IT 10 016
Examensarbete 30 hp
Maj 2010

Multimedia Messaging Service Front End for Supplementary Messaging Services

Robert Andersson
Larry Canady

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Multimedia Messaging Service Front End for Supplementary Messaging Services

Robert Andersson & Larry Canady

The standardization forum 3GPP has specified a Multimedia Messaging Service (MMS) standard including an MMS Center (MMSC) that allows users to send and receive messages including e.g. text, images, audio and video. The 3GPP forum has however not standardized any MMSC provided supplementary services in relation to MMS.

The goal of the thesis is to create a Front End (FE) prototype which is capable of supporting a given set of MMSC supplementary messaging services for MMS and to develop and test an MMSC. The set of supplementary messaging services to be supported are auto-reply to an MMS message, auto-forwarding, convert an email to a MMS message, and cloning a MMS message.

The final results for the project are mixed. The prototype was not implemented entirely mainly due to lack of available software and not having access to an operators MMSC, which would be needed to push the supplementary services to mobile phones. However substantial progress was achieved. All of the supplementary messaging services were implemented and made compatible with Mobile Arts existing system. A testing system was also designed capable of processing simulated incoming MMS messages.

Handledare: Lars Kari
Ämnesgranskare: Ivan Christoff
Examinator: Anders Jansson
IT 10 016
Tryckt av: Reprocentralen ITC

Acknowledgments

We would like to thank Mobile Arts who provided us with a tremendous amount of support and advice throughout this project.

Contents

1 Introduction **4**
- 1.1 Previous work . . . 6
- 1.2 Erlang . . . 6
- 1.3 The Front End . . . 7
- 1.4 Thesis Outline . . . 9

2 Implementation **10**
- 2.1 Division of Labor . . . 10
- 2.2 Equipment . . . 10
- 2.3 Addressing . . . 11
- 2.4 Messages . . . 11
- 2.5 The Subscriber Database . . . 15
- 2.6 The Message Transfer Store . . . 15
- 2.7 The Message Handler . . . 17

3 Conclusion **20**
- 3.1 Problems . . . 21
- 3.2 Future work . . . 23

A Figures **25**

B Sample messages **28**

C Requirement Specification **32**
- C.1 Required Parts . . . 32
- C.2 Desired Parts . . . 34

Definitions and Abbreviations

MMS	Multimedia Messaging Service
3GPP	3rd Generation Partnership Project
MMSC	Multimedia Messaging Service Center
SOAP	Simple object access protocol
SMIL	Synchronized Multimedia Integration Language
WTP	Wireless Transaction Protocol
WSP	Wireless Session Protocol
VASP	Value Added Service Provider provides services other than basic telecommunications service for which additional charges may be incurred
MMS Relay/Server	A server responsible for storage and handling of incoming and outgoing messages and for the transfer of messages between different messaging systems
MMSE	A collection of MMS specific network elements under the control of a single administration
MIME	Multipurpose Internet Mail Extensions
Abstract Message	Information which is transferred between two MMS entities used to convey an MM and/or associated control information between these two entities
User Agent (UA)	An application residing on a User Equipment (UE), an MS or an external device that performs MMS-specific operations on a user's behalf and/or on another application's behalf

Chapter 1

Introduction

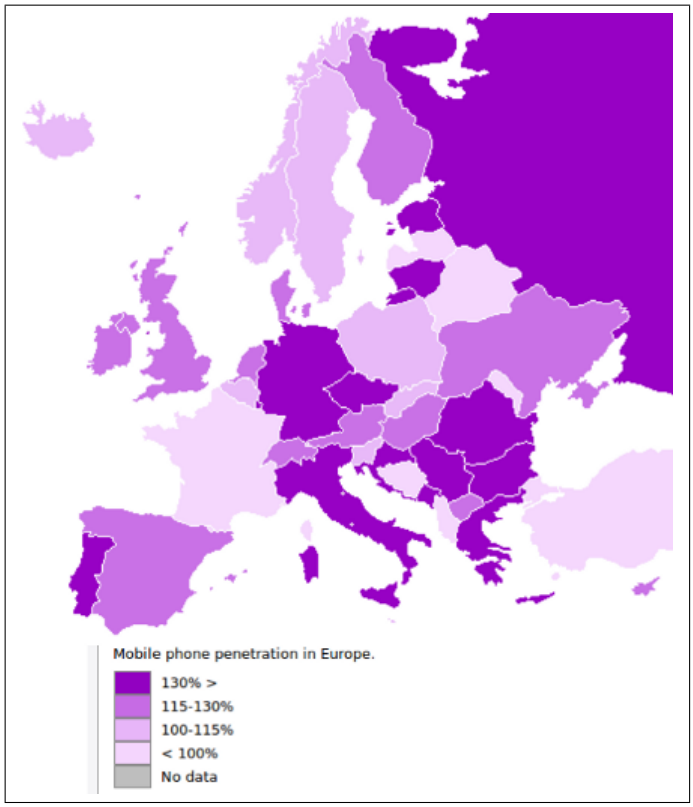

Figure 1.1: European Mobile Phone Penetration [10]

The idea of mobile phones has been around for ages. The predecessors to todays mobile phones were two-way radios and walkie-talkies. The use for these devices was varied, from military, police and taxi cabs to name a few. The difference between two-way radios and telephones are that they were not connected to telephone networks, meaning you could not dial to other telephones. The first phone made as a "portable cellular phone" was made by Martin Cooper,

a Motorola employee, in 1973 [6]. These early mobile phones were big, bulky, expensive and mainly used by professionals. The mobile phone made tremendous progress in the following years. In 1991 the 2nd Generation mobile phones networks were introduced. These GSM networks included the ability to send SMS messages. Nowadays mobile phones are a necessity. They are smaller, more stylish, and offer an array of features, including address books, alarms, calendars, calculators and many more. There are over 4 billion mobile phones in use throughout the world today[1]. The number of mobile phones throughout Europe, the United States and the rest of the world has steadily increased. Figure 1.1 shows the penetration of mobile phones throughout Europe. With the average cost of making phone calls falling, mobile phone operators are constantly looking for new ways to increase revenue. One service that operators were able to take advantage of is the SMS or short message system. SMS are text messages which can be sent and received on mobile phones. However this service was never intended to be a commercial product. It was initially intended to be used by operators to inform their customers about network problems (and possibly for advertising). This was due to the fact SMS's could only be sent within the same network at that time. Teenagers were the first users to use the system and take advantage of it. They discovered that SMS's could be sent to friends for free since the service was never intended to be used by individuals there was no mechanism at the time to charge customers for using it. This quickly changed, mobile operators would soon establish standards for sending SMS between networks and mechanisms for charging.

In 2008 an estimated 4.1 trillions messages were sent throughout the world earning telecoms operators 81 billion dollars[2]. Since any binary data can be sent via SMS other services also became available. One of the most successful services is the selling of ring-tones and screen-savers via SMS. However, SMS messages have some limitations as to what type of content can be sent. Messages are restricted to 160 characters and multimedia content is limited.

With the implementation of the 3rd generation GSM systems a new standard for MMS (Multimedia Message Service) was added. The standardization forum 3GPP has specified a Multimedia Messaging Service standard including an MMS Center (MMSC) that allows users to send and receive messages including but not limited to:

- Text
- Audio
- Synthetic audio
- Still Image
- Video
- Vector graphics

Figure 1.2 shows how MMS messages can encompass a variety of different networks types.

