38
4.1.3 TCP tunneling ............................................. 38

4.2 Topological solution, everything in DMZ ................... 38
4.2.1 Conclusion ................................................. 38

4.3 Network deployment solution ............................... 39
4.3.1 Software components ................................. 39

5 Implementation ................................................. 42
5.1 Client software ......................................................... 43
  5.1.1 Target platform ................................................. 43
  5.1.2 MSRP Stack ...................................................... 44
  5.1.3 IMS Stack ....................................................... 47
  5.1.4 Java client ...................................................... 50

5.2 Server software ..................................................... 51
  5.2.1 SIP focus for MSRP sessions .................................. 51
  5.2.2 Billing proxy ................................................... 52

5.3 Client based test bots .............................................. 53
  5.3.1 MSRP echo bot .................................................. 53
  5.3.2 Eliza bot ....................................................... 53
  5.3.3 Inviter bot ..................................................... 53
  5.3.4 Conference bot ................................................ 53
  5.3.5 Conference inviter bot ...................................... 53

6 Summary and future work ............................................ 54
  6.1 Summary .......................................................... 54
  6.1.1 Present status ................................................ 54
  6.1.2 Technologies used ............................................ 54
  6.2 Future integration ................................................ 55
  6.3 Future development .............................................. 56
    6.3.1 MSRP Stack .................................................. 57
    6.3.2 IMS Stack ................................................... 57
    6.3.3 Java Client ................................................ 58

7 Results ................................................................. 60
  7.1 Discussion ........................................................ 60
7.2 Conclusions ........................................... 61

Bibliography ............................................. 62

A Links .................................................. 65

B Establishing SIP focus session .................. 66
  B.1 Alice creates the session ......................... 66
  B.2 Alice creates a state subscription to the session .......... 70
  B.3 Alice invites Bob ................................ 71
  B.4 Bob invites himself and joins .................. 71

C Class diagrams ..................................... 73
  C.1 MSRP stack API ................................ 73
  C.2 MSRP stack implementation ..................... 73
  C.3 IMS stack API ................................ 73
  C.4 IMS stack implementation ..................... 73
  C.5 Java client ..................................... 73

D Activity diagrams ................................ 87

E State diagrams .................................... 91
  E.1 MSRP stack API ................................ 91
  E.2 IMS stack API ................................ 91

F Sequence diagrams ............................... 94
List of Figures

2.1 IMS functions overview ........................................... 20
2.2 Different modes of operation for the SIP-AS .......................... 24
3.1 SBC on private/public network border ................................. 29
3.2 The SIP-AS function with MRFC and MRFI .......................... 33
3.3 RTP mixer in a 3-part conference .................................... 34
4.1 Network with SBCs .................................................. 37
4.2 Network with multi-host component ................................. 37
4.3 Deployment example ................................................ 39
5.1 Model-view-controller architecture behind the java client ........ 50
6.1 Present deployed network ............................................ 55
6.2 Future extension examples ........................................... 56
6.3 Client screen shot .................................................. 59
C.1 MSRP base class diagram ........................................... 74
C.2 MSRP header class diagram ........................................ 75
C.3 MSRP packet class diagram ........................................ 75
C.4 com.atosorigin.msrp package class diagram ..................... 76
C.5 com.atosorigin.msrp.data package class diagram ............... 77
Acknowledgments

I would like to thank Régis de la Broise for supervising my project at Atos Origin, helping me with guidance, discussions and proofreading.

My internship supervisor at the University of Rennes, Professor Cesar Viho who made my internship and thesis work possible in France by accepting me as a student at his institution.

My thesis supervisor at Luleå University of Technology, Kåre Synnes.

My family and the family of my beloved girlfriend for supporting me during my time in France.

Celine Schmidt for receiving me with open arms and taking care of me during my first trembling steps on French soil.

My collective friends and their friends for accepting me and helping me during my time of integration into the French society.
Chapter 1

Introduction

1.1 IP Multimedia Subsystem

The IMS architecture provides a multi-service, multi-access and multi-session IP network which offers enablers to real-time, pseudo real-time and reliable-transmission application servers. The architecture also allows IT actors to take a more important place than before in the development of the communication services.

1.2 Assignment background

A large part of the capital expenditures (CAPEX) of the telecom operators in the IMS architecture takes place in the application services and in the integration of supplies from multiple providers. The related activities are Atos Origin’s strengths.

In order to be prepared for this new evolution of the communication networks, Atos chose in 2006 to develop strong SIP-IMS expertise by creating a pre-IMS platform within its French labs. This platform is used to develop and demonstrate several Value Added Services based on SIP-IMS capability. Some Added Value partners and Open Source projects are integrated in some prototypes realized.

The goal is to have a better knowledge of the corresponding technology, a better answer to customer expectations and a better understanding of operators business to help them anticipate the next generation services. Currently an Atos Origin communication prototype using a soft-phone (standalone VoIP capable application) and an instant messaging application running in a browser exists. This prototype is based on the following technologies: SIP (using BEA WLSS), Java, Ajax, JSP, Struts, Spring, Hibernate, PostgreSQL. Within this prototype, some IMS functions have been simulated.
1.3 Problem

The subject of the thesis is to study, identify and test new innovating services taking advantage of the new capabilities of the IMS architecture. The way to test these new services will need to be defined and an IMS platform allowing running these new services will have to be built.

1.3.1 Prototype

This platform could be resulting from evolutions of the existing prototype making it more IMS compliant by replacing some simulated IMS functions by components that will make the prototype interact with a core via the standardized interfaces. These IMS components could consist of Open Source software components acting as IMS components, such as the CSCF (Call/Session control function) and the HSS (Home Subscriber Server).

Additional basic services could be added such as the integration of a presence server and/or an XCAP server provided by an editor (an Atos partner) or coming from an Open Source project.

1.3.2 Requirements

In order to facilitate demonstrations on the Internet, NAT traversal solutions is also to be studied. The different ways to achieve this feature should be studied and the selected solution will have to be integrated with the prototype.

For the prototype platform to be useful in demonstration purposes an important requirement is that demonstrations of applications can be performed at locations which are not completely bound to geographical locations. At least the aim is to have a goal where two clients on visited network may communicate. In this case, between a local private network and the public routable network.

This means that one private network should host the IMS home network, where user profiles are stored and the IMS mechanisms come in action. The only SIP interfaces outside the home network is via the L-CSCF and media via the MRFP.

Network restrictions

The two networks involved in this case are the local private network and a remote private network. They both reside behind NAT and has a DMZ (DeMilitarized Zone). However, a host in the DMZ may not connect to a host in the private network. In this case even the DMZ is behind a NAT, although the address space behind the NAT is not in the private range and potentially routable, they have however an external representation.
1.3.3 Summary

To summarize, the objectives of the thesis is to

- Analyze the present demonstration platform to determine its re-usability and extensibility.
- Integrate a prototype platform providing the CSCF and HSS IMS functions.
- Enable IMS instant messaging services on the IMS platform.
- Propose a solution reducing or eliminating connection issues concerning networks behind NAT.
- Design and implement necessary missing software components to enable instant messaging services.
- Add, or suggest possibilities to add, additional services to the IMS platform. Most notably presence services and group management.

1.4 Emphasis

The subject of IMS is very vast and in term of research, design and development, it covers an area which is beyond the scope of this thesis. Therefore the emphasis of this work will be limited to certain aspects in IMS, summarized as

- The study of the core IMS system and services
- The study and deployment of instant messaging services in IMS
- Theoretical study and solutions of network topology issues for IMS deployment on unmanaged networks.

1.5 Definitions

The base of this work is mainly specifications and proposals from organisations like IEEE, IETF, 3GPP and ETSI. They are not only very dubbed in using acronyms and abbreviations aggressively (perhaps due to necessity) but they also tend to differ in vocabulary between the organizations. I therefore see a great need of pre pending this report with a list of definitions of words, expressions, acronyms and abbreviations. The ETSI also has its own vocabulary list [18].
1.5.1 General abbreviations

ETSI European Telecommunications Standards Institute
3GPP 3:rd Generation Partnership Project
TS 3GPP Technical Specification
IETF Internet Engineering Task Force
RFC Request For Comments
IMS IP Multimedia Subsystem
QoS Quality of Service

1.5.2 IMS abbreviations

CSCF Call/Session Control Function, SIP routing network routing component.
P-CSCF Proxy CSCF, IMS network entry point
I-CSCF Interrogating CSCF, inter-network routing component
S-CSCF Serving CSCF, service aware router and registrar
HSS Home Subscriber Server, subscriber information database
SLF Server Locator Function, registered user to S-CSCF mapping
MRFC Media Resource Function Controller, signaling layer media server
MRFP Media Resource Function Processor, media layer media server
MGCF Media Gateway Control Function, inter-network media gateway
MGW Media Gateway
UAC User Agent Client, originating user agent
UAS User Agent Server, terminating user agent
UE User Equipment, the end terminal where a client interacts with the IMS
UMTS Universal Mobile Telecommunication System (3G)
GGSN Gateway GPRS Support Node
OSA-CSC Open Service Access – Service Capability Server
IM-SSF IP Multimedia Service Switching Function
TP Trigger Point, logic expression identifying SIP packets
iFC Initial Filter Criteria, pairs binding trigger points to application servers
1.5.3 Protocols

SDP  Session Description Protocol [21]
SIP  Session Initiation Protocol [10]
MSRP  Message Session Relay Protocol [6]
Diameter  Authentication, authorization and accounting protocol [14] [21]
RTP  Real-time Transport Protocol [8]
UDP  User Datagram Protocol [30]
PIDF  Presence Information Data Format [9]
RPID  Rich Presence Information Data format for presence [38]
XCAP  XML Configuration Access Protocol [36]
AKA  Authentication and Key Agreement [4]
Chapter 2

IP Multimedia Subsystem

IMS is the response by 3GPP (3:rd Generation Partnership Project) to the clear communication evolution where data transport and communication services are going to separate. IMS provides a flexible and modular way of controlling and providing services to the clients. The IMS network lives on top of a general data transport network, either IPv4 or IPv6. The network is defined as functional nodes, gateways and interconnects and the actual specification does not require functions to be implemented in particular fashions, only which protocols they follow.

The IMS is specified within numerous documents by ETSI (European Telecommunications Standards Institute) and IETF (Internet Engineering Task Force). This because IMS provides a way to use existing open technologies and ties them all together, to answer needs of service providers and for the flexibility of User Equipment.

IMS is usually said to consist of a signaling plane and a media plane. They both reside on the general network and they are not depending on each other. They are however both necessary to provide IMS services (with the special exception of services using only the signaling plane).

This chapter roughly presents the general functionality and technologies involved.

2.1 IMS signaling

Two main protocols are used for the signaling part of IMS. SIP for the interconnect of UE (User Equipment) and management of communication. Diameter is used for AAA (Authentication, Authorization and Accounting) in the network to support the network functions with user information and billing capabilities.
2.1.1 SIP - Session Initiation Protocol

SIP has become very popular for VoIP (Voice over IP) services since it is an open protocol which has reached broad recognition and many free and open source software solutions exists (see Appendix A). SIP provides the functionality of setting up, managing and tearing down sessions between clients in the network and is generally transport protocol agnostic. The main functionalities in SIP are specified in RFC 3261 [10].

Within IMS, SIP has extensions to provide signaling for the QoS functionality. For the UE the SIP part strictly follows the specification and theoretically the mechanisms of IMS is transparent for the terminal application. But within the IMS network, the SIP signaling is managed to adapt the client communication to the underlying network, to control subscriptions and billing functions.

