An MMST-Integrated MMS Editor

Petra Winberg

April 27, 2006
Master's Thesis in Computing Science, 20 credits
Supervisor at CS-UmU: Lars-Erik Janlert
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

This report describes a master's thesis performed at Obigo AB in Umeå between November 2005 and April 2006. The main purpose with the project was to create an MMS editor and to integrate it with the company's MMST. The editor was supposed to be a tool for creating test messages to be used in the development of MMS applications.

The created editor provides a usable interface, that makes it possible to use by both advanced and novice users. The editor has a core set of functionality implemented, but the structure allows more advanced functionalities to be added in an easy way.

A protocol in XML has been deigned to solve the communication between the editor, which is implemented in Java, and the MMST, which is an Apache Web Server.

The report also provides an in-depth study of the MMS. It provides background knowledge of the service, together with a prediction of the future for it. The service is estimated to continue to gain ground and to become the leading messaging service by 2008. This is probably possible if the problems associated with it are solved. Some problems that it neads to deal with are transmission delays, compability between content classes and added value for the end user.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The work with this master's thesis was performed at Obigo AB in Umeå between November 2005 and April 2006. Obigo is a company within the Teleca group.

1.1 Background

MMS (Multimedia Messaging Service) is one of the fastest growing services of today. It is a service that allows users to interchange messages containing text, sound, image and video content. The service was introduced to the market in 2002, and has grown at a rapid rate since then.

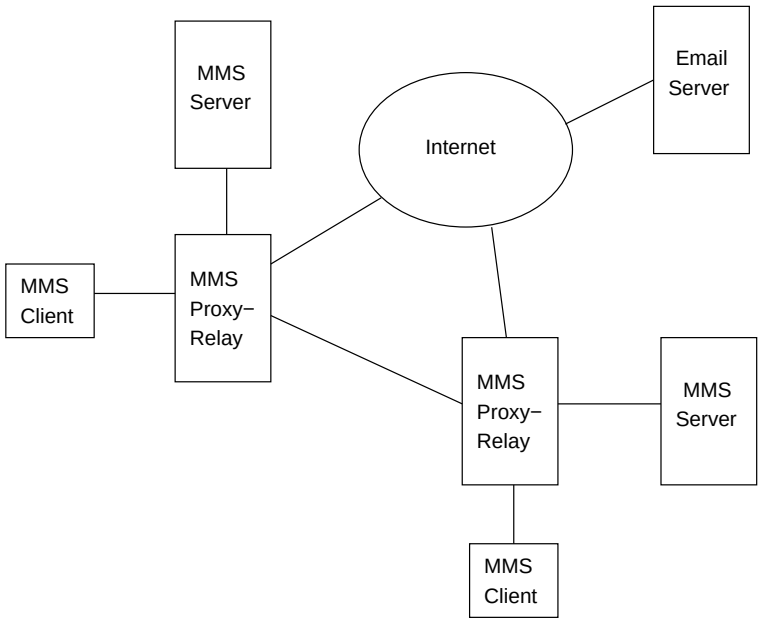

Figure 1.1: Exchanging multimedia messages

The interchanging of MMs (Multimedia Messages) is performed in the way displayed

in figure 1.1. The figure contains the following elements[2].

- **MMS Client** The device that the user interacts with, eg a mobile phone.
- **MMS Proxy-Relay** The server that the MMS Client interacts with. The Proxy-Relay is responsible for handling send and retrieve requests from the MMS Client.
- **MMS Server** The server that provides persistent storage of messages.
- **Email Server** The server that provides traditional email services.

The MMS Server and the MMS Proxy-Relay are often combined. The server is then referred to as an MMSC (Multimedia Messaging Service Center).

In the development of MMS applications, a large amount of test data is required. The testing is mainly performed using dedicated MMs. Since new features are added to the service fairly often, new MMs are often required. During the development, the test data needs to be stored at a location accessible for all parts. That is achieved by storing the test data at the MMSC.

Since new messages often are needed, there exists a desire for a program that provides the possibility to create them in an easy way. In this thesis, an MMS editor has been implemented and integrated with an MMSC. The editor provides a usable interface, in combination with the possibilities to store and send messages.

1.2 Layout of the thesis

Chapter 2 contains the problem description. It gives background information and guidelines for the task that has been done. It also describes the methods that was planned to be used during the work.