[1] International Telecommunication Union (ITU) estimate
[2] ITU Report 2006

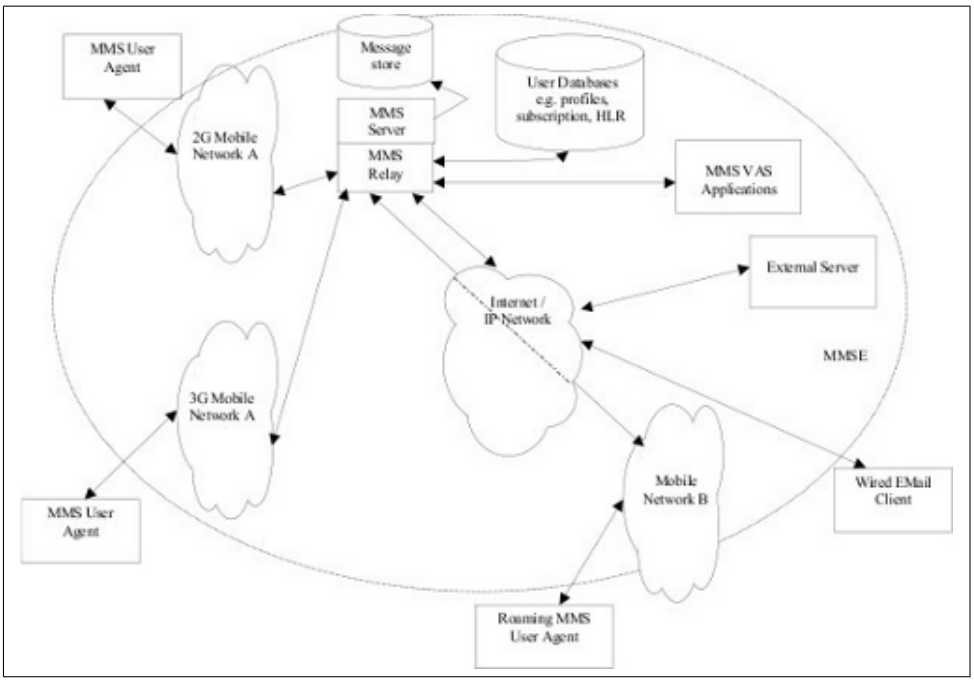

Figure 1.2: This figure represents a view how different networks handle mms messaging. [3]

The purpose of this thesis it to first specify an MMS Center Front End (FE) that is placed in front of an existing MMSC. The second part includes implementing the MMSC FE prototype to verify its characteristics.

1.1 Previous work

There are commercial products available to transform incoming MMS's into an array of other services, including email. An example is MMSNOW. Our work is built upon a product offered by Mobile Arts, Short Messaging Center (SMSC). This product supports the same supplementary services this thesis is based upon however for SMS's.

1.2 Erlang

As mentioned in the previous section the Front End (FE) is based on a similar product offered by Mobile Arts called a SMSC. The SMSC product is based on Erlang/OTP. Therefore Erlang was the chosen programming language for this project. Erlang is a functional programming language developed by Ericsson in the 1990's. Erlang is a general purpose concurrent programming language. It contains a small library of primitives for creating processes and passing messages between them [1]. There are also built in mechanisms for fault tolerance. These properties worked well for the design of the prototype. Since there were multiple abstract messages transmitted between the MMS Relay/Servers and the pro-

totype, not having to worry about synchronization eliminated some potential problems.

Erlang records have a very simple syntax for creating data types. They are essentially tagged tuples which can be easily manipulated. An example of a record (MM7_forward.res{message_type=MM7_forward.rep,<other fields>}) where the record name is given first and the fields are placed inside curly brackets. Notice that no data types are given for the records fields or values nor are they declared anywhere else. This is part of the Erlang language which is non-typed. Non-typed languages have some disadvantages especially with run time errors. However, Erlang has built in guards that can be used to type-check. These guards can also be user-defined. Erlang has its own built in DBMS called Mnesia, which has it own syntax for queries. (i.e. no similarity with SQL queries.)

1.3 The Front End

The FE was intended to act as a gateway between an user agent and originating MMS Relay/Server or a terminating MMS Relay/Server. The MMS Relay/Sever is responsible for the storage and handling of incoming and outgoing messages and for the transfer of messages between different messaging systems. There are several dozen abstract messages that a MMSC is required to support. For this project the FE was designed to accept only a subset of the MMS abstract messages.

- MM1 to/from subscriber equipment
- MM4 to/from MMSC
- MM7 to/from MMS Value Added Service Provider(VASP)

MM1 messages are used to submit Multimedia Messages from MMS User Agent to the MMS Relay/Server and to deliver messages to a UE from a VASP or another UE. MM4 messages use the reference point between the MMS Relay/Server and another MMS/Relay Server that is within another MMSE. MM7 messages use the reference point between a MMS Relay/Server and MMS VAS Applications. Figure 1.3 shows the FE architecture.

The FE would then apply one or several supplementary services to each message. The supplementary services that the FE supplies are:

- Email copy of a sent Mobile Originating (MO) MMS message to subscriber specific email addresses
- Email copy of a received Mobile Terminating (MT) MMS message to subscriber specific email addresses
- Auto Reply to a received Mobile Terminating MMS message to the senders addresses
- Forwarding of a received Mobile Terminating MMS message to subscriber specific forwarded-to subscriber numbers
- Cloning of a received Mobile Terminating MMS message to subscriber specific set of cloned-to subscriber numbers

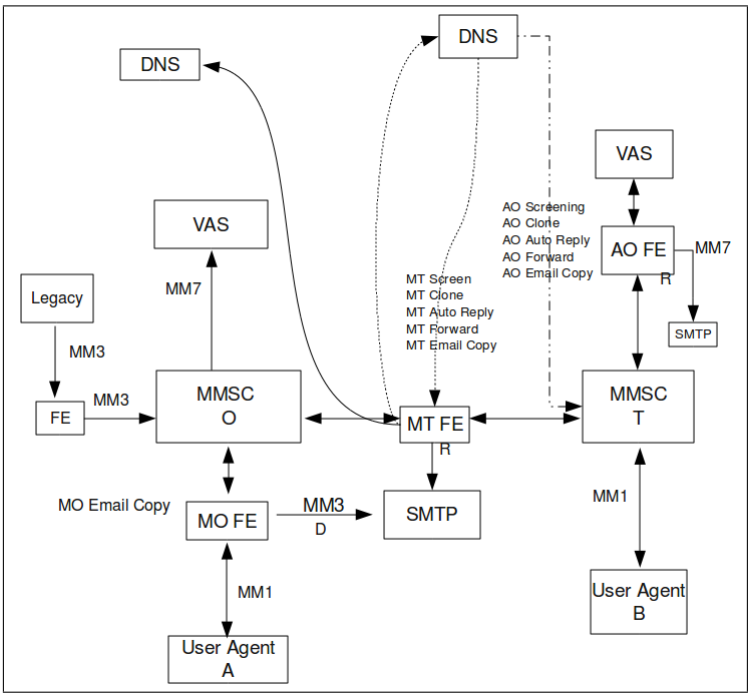

Figure 1.3: This is a abstract representation of different messaging protocols handled by the FE

MMS

These figures were taken from the document 3GPP TS 23.140 [3]. Figure 1.2 shows how MMS messaging can function with both 2G and 3G since Internet transport protocols such as SMTP and HTTP are used. Figure 1.4 show various uses of the messaging protocols specified in 3GPP TS 23.140. There are several other protocols for MMS messaging. However, for the purpose of this project the only MM1, MM4 and MM7 were used.