2.1.2 Diameter

For the service and/or network operator to control the usage of services and the connectivity of users the network uses the functionality of Diameter. Diameter is a AAA protocol defined in RFC 3588 [14]. It is a successor to RADIUS (RFC 2865 [31]).

It is used in a number of forms, named interfaces, between the different functions of the IMS network. This to provide the specific information at the points where they are needed. This report covers Diameter very lightly since it focuses on service enabling, signaling and media transports.

2.1.3 Quality of Service

QoS is provided by interfacing the IMS components towards components in the different network architectures which provides QoS functionality. This is mainly done by P-CSCF which has the knowledge of the local network and a policy description function attached to regulate the use of QoS. One example of this is a P-CSCF providing IMS connection in a UMTS network. The P-CSCF tracks establishment of session to terminals in the network and controls the UMTS GGSN to open and close streaming channels to the clients.

To provide easier QoS setup in networks SIP session negotiations in IMS uses PRACK (PRovisional ACKnoledgment) to separate session acceptance and codec agreement from network specific negotiation.
CHAPTER 2. IP MULTIMEDIA SUBSYSTEM

2.1.4 Security

For clients registering its location to the “registrar” (the S-CSCF in IMS) a challenge authentication is of course required. This is done with the HTTP digest function to avoid sending the shared secret over the public network [19]. Specification for sending clear-text passwords exists but is strongly discouraged and most SIP software suites does not allow it.

Network authentication is done using the AKA [4] protocol. This way the client can authenticate the connection to the home network when in a visited network. This is done using a shared secret and a sequence number.

To encrypt inter IMS traffic TLS can be used. TLS is depending on certificates which makes it more powerful between stationary units such as proxies. TLS can however still be used as a SIP encryption method for clients, although it does not easily permit the use of certificates to authenticate clients.

For an even higher degree of encryption, provided by network borders (perhaps regulated by an SBC, see Section 3.4) provides a more general encryption of all UE traffic between networks. This is transparent but requires a more complex configuration and maintenance between each interconnected network.

2.2 Single client services

The different services in the IMS network can roughly be categorized as single- and multi-client services. In the case of single client services we have all those that at the point of service access only involve one user. These are functions like centralized configuration information, presence monitoring and call forwarding.

2.2.1 Client configuration data

The client will most probably need and want to store data on the network instead of its (many) devices. This can for example be contact and group lists, chat and call history. Maybe even personalized themes and media.

This together with information which controls network behavior towards other clients, such as call forwarding, availability and preferred terminals for certain types communication medium is all stored on an XDM (Xml Document Management) server and configured from the client via the XCAP (XML Configuration Access Protocol) protocol, defined in RFC 4835 [36].
2.2.2 Presence service

Presence information is being regarded as vital information in today’s communication systems. The presence handling in IMS is defined in TS 23 141 [15]. It is mainly based in Application Servers handling PUBLISH (RFC 3903 [27]) , SUBSCRIBE and NOTIFY (RFC 3265 [32]) requests from user agents, however an important feature is the ability to subscribe to a resource list in an RLS (Resource List Server). The resource list is manipulated by the client via XCAP. The XCAP server may then provide the resource list via a URI to the presence server.

Presence servers normally operate using internal presence documents [9]. The presence document contains the presence information for a certain presentity, the UA (User Agent) publishing presence information. The document is updated by the presentity using the PUBLISH SIP method [27], containing the presence document for the presentity. When the presence server receives this update it sends out notifications, using the NOTIFY [32] method, to all UAs which have previously subscribed to the presentity using the SUBSCRIBE SIP method [32]. The NOTIFY may contain either the entire presence document or an update (diff) from the previous version.

In IMS many different type of notification services is provided, not only presence. For each such service an “event package” is defined, which specifies which information is provided. For user presence, using RPID or PIDF presence documents, the “presence” package is used (RFC 3865 [33]). The S-CSCF holds a special package, “reg” (RFC 3680 [34]), to which P-CSCF subscribes to be able to refuse client packages if the client is not registered. The registration state may also be used by registered clients to monitor their own registration, in the same manner as portable devices today monitor signal strength and roaming network. For focuses, the package “conference” (RFC 4575 [12]) is provided as mentioned in Section 3.6.2.

Application servers holding pager-mode messages for offline clients can also chose to subscribe to notifications when that users comes on line. In this case the Message Application Server acts as a User Agent in the presence system.

2.3 Text services

Instant message service mechanism in IMS is covered in TS 124 247 [16]. This contains the methods and actions taken by the functions in the IMS system to handle pager-mode, session-mode and session-mode conference instant messaging.

2.3.1 Pager-mode instant messaging

This is like good old way of sending messages like SMS. You send one and (maybe) get a delivery report when the destination has received it. This is generally called pager-mode messaging and is deemed to be session-agnostic unless some client developer intends to build a client handled and dependent session based messaging.
This is generally a bad idea.

This is done using the SIP MESSAGE request. The S-CSCF should deliver the MESSAGE request to an AS for handling. The AS determines the current state of the recipient. It should determine whether the message can be delivered at the moment and to which terminals which the user is registered to it should be delivered. The receiving terminal(s) must have registered support for MESSAGE in the HSS. If the message cannot be delivered at the moment, action will be taken later when there is a terminal capable of receiving it. Failure should be reported when the message expires, this is set by headers by the sending client (and might be moderated by the AS). Delivery report of page messages is done using the default SIP response codes, i.e. 200 OK for successful delivery. The SIP MESSAGE method is an initial SIP request and thus subject to SIP proxy forking. If the AS chooses to fork the message to multiple receiving terminals, it will still only report one SIP response to the sending UA. (RFC 3428 [5]).

2.3.2 Session-mode one-to-one instant messaging

This is, with today’s aspects, the general way of doing instant message textual communication. Using a chat session with exactly two participants. This is done in IMS by setting up a SIP session between the terminals and using a direct media connection or connecting directly via UE chosen by the terminals.

In IMS the media layer for session based instant messaging is MSRP [6] running on TCP.

2.3.3 Session-mode instant message conferencing

A conference session is a session based textual communication where the number of participants are two or more. One-to-one can because of this be considered a special case of conferencing and considered to be the general future form to use. However, for interoperability reasons we must assume that other terminals may not completely support the one form or the other.

Using a MRFC (Media Resource Function Controller) and a MRFP (Media Resource Function Processor) as relays and session managers to establish connections. In this case, the MRFC is an AS which handles and manipulates invites to direct UE to the MSRP MGCF instead of to the end clients.

Camarillo suggest in chapter 19.2 [3] that it is for each UE to perform SIP INVITE to the MRFC and for the MGCF in turn to perform the MSRP connection to each UE. Poikselkä [28] extends this idea briefly in chapter 8 by introducing the concept of tightly coupled conferences using a focus session concept. Requirements of the focus is defined in RFC 4245 [23].

Section 8 and 9 in TS 124 247 [16] covers the specification of SDP session creation and MSRP session progress, respectively. This specification must be followed for
any IMS terminal supporting the feature.

2.3.4 Single-client text services

Text services such as searching for train departures or having asynchronous information sent to your device can be implemented using either session based connections to information “bots” or as pager-mode messages for requests and responses.

2.4 Audio and video services

Audio and video conversation is the classic way of human interaction. Audio has for a long time formed the very foundation of human interaction over distances since the telephone was invented. We will without any doubt continue with this. To add to this the communication industry has for a long time been trying to introduce video services. This has never reached any broad use, although most clients are not opposed to using it.

2.4.1 Single client media service

In IMS there is nothing limiting the client from receiving audio and video media which does not originate from a conversation. Music, pictures and movies can be delivered from the IMS system, just as an ordinary media stream. For this service the SIP-AS will act as a UA see Section 2.5.4. The system will provide a MRFC and a MRFP, where the MRFC is the actual AS. The client establishes a session to the UA, which then would have a SIP-URI, like perhaps sip:radio@ims.net, and through the media connection have access to media content by the service provider.

2.4.2 Multimedia conversation session

Today, the most common way of telecommunication is in one-to-one conversations. This is most likely due to traditional services supplied by the telecom service providers. This is also one of the simplest services to provide in the IMS network. In this case the SIP-AS works as a proxy, see Section 2.5.4 transparent to the UE which setup their media connection as a usual SIP initiated media stream session. The AS main function is to provide billing functions to the sessions. The AS can also provide, by SDP manipulation, transparent RTP proxy incorporation solving issues with UE on NAT networks. Using a B2BUA (Back-to-Back User Agent) AS is also a solution where separate session are maintained between the two participating UA.
2.4.3 Multimedia conference

The conference perspective of media services is a much more complex situation than for textual conferences. The AS operates as a B2BUA (see Section 2.5.4) providing a focus session in the signalling layer and an RTP mixer in the media plane. Real-time media will not be covered in more detail as it is out of scope for this thesis.

2.5 System topology

Figure 2.1 shows a general overview of some of the functional nodes in the IMS network. It is important to understand that there is a big difference between a network function and an actual piece of software or machine running and performing it. One server, in the aspect of machine or software, can perform many functions or one function can be spread over multiple machines with any non-defined interconnect in between them. As long as the function rest atomic in the network view, this is very possible. Later on, the network functions will be put onto pieces of software performing them, which somewhat will change the view of the system. Here, the tasks of certain functions are presented.

![Diagram of IMS network functions](image)

Figure 2.1: Overview of the basic functions in the IMS network

2.5.1 Call/Session Control Function

The CSCFs provide the main signaling functions for the UE. From the UE perspective it is just a set of SIP proxies and registrars and provides for the UE the
possibility to locate and set up session to other UE. The CSCF are split into three functions, P, I and S. For proxy, interrogation and serving, respectively.

Proxy - CSCF

For each IMS network, the proxy is the entry-point for the UE. Its function is obviously just a proxy for the UE but within the IMS network it carries many more functions. For a visited network, its service provider has the chance to keep control over its network resources and traffic thanks to the mandatory traversal of the proxy. One might think that in a free IP network the proxy in visited networks is useless, which is right, but the service provider has the possibility to enforce it due to this network topology.

To provide the control over network resources, we might assume that some client is connected to IMS via a proxy in a visited network (or maybe even the home network). This network may be behind a NAT or heavily filtered to enforce P-CSCF traversal. When setting up sessions, the P-CSCF provides two major mechanisms. It lets the network service provider have control over network resources used by clients of the service providers. It also gives the ability to proper control of the SBC (Session Border Controller), see Section 3.4 component and other network specific gateways during session negotiation. The SBC serves a purpose of media proxy where it lets packages traverse NAT obstacles, it opens channels in otherwise filtered firewalls and foremost it provides the mechanisms of QoS on the media layer. Other components can be support nodes which enables IP traffic with QoS in cellular networks.

Interrogating - CSCF

The I-CSCF provides the entry point to a home IMS network. For any P-CSCF residing in visited networks, they will forward request to the UE home network via the I-CSCF. The I-CSCF is located via DNS and has to be on the public routable network. The I-CSCF has information about S-CSCF in the home network and also knows how to contact UE residing in visited networks. It is also capable of authenticating incoming requests, and deny them if the client is not welcome.

Up to 3GPP release 6 the I-CSCF also provided topology hiding and the main gateway control functions at the entry point of the Home IMS system. In release 7 this function has been separated from the I-CSCF and has been assigned to a particular function namely the I-BCF (Interconnection Border Control Function), which provides similar functionality as the SBC, see Section 3.4.