In chapter 3, the theoretical part of the thesis is presented. It contains an in-depth study of the service. Problems associated with the service are presented, together with proposed solutions.

Chapter 4 describes the implementation part of the thesis. It presents the different techniques that have been used to create the system.

Chapter 5 presents the results of the thesis. A description of the system together with a presentation of the GUI is given.

In chapter 6, conclusions and limitations are discussed. Some ideas of future work are given.

Chapter 7 contains acknowledgments to the people that have been of help during the work with this master's thesis.

Chapter 2

Problem Description

Creating MMS applications for mobile phones is an extensive task. New features are constantly added. At this point in time, features like postcard, delivery and read reports, digital rights management, immediate and deferred retrieval are supported[3]. New features are added as new standards for MMS are released. This indicates that severe changes often are required for the MMS application to be compliant with new standards.

As for all software developed, thorough testing is required to ensure the quality of the MMS application. Since it is a messaging service, testing is mainly performed by sending and receiving messages.

The communication between the MMS Client and the MMSC (see figure 1.1) is normally done in the manner presented in figure 2.1. That is, the MMS Client communicates with the MMS Proxy-Relay through a Gateway. The transport protocol between the MMS Client and the Gateway is WSP (Wireless Session Protocol). The Gateway and the MMS Proxy-Relay communicate using HTTP (Hypertext Transfer Protocol).

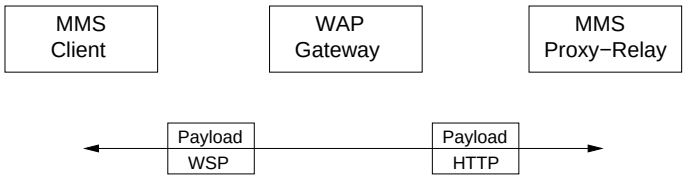

Figure 2.1: Communication with MMSC using WSP

Communicating with the MMSC using WSP can be slow when there is much traffic on the net. Because of this, another way of communication is often used while testing. Then, an MMST is used instead of an MMSC. This server is an MMSC that uses TCP as transport protocol instead of WSP. The communication is illustrated in figure 2.2.

For quality assurance issues regarding the MMS application, a large amount of test content is needed. Since the development is performed in an iterative way, test contents are often reused. After adding new features, it is necessary to ensure that old functionality still works properly. To simplify the reuse of MMs, a web interface is connected to the MMST, which is used to manage stored MMs. This way, everyone involved in the development process can access the same content.

Since functionality is added to the MMS application in a rapid way, new test contents are constantly needed. For instance, the maximum allowed message size has been

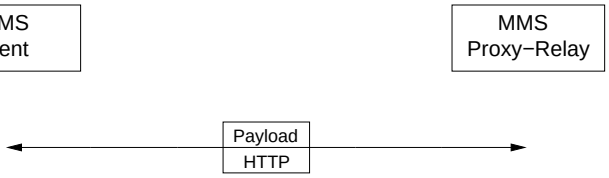

Figure 2.2: Communication with MMSC using HTTP

increased after nearly every released MMS standard. An effect of this is that test contents used to verify the handling of maximum sized messages often need to be altered. This indicates that there is a desire for a way to create new MMs easily and quickly.

To summarize, a good and usable MMS editor is needed. The editor should preferably be integrated with the MMST. This way, the messages present at the MMST could be used as a basis when creating new messages. Created messages can that way also be stored directly to the MMST. The editor could then also make use of functionality already implemented at the MMST.

2.1 Problem Statement

Previously, Obigo basically had two ways of creating new test data. The first variant was to use an MMS creator located at their MMST. This creator was very basic. It had a small set of contents that could be added to an MM. It did not provide the possibility to edit all headers in the MM.

The second variant was to use a protocol based program that had been developed within the Teleca group. This program provided the possibility to edit all parts of the MM. However, to be able to use it, quite a lot of knowledge was required.

As one can imagine, none of these variants was particularly good in the long run. It would be better to have a program with a usable interface. So, this is why this thesis had been proposed. The aim was to create such a system for the company.

Another office within the Teleca group had developed a variant of an MMS editor. This editor could possibly be used as a basis in this thesis. If the editor was suitable for this purpose and if the internal policy allowed it, it would be made use of. This way the basic functionality was already implemented, and the work could be focused on extending the editor with more advanced features.