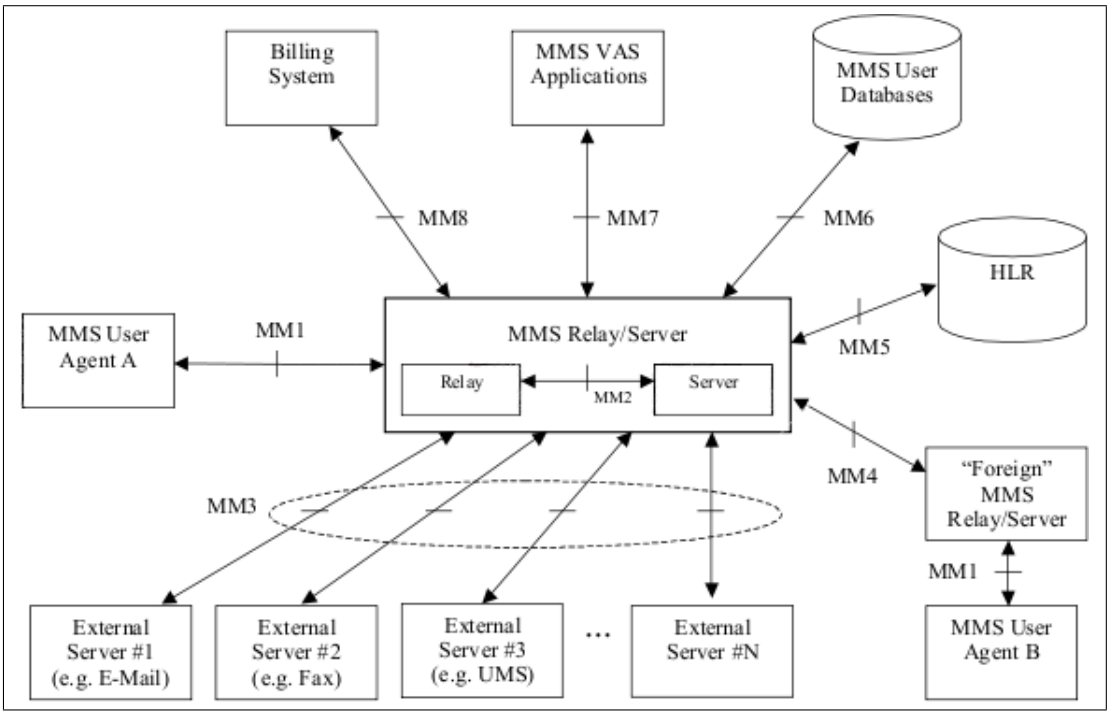

Figure 1.4: Reference Architecture [3]

1.4 Thesis Outline

In chapter 2 we discuss how we implemented the FE. The areas we focus on are the different addressing formats compatiable according to 3GPP standards. In this chapter we also describe the different message types which are supported by the FE and the different modules contained in the FE. They include Subscriber Database, Message Transfer Store, and the Message Handler. In chapter 3 we present our final conclusion and the problems we encountered during the project. The appendix is made of 3 sections. Appendix A contains figures describing different parts of the FE. Appendix B has different sample message types and appendix C contains the requirement specification for the thesis.

Chapter 2

Implementation

The initial implementation plan was divided into 5 parts. The first part of this project would be the only part which would have to be done together. Once the requirement specification was complete actual implementation could begin.
A preliminary copy of the requirement specification can be found in Appendix C. Items 2-5 were coded in parallel, with the work divided more or less evenly. The description in Section 2.1 specifies who was responsible for which individual modules. The plan for the project was as follows:

1. Write Requirement Specification
2. Create Message Types
3. Create Internal Modules
   - Message Transfer Store
   - Message Broker
   - Message Handler
4. Content Parser
5. Testing

2.1 Division of Labor

After the implementation and testing were completed, the report was written by both of us. For the coding we decided to divide up the modules between us.

Robert	Larry
MTS, SDB	Content Parser
Message handling	MM4 - Decoding/Encoding/Converting
MM1 - Encoding/Decoding	MM7 - Encoding/Decoding/Converting

2.2 Equipment

The equipment used was two laptop computers both running Ubuntu Linux 8.04. Two mobile phones were used for testing, one Sony Ericsson v600 and one Samsung SGH-X480. Emacs and Erlang/OTP R12 were used as the development environment.

2.3 Addressing

The 3GPP standards for addressing formats for MMS messages are E.164, MSISDN and E-Mail addresses [3]. E.164 is an ITU-T recommendation which is defined by the international public telecommunications numbering plan while E-Mail addresses are defined in RFC 2822 [7]. Service provider specific addresses are also allowed to be used. These addresses must be convertible to E-Mail addresses. The FE will have the ability to support all of the address formats.

2.4 Messages

There are eight types of abstract message (MM1 through MM8), which are sent between or to and from MMS Relay/Sever. Figure 1.4 shows which servers MMS messages are transmitted between. Each message has a subset of messages used to transmit specific information such as delivery information for MMS, read reply information and a variety of other information. Each request also requires a reply. The requirement for the FE only specifies that a limited number of messages be supported which are listed below. The remainder of messages which pass through the FE will simply be forwarded to the nearest MMS Relay/Server. The following list shows the FE supported messages.

- MM1_submit.REQ
- MM1_submit.RES
- MM1_delivery_report.REQ
- MM1_delivery_report.RES
- MM4_forward.RES
- MM4_forward.REQ
- MM4_delivery_report.REQ
- MM4_delivery_report.RES
- MM7_submit.REQ
- MM7_submit.RES
- MM7_delivery_report.REQ
- MM7_delivery_report.RES

MM1

MM1 messages are used between a user agent and a MMS Relay/Server. There are two protocols use to set up this communication. One is Wireless Session Protocol and the other is Wireless Transport Protocol. A MM1 abstract messages contain 29 informational elements. Table 2.1 shows the five mandatory elements in a MM1_submit.REQ message, the rest are optional. The entire list of elements of a MM1_submit.REQ can be found in Appendix B.1. Figure 2.1 shows the traffic flow when Mobile Originating FE and Mobile Terminating FE are added to the network.

Information Elements	Presence	Description
Message Type	Mandatory	Identifies this message as MM1_submit.REQ
Transaction ID	Mandatory	The identification of the MM1_submit.REQ or MM1_submit.RES pair.
MMS Version	Mandatory	Identifies the version of the interface supported by the MMS UA.
Recipient address	Mandatory	The address of the recipient(s) of the MM. Multiple Addresses can be used
Content type	Mandatory	The content type of the MMs content.

Table 2.1: MM1 Informational Elements [3]

MM4

MM4 messages are transmitted between two MMSC's. For each MM4 message sent to an MMSC a response is anticipated in order to confirm that the request has been received. MM4 messages use SMTP as it's transport protocol. Table 2.2 shows the header mappings used by MM4 when sending messages via the SMTP protocol. Figure 2.2 shows how the MM4 protocol is used to connect between MMSE service providers. As stated earlier in this thesis we are only implementing a subset of the MM4 abstract messages. The following MM4 messages are the ones which will be implemented and supported by the FE.

- MM4_forward.RES
- MM4_forward.REQ
- MM4_delivery_report.REQ
- MM4_delivery_report.RES

Information Element	STD 11 Header
3GPP MMS Version	X-Mms-3GPP-MMS-Version:
Message Type	X-Mms-Message-Type:
Transaction ID	X-Mms-Transaction-ID:
Message ID	X-Mms-Message-ID:
Request Status	X-Mms-Request-Status-Code:
Request Status text	X-Mms-Status-Text:
-	Sender:
-	To:
-	Message-ID:
-	Date:

Table 2.2: MM4 STD 11 Header Mappings [3]