Serving - CSCF

In the home network the S-CSCF is the main manager of SIP-signaling between UAs. The S-CSCF provides all the filtering need for propagating SIP-messages
their respective application servers and it also holds information about clients in visited network. All SIP traffic related to a certain client passes the S-CSCF in the home network.

For all initial SIP requests the S-CSCF applies a set of routing rules provided by the HSS for each subscriber. This rules-set is called a subscriber profile and contains an order list of Initial Filter Criteria (iFC). Each iFC contains one trigger point and one application server. The trigger point contains a set of logical expressions which is used to analyze the request, looking at request method, URI, headers and content. If the logical expression matches the request it is routed via the application server attached. After the request has been processed by the application server the S-CSCF continues processing the IFC list.

### 2.5.2 Home Subscriber Server

The HSS is basically a database storing user-related data. If a network has multiple HSS a SLF (Subscriber Locator Function) is required to locate the necessary data and preferred S-CSCF.

### 2.5.3 Media Resource Function

The MRF provides media resource services in the network. Examples of such services may be streaming media and conference sessions. Common for those services are that the UE sets up a SIP session to an AS and accesses the media resource on the media plane.

The MRF is split into two parts, MRFC (MRF Controller) and MRFP (MRF Processor). The MRFC resides in the SIP signaling plane and has also access to the HSS. The MRFC can be implemented directly as a SIP application server. The second part, the MRFP resides in the media plane and serves the client/UE with the media content. The interconnect between MRFC and MRFP is undefined and they may be implemented as one piece of software fulfilling the functionality of SIP-AS, MRFC and MRFP. There exists however a IETF draft (now expired) proposing a protocol MSCP (Media Server Control Protocol) to provide a standard for the MRFC-MRFP interface.

### 2.5.4 Application Server

There are three types of application servers; SIP-AS, OSA-SCS, IM-SSF. Only the SIP-AS will be covered here since it is the most flexible and least complex. The Application server provides for the service mechanisms in the IMS network. Each SIP-message traversing the S-CSCF is subject to a filter-set which may reroute the messages to one or many AS serving different services. The SIP-AS may operate in three different roles; user agent, proxy and Back-to-Back user agent. Figure 2.2 shows a simple view of the three roles.
User Agent mode

For services where the a session is established between the UE and the AS the AS acts as an UA towards the UE and is also viewed as one from the UE point of view. This is useful for services involving only one client. For example a film-streaming service might be addressed with sip:film@ims.net to which the UE establishes a SIP session to access media streams (on the media layer) on an MRFP.

Proxy mode

For services which take form of one-to-one sessions between end-clients, the AS is often an interceptor and manager of the SIP-sessions. Upon session setup, the AS will handle the Authentication, Authority and Accounting for the clients involved. The AS of course has the possibility to deny services, and to interrupt them. If an AS decides to interact as proxy in the SIP Dialog it must add itself to the Record-Route header of the initial request. This way clients are aware of the traversal but not of its effects.

An example application for proxy AS is a billing proxy. It keeps track of conversations and reports, via the Diameter interface, call starts, destination, ends and other interesting information (geographical information, identity usage, etc). For the online charging case it may refuse a call and end it prematurely when the caller has no more credit.

Back To Back user agent mode (B2BUA)

In services involving two or more UE this model is used to manage for example conferences. In this case each UE has one established SIP-session towards the AS which acts as a UA to each. The AS then has abilities such as inviting more UE, disconnecting UE and charging for service usage.

An important ability for a B2BUA is also to be able to redirect calls in a client transparent way. This is useful for call-centers where a calling party may first perform certain choices via DTMF, then be in a waiting line with some fancy music and finally arrive at an operator. For the UE this is only one SIP session but the B2BUA may jump between many on the other side.

2.5.5 Media Gateway Function

For interfacing IMS with other communication networks there is a need for gateways. In the case if IMS the media gateway is split into three parts; media gateway function controller (MGFC), media gateway (MGW), signaling gateway (SGW). Those three components provide the signaling and media interworking between the networks.
The perhaps most common usage of this is the interconnect between IMS and PSTN allowing IMS client to make calls to the old telephone system, and vice versa. As a possible example for instant messaging emphasis such a gateway may provide presence, contacts and chat-rooms between IMS and other IM protocols such as XMPP/Jabber and Oscar/ICQ.
Chapter 3

Technical aspects and solutions

This chapter covers the technical solutions and considerations which have to be taken into account; on the basis of today’s structure of the Internet and entities tying it together, serving and restricting it.

3.1 Network restrictions

IMS was in its early stages supposed to run on IPv6 networks, where UE addressing does not pose a problem as in IPv4, where address space has forced the Internet systems into radical measures. The delayed adoption of IPv6 has forced the IMS work groups to adapt IMS to work with IPv4 systems as well. The main issue is network topologies containing NATs. IMS does not present any solutions on how such networks restrictions should be handled, it has however been considered and is mentioned very briefly in Annex G of TS 23.228 [17].

It has to be mentioned that at a grand deployment stage, most IMS networks will run on transport networks restricted to such a degree that it can be considered to not be apart of the general Internet but built exclusively for IMS. This is mainly due to the willingness to be able to control services on the network and to keep control over complex mechanisms such as QoS and priority. It must also be assumed that IMS network providers does have control over the transport network they are running on.
3.2 Network Address Translation

Currently most mobile networks providing Internet connections do so by using NATs (network address translation) [39]. Those provide extended address spaces within private networks who are not routable from the public Internet.

Generally, the types of NAT that exists impose a certain level of restriction. The cone NAT maintains a mapping based on previous traffic from a source on the private network to an address and port on the public network. The mapping is initiated when a packet passes the NAT to the public network and is dropped after a certain timeout. Any packet arriving from the public network on that port will be translated and sent to the host on the private network, thus allowing traversal of the NAT if the host in the private network knows its public IP, there are multiple technologies to solve this, using STUN, ICE, TURN or information in SIP messages.

The “worst” kind of NAT is the symmetric NAT. This type of NAT saves additional information in the mapping and filters out packages which do not have the same source address and port (public) as the destination of the originating packet. This restricts the host on the private network to only being able to communicate to hosts it can establish a connection to. Thus, the private host can only connect to a routable address on the public address space and two private hosts can never reach each other.

3.2.1 NAT and UDP

There exists multiple protocols for providing traversal over NAT. Mainly, this is an important feature for audio and video media using RTP [8] when providing such services over Internet. RTP over IP is a user-space extension to UDP. It adds framing mechanisms where sequence and timestamp information is incorporated (this does not exist in UDP). The protocol permits for many media streams to be transmitted over one RTP stream and to send data stream information and feedback in a separate stream (RTCP, RTP control protocol).

STUN is defined in RFC 3489 [11] and provides functions such as determining a client connection to the public IP address space. The protocol helps determining properties of the connection, such as filtering and presence of NATs and their properties. It also provides information to the client of its public address. This permits a client behind a NAT to offer its routable address during SDP negotiating and may thus permit UDP hole punching attempts.

TURN is still an “internet-draft” currently documented in [13]. It provides means of allowing any client to connect to any host, even on a private network, independent of protocol or NAT type. The main principle is to provide a dynamic way of setting up a relay on a public routable address. This is of course costly for the provider of the TURN server and may cause longer propagation delays.

ICE, just as TURN is also still in drafting, see [37]. ICE is a protocol defining a process to use STUN and TURN to provide NAT traversal for offer/answer session
initiation. The method is fairly complex and there is nothing in IMS that specifies supporting it (as with STUN and TURN) and the IMS network has to be ICE-aware not to start billing users for calls not made etc.

Many proprietary solutions use hole-punching to ease up NAT traversal. The technique is very simple. Both endpoints perform multiple re-sends and accepts initial failures to establish the association in the NAT. This method is however completely client specific.

RTP proxy

Providers of SIP based VoIP services use a fairly simple solution to avoid NAT issues. This solution is provided by two open source community products, RTP Proxy and Media Proxy (see Appendix A). Their function is very simple. They integrate with an outbound SIP proxy which performs SDP manipulation to route UDP traffic via the proxies. This is completely transparent to the client and they always send their media to and from the public routable proxy.

Extreme case: asymmetric client

There is yet one even worse case where not even the proxy based systems helps; asymmetric clients behind NATs. An asymmetric client is a client which sends and receives UDP data on different ports, for a certain session. This is possible since UDP does not create any connection state but just sends datagrams and lets the client handle the datastream. Due to this the outgoing connection does not create any associations related to the incoming stream, independent of the NAT type. Thus the incoming packet stream cannot be mapped back to the proper UA. This case is out of scope of this thesis and will not be covered any further.

3.2.2 NAT and TCP

Since TCP is connection oriented (as opposed to UDP which uses datagrams) it can not easily trick NATs to let it traverse. Hole punching is something that is sometimes used but rarely works. Any type of state-full TCP information in the NAT router causes TCP hole punching to fail. TURN is of course a general solution to this, but as mentioned in the previous section, it is bandwidth and time costly. It also adds yet another layer that clients and network has to be aware of and support.

3.3 General requirements in restricted networks

Restricted networks containing IMS entry-points have to provide for their visiting clients to utilize IMS services. For general network access it is simply required that the UE is on a network that “supports” IMS. Either a restricted network with an
P-CSCF and SBC, covered in Section 3.4, or on the public Internet with knowledge of at least one public P-CSCF.

As a general point of view this is of course a network requirement but it might not suffice in our case where we have limited control over the network due to policies.

### 3.4 Generic solution to IMS on restricted networks

As an IMS network provider normally has control over the underlying transport layer network, it can, and should, deploy network mechanisms which lets a client interface transparently over the IMS network without the need to use methods like STUN, TURN and ICE. This general system is called the Session Border Controller, SBC.

#### 3.4.1 Session border controller system topology

The main ability of the SBC is that it has information about both the private and the public network. This provides the ability for the SBC to act as what could be called a layer 5 NAT. This is covered more in detail in Section 3.4.2. The SBC might also have functionality providing QoS and network usage monitoring and restrictions. Figure 3.1 shows a typical visited network. It has a P-CSCF, which is reachable and the only IMS point that the UE sees. The UE finds the P-CSCF via DHCP information or DNS resolve. In the figure both a GGSN and a general SBC is shown. The P-CSCF provides information to the SBC and GGSN by relaying SIP messages via them. The normal SIP communication traverses the NAT, to corresponding I-CSCF, as usual.

Since release 7 a function of the I-CSCF have had one function separated from it which is now called IBCF (Interconnect Border Control Function). It has similar functionality as the SBC but mainly provides inter-IMS-network encryption and network resource policy enforcement.

#### 3.4.2 SBC as NAT/Proxy

To provide a transparent network for the UE in the IMS the SBC monitors and manipulates the SDP information for session establishment. In this way it will transparently place itself as a media proxy between the two participating parties. When a UE on the private network sends an INVITE the SIP messages traverses the SBC. The SBC stores the given data reception information for the UE and replaces it with an assigned public address, of the SBC itself. When the OK from the opposing party arrives it may also, if it is necessary, make the same mapping for outgoing data. Finally, when the ACK passes the SBC it opens the necessary ports for the media.
Generally, the SBC may be incorporated into any CSCF and thus reduce SIP message processing and provide a dynamic flow control for security purposes.