2.2 Goals

This master's thesis consisted of two parts, one theoretical and one practical. The theoretical part contained an in-depth study of MMS and what the future will hold for it. Some of the questions that the study would try to answer were the following.

- How is an MM structured?
- How is the service standardized?
- What changes are made to the message structure with new standards?
- What problems are associated with the service today?

– What new problems may be introduced in the future?

– How could these problems be overcome?

The conclusions of the study would hopefully provide a basis for the design of the MMS editor. If the editor was designed in a way that made it easy to alter to comply with new standards, it would be more useful in the long run.

The practical part involved creating the MMS editor and integrating it with the Obigo MMST. The part was broken down into the following two main tasks.

– Expose an API in the existing MMST.

– Implement the MMS editor.

2.3 Purposes

The purpose with this master's thesis was to create an MMS editor integrated with the Obigo MMST. To perform the work, the following issues had to be considered.

– How to achieve simplicity and flexibility for end users.

– How to design the editor in a way that made it possible to, with small adjustments, prepare it for new MMS standards.

– How to make use of existing functionality at the MMST as much as possible.

– How to handle the documentation, to simplify updating the editor.

2.4 Methods

To be able to perform this task, a lot of studying had to be done to gain the background knowledge that was required. Therefore, the first period of the work was to be dedicated to this task. The areas that had to be studied were the following.

– **The Obigo MMST**

  • What functionality exists?
  • How should the API be designed?
  • How should the editor communicate with it?

– **The MMS structure**

  • How is an MM structured?
  • What parts are necessary/optional?

To determine the functionality of the MMS editor, the end users would be taken into account. The plan was to send a survey to them regarding what functionality they desired. The result from the survey would then be used as a basis for specifying the requirements for the editor.

The implementation strategy was depending on the decision made regarding the existing editor. If it was to be used as a basis, the starting phase of the implementation would be quicker. If it was not be used, a basic editor needed to be implemented.

In each case, the implementation of the functionality was to be performed iteratively. Each iteration would add functionality to the editor. A core set of functionality would be implemented at the first iteration. However, quite a lot of the system needed to be implemented before the entire program flow could be tested. The initial part of the implementation would therefore be performed according to the following plan.

1. Implement an MMS editor with basic functionality, if the existing editor was not to be used.
2. Create an API at the MMST.
3. Design a protocol for the communication between the editor and the MMST.

These steps would set up the communication between the parts. After this had been done, more advanced functionality would be added in iterations.

The design of the user interface would be performed using a review group. The review group was to be made up by both developers and testers, to get a broad perspective.

The entire implementation was planned to be done with continuous feedback from future users, to ensure that the end product would be satisfactory.

Chapter 3

An in-depth study of MMS

MMS is one of the fastest growing messaging services of today. In 2004, the compound annual growing rate was estimated to be 82.2 percent[14]. This chapter will provide a deeper look at this service. The first parts of this chapter will be dedicated to provide some background knowledge, that will be a help in understanding the rest of the discussion. Short introductions to the following areas will be given.

- The history of MMS.
- How standardization of the service is being performed.
- The structure of an MM.
- The way in which new features are added to the service.

After the introduction, a deeper discussion will be made regarding the future of the service. During the past decade, innumerable new services have been released on the world market. But not all of them continue being successful. The discussion will concern how the service should cope with all the competition and problems on the market. The chapter will try to provide an answer to the following main questions.

- What problems are associated with MMS today and what problems may arise in the future?
- How could these problems be solved?
- What changes are required to solve the problems from a technical point of view?

Hopefully, this chapter will provide some useful knowledge of the service, and hence also be a good foundation for making decisions regarding the design of the system.

3.1 Background of MMS

MMS is a continuation of some earlier released messaging services. Some services that can be mentioned are the SMS (Short Messaging Service), the EMS (Enhanced Messaging Service) and the Internet mail[5]. The SMS service has been a huge success since the introduction in 1992. The service was initially designed to handle short text messages

with a maximum of 160 simple characters[10]. It has later on been enhanced to be able to send larger messages. This has been realized by concatenating several short messages into larger ones[5].

The EMS service is an extension of SMS that allows users to add rich media, such as polyphonic melodies, simple black and white images or animations to messages[10]. The service can be seen as a light version of the MMS. Despite the improvements compared to SMS, EMS has not been the success that was expected.