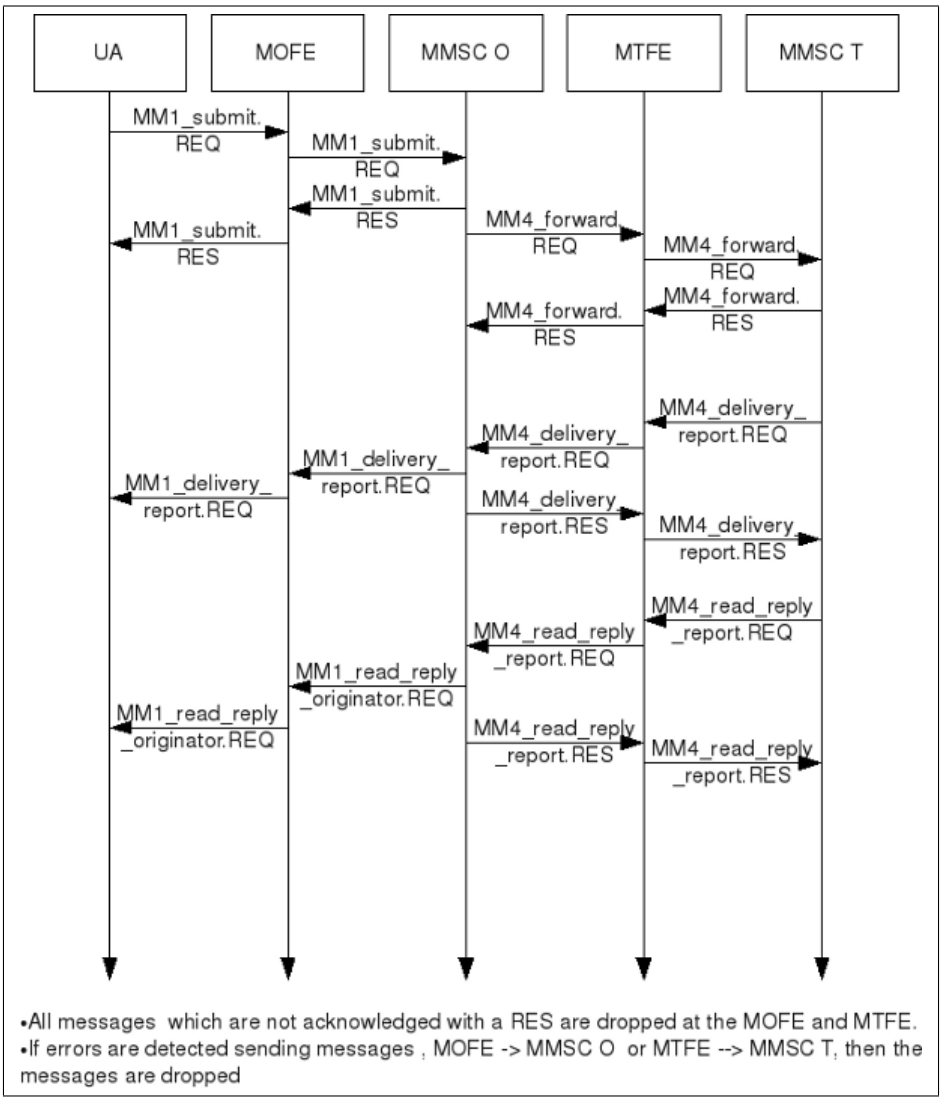

Figure 2.1: MM1 & MM4 Abstract Message Flow

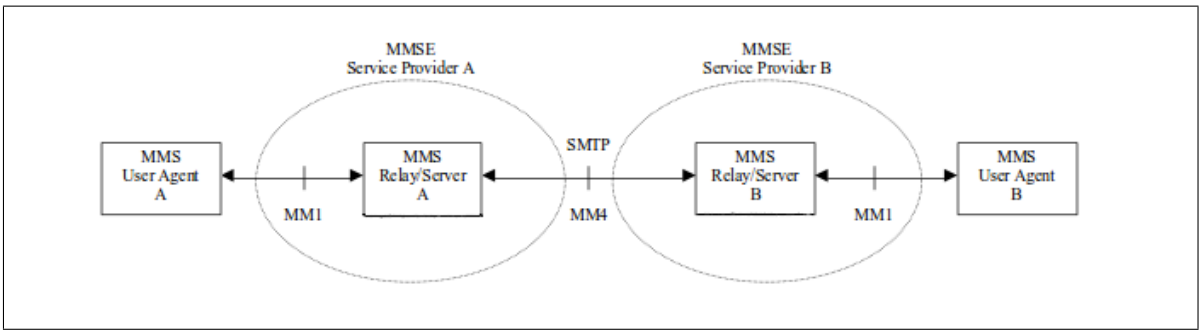

Figure 2.2: Networking of different MMSE's [3]

MM7

MM7 messages are transmitted between VASP and a MMSC. A VASP allows operators to better control content distribution. The transport protocol used for MM7 messages is HTTP POST. The message is encapsulated in a SOAP message with attachments. Table 2.3 shows a complete listing of the information elements found in a MM7_submit.RES.

Informational Element	**Presence**	**Description**
Transaction ID	Mandatory	The identification of the MM7_submit.REQ/ MM7_submit.RES pair.
Message type	Mandatory	Identifies this message as a MM7_submit response.
MM7 version	Mandatory	Identifies the version of the interface supported by the MMS Relay/Server,
Message ID	Conditional	If status indicates success then this contains the MMS Relay/Server generated identification of the submitted message. This ID may be used in subsequent requests and reports relating to this message.
Request Status	Mandatory	Status of the completion of the submission, no indication of delivery status is implied.
Request Status text	Optional	Text description of the status for display purposes, should qualify the Request Status.

Table 2.3: MM7 Submit Response Informational Elements [3]

2.5 The Subscriber Database

The Subscribers Database (SDB) is used to store information concerning subscribers handled by the system. For our implementation we decided to create our own simplified module to handle SDB tasks. It consists of a Mnesia table to store data and functions to manipulate and read the records. Below is a sample which shows the implementation of SDB records, where the status fields are used to indicate the user's active services. The different lists are used to store one or several addresses used as receivers of the messages created. The from email address is used as sender address when doing email copy.

In our FE the SDB is used by the Message Control module. The SDB is where the sender or receiver for an incoming messages will be looked up. Depending on which services are enabled new messages are then created. The following information is contained within a SDB record.

```
-record(sdb,
       {scr_addr,from_email_address,
        mo_email_status,mo_email_list=[],
        mt_email_status,mt_email_list=[],
        forward_status, forward_list=[],
        clone_status, clone_list=[],
        autoreply_status,autoreply_message})
```

2.6 The Message Transfer Store

The content within MMS messages needs to be removed and stored for retrieval at a later time in order and reattached to the messages which are generated by the supplementary services in the message broker module. When the front-end receives a message that has some content attached, the first step is to detach the content and instead include a content-id to the message. This content-id can later on be used to locate and access the content, this needs to be done before the message is leaving the front-end. The reason for doing this is to reduce the size of the messages that are passed around between the different processes in the FE. Figure 2.3 shows how messages are sent in parallel using a reference pointer to content.

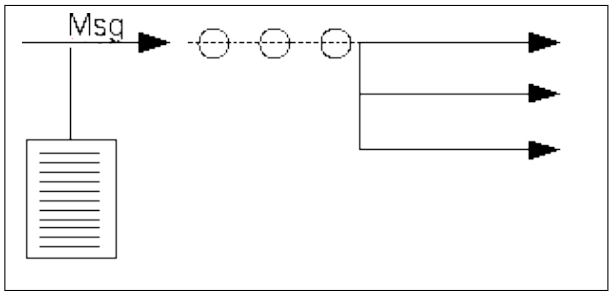

Figure 2.3: Message Transfer Store Synchronization

The MTS is built using the gen_server behavior and starts as a registered process (mts) when the mms_fe application is started. All operations to MTS are issued through `gen_server:call(mts,{<operation>, [value]})`. Content records are stored in Mnesia with a unique key to identify each record.