### 3.4.3 SBC for QoS and network resource monitoring

Since QoS may be critical for real-time applications on highly loaded networks, the GGSN provides this functionality in a similar fashion as an SBC. Although the GGSN provides a more particular functionality. This component interfaces towards the local networks QoS enabling facilities and provides a means for the network provider to monitor and regulate network usage.

The major extension to SIP to provide QoS support and resource reservations are the extended form of session establishment using provisional responses. This splits the session establishment into two phases. First the initial \texttt{INVITE} is sent. The response to this invite determines whether the peer accepts the session and which common codec’s to use. The response is a \texttt{PRACK} (PRovisional ACKnowledge-ment). In the next phase the originating UE sends an \texttt{UPDATE} which is, if necessary, manipulated by SBC and Gateways to provide correct IP and port mappings for network resources. This determines the media path.

### 3.5 Volatile networks

Future IMS clients will live in IP networks connected to the Internet using cellular networks (beginning with 3G). Those networks are subject of being extremely unstable and unreliable. At any point a connections might be lost (i.e. passing through a tunnel) or the host IP and network route may change (i.e. network roaming access).
3.5.1 Signaling connection instability

Today SIP applications are used on stable networks connections. The IETF RFCs often suggest default timeout constants in their standardization documents. Those numbers are often in the order of hours. This is of course not applicable to IP in cellular networks, where the connection is subject of changes as soon as the client is changing location. For those networks the timeout should be in the ranges less than a minute. The client is then required to refresh their registration and their published presence information constantly.

3.5.2 Media connection instability

The SIP \texttt{UPDATE} method provides the signaling means to change media connectivity as the IMS devices traverses networks. It requires high awareness of possible networks changes in the IMS device to be able to predict and prevent connection loss.

Real-time media

RTP transmission carrying audio and video must function seamless in roaming situations. This is a known problem since the very first packet switched cellular networks. Since IMS operates in the application layer of the IP stack it increases the complexity of media reliability in roaming situation. This increases the demands both for the transport network as well as the IMS media and signaling networks.

In many cases the real-time media does not suffer from the same restraints as reliable media. When passing through short tunnels or losing the stream for shorter moments it does not impact the user more than today’s GSM systems does.

Reliable media

Reliable media, transmitted over TCP streams can not easily be reestablished and continued upon changes in network and any type of connection loss. Using direct TCP transmissions in highly mobile IMS clients is most likely an issue to be avoided.

MSRP may be a solution to this. There is no proposed standard for this from the IETF but, one might ponder, that letting relays offer buffers for packets being retransmitted and delivered later when connection is dropped.

These issues may form the subject for a Master Thesis themselves and are somewhat out-of-scope here. The matter will not be covered any further.
CHAPTER 3. TECHNICAL ASPECTS AND SOLUTIONS

3.6 Conference sessions in SIP

Extensions to SIP to support conference sessions are in the works, but they are still on the working table and it might take a long time before we start seeing standards and implementations supporting it. The IETF MMUSIC working group is looking into this. Therefore, were not going to get involved into this at the moment.

At present, many proprietary solutions has been developed for providing conference sessions within SIP, such as Genesys ACD and Streamwide SVI (see Appendix A). This is done using the focus session concept where one client, be it a participator or a server acts as a focus, letting many connect to it and informing the others. The focus session is just what SIP-AS B2BUA is all about, see Section 2.5.4.

SIP focus conferencing has been so widespread that IETF issued RFC 4245 to define requirements of this model. This to regulate future implementations and permit higher degree of interoperability. Later IETF specified the conferencing in SIP in detail in RFC 4353. However none of these RFC’s are standards track and does not actually specify in detail how focus mechanisms should work in practice.

Although very shallow, Both Camarillo (chapter 19.2) and Poikselkä (chapter 7) touches this concept in IMS. Camarillo only discusses the SIP-AS as an entity to where clients invites themselves but Poikselkä touches the concept of a focus and suggests an IMS compliant SIP aspect. He does however not go into depth covering session information exchange or authorization.

3.6.1 Creating the focus/session

The actual traffic sent is covered in detail in Appendix B. The session is initiated by a UE which wishes to participate in a conference with one or more other participants. The UE generates a conference URI via a conference factory. The conference factory may have a sip uri like sip:imschat@ims.net. This process is done by sending an invite to this address. At this point, the S-CSCF must determine which SIP-AS should handle this request, depending on the URI and the media suggested in the SDP offer.

The conference factory in this case is the application server. It will respond to the INVITE with an OK with a Contact header which is the URI of the session. The URI should contain the parameter isfocus to indicate that it is a focus. At this point the application server also takes measures needed to set up media resources, such as for MSRP, see Section 3.7.

In Section 5.2.1 a very critical limitation in SipServlet is presented which makes the use of Contact header for focus URI information impossible.
3.6.2 Subscribing to session changes

To receive notifications about state changes (joins/leaves etc.) in the session each participant must send a \texttt{SUBSCRIBE}. The subscribe should be to the conference URI and the event package “conference”. The focus notifies subscribed clients when the conference state changes according to RFC 4575 \cite{rfc4575}. The notifications provide information about the conference itself, such as textual information, connection URIs and media. It also provides information about the current participants, such as names, contact information and connection status.

3.6.3 Inviting new participants

Inviting other clients to the session can be done in two ways. Either by sending a \texttt{REFER} to the focus, which in turn will invite the URI suggested. Or by sending a \texttt{REFER} to the client, with enough information for the client to invite himself to the session.

The \texttt{REFER} method is very complex and contains mechanisms to inform the originator of the session progress of the referee. This is not very useful since the focus provides this information using presence information. An alternative would be to exchange the focus information in a \texttt{MESSAGE} request or using SIP \texttt{INVITE} to the focus sip-uri.

3.7 MSRP conferencing

Where the SIP focus provides conference management on the signaling plane, there is a corresponding communication in the media plane. This can be provided in a series of ways using a distributed relays or central relays. Only the central solution is covered here since it does not pose any problems whatsoever for hosts behind NATs.

The MSRP session is handled in an MSRP session server. In the IMS system it falls under the function of a MRFP (Media Resource Function Processor). To provide security mechanisms and not just free-for-all access it may be tightly coupled with its signaling counterpart MRFC (MRF Controller). If implemented as one piece together they provide the functions SIP-AS, MRFC and MRFP.

Using the SIP session state information the MRFP can create and destroy sessions and control who may \texttt{SEND} and \texttt{VISIT} MSRP sessions.

3.7.1 Connecting to the session

From the client side the process of connecting to the MSRP is fairly simple. The UE is responsible for establishing a TCP connection to the MSRP, defined in section
8.5 in RFC 4975, since it is the sending the initial offer. This means simply that the one sending the initial INVITE, that is, the originating client is the one that should provide the MSRP path between the endpoints. The MMUSIC group did propose methods for negotiating connection direction for TCP session but it was discarded due to complexity.

When the TCP connection is established, the UE sends either a SEND or a VISIT. A VISIT is like a SEND but without content. The VISIT request is not specified in the MSRP standards, however it is commonly used in discussions and Poikselkä also mentions it as a method for client to confirm there presence in the media. The message contains a To-Path header containing the MSRP URI sent by the SDP answer from the SIP focus. The From-Path header contains a the MSRP URI sent in the SDP offer, by the UE. This way the two parts can identify the TCP connection as bearer of the messages to that those recipients, using only a temporary credential.

The base MSRP as defined does not contain any extensions for session state monitoring. The only requests are SEND and AUTH. SEND is replied to with an OK, or an error response. The MSRP session server only manages the session and relays all incoming SENDs to participators.

### 3.7.2 Authentication mechanisms

As a MSRP session server can be either central or distributed in the network it does not serve well for authentication purposes. If a client uses an MSRP server...
CHAPTER 3. TECHNICAL ASPECTS AND SOLUTIONS

not on the home network, the session server does not have access to the shared secret required for authentication.

Instead, MSRP relays can provide authentication using the AUTH method defined in RFC 4976 [7]. This would require all clients to establish connection and authenticate towards the relay in the home network. This would be a potential solution to one-to-one MSRP around NATs but since we already have a potential solution no more elaboration is put into this case.

3.8 Audio and Video conferencing

Just as the case with MSRP there are a multitude of solutions for providing the media plane communication in a multimedia conference. Using many of the decentralized solutions such as meshes, multiple point-to-point or decentralized bridging poses great requirements on the underlying network structure.

Therefore a centralized bridge is preferred, since it helps avoiding the NAT problem. The centralized bridge is often called an RTP mixer and like the MSRP session server it keeps track of members in a focus based conference. It provides both direction of the RTP data flow and transmits all incoming data to the other participating parties. Mixers are covered in the definition documents of RTP, see RFC 3550 [8].

![RTP mixer in a 3-part conference](image)

Figure 3.3: RTP mixer in a 3-part conference
3.8.1 RTP media and conferences

One issue with RTP and NAT traversal is the requirement of sending and receiving towards one host on the same port to achieve traversal of symmetric NATs. This works fine for any one-to-one stream where one side is public (i.e., proxy or mixer). RTP media is however streamed on separate UDP port pairs (one port media, one port control) for different streams, and they cannot easily be combined. Participating in a conference where media is separated which needs NAT traversal on ports where the client is not sending is not possible.

One solution would be that the client does a NAT hole punching by sending some bogus data on the ports it wants to open. The proxy might need to be aware of this and it is client specific. It results in an unorthodox (in the IMS sense) way of solving this problem.

Another solution would be that the RTP mixer joins the all streams too one image, one stream. This would simplify for clients having possibility of handling only one image, such as a small device. Combining this by sending dimension information in the SDP offer to the focus would enable the mixer to adapt the image size or video encoding for each client and enable the NAT traversal as usual. This however poses great complexity and processing power requirements on the mixer. Developing an RTP mixer is not in the scope of this assignment but a deployment of present software is still possible if suitable software is to be found.

3.9 Conclusions

For a prototype system, a general solution to the NAT problem would be to simply not use the peer-to-peer method for any conversation. Instead always treat one-to-one communication as a conference with only two participants. In this case it would simply utilize the MSRP session server for all instant messaging services (except pager-mode) and require a RTP mixer only supporting two participants, thus act as RTP Proxy but comply better with the IMS since it avoids the insertion of a hidden RTP Proxy in the network. Of course; such a system would not support a client attempting to perform a regular peer-to-peer connection.

Developing a limited RTP mixer for this purpose would enable a future upgrade of the mixer without actually changing the IMS network structure. Such an upgrade could contain many features such as image resizing, bandwidth adaption, image merging (many sources in one stream/image), audio mixing, etc.
Chapter 4

System design

In this chapter, the functional model of IMS (see Section 2.1) will be converted into a practical and technical model based on networks, machines, and server software. Later on, specific software will be deployed to the technical model. The practical model considers requested capabilities of the platform as well as known network restrictions. The requirements and restrictions are covered in Section 1.3.2.

4.1 Topological solution, DMZ to Private network border controllers

Figure 4.1 depicts a possible practical solution that avoids putting the entire IMS home system in the DMZ, instead some of the functions must act as border controllers. The main issue here is the security level in the border controllers. By enterprise security policies a non-validated software can never be put to provide access to the private network of the enterprise. Most open source software (such as the I-CSCF) and software developed in the time-frame of this thesis fall under this category. Hence this is not a good solution. This is however how a true IMS system for full deployment purposes would have to be designed.