In 2002, MMS was introduced to the market[10]. At early stages, there were not many mobile terminals available that handled the service. This fact in combination with various interoperability problems between mobile operators, caused the service not to become widely spread until in 2003[10]. The first version of MMS allowed users to interchange messages containing text, images and sounds[5]. Since this initial release, the service has been modified and improved several times. Features like, for instance, delivery and read reports have been added together with the possibility to add video clips[5]. The maximum size of an MM has increased several times during the past years. As of 2005, the maximum size is set to 300 kB[3].

MMS is, as earlier mentioned, growing at a rapid rate. If the service continues to grow at this pace, analyzers believe that it will replace SMS as the leading messaging service by 2008[14].

3.2 Standardization of MMS

Since the mobile market is a multi-vendor environment, the problem with interoperability early became an issue for MMS. At the initial stage of the service, there existed no widely accepted protocol for the communication. Operators within the field feared that users might become discouraged to use the service if it was not working properly.

To achieve interoperability, several associations began working for a standardization. Some of the more well-known organizations that have been involved in the standardization process are the 3GPP (Third Generation Partnership Project) and the WAP (Wireless Application Protocol) Forum[5]. In 2002, 200 operators within the mobile market created OMA (Open Mobile Alliance)[1]. This organization later incorporated the WAP Forum.

There are also other organizations working with standardization issues. For instance, the 3GPP2 (Third Generation Partnership Project 2) produces specifications for the North American and Asian markets with focus on CDMA technology[5]. The issues from the European perspective, though, is based on specification produced by OMA. Therefore, specification other than that coming from OMA will not be covered in this paper.

3.2.1 OMA's standards for MMS

The release of a new MMS standard is initiated by the delivery of a set of technical requirements from 3GPP[5]. These requirements define the functionality and features that shall exist, but they do not specify how this should be solved technically. This task is given to the OMA. The OMA takes the requirements from 3GPP and determines the way they should be implemented. That is, they decide on the protocol to use for the communication. The result is then released as a set of technical specifications from OMA at two stages[1].

– **Phase 1 (Candidate Enabler Release)**. This release contains an approved set of technical specifications, that can be used in implementation and for interoperability testing.

– **Phase 2 (Approved Enabler Release)**. This release has undergone a 30 days period of public review and has passed interoperability validation.

As of the time of writing, three approved enabler releases exist: version 1.0, 1.1 and 1.2[1]. Version 1.3 is at the moment a candidate enabler release. The approved version is estimated to arrive during 2006.

3.3 The MMS presentation format

An MMS can be presented in a couple of different ways. The most common variant, and the only presentation model that will be discussed here, is SMIL (Synchronized Multimedia Integration Language). SMIL is a presentational language based on XML that was published by W3C[5]. The SMIL language allows handling of timing, synchronization and preferred layout between operators. Different versions of SMIL have been released during the years. The most recent one is SMIL 2.0.

There exists a large amount of features attached to SMIL. Depending on what device that is used, a subset or all of the features may be supported. Mobile devices often have small displays in combination with a limited amount of memory. A consequence of this is that the devices have problems with handling all of the SMIL features. Therefore, a limited SMIL profile was created, known as SMIL MMS[5]. The standardization of the SMIL MMS language is handled by the OMA. A requirement to be conformant to OMAs MMS standards is that each MMS contains a SMIL presentation[3].

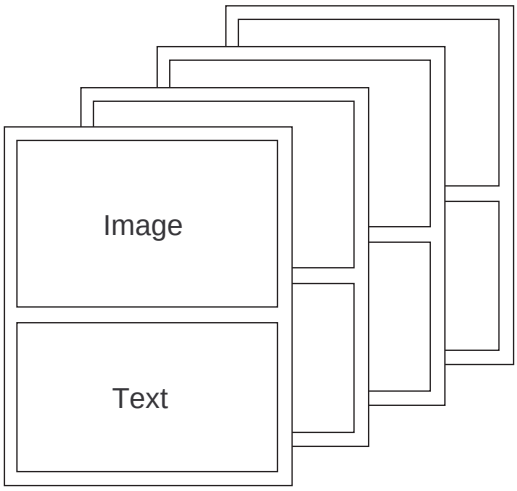

Figure 3.1: SMIL layout in an MMS

An MMS compliant with OMAs standards must consist of a slide show[3]. Each slide should be made up by at most two regions. One of the regions is supposed to hold text,

and the other either an image or a video clip. Hence, no slide can hold both image and video content. A slide can also contain a sound content. Since sounds do not need to be displayed visually, they will not be connected to a region. It is however not allowed to have both a video clip and a sound on the same slide even if the video does not contain an audio part[3].