Usage

Each message has one record in the MTS database. The record is called mts and consists of three different parts

- cid - this is the content id, the unique key used to keep track of which content belongs to a specific message
- content - this is the content of the message, it can be the one big chunk of data, but it can also be a list of several data pieces that together form the entire content of a message
- count - this is the number of virtual copies of the content

Operations

These are the functions used to store and retrieve information using MTS.

store

`gen_server:call(mts, {store, Content})` will store the data in variable Content in MTS. If the operation is successful the response will be the tuple {ok, Cid} where Cid is the content id used to later request the content. Failure is indicated by the tuple {error, Reason} where reason will describe the reason for a failed store. (For example the disk is full).

get

`gen_server:call(mts, {get, Cid})` is the request to get content from MTS identified by the unique key in Cid. The response will be the tuple {ok, Content} where Content is the content, failure will be indicated by the tuple {error, content_missing}

delete

`gen_server:call(mts, {delete, Cid})` will delete one virtual copy of the content record identified by content id Cid. If there is only one virtual copy the entire content record will be removed. A successful request will result in ok, failure to delete will be indicated by the tuple {error, delete_failed}

copy

`gen_server:call(mts, {copy, Cid})` will create one virtual copy of the content record identified by Cid. The number of copies is indicated by the count field in the record. Return ok on success and the tuple {error, copy_failed} on failure

Virtual copies

For each new message that is created the MTS must make a virtual copy of the content, otherwise the risk is that the content will be removed from the database before each message has had time to add the content to the message. Figure 2.4 shows how several virtual copies are referencing the same content. A real copy (Rc) can be created if the content needs to be converted when creating another kind of message. The same function is used to delete a virtual copy as to delete the real content record. Each delete call issued will decrement the counter with one, when the counter reaches zero, there are no more copies, hence the entire content record is removed from the content database. It is very important to issue the calls in the correct order, i.e the different services must be activated before the original message is sent, since the sending part of the FE will delete one content copy, and initially this is the only copy. This could lead to content needed by other messages being lost. Because of this each new message created will have to request a copy of the content, to ensure that the content is available later on in the process.

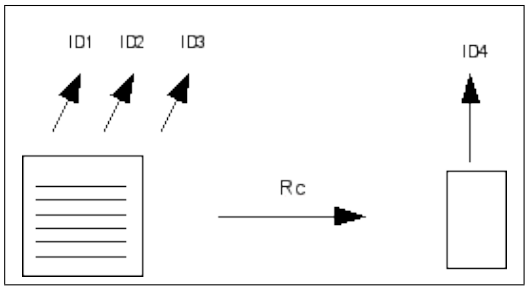

Figure 2.4: Virtual copies

2.7 The Message Handler

For MM1 messages we had problems getting a WAP gateway to work as intended. We also had some problems to get the message from the test phones that we were using. In order to get around this problem we decided to implement our FE with a message handling API, that would later on be called by the protocol handling functions. In this way we could isolate the internal workings of our FE, and let the other well defined parts of sending and receiving messages be handled by other mechanisms.

Message Type	Protocol
MM1	WAP and WSP
MM4	SMTP
MM7	HTTP POST

Table 2.4: Message Type and the corresponding protocol

We decided to go with this solution for MM1 messages. Moreover, we felt that we wanted to implement all parts in the same fashion, hence we did something we called handlers for the three protocols used, which expect something to handle the communication over WTP/WSP, SMTP and HTTP respectively. Our solution was to simply have separate ip addresses for each protocol, as shown in Table 2.4

Message Control

Message Control is the first part for handling message services. First a control is done to make sure the incoming message is of the right type (m-send.req, mm4_forward or mm7_submit). If the message pass this initial test the sending subscriber address is retrieved from the message and with this a request is made to the SDB to get the entire subscriber record for this user. Active services are queried and corresponding addresses are stored in the internal record that is passed on together with the message. Our FE also supports the possibility to suppress certain services on a per subscriber basis. One example might be a pre-paid customer who has activated some services, but has insufficient funds to send the number of messages configured. In this case one or more messages would be suppressed in order to stay within the limits of account balance.

Content Screen

Our FE features the possibility to screen message subject lines for certain words. We decided to only implement this small part as an example, since we had limited time for doing the implementation. However, there are many different parts of a message one might like to screen. For example, it would be possible to screen message text or attachments of different types for inappropriate, illegal or in other ways undesired message content.

In our FE we only screen MM4 and MM7 subjects for `autoreply:` and if found we remove this prefix from the subject and suppress auto-reply messages. This is to prevent auto-reply message loops that could occur if two parties are sending auto-reply messages to each other over and over. Since MM1 does not support auto-forward, no MM1 screening is done in our FE.

Message Broker

This module is where all active services will generate new messages. The message control step has previously figured out which message services are active and now it is just a matter of fetching the info from the internal record and create the new messages. For MM1 the only possibility is to create a e-mail

copy, while MM4 and MM7 have several different services that might need new messages to be created.

E-mail copy

The message subject and body is copied from the MMS message to a SMTP message. The sender and destination e-mail addresses were stored in the SDB, and hence located in the internal record. Content is collected from MTS and a message is sent.

Autoreply

A new message is created with the original sender address as recipient, original recipient as sender, subject is the original subject prefixed by "autoreply:". The autoreply message is stored in SDB and internal record.

Forward

The original message is copied to one or several new recipients from SDB. Subject is prefixed with "fwd:" Sender is changed to be the same as the recipient of the original message, for each recipient a attachment copy is made through MTS.

Clone

The original message is copied, the sender remains the same, subject is prefixed with "clone:" for each recipient a attachment copy is made through MTS. See Figure 2.4

Chapter 3

Conclusion

This project proved to be a very challenging and educational experience. At the beginning we were not sure whether we would be able to implement all of the specified features, and this proved to be the case. Because of a series of issues, primarily due to lack of access to a MMSC Relay/Server and time constraints we were not able to get a fully working prototype. We successfully implemented all the necessary modules according to the thesis specification.

- Protocol for MM1 to/from subscriber equipment
- MM4 to/from MMSC
- MM7 to/from MMS Value Added Service (VAS) Provider VASP
- MTS - Message Transfer Store
- SDB - Subscriber Database
- Message Broker
- Content Parser
- Message Control

Figure 3.1 represents an internal description of all the implemented modules. This figure also shows the abstract message flow though the FE.

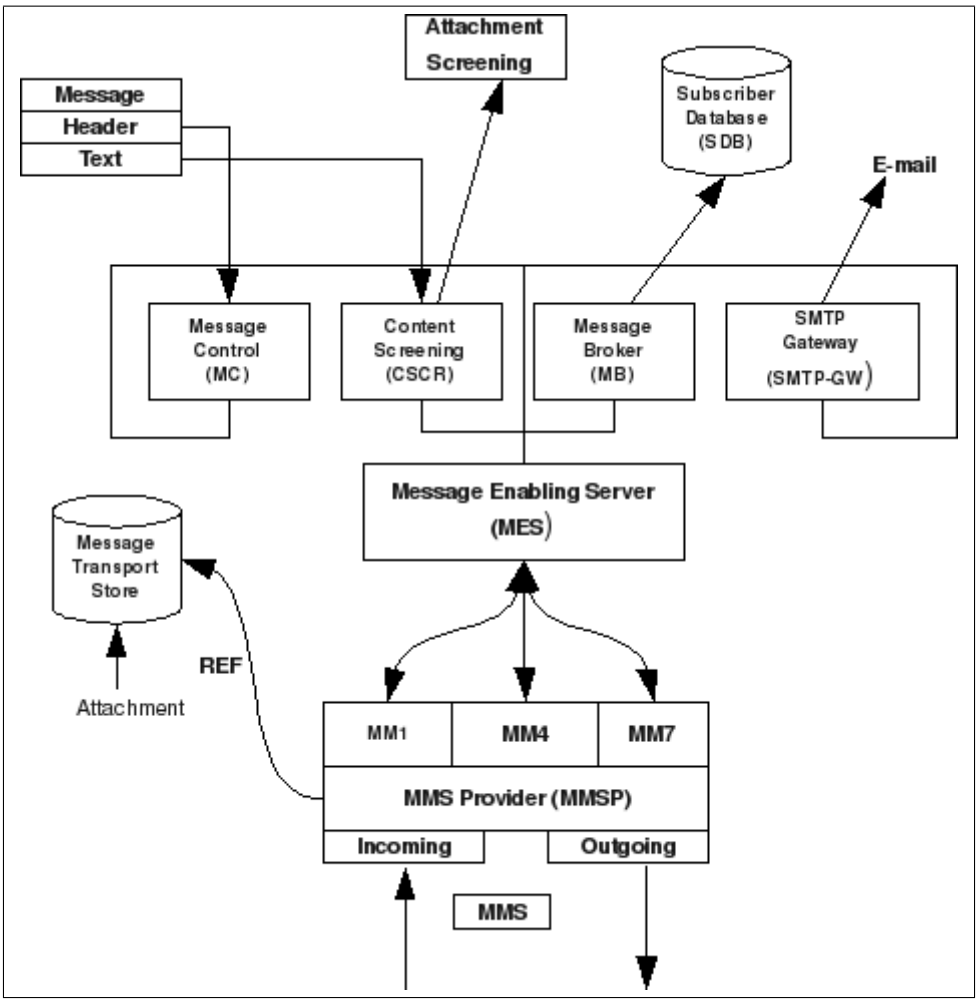

Figure 3.1: The Internal Modules

3.1 Problems

The main part of the project was the implementation and testing of the FE. In order to be able to send and receive messages we would need to add our node to a telecoms operator's mobile network. However, we were unable to accomplish this. Here are some of the other problems which prevented us from fully implementing the FE.