4.1.1 Multi-host systems

Putting certain functions on system having two IP-addresses lets the software act as a border controller. One address is public and routable, residing in the DMZ, perhaps also behind a NAT in which case the software has to be aware of its IP or its DNS name. The private address allows the software to connect to servers residing in the private, otherwise, unreachable domain. One must of course take care that the machine does not route unhandled packages and such to the private network. If the border machine is compromised, the private network is as well.
Figure 4.1: Network topology for the network using SBCs as entry-points

Figure 4.2 explains this, where the I-CSCF, MRF and a dedicated SBC acts as border controllers.

Figure 4.2: Network topology for the network with software components residing in multiple networks
4.1.2 TCP initiation from private network

In this case the host residing on the private network must establish and maintain TCP connections to servers in the DMZ. This is fairly feasible concerning the IMS system, but for the UE it is much less practical. They must employ some form of discovery to use proxies/SBCs available. It is not flexible and does not fall under the specifications of IMS and is therefore discouraged.

4.1.3 TCP tunneling

By setting up SSH (or similar) tunnels from hosts on the private network to hosts in the DMZ connections are allowed to be initiated from the DMZ to the private network. This features the same security risks as with multi-host systems. Generally the only gain of using this method is to bypass network security policies and is therefore strongly discouraged. Concerning UE side, it poses of course the same problems as TCP initiation and is therefore not applicable to the general case.

Tunneling RTP traffic over TCP also have the effect the RT (Real-Time) property of the protocol is completely lost. The RTP stream will suffer under congestion effects (RFC 2581 [1]) in TCP which will increase propagation delays. Also, the phenomena called “TCP slow start” (RFC 3390 [2]) causes inconvenient problems when establishing sessions. Sharing TCP connection for multiple UDP protocols will have side-effects like SIP traffic being queued behind RTP streams on congested TCP connections.

4.2 Topological solution, everything in DMZ

Placing the entire IMS network in the public routable address space solves all the central problems. It is however not recommended for security reasons. Using an MSRP relay to provide a MSRP entry-point in the public address space and an RTP Proxy (see Section 3.2.1) for RTP streams gives a system which will work for UE on private NAT systems.

4.2.1 Conclusion

If we want to follow the specification of IMS and allow any generic IMS UE to connect to the network, we are forced to use SBC solutions in NAT-ed networks. This is also the general solution in IMS. However, from network restrictions and policies for this particular case it is not possible. Both UDP and TCP proxies must be integrated into the network to completely bypass the problem.
4.3 Network deployment solution

For the deployment, the system will have to be built on a machine connected on the DMZ. This is not completely recommended but internal functions will be a little more protected in the sense that only the P-CSCF, I-CSCF, MRFP and RTP Proxy will be connectable from the outside.

To build this simple system; the only thing needed is a machine hosting the system, connected to the DMZ which must be configured to not to filter the protocols we are going to use. In this case; UDP port 5060 for SIP and TCP port 2855 for MSRP.

![Deployment Diagram](image)

Figure 4.3: A deployment example for the IMS system, still in functional notation

4.3.1 Software components

The main IMS system will consist of a number of different software components inter-operating to provide services and management to the network. It should as far as possible be built on preexisting software from the free software community.

The IMS core

The FOKUS work group at TU Berlin has developed an IMS testing playground. For the CSCF and HSS they have used and extended SER. Since SER is free and open software FOKUS are redistributing their software for free use and modification. They do however not distribute other pieces of their IMS system which they have produced.
Presence and contact service

The OpenSER community provides a solution for SIP/SIMPLE presence services. They use OpenSER in conjunction with OpenXCAP to provide this, using OpenSER for monitoring services and OpenXCAP as a database back-end. This same XCAP server can later be reused for other services depending on XCAP capabilities, such as contact list management. In the IMS system OpenSER will act as a SIP-AS.

This solution is actually not IMS compliant. The OpenXCAP server does not support Diameter for authentication. Also the documents are strictly bound to a sip-URI which results in issues when one private identity (which may be in any format) has multiple public identities.

The OpenSER community provides a SIP server core and a multitude of extension modules. It can be used as SIP proxy, registrar, presence. It can be configured to provide user agent services, supporting SDP manipulation, different database back-ends and direct XCAP integration. The OpenXCAP + OpenSER suite is a very interesting combination where the two servers shared database tables, this is also the variant promoted by the OpenSER community. Together they provide resource-list manipulation and subscription, group management and presence-rules for privacy policies.

OpenXCAP is an HTTP server providing GET and PUT methods to manipulate XML documents stored in a MySQL database, where it may share its table with an OpenSER presence module. It was originally developed as a graduation thesis which is obvious when you start working with it. The server is written in Python and is more or less just a combination of Python frameworks; twisted web2, gnutls and MySQL. It is developed under Debian and does not work properly on other systems. In my case, CentOS, the server crashes due to run-time linking errors during start up. This is fixed by removing the TLS support in the source code. The server does not refresh the MySQL connection and stops working when the connection times out due to inactivity. The development of the server is tracking the very latest of its dependencies which makes maintenance difficult. Unfortunately due to time-frame reasons the OpenXCAP + OpenSER solution is kept, but one must be aware of the daily need to restart the XCAP server. A future option would be to replace the OpenXCAP with the java.net project OpenXDM, which also provides a framework for client development.

Audio and video conversations, with NAT solution

OpenSER in conjunction with MediaProxy solves the issue with clients behind NATs in a transparent way. OpenSER provides the mediaproxy module which transparently modifies the SDP in a SIP offer/answer to direct the media-stream via a RTP proxy. This is completely transparent to the UE and solves the NAT issue for UDP media in one-to-one conversations. However, it is somewhat of an
unorthodox solution for an IMS system. Section 3.9 suggest a possible alternative to this.

The MediaProxy does however only support UDP streams is thus not a solution for MSRP one-to-one conversations. If using the media proxy from OpenSER this SIP server must be the one that the UE is connecting to (the P-CSCF) to be able to detect the originating address.

**Textual messaging**

For each type of textual (see Section 2.3) messaging services we need one application server.

Pager-mode messaging is supported by an OpenSER server configured to provide long-time delays and delivery reports. The AS intercept and treats MESSAGE’s sent.

One-to-one session messaging is supported by a SIP-AS working in proxy mode. This can most probably be solved using an OpenSER server but since the functionality in it is simple, it could be a good opportunity to develop the functionality in a SIP-supporting framework, like BEA. It should be noted that it might be necessary to develop a TCP TURN proxy, similar to media-proxy but for MSRP connections.

Textual conferencing is supported by a SIP-AS acting in B2BUA mode. If OpenSER requires to much modification it could also be deployed on the BEA server. This does however have much higher complexity than the one-to-one messaging SIP-AS.

The MSRP session manager (MRFP) need to be developed entirely. But the development of the MRFP should directly provide the MSRP stack which will also be used when developing the client.

Textual messaging history is a function controlled by the client, if it supports it, and is implemented as a service towards the XCAP server.
Chapter 5

Implementation
5.1 Client software

The client should provide a flexible way of demonstrating the system. It is developed in Java which provides a multitude of possibilities concerning platforms and environments.

This chapter covers the choice of target platform. It covers the design considerations for the implementation of the communication stacks and specifies which existing frameworks and libraries are used within the implementation.

5.1.1 Target platform

There already exists a demonstration client for the system simulating IMS services. This client is a web-based solution to provide the flexibility of running the application independent of platform or location. The system consists of an AJAX interface containing:

- Contact list, providing presence information via SIP/SIMPLE
- Chat window, enabling SIP/SIMPLE based textual messaging
- Shared white-board for Internet collaboration
- Java application (soft-phone), running as an applet, to provide audio and video communication. This is the only way for the application to interface with the system microphone and camera using Java media framework.

AJAX Client

At this point, an embedded java applet will not be considered as an AJAX component, since it is just a program running in a browser.

The most particular point in using an AJAX client is the fact that all communication between the client and IMS will not traverse the Internet, if the client is hosted on the same machine as the IMS system. This is a most likely case here. Since the communication runs via servlets on the website host it is independent of the location of the user.

A web-client is limited in the sense that it cannot communicate with a web-cam or a microphone of the host system, and therefore can only support textual services.

Web embedded applet

As the case with the soft-phone a single fully featured client can be embedded as an applet on a website. This does however put certain restraints on the client. It must
all be implemented as one “window”. This will not solve any problems concerning
firewalls or NATs. To be able to connect to network services the applet has to be
signed and the user must accept the certificate.

Standalone Java application

The main advantage of a standalone client, compared to an applet, is the flexibility
of the UI design. The interface can be split up into multiple windows. The main
drawback of this is that the user must download and run the application locally.
This does however not require any signature and may be considered less secure.

Java ME application for mobile devices

The last and most complex aspect of the client software is the possible requirement
that is should be able to run on a mobile device. Mobile devices, like smart-phones
and most newer cell-phones, mostly support only Java Micro Edition. Java ME
only contains a subset of the Java Standard Edition API. This puts restraints on
components shared among software components, like the MSRP stack (see Section
5.1.2).

The Java ME client most likely have to developed separately since it has a com-
pletely different screen and input aspects to be considered.

Issues with Java ME  There are a number of serious issues which have to be
considered before the decision to develop Java ME versions of the shared software
components, which are mainly the network protocol stacks and the client core.

The Java ME API has much fewer classes. For example there is only one List
implementation, the Vector, and it does not inherit List but Object. This poses
limitations on the choice of data structure to use. This is an important aspect
in designing software that is also going to run on servers, possibly handling great
datasets.

The set of methods for classes are also reduced. For example the String.split()
method which provides powerful parsing capabilities is missing.

The language support is also limited. Generics, annotations and enumerations are
absent. This does not impact on the programming capacity of java but does make
it more error prone and will cause an increase in time spent on bug searching.

5.1.2  MSRP Stack

The MSRP is a component in the system which is going to be used both by the
server and client software. A stack is not available in an open-source form or as
a Java API at the moment, therefore it will have to be designed and developed. Important is that the stack may function both for clients and servers (clients may also serve in peer-to-peer) and that it must also respect the limited API of Java ME if the choice to develop such a client is taken.

Statefullness hierarchy

Three levels of statefullness is provided by the stack. Channel, connection and transmission. Those levels are directly related to how the MSRP specifications suggest that MSRP data transfers are performed.

The Channel  The TCP socket is wrapped into a Channel. The channel symbolized the peer-to-peer relation between two IP hosts. The channel contains a statefull, non-locking parser and enforces rules upon the TCP connection to provide fault-condition handling. If the stack is running as Channel-statefull the channel accepts MSRP packets directly and which are sent to the underlying socket. Incoming packets are only parsed and delivered the channel listener.

The Connection  Each channel may (and should preferably) carry many MSRP connections for which the next hop is the same host. The Connection interfaces models this mechanism and provides a means of sending packets over the next, without knowledge directly of the next hop. The channel maps the destination URI to a connection and packets are delivered via the connection.

The Transmission  Transmissions are pieces of data which should be transmitted and presented at the peer as an entity. Small (¡ 2048Kb) transmissions are performed using only one MSRP packet. Larger transmissions are chunked into multiple transactions, MSRP packets. The transmission contains the mechanisms to provide transmission progress reporting. This is also the main model for the stack listener to receive data from connections.

The Transaction  This interface exists only to provide a complete model of the MSRP mechanisms to the implementer. The implementer may, using it, inspect transaction information (mainly packets transmitted and progress). No practical interests exists for not having this layer stateless if the transmission is statefull.