The regions used within MMS applications are labeled "Image" and "Text"[3]. The regions are shown in figure 3.1. The two regions can be arranged in any way convenient for the device using it. If, for instance, the device is built with a landscape layout of the display, the most suitable way would be to place the two regions beside each other. All the slides in the message do however have to use the same layout[3].

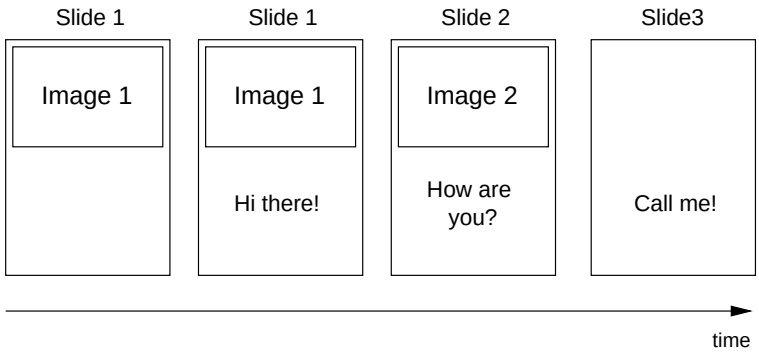

Figure 3.2: Timing functionality in SMIL

SMIL also supports timing settings for contents attached to the presentation. This basically means that it is possible to set starting and ending times for the contents. Figure 3.2 is an example of this. There, the first slide contains contents with individually set start times. The image starts playing instantly, but the text is added after a while.

3.4 The MMS structure

An MMS is mainly made up by two parts, the message header and the message body, as shown in figure 3.3[10]. The message header contains information about the message in general. It specifies, for instance, message recipients, message originator and subject. The message header also contains more advanced information. As more features have been added to MMS, more header fields have been added to support them. As one might understand, all of the header fields are seldom used at the same time. Therefore, the fields are divided into the following three categories[3].

- **Mandatory** The mandatory fields have to be filled in for all messages. This category contains fields like From, MMS-Version and Content-Type.
- **Conditional** The conditional fields may have to be filled in. To, Cc and Bcc are examples of fields that apply to this category. They are conditional since at least one of them need to be filled in.
- **Optional** The optional fields do not have to be filled in. These fields are used to control extra features to the service. For instance, Read Report and Delivery Report are optional fields.

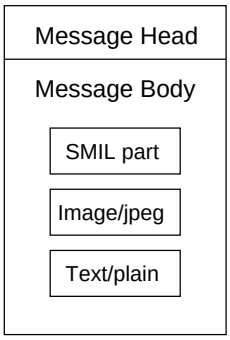

Figure 3.3: Message structure for an MM

A description of some of the more common header fields in version 1.3 of OMA's MMS standard is included in Appendix C.

The message body contains all the contents added to the message and a SMIL presentation part.

3.5 MM content classes

Messages in the MMS application are divided into content classes. In version 1.3 of OMA's MMS Standard, there are six content classes: Text, Image Basic, Image Rich, Video Basic, Video Rich and Megapixel[4]. These content classes are displayed in table 3.1.

The content classes are used to provide a help in interoperability issues. To be conformant to OMA's standards for MMS, at least Text and one of the other content classes need to be supported[4].

3.6 Problems for MMS today and in the future

MMS does, as all services, encounter problems in the quickly changing market of today. To be able to survive and not be replaced by a new service, it needs to be aware of changes at the market and be quick to evolve. This section will highlight some of the problems that are present today and that might arise in the future.

3.6.1 Not enough value added from earlier services

One big weakness for MMS is that it basically can be seen as an improved variant of the SMS[6]. It is basically a service that offers the possibility to create a message that will be played for the recipient. Of course, it has some more advanced functions compared to the SMS, but nothing that offers easier interaction between sender and recipient. To reply on a message, one must create a new MM, add the content you want, and send it. This is a time consuming task, that the service may not benefit from in the long run.