- We did some testing to receive MM1 messages sent from our test phones. We configured our phones to use the address of our very basic test server. We concluded that we managed to receive a message from the Samsung phone using Tele2. On the Sony Ericsson using Telenor we never managed to send a test message, the sending operation timed out. We never figured out if this was due to some problem in the phone or if the operator actively prohibited it. The small test to receive these test messages was very time consuming. We also spent a considerable amount of time analyzing data packets we received on the test server.

- We tried to find a stack to handle the WTP/WSP messages. When we could not find a working stack, this also prevented us from being able to fully test MM1 messaging. It should also be noted that implementing a MM1 stack was out of scope for our thesis and would have taken a considerable amount of time to do. Instead we took this part out and concentrated on handling MM1 messages internally without receiving them from a UE.

- To test MM4 and MM7 properly we needed access to an operators MMSC to which we could submit our messages. In order for the messages to be deliver or push a mms message to a UE we would also need access to an operator MMSC. Again we were unable to get such access. In an initial test conducted early in the project we set up a simple server which would act as a MMSC and forward a message (attempted only MM4) to an operators MMSC (we tried both Telenor and Tele2). After several attempts, each resulting in a timeout we came to the conclusion that an operators MMSC only accepts incoming traffic from approved servers.

- In order to get around the need for an operators MMSC we tried to use Mbuni, which is an open source MMSC. We attempted to get the MMSC to work, but due to lack of documentation and support we were unable to get it working properly. We then tried a commercial SMSC, NowSMS, which also supports MMS. NowSMS offered a 60 day test version, yet again the testing was too time consuming and in the end proved unsuccessful.

Although this part of the testing was not completely successful it still helped us in getting a understanding of the different protocols and nodes involved in MMS message communication.

Testing

Due to time restraints and other unforeseeable problems there was no extensive testing of the FE. Each module was tested during implementation and interacting modules were tested together as soon as they were implemented. For testing we created a series of handler modules see Figure 3.2. These modules performed as a simulated MMSC, which for our purposes simply received messages and forwarded them to the correct handler. We also tested the entire implementation, but unfortunately we did not have enough time to do any extensive testing.

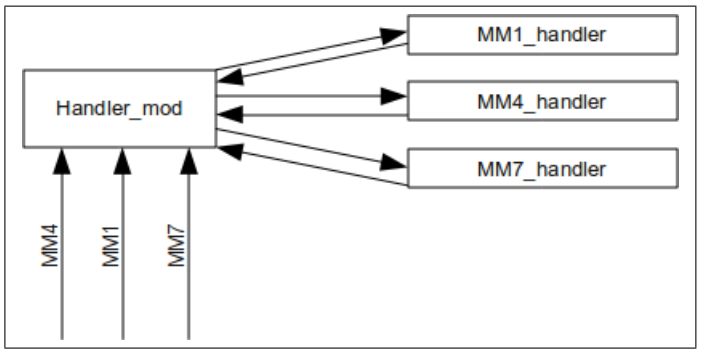

Figure 3.2: The Handler Module

3.2 Future work

Message content should be stored in some internal representation, such that it could easily be converted to suit any of the message types supported by the FE. This would help when we want to convert from one message to another, for example when we want to create a SMTP mail message from a MM1 originating message. Unfortunately, we did not have enough time to implement all of this functionality. We did implement a MTS record to keep track of different message parts, but we never got the time to implement the functions to encode content to the correct format. Instead, we just take the content and store it in MTS. We would at least need support for the following content conversions

- Convert and Forward an incoming Email message to a mobile subscriber
- MM4 to MM7
- Email copy of an incoming VAS message to a subscriber specific email address

Bibliography

[1] Armstrong, Joe. Programmering Erlang. Pragmatic Programmers, 2007.

[2] 3rd Generation Partnership Project. March 2006,
3GPP TS 22.140 V6.7.0 Group Services and Systems Aspects: Multimedia Messaging Service-Media formats and codecs April 11, 2010. 13:45 CET
<http://www.3gpp.org/ftp/Specs/html-info/22140.htm>

[3] 3rd Generation Partnership Project. September 2006
3GPP TS 23.140 V6.14.0 Multimedia Messaging Service: Functional description, April 11, 2010. 13:55 CET
<http://www.3gpp.org/ftp/Specs/html-info/23140.htm>

[4] 3rd Generation Partnership Project. March 2006
3GPP TS 26.140 TS 26.140 V6.3.0 Media Format and Codecs, April 11, 2010. 13:35 CET
<http://www.3gpp.org/ftp/Specs/html-info/26140.htm>

[5] RFC 821 Simple Mail Transfer Protocol,April 11, 2010. 16:35 CET
<http://www.faqs.org/rfcs/rfc821.html >

[6] Martin Cooper History of Cell Phone, March 11, 2010 16:00
<http://inventors.about.com/cs/inventorsalphabet/a/martin_cooper.htm>

[7] RFC 2822 -Internet Message Format , April 11, 2010. 11:35 CET
<http://www.faqs.org/rfcs/rfc2822.html>

[8] List of countries by number of mobile phones in use-Wikipedia,the free encyclopedia, April 16,2009. 11:00 CET
<http://en.wikipedia.org/wiki/List_of_countries_by_number_of_mobile_phones_in_use>

[9] textually.org:SMS a little history. April 11,2010 16:05 CET
<http://www.textually.org/textually/archives/cat_sms_a_little_history.htm>

[10] Europe mobile phone penetration map.png. April 11,2010 15:47 CET
<http://commons.wikimedia.org/wiki/File:Europe_mobile_phone_penetration_map.png>

[11] SMS - Wikipedia, the free encyclopedia. April 11, 2010, 16:05 CET
<http://en.wikipedia.org/wiki/SMS>

Appendix A

Figures

Figure A.1 represents the flow of traffic when a UA retrieves a message and how the MM4 delivery report is handled. Since we did not have access to any operators seves to conduct actual test. This figure is included as a reference to show where the FE should be placed.