Threaded model

The stack is implemented using two threads. One working thread which performs all operations such as queue flushing and data buffer checking, in a round-robin fashion with limitations for each step. It is the provider of execution time for
parsing incoming messaging and encoding messages sent. The second thread serves only as the waiting thread for listening to and handling incoming TCP connections.

The listener thread is separated from worker thread using a thread-safe queue, where sockets for new connections are left for the worker to handle.

To provide locking avoidance for the parser a statefull parser is implemented. A unique parser is assigned to each stream. The parser can be called to perform parsing of present data in the incoming buffers, to a certain limit, or to parse until the buffer is empty. For large incoming packets, the latter will block all other operations in the stack until the buffer is empty.

Encoding data works similarly. All packets requested to be sent are put in a send-queue by the software using the stack. The send-queue is emptied when the worker arrives at this operation and messages are moved to the send-queue for there TCP connection. Those queues are then emptied either fully or partially. At this point packets are also subject to fragmentation, or chunking as expressed in section 5.1 in RFC 4975 [6].

Configuration and logging interfaces

The MSRP stack is configured using the Properties class. Properties can be hard coded on in the application or loaded from an XML or Properties (key=value) formatted stream. The properties are fed to the stack upon instantiation and are not subject to change in the present implementation.

Commons Logging by the Apache Commons community is used for logging. It provides a convenient interface for libraries and components used by applications. The logging facility itself does not provide a connection to logging output but must be used together with a “native” logging facility, in our case Apache Log4j managed by either the client or the SipServlet applications.

TCP connection direction and usage

RFC 4975 [6] slightly covers certain aspects of the problems posed by peer-to-peer protocols over persistent channels. In contrast to other TCP driven protocols like HTTP, MSRP maintains persistent TCP connection with request in both directions. This requires reactivity at both endpoints, as well as all relays on the path, as defined by the MSRP relay extension RFC 4976 [7].

Where both peers are servers, the first issue is which endpoint should establish the TCP connection. This part is clearly specified in RFC 4975 is being the originating client, i.e. the client sending the INVITE.

RFC 4975 implies the preferred usage of shared TCP connections where many MSRP connections are handled with the same TCP peer. The mechanism providing this capability in the stack is covered in detail in Section 5.1.2.
The problem arises upon session negotiation. The persistent MSRP relay scenario will not be covered in this report. To keep the scenario understandable for the reader I will cover only the most simple scenario where client incompatibility may occur. Assume Alice and Bob have established an MSRP peer-to-peer channel and connection. They are transmitting text messages in a session. For some reason they are negotiating a new connection to transmit a file. By RFC 4975 [6] recommendations they should reuse the existing channel, however this is only recommended and if the two clients make different choices the connection will fail. Alice will offer a path and await packages on the present channel and Bob may attempt to establish a new channel, which is not expected by Alice and thus denied.

At this point a scheme for determining form of connection establishment at session negotiation should be required. This is however undefined and subject of client incompatibility. This issue is very probable to arise for any type of TCP protocols under SIP. In our case, since the same stack is running in all clients, we enforce the reuse of channels since it is more “sexy” (common expression in European IT development projects). The reuse is determined by maintaining a mapping of destination hosts for both incoming and outgoing connections.

5.1.3 IMS Stack

The Java Community Process JSR 281 [20] is a draft of a Java Micro Edition API for IMS application development on JME ready devices. The present public draft was approved as late as October 29 2007. The purpose of the JSR is to specify an API which enables non-IMS-experts to develop cross-device IMS applications. The interface abstracts the SIP and SDP layers for session negotiation with a set of interfaces.

A very important note is that JSR 281 is merely an interface an underlying IMS layer. This layer is most probably (on a portable device) a native implementation. The interface specifies the java application view and is thus very limited and is not aware of the IMS layer implementation and mechanics.

This interface model, even if not targeted for the JSE platform, is chosen as the base of the standalone application development. A few extensions and limitations are made in the implementation to create a stack adapted for the JSE platform.

**JSR 281 shortcomings**

Here are issues and proposals to adapt and extend the JSR 281 interface for JSE usage.

**TCP connection state** For reliable media transmissions, modeled as interfaces BasicReliableMedia for pure TCP and FramedMedia for MSRP over TCP connection are missing information for connection state. In contrast to RTP/UDP
where data is sent without confirmation to a supposedly reachable address the TCP transmission are dependent on a confirmed transmission channel.

For BasicReliableMedia the Listener should be extended to be notified of such changes. As of the current JSR 281 draft [20] is only notified of failure in establishment and never alerted of successful connection.

MSRP transmissions are dependent of several TCP connections as its path to the peer may be relayed. It is therefore important to have state information where the path to the peer is in a functional state. At MSRP connection establishment the MSRP stack is implemented to send a VISIT package to confirm the connection. After such a successful transmission, the Listener should be alerted of the state change in the FramedMedia.

Session/focus model Sessions are modeled by the Session interface. Implementations are instantiated using a CoreService which in turn originates from the Stack. The sessions are created using a from-user to-user aspect suitable for one-to-one conversations. It is of course also applicable to focus sessions (Section 3.6) since the actual sip session is established between the UE and the SIP focus.

Conference sessions are going to become much more usual as they are more easily built using IMS than older more rigid communication systems. I think it is therefore important to make difference in the API between sessions and focuses. The focus contains much more dynamics than the session, foremost the state subscription and participant management. A session is merely a point-to-point connection and has generally only on/off state. A focus has many users, they may join or leave or publish presence information to the focus. Creating a focus model in the API will simplify the view of the application developer.

Engine control The IMS engine of a portable device is a native management layer on the device which handles network connections for the subscriber. It uses subscriber information from the ISIM and not from user-space. The engine is of course always running on the device, similar to the GSM connection management of 2G devices.

This layer does not exist for the JSE platform. Therefore a tiny extension to bootstrap the engine is done. This is with the Manager.loadProperties(Properties p) method. It will start the “background” engine with the given properties. The client can after that use the JSR 281 interfaces as specified. Manager creation and engine init can be done manually but also with the ImsFactory which permits implementation selection.

This solution leaves a lot of space for improvements since the Manager is static and jvm global and multiple jvm may have port clashes. Ericsson offers a product containing what they call a ICP (IMS Client Platform). This is a background process in the machine which provides the IMS connection. Applications then connect to it to use the IMS core. The API for programming with the ICP is not related to JSR 281.
**Group management**  What is often mentioned as group management can generally be referred to as non-client bound information such as contacts and groups. The OMA (Open Mobile Alliance) refers to this as XDM (XML Document Management) which is related to the IMS way of doing this.

For group management in IMS the XCAP protocol is used to manage central configuration files in XML format. The JSR 281 does not handle this issue, most probably because of the native device function which does this and later offer a separate interface to device contact lists to the application.

For a standalone applications this is however highly interesting and a particularity is the fact that user presence is offered in the interface, but not the user account information itself, whereas they are actually related.

An interface providing a high level of abstraction for this is provided in the resource package extension detailed in Figure C.11.

**Symmetric client enforcement**  Due to all the network issues (Section 3.1) over non-manageable networks (Section 3.4) and to simply avoid stumbling over a clients wishing to run asymmetrical (Section 3.2.1) data streams the stack enforces symmetrical communication.

**Rich presence information**  The JSR 281 interface for event subscription is extremely limited. The only information notification information that the interface provides is a raw String provided via the SubscriptionListener when a notification arrives. This leaves all the data treatment to the client and somewhat loses some of the purpose of the abstraction layer which is supposed to hide the IMS specific parts to the implementer.

This is more of a general issue for subscription information. Since subscriptions can be done to various different event packages one might consider designing one particular interface for each one which belongs to the IMS standard. The three most interesting would probably be “presence”, “reg-state” and “conference”. For any other extension, the present, general, interface would be used.

**SessionDescriptor interfaces**  For session negotiation, the java community process defined JSR 141 clearly defining an interface for SDP programming. Reading the specification of JSR 281 you get the impression that the working group got tire when arriving at session descriptor. JSR 141 offers an easy to use interface to SDP whereas JSR 281 only has a number of getters giving raw Strings. This leaves much of the work to the implementer, which is perhaps not the purpose of the JSRs.
5.1.4 Java client

The main purpose of the java client is the ability to demonstrate the capabilities of the network and the implemented software components. Since the IMS stack hides everything concerning SIP, MSRP and other protocols used for the IMS connection the implementation of the client is quite straightforward. The class diagrams of the client is presented in Appendix C.5.

Model, view, controller

The model-view-controller development pattern is popular when developing applications with human interfaces. The main principle of the model is to separate the visual and IO components, the view, from the data delivering back-end, the model. This is done by introducing a new intermediate part, the controller.

Figure 5.1 shows the principles used for the implementation. The view, what the users see, is implemented in Java Swing. Swing is the second generation of GUI (Graphical User Interface) libraries provided with J2SE. It provides all the mechanisms and widgets to build just about any application.

For all components in Swing, callbacks can be registered. Callbacks are listeners, triggered when actions are performed on the GUI. Example of such are when keys are pressed or when the mouse pointer enters or leaves areas. The callbacks are the way that the controller is alerted of changes in the view. The controller then takes necessary actions. When user inputs actions the main work done by the controller is to send commands to the model, to request changes in the stack.

The IMS stack also works with callbacks. Mostly upon state changes for the different components in the stack the listener is notified. Such events occur when invitations arrive, media is transmitted or when services requested are successful or fails. At this point the controller notifies the user by changing the view.

![Model-view-controller architecture behind the java client](image_url)
5.2 Server software

On the server side, most components are already available. The CSCF and HSS are provided from the OpenIMSCore project at FOKUS Berlin (see Appendix A). It uses SER and FHoS for these functions. The presence server is deployed with OpenSER and OpenXCAP. OpenXCAP can also be used for the general XCAP access. The parts that are missing are SIP-AS for focus and media resource management. Those parts will have to be designed and developed. For SIP-AS development a server with a servlet container supporting the JSR 116 SipServlet API is suitable. In this case the BEA SIP server is chosen, the main reason for this is that it is already used and the company has licenses for it.

5.2.1 SIP focus for MSRP sessions

The SIP-AS providing instant messaging conference capabilities is developed as one piece of software working as SIP-AS, MRFC and MRFP. The activities that should provide are described in Figure D.1. The server is implemented as a servlet in the BEA servlet container. The container provides a SIP interface where the BEA SIP Server provide most of the SIP functionality leaving only small pieces required to develop for the servlet. The BEA server also provides the necessary Diameter components needed by the MRFC.

The MSRP stack must however be completely developed. The MSRP stack is shared among many platforms requiring it to provide a flexible interface for usage, but also a flexible interface towards external facilities such as logging and configuration. It loads configuration in the form of Properties and uses commons-logging as logging facility. The stack is covered more in detail in Section 5.1.2. The MSRP stack provides the media layer interface for the part of the server which in IMS would be called MRFP (Media Resource Function Processor).

Limitations

The current SipServlet specification reserves headers such as From, To and Contact as so called system headers. This means that the possibilities to modify them is reserved for the servlet container. For the focus model described in Section 3.6, where the Contact header provides the means to communicate the focus URI to a client creating it this is an issue. The Contact header may only be set by the servlet for initial REGISTER requests and 3xx and 4xx responses.

For this reason, implementing the focus mechanisms like this is simply impossible. Instead the focus is implemented to transmit this information in the Reply-To header.

The BEA SipServlet container does not implement all Privacy header extensions in IMS. In this case the P-Asserted-Identity set by the P-CSCF is not implemented and thus cannot be used for client identification. Instead the public user identity
must be taken from the From header. This is much less secure since it may be forged by the client and manipulated.

Focus model

The focus is modeled according to class Diagram C.13. This first version of the model does invite for some future extensions. It manages focuses. Each focus may have one or more users. An empty focus may be considered dead. Each user may be connected to the focus using one or more sessions. The model suggests a possible extension for SMS. In a next-generation model the session can be made more generic, supporting back-to-back sessions to provide RTP media conferences (in combination with MSRP).

In the focus. Each message arriving to the focus is retransmitted to all other Users. It is up to the implementation for each sessions which messages to transmit and how to provide status notifications. This permits a simple and powerful differentiation between SMS and MSRP messaging.

5.2.2 Billing proxy

Simple peer-to-peer session does not need the support of an application server in the bare IMS core network. However in a deployment situation the billing proxy provides a monitoring role to provide tracking of calls and reporting of billing information to the HSS.

The proxy is implemented in one single servlet listening for new INVITE request on the network. When an initial request arrive, the AS may determine whether the call may continue, depending on user credits or permissions. The proxy always records its route for initial request to be able to follow the entire SIP dialog.
5.3 Client based test bots

For testing purposes a simple bot framework is developed to quickly be able to implement autonomous clients imitating particular behaviour. These simple bots lets the implementor and integrator to easily test basic functions and client behaviour.

5.3.1 MSRP echo bot

Simple bot accepting all MSRP connections and echoing all messages sent to it. As an extension it may also accept invitation to focus sessions.

5.3.2 Eliza bot

Eliza is computer program imitating a therapist by replying to questions and posing follow-ups, imitating artificial intelligence in a simple way. The Eliza bot provides a connection between an MSRP session to a Java implementation of the Eliza program.

5.3.3 Inviter bot

The inviter bot sends an invitation every 30 seconds to sip:alice@ims.ao (or any other configured public identity). It sets up an MSRP connection, transmits a few messages (echos messages received) and then disconnects. This enables some client testing of session management.

5.3.4 Conference bot

The conference bot provides a network user which always accepts invitations to MSRP conference sessions, chat rooms. When in the chat room, The bot replies to all messages received with an echo. Also some additional info is added to the echo to notify participants, via the media, of the information that the bot has about the session.

5.3.5 Conference inviter bot

This is similar to the inviter bot. But instead of inviting the peer to a session it invites to a conference focus using a REFER request. This gives an opportunity to test the stack and client reactions to arriving invitations. After the invitation is done, the bot echos messages received.
Chapter 6

Summary and future work

6.1 Summary

6.1.1 Present status

As of finalizing this document, the deployed system and the implemented software components supports the main core features specified in the initial project description; instant messaging with MSRP, presence and central contact lists (group management). The system also supports IMS compliant clients to establish session of other media types (such as audio/video) which do not require any network specific handling. This is however not implemented in the IMS stack.

6.1.2 Technologies used

The components of the system relies on different technologies enabling their communication frameworks and run-time environments.

**IMS Core:** OpenSER, FHoSS, OpenIMSCore, OpenXCAP, Tomcat, MySQL, Named, Python, J2EE

**IMS Stack:** Apache Commons-Logging, Apache HttpClient, JAXB, JAIN-SIP, J2SE

**jClient:** J2SE, Apache Log4j, Swing
CHAPTER 6. SUMMARY AND FUTURE WORK

6.2 Future integration

Now that the core services of the IMS system is in place it is fairly easy to extend it with new services and interfaces towards other networks. The most interesting extension as a continuation to this project is the implementation of conference services for MSRP. Also the development of an MSRP relay for a consistent NAT traversal solution is of high interest.

Adding an RTP proxy to help NAT traversal for real-time media is an extension which already exists and should not be too hard to add. For RTP media conferences there exists numerous enterprise solutions. Examples of those are the MRF servers...
from Converse and Streamwide. They provide both multimedia conferencing as well as B2BUA capabilities and VoiceXML controlled services.

An interesting subject for future thesis work is to implement ultra-multimedia sessions supporting both real-time and reliable media. For example sessions supporting audio, video, textual messaging, central file transfers/storage and co-editing.

![Diagram](image-url)

**Figure 6.2**: Example future extension possibilities to the presently integrated components. Components named with software suggestions.

### 6.3 Future development

The software components implemented has been designed with concern to future implementation interests. Due to time and subject limitation only a small subset of the functionality has been implemented to support the required functions.

This section does only cover the currently most interesting features which may serve well for developing new services.
6.3.1 MSRP Stack

The statefullness hierarchy

The stack is designed to provide statefullness on different levels depending on the wishes of the developer. The present implementation only provides complete statefullness which in most cases is a good option as it completely hides the MSRP mechanisms to the implementer.

Supporting stateless connections and transmission would allow implementers to control connections and ease up implementation of very specific extensions.

Relay support

One of the most central important design concepts of MSRP is the possibility to connect via relays. In fact MSRP offers a TCP-like transmission scheme where an application layer supporting authentication, relaying and multi-leg progress reporting is added.

This is not implemented into the stack but would be vital for many applications running behind strict network policies.

Connection reestablishment support

On cellular networks, where the network connection is not reliable and subject of disconnections and IP reassignments there is a true issue with reliable media transmissions. MSRP has not specifications (yet) how to handle this issue which may become an outstanding issue in future implementations.

6.3.2 IMS Stack

As a starting point the JSR 281 specification for IMS services in java micro edition was chosen. The base interfaces have been changed to provide a “stack” instead of a “manager” interface towards the native implementation. Also, some functionalities have been added to support more features in reliable communication. A service have been added to provide XCAP support directly in the stack.

Present implemented features

The current implementation supports:

- Session (SIP peer-to-peer sessions)
• Publication (presence package)
• Subscription (presence package)
• FramedMedia (MSRP)
• Resource lists (group management) for a single contact list
• Multiple public identities per private identity

Focus support

An initial draft interface has been added for supporting focus sessions in SIP. This provides an abstraction of the otherwise complex mechanisms of invitations, referrals and event subscriptions present in focus session. It is partially implemented.

Page messages

SMS has for the last 10 years been the most widespread of instant messaging on mobile devices. This way of communicating won’t go away tomorrow and demonstrating this is interesting maybe most importantly to make the distinction from between session-based massaging and pager-mode messaging.

Capabilities

In the extension, the IMS system will provide all communication methods for it’s subscriber. Be it instant messaging, voice calls or video-on-demand. Capabilities announcement is critical to enable the IMS network to be aware of which media is accepted/supported at user endpoints.

6.3.3 Java Client

The Java client has been implemented fairly fast to make use and visualize the implemented features of the IMS stack. Today it provides a login window, central contact list (with manipulation of course) and the possibility to participate in peer-to-peer message session. The interface has been designed with the addition of more basic media and services in mind. When those have been implemented in the IMS stack the step to usage in the client is not far away.
Figure 6.3: Screen shot from the Java Client version 0.1 running with a chat session to the Eliza chatterbot open.
Chapter 7

Results

7.1 Discussion

This thesis provides the standing ground for developing IMS services beginning at a knowledge base limited to basic IP networking knowledge. By starting at zero and working the way towards the goal, instant messaging in IMS, covering all steps, the basis is given for understanding of the IMS network, how it works, and how it provides services.

Obviously, the work done follows certain predefined milestones. There is a need for the prototype to work under certain network conditions. Certain services should be provided, namely instant messaging, which should be demonstratable. The platform should be extensible for future use and general service deployment should be possible. These issues are all covered and an implementor of other services than instant messaging can draw experience from the results presented by this thesis.

With the evolution and convergence of network societies on the internet instant messaging is now at an interesting point. Jabber/XMPP has been embraced by the open source community as a protocol which is simple yet powerful and has an open specification with a clear evolutionary process. On the proprietary side we have Microsoft, which with their protocol for instant messaging are offering integration of “Live Messaging” in the service platforms of their clients. This is something France Telecom (Orange) is offering at the moment. The third protocol is that specified in IMS. IMS uses the combination of SIP and MSRP separating signaling and media, just as real-time media streams. But yet, IMS has no client deployment and the specification lies dormant until the day it arrives. The question is only, will the telecom operators chose to use the instant messaging in IMS?
7.2 Conclusions

The direct results of these thesis work is based on the problem definition. Therefore the summary of problem definitions is repeated here.

- Analyze the present demonstration platform to determine its re-usability and extensibility.
- Integrate a prototype platform providing the CSCF and HSS IMS functions.
- Enable IMS instant messaging services on the IMS platform.
- Propose a solution reducing or eliminating connection issues concerning networks behind NAT.
- Design and implement necessary missing software components to enable instant messaging services.
- Add, or suggest possibilities to add, additional services to the IMS platform. Most notably presence services and group management.

The present demonstration platform is a hybrid solution where one application server drives all IMS services and the web client itself. The web client also contains a java applet. The java applet provides advantages in the audio video interface, but also brings along disadvantages in terms of NAT issues. This previous demonstration prototype was not used in the developed platform.

The new IMS prototype platform was based on the open source project OpenIMSCore by TU Berlin. It provides the SIP routing elements (P/I/S-CSCF) and the HSS. The presence service was added using the OpenSER presence module, not being targeted for IMS systems, proving the possibility to integrate present SIP services in IMS networks. An XCAP server (OpenXCAP) was also added to provide group management capabilities.

Concerning network restrictions, focused on NAT (Network Address Translation), the best way to deploy the core network is in the DMZ (DeMilatarized Zone). This is mainly due to company security policies and the absence of suitable SBC (Session Border Controller) software.

Since no MSRP java component is available in the open source community, this part has to be developed. An MSRP stack is developed in two separate components; API and implementation. It is designed to be used both in clients and servers.

The IMS instant messaging server is developed in a SipServlet (see Section 5.2.1) container, providing the SIP interface, and with the instant messaging session/chat-room manager using the MSRP stack.
Bibliography


Appendix A

Links

This is a collection of Internet links which does not qualify as formal references. Here you can find resources for software downloads and project information.

- http://www.openxcap.org/
- http://openser.org/
- http://www.openimscore.org/
- http://www.asterisk.org/
- http://www.mobicents.org/
- http://www.rtpproxy.org/
- http://www.ag-projects.com/MediaProxy.html
- http://commons.apache.org/logging/
- http://maven.apache.org/
- http://hc.apache.org/httpclient-3.x/
- https://jaxb.dev.java.net/
- https://openxdm.dev.java.net/
- http://www.bea.com/sip/
- https://jain-sip.dev.java.net/
- http://junit.sourceforge.net/
- http://www.genesyslab.com/
- http://www.comverse.com/
- http://www.streamwide.com/
Appendix B

Establishing SIP focus session

This document describes in detail SIP and MSRP messages passed over the IMS network when establishing, participating in, and leaving an MSRP conference, driven by a SIP focus.

In this case the system contains a P, I and S-CSCF. Both participants are separate private networks behind NAT routers. Their UE are unaware of any particular methods of NAT traversal, they dint even know their public IP.