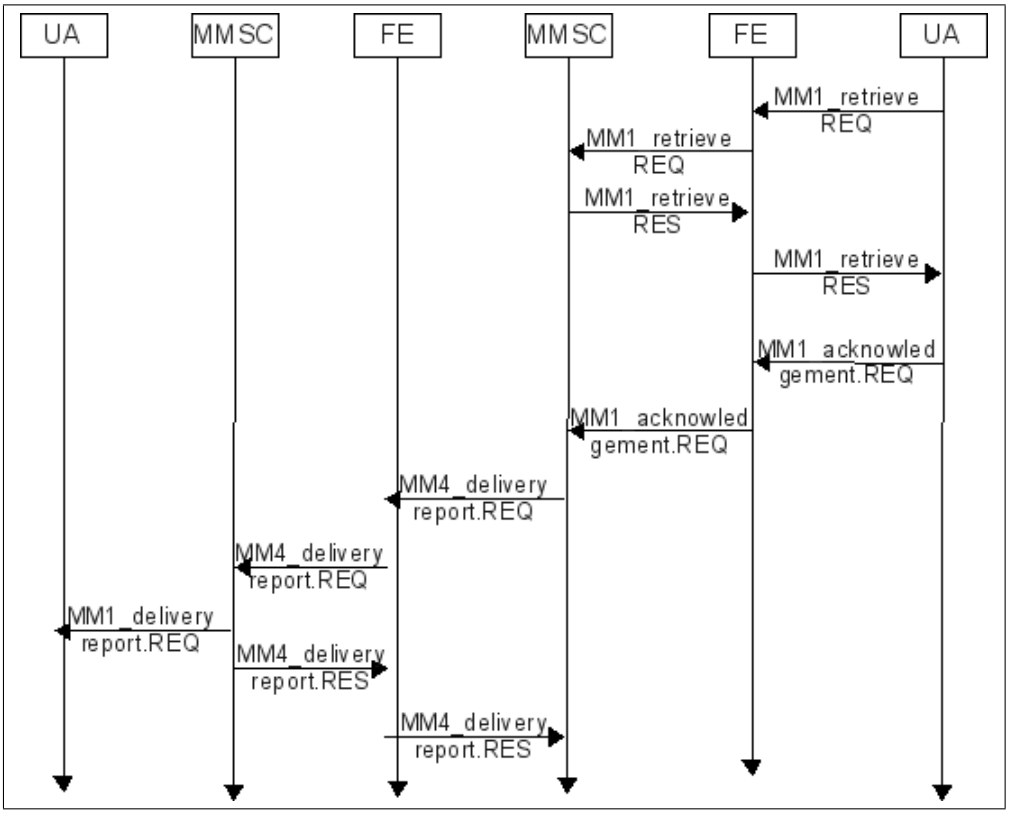

Figure A.1: MM1 & MM4 Data flow message retrieval and delivery report

Figure A.2 is another reference of traffic flow when sending and retreiving MM7 messages between a UA and a VASP

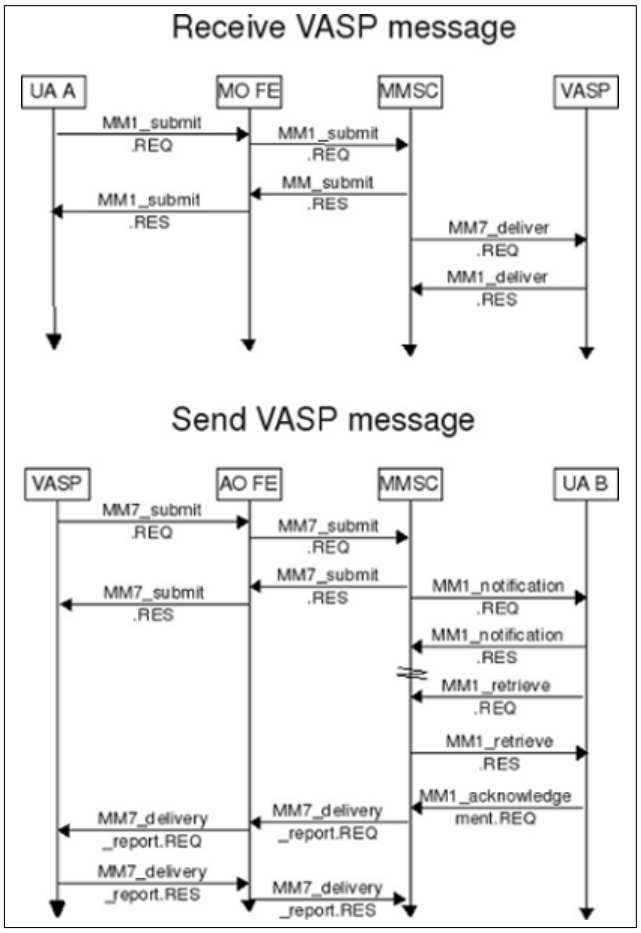

Figure A.2: MM1 & MM7 Abstract Data Flow, send and receive VASP messages

Figure A.3 shows the abstract network architecture and how different message types pass through it.

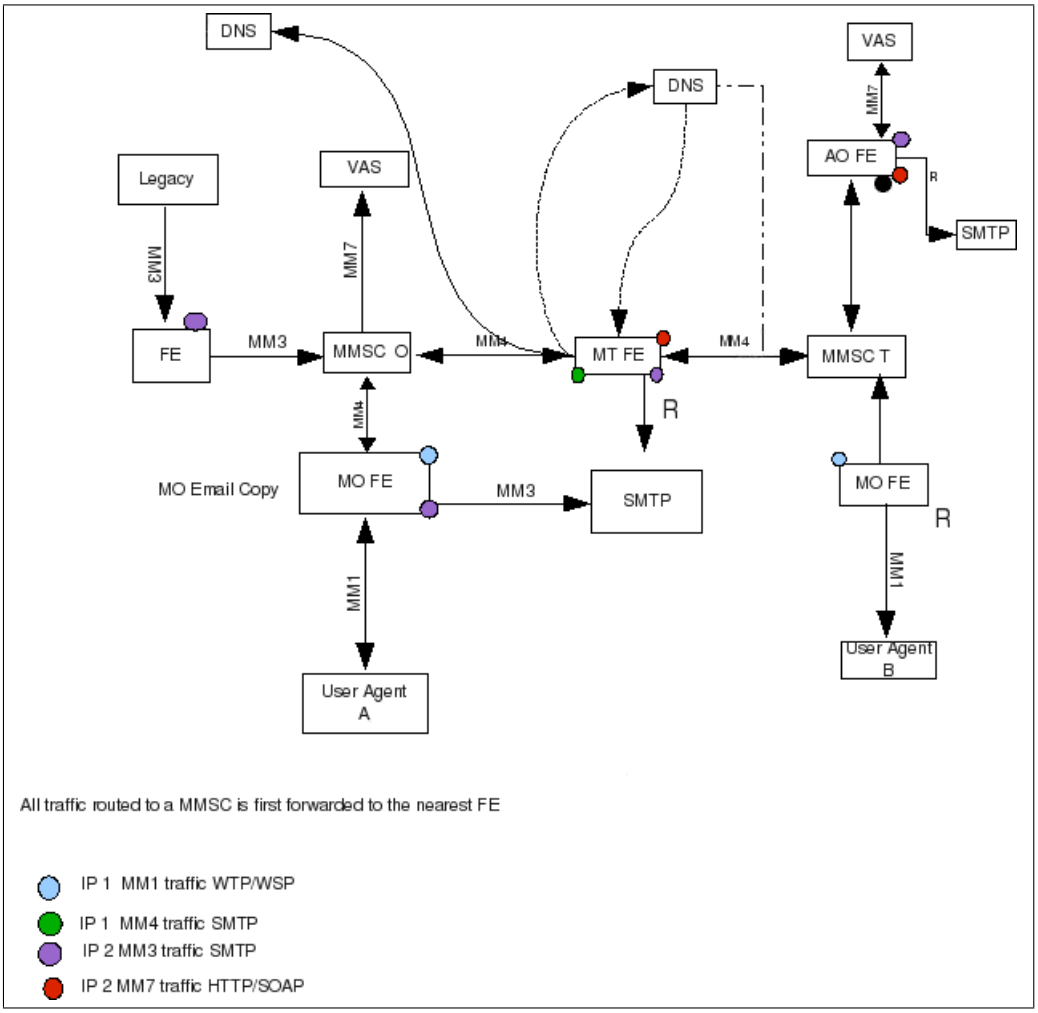

Figure A.3: FE Abstract Addressing

Appendix B

Sample messages

The following tables show all of the informational elements for MM1_submit.REQ and MM4_deliver_report.REQ. Table B.3 is a listing corresponding elements and where there are located within a SOAP message.