- P-CSCF proxy.ims.net : 192.168.5.1 @Public 5.5.5.1
- I-CSCF inter.ims.net : 192.168.5.2 @Public 5.5.5.2
- S-CSCF server.ims.net : 192.168.5.3
- SIP-AS/MRFC mrfc.ims.net imschat@ims.net : 192.168.5.4
- MRFP-MSRP msrp.ims.net : 192.168.5.5 @Public 5.5.5.5
- Alice alice@ims.net alice.athome.net : 192.168.1.1 @Public 1.1.1.1
- Bob bob@ims.net bob.athome.net : 192.168.2.1 @Public 2.2.2.1

B.1 Alice creates the session

Alice starts by sending an INVITE to the conference factory. In this case it is the SIP-AS that handles this. She knows the URI of the conference factory imschat@ims.net by UE configuration. In this first step every jump in the IMS network traversal is included, later it will be omitted.
The P-CSCF records the route and forwards it to Alice’s home network ims.net:

INVITE sip:imschat@ims.net SIP/2.0
Via: SIP/2.0/UDP proxy.ims.net
Via: SIP/2.0/UDP alice.athome.net;received=1.1.1.1
To: IMSChat <sip:imschat@ims.net>
From: Alice <sip:alice@ims.net>;tag=2345
Call-ID: a3456@alice.athome.net
CSeq: 1 INVITE
Route: inter.ims.net server.ims.net
Record-Route: <sip:mo@proxy.ims.net:5060>
Contact: <sip:192.168.1.1:5060>
Content-Type: application/sdp
Content-Length: ?
v=0
o= 9876 1 IN IP4 alice.athome.net
s=Our chatroom
c=IN IP4 alice.athome.net
t=0 0
m=message 7394 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://alice.athome.net:7394/2s93i9ek2a;tcp

The I-CSCF can now determine if Alice is a valid user in ims.net and which S-CSCF to use, if more than one exists. It records the route and forwards:

INVITE sip:imschat@ims.net SIP/2.0
Via: SIP/2.0/UDP inter.ims.net
Via: SIP/2.0/UDP alice.athome.net;received=192.168.5.1
The S-CSCF uses its internal filtering condition(s) to determine if and which AS (Application Server) should process this Request. It records the route and forwards:

INVITE sip:imschat@ims.net SIP/2.0
Via: SIP/2.0/UDP server.ims.net
Via: SIP/2.0/UDP inter.ims.net;received=192.168.5.2
Via: SIP/2.0/UDP proxy.ims.net;received=192.168.5.1
Via: SIP/2.0/UDP alice.athome.net;received=1.1.1.1
To: IMSChat <sip:imschat@ims.net>
From: Alice <sip:alice@ims.net>;tag=2345
Call-ID: a3456@alice.athome.net
CSeq: 1 INVITE
Record-Route: <sip:mo@proxy.ims.net:5060>
Record-Route: <sip:mo@inter.ims.net:5060>
Record-Route: <sip:mo@serv.ims.net:5060>
Contact: <sip:192.168.1.1:5060>
Content-Type: application/sdp
Content-Length: ?
v=0
o=- 9876 1 IN IP4 alice.athome.net
s=Our chatroom
c=IN IP4 alice.athome.net
t=0 0
m=message 7394 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://alice.athome.net:7394/2s93i9ek2a;tcp

The request now arrives at the SIP-AS serving SIP focuses and MSRP sessions. It will create an MSRP session, identified by “abcd”, and a SIP focus, identified the
same way, this is not necessary. The SIP-AS responds with an OK containing the SIP focus address in the Contact header and the connection information for the MSRP session in the SDP answer. Note also how session-version in the “o=” field is incremented.

SIP/2.0 200 OK
Via: SIP/2.0/UDP server.ims.net;received=192.168.5.3
Via: SIP/2.0/UDP inter.ims.net;received=192.168.5.2
Via: SIP/2.0/UDP proxy.ims.net;received=192.168.5.1
Via: SIP/2.0/UDP alice.athome.net;received=1.1.1.1
To: IMSChat <sip:imschat@ims.net>;tag=3456
From: Alice <sip:alice@ims.net>;tag=2345
Call-ID: a3456@alice.athome.net
CSeq: 1 INVITE
Record-Route: <sip:mo@proxy.ims.net:5060>
Record-Route: <sip:mo@inter.ims.net:5060>
Record-Route: <sip:mo@server.ims.net:5060>
Contact: <sip:imschatabcd@192.168.5.4>;isfocus
Content-Type: application/sdp
Content-Length: ?
v=0
o=- 9876 2 IN IP4 msrp.ims.net
s=Our chatroom
c=IN IP4 msrp.ims.net
t=0 0
m=message 2588 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://msrp.ims.net:2588/abcd;tcp

Alice must now only connect to the MSRP server and VISIT or SEND to the session to be a part of it. RFC 4975 section 8.5 specifies that the inviter is responsible for creating the connection and thus Alice can connect as long as the MSRP server is reachable from the outside, she may also use an already present connection to that MSRP server. Her next step is joining the MSRP session.

MSRP dkei38sd VISIT
To-Path: msrp://msrp.ims.net:2588/abcd;tcp
From-Path: msrp://alice.athome.net:7394/2s93i9ek2a;tcp
Message-ID: 456s9wlk3
-------dkei38sd$

The MRFP replies

MSRP dkei38sd 200 OK
To-Path: msrp://alice.athome.net:7394/2s93i9ek2a;tcp
From-Path: msrp://msrp.ims.net:2588/abcd;tcp
-------dkei38sd$
B.2 Alice creates a state subscription to the session

Subscribing to events in a focus does not function in the same way as subscribing to user events in IMS. In the IMS system user events are handled by presence servers and not by subscribing directly to the users themselves. In focus sessions, the presence is handled internally in the focus, but with the same protocol as a normal subscription.

```
SUBSCRIBE sip:imschatabcd@192.168.5.4 SIP/2.0
Via: SIP/2.0/UDP alice.athome.net
To: IMSChat <sip:imschat@ims.net;session=abcd;isfocus>
From: Alice <sip:alice@ims.net>;tag=76283
Call-ID: aq528f
CSeq: 1 SUBSCRIBE
Event: ?
Contact: <sip:192.168.1.1:5060>
```

She then gets two replies. One notification and one OK

```
SIP/2.0 200 OK
To: Alice <sip:alice@ims.net>;tag=76283
From: IMSChat <sip:imschat@ims.net;session=abcd;isfocus>
Call-ID: aq528f
CSeq: 2 SUBSCRIBE
Contact: <sip:imschatabcd@192.168.5.4>;isfocus

NOTIFY sip:alice@ims.net SIP/2.0
From: IMSChat <sip:imschat@ims.net;session=abcd;isfocus>
;tag=893790
To: Alice <sip:alice@ims.net>
Call-ID: aqsd38907
CSeq: 1 NOTIFY
Contact: <sip:imschatabcd@ims.net>;isfocus
Content-Type: application/pidf+xml
Content-Length: ?

<?xml version="1.0" encoding="UTF-8"?>
<conference-info
   xmlns="urn:ietf:params:xml:ns:conference-info"
   entity="sip:imschatabcd@ims.net"
   state="full" version="1">
   <conference-description>
     <subject>Just another chat room</subject>
   </conference-description>
   <conference-state>
```
Yes, she got her own information... NOTIFY is a request and is therefore answered with an OK.

B.3 Alice invites Bob

Alice doesn’t really feel like chatting for herself, therefore she invite Bob as well. This is done using the REFER SIP method which is sent to Bob. It contains the URI of the SIP focus which lets Bob invite himself. The REFER will itself result in a set of messages exchanged between Alice and Bob but they’re not really interesting at this point.

REFER sip:bob@ims.net SIP/2.0
From: Alice <alice@ims.net>;tag=9289
To: Bob <bob@ims.net>
Refer-To: IMSChat <sip:imschatabcd@ims.net>;isfocus

B.4 Bob invites himself and joins

INVITE sip:imschatabcd@ims.net SIP/2.0
From: Bob <bob@ims.net>;tag=8927208
To: IMSChat <sip:imschatabcd@192.168.5.4>;isfocus
Call-ID: 987097865
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: ?
v=0
o=- 9876 1 IN IP4 msrp.ims.net
s=Our chatroom
c=IN IP4 msrp.ims.net
t=0 0
m=message 2588 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://bob.home.net:2588/572424;tcp

The MRFC replies
SIP/2.0 200 OK
From: IMSChat <sip:imschat@ims.net;session=abcd;isfocus>
;tag=976367891912
To: Bob <bob@ims.net>;tag=8927208
Call-ID: 987097865
CSeq: 2 INVITE
Content-Type: application/sdp
Content-Length: ?
v=0
o=- 9876 2 IN IP4 msrp.ims.net
s=Our chatroom
c=IN IP4 msrp.ims.net
t=0 0
m=message 2588 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://msrp.ims.net:2588/abcd;tcp

Bob sends an ACK back and is ready to VISIT or SEND to the MSRP conversation. As mentioned, since he is the inviter in this case he is also responsible for the connection. This avoids the NAT problem he might have been posed to at this time.

At this time a notification will go out to Alice about Bob joining. She will also receive information in the REFER dialog about the success for Bob. The focus state information may reveal differences between the MSRP and the SIP session if Bob has not joined the MSRP session yet. This may be discussed at a later time.
Appendix C

Class diagrams

C.1 MSRP stack API
C.2 MSRP stack implementation
C.3 IMS stack API
C.4 IMS stack implementation
C.5 Java client
Figure C.1: javax.msrp and javax.msrp.data package class diagram
APPENDIX C. CLASS DIAGRAMS

Figure C.2: javax.msrp.data.header package class diagram

Figure C.3: javax.msrp.data.packet package class diagram
Figure C.4: com.atosorigin.msrp package class diagram
Figure C.5: com.atosorigin.msrp.data package class diagram
Figure C.6: com.atosorigin.msdp.data.header package class diagram, only implemented classes included
Figure C.7: com.atosorigin.msrp.data.packet package class diagram, only implemented classes included
Figure C.8: javax.ims package classes
Figure C.9: javax.ims.core package classes
APPENDIX C. CLASS DIAGRAMS

Figure C.10: javax.ims.core.media package classes

Figure C.11: javax.ims.resource package classes
Figure C.12: The SipServlet API. Taken directly from the API specification. This is just for clarification.
Figure C.13: SipServlet focus for MSRP conference sessions class diagram.
Figure C.14: Java client controller classes
Figure C.15: Java client view classes
Appendix D

Activity diagrams
Figure D.1: Message handling of the MSRP/SIP focus server
Client constructs MsrpStack

Figure D.2: Client initiating the MSRP stack
APPENDIX D. ACTIVITY DIAGRAMS

Figure D.3: Client subscribing to focus state information

Figure D.4: Client joining an SIP focus and an MSRP session
Appendix E

State diagrams

E.1 MSRP stack API

Figure E.1: MSRP Channel (TCP connection) state diagram.

E.2 IMS stack API
Figure E.2: State diagram for the IMS session. INVITE, UPDATE and BYE originating at the UE. Session/listener calls and SIP messages displayed.

Figure E.3: State diagram for the IMS Session for INVITE, UPDATE and BYE terminating at the UE. Session/listener calls and SIP messages displayed.
Figure E.4: Datagram for publications. This diagram presents both the major states visible to the API user as well as the internal states used in the implementation.

Figure E.5: State diagram for subscriptions. This diagram presents both the major states visible to the API user as well as the internal states used in the implementation.

Figure E.6: State diagram for the life cycle of a Media
Appendix F

Sequence diagrams
Figure F.1: IMS Network level sequence for establishing a peer-to-peer session between two IMS terminals.
Figure F.2: UE component level sequence diagram for session establishment.