Table B.1: MM1 Informational Elements

Information Element	**Presence**	**Description**
Message Type	Mandatory	Identifies this message as MM1_submit.REQ
Transaction ID	Mandatory	The identification of the MM1_submit.REQ and MM1_submit.RES pair.
MMS Version	Mandatory	Identifies the version of the interface supported by the MMS UA.
Recipient address	Mandatory	The address of the recipient(s) of the MM. Multiple addresses are possible.
Content type	Mandatory	The content type of the MMs content.
Sender address	Optional	The address of the MM originator.
Message class	Optional	The class of the MM (e.g., personal, advertisement, information service)
Date and time	Optional	The time and date of the submission of the MM (time stamp).
Time of Expiry	Optional	The desired time of expiry for the MM or reply-MM (time stamp).
Earliest delivery time	Optional	The earliest desired time of delivery of the MM to the recipient (time stamp).
Delivery report	Optional	A request for delivery report.
Reply-Charging	Optional	A request for reply-charging.
Reply-Deadline	Optional	In case of reply-charging the latest time of submission of replies granted to the recipient(s) (time stamp).
Reply-Charging-Size	Optional	In case of reply-charging the maximum size for reply-MM(s) granted to the recipient(s).
Priority	Optional	The priority (importance) of the message.
Sender visibility	Optional	A request to show or hide the sender's identity when the message is delivered to the recipient.
Store	Optional	A request to store a copy of the MM into the users MMBox, in addition to the normal delivery of the MM.
MM State	Optional	The value to set in the MM State information element of the stored MM, if Store is present.
MM Flags	Optional	One or more MM Flag keywords to set in the MM Flags information element of the stored MM, if Store is present
Read reply	Optional	A request for read reply report.
Subject	Optional	The title of the whole multimedia message.
Reply-Charging-ID	Optional	In case of reply-charging when the reply-MM is submitted within the MM1_submit.REQ this is the identification of the original MM that is replied to.
Content	Optional	The content of the multimedia message

Table B.2: MM4 Informational Elements

Information Element	**Presence**	**Description**
3GPP MMS Version	Mandatory	The MMS version of the recipient MMS Relay/Server as defined by the present document.
Message Type	Mandatory	The type of message used on reference point MM4: MM4_delivery_report.REQ.
Transaction ID	Mandatory	The identification of the MM4_delivery_report.REQ/ MM4_delivery_report.RES pair.
Message ID	Mandatory	The identification of the original MM.
Recipient address	Mandatory	The address of the MM recipient of the original MM.
Sender address	Mandatory	The address of the MM originator of the original MM.
Date and time	Mandatory	Date and time the MM was handled (retrieved, expired, rejected, etc.) (time stamp).
Acknowledgment Request	Optional	Request for MM4_delivery_report.RES
Forward to Originator UA	Optional	If No, indicates that the originator MMS Relay/Server is not allowed to forward the Delivery Report to the originator MMS User Agent.Interpret as Yes in the absence of this Information element.
MM Status	Mandatory	Status of the MM, e.g. retrieved, expired, rejected
MM Status Extension	Optional	Extension of the MM Status, to provide more granularity.
MM Status text	Optional	Status text corresponding to the MM Status

Table B.3: MM7 Informational Elements

Information Element	**Location**	**ElementName**
Transaction ID	SOAP Header	TransactionID
Message-Type	SOAP Body	MessageType
MM7 Version	SOAP Body	MM7Version
VASP ID	SOAP Body	VASPID
VAS ID	SOAP Body	VASID
Sender Address	SOAP Body	SenderAddress
Recipient Address	SOAP Body	Recipients
Service code	SOAP Body	ServiceCode
Linked ID	SOAP Body	LinkedID
Message class	SOAP Body	MessageClass
Date and time	SOAP Body	TimeStamp
Time of Expiry	SOAP Body	ExpiryDate
Earliest delivery time	SOAP Body	EarliestDeliveryTime
Delivery report	SOAP Body	DeliveryReport
Read reply	SOAP Body	ReadReply
Reply-Charging	SOAP Body	ReplyCharging
Reply-Deadline	SOAP Body	replyDeadline
Reply-Charging-Size	SOAP Body	replyChargingSize
Priority	SOAP Body	Priority
Subject	SOAP Body	Subject
Adaptations	SOAP Body	allowAdaptations
Charged Party	SOAP Body	ChargedParty
Message Distribution Indicator	SOAP Body	DistributionIndicator
Content type	MIME header Attachment	Content-Type
Content	SOAP Body	Content

Appendix C

Requirement Specification

This is the requirement specification that we wrote after meeting and discussing our thesis with our supervisor at Mobile Arts. It includes both required and desired parts.

C.1 Required Parts

R-1. Message Transport Store

The message transport will store multimedia attachment e.g movie clips, images, audio This will decrease the load on the system while the message is passed through Message Enabling Server

- Create a unique identifier for referencing the multimedia content of the MMS
- Store the multimedia content in the MTS
- It shall be possible to retrieve the multimedia content
- It shall be possible to delete records

R-2. Message Control

Checks the headers of the message for type and handles it accordingly

- Checks the MMS for the correct format
- Keeps track of read Reply and delivery receipt requests
- Place attachments in MTS (Message Transport Store)
- Add reference to the message so later instances can retrieve the attachment belonging to this message If the incoming message is a return receipt or a read reply it is then it is sent to the originating MMSC for delivery to the originating UA

R-3. Message Broker

- Look up terminating UA address using DNS queries
- Check which services are available for UA
- Create Clone/Forward/Auto Replay/Email copy
- For email copy send to SMTP Gateway
- For Forward change recipient
- For Clone created new Message
- For Auto Reply send Auto Reply to sender UA

R-4. SMTP Gateway

- Convert MMS to email
- Get Attachment from MTS
- Get Headers from MMS
- Create email conforming to SMTP message standards
- Send to SMTP server

R-5. Subscriber Database

The SDB shall support MMS by having the following records

- One field for email address (Email Copy MO)
- One field for email address (Email Copy MT)
- Field /Fields for phone numbers (MMS Forwarding MT)
- Field /Fields for phone numbers (MMS Cloning MT)
- Field to indicate MMS auto reply (MMS Auto Reply)

R-6. MMS Forward

- Retrieve MMS forward field from SDB
- Change recipient of the message to the recipient in the MMS forward field
- Send the message to the new recipient

R-7. MMS Clone

- Retrieve MMS clone field/s from SDB
- Create n copies of the original messages with new recipient addresses
- Send original and cloned messages

R-8 Simulating

Simulating shall be done by using regular phones set to use the MMS FE as message server. MMSC, DNS and SMTP has to be in sourced.

- MM1 simulating shall be done using actual mobile phones, but to save costs we should implement a MM1 message generator that will make it possible to measure MM1 throughput
- MM3 simulating shall be done by setting up a SMTP server to generate traffic
- MM4 simulating shall be done using an existing MMSC server
- MM7 Might have to be written from scratch or in-sourced from an existing MM7 module

C.2 Desired Parts

D-1. Email Copy from Mobile Originating (MM1→MM3)

- Convert incoming MMS to SMTP standard format
- Add attachment to email
- Look up email address in SDB
- Add email recipient
- Connect to SMTP Gateway via MM3 and send email
- Forward MMS message to MMSC O

D-2. Email Copy from Mobile Terminating (MM4→MM3)

- Convert incoming MMS to SMTP standard format
- Add attachment to email
- Look up email address in SDB
- Add email recipient
- Connect to SMTP Gateway via MM3 and send email
- Forward MMS message to MMSC O

D-3. Content Screening

- Subscriber Blocking of MMS from certain phone numbers
- Subscriber Blocking of MMS attachments exceeding a specified size

D-4. Message Management

Allows the user Agent to update services on the Subscriber Database (SDB)

- Add support for new services for handling MMS messages e.g (clone, forward, auto reply, email copy)
- Update SDB with new settings
- Send notice to user agent that service has been activated

D-5. Message Management (MM)

- It is desirable that a subscriber can update records in the SDB using MMS
- For specification for which records should be supported see R-5