Content Class	Size (kB)	Image	Image Resolution	Sound	Video
Text	≤30	N/A	N/A	N/A	N/A
Image Basic	≤30	JPEG, GIF87a, GIF89a, WBMP	≤160*120	AMR	N/A
Image Rich	≤100	JPEG, GIF87a, GIF89a, WBMP	≤640*480	AMR SP-MIDI	N/A
Video Basic	≤100	JPEG, GIF87a, GIF89a, WBMP	≤640*480	AMR, SP-MIDI	H.263 with AMR
Video Rich	≤300	JPEG, GIF87a, GIF89a, WBMP	≤640*480	AMR, SP-MIDI	H.263 with AMR
Megapixel	≤600	JPEG, GIF87a, GIF89a, WBMP	≤1600*1200	AMR, SP-MIDI	H.263 with AMR

Table 3.1: MM content classes

3.6.2 Transmission problems

Another problem for the service comes directly from the fact, that the service has been improved a lot since its introduction. Today, it is possible to send large MMs over the net. This way, if the service continues to grow at the same pace, the service may encounter problems with severe delays while transmitting messages over the net[14]. Delays and failures during transmission is a problem that might make users turn away from the service.

3.6.3 Compatibility between MMS standards

The pace of which MMS is evolving at does also give the service other problems. The number of released MMS standards are a problem. Mobile phones of today need to be backwards compatible, which means that they need to be able to receive and handle all kinds of messages. Handling older types of messages may not be that big of a problem, since no header fields generally are removed from a new MMS standard.

Phones using an older version of the MMS standard may encounter more problems, though. OMA state that unknown header fields in an MM should be discarded[3]. This solves the problem with new features that are added to newer standards. They are simply ignored. However, the problem will become more obvious regarding the message size and content classes. Newer standards allow MMs to be larger, which will cause problems when transmitting such messages to a phone using an older standard. When such a phone receives a large message, it will notice that the message is larger than the

allowed size. In most cases (there are exceptions) the user is not allowed to download an MM that is larger than the allowed maximum message size[3]. This would, of course, be annoying for the user.

3.6.4 Compatibility between content classes

As earlier mentioned, there exist several content classes for MMS. This indicates that problems will occur when a user tries to send an MM of a content class that the receiving phone can not handle.

Since operators only need to support a minimum of two content classes, this indicates that a lot of problems may arise. Interoperability is only guaranteed within content classes, not between them.

3.6.5 Entertainment services sensitive to delays

The main usage of MMS today is for P2P (Peer-To-Peer) messaging, eg a user creates an MM and sends it to a recipient[14]. However, the other type of MMS usage, the entertainment services, are growing strongly. This type of messaging, that also is called A2P (Application-To-Peer) messaging, is estimated to have a compound annual growing rate of 96,5 percent[14]. This can naturally cause problems in the future. These services are often dependent of the sending to be performed in real-time. As one might understand, no one is willing to pay for live updates from a game, unless the updates arrive instantly. This indicates that solving the network issues is crucial to make this type of service survive.

3.7 Possible solutions to the problems

This section will provide some ideas on how to solve the problems. Naturally, there are numerous variants, and this paper can not cover them all.

3.7.1 Separating P2P and A2P traffic

As mentioned in the last section, problems might arise when A2P traffic need to be sent through the same channels as P2P traffic. The P2P messages are often big, and will become bigger and bigger as new MMS standards are released.

To solve the problem with providing a proper A2P service, some experts suggest that the two service should be transmitted over different channels[14]. This way, the A2P messages would not be delayed by the P2P messages.

3.7.2 Content adaptation

To solve the problem with interoperability between the content classes, a technique called content adaptation could be used[3]. Content adaptation means that content that can not be handled by an MMS Client is altered into a format that is supported. The changes that have to be made can be, for instance, to resize the content, convert between media formats or, if nothing else can be done, to remove the content. The content adaptation can be divided into the following two categories[3].

- **Minor Adaptation** - The adaptation is mainly focused on adapting message size, image resolution and the quality of sounds and videos.

- **Major Adaptation** - More drastic adaptations are performed, like for instance conversions that will cause losses of quality.

Content adaptation is, as mentioned, a way to bridge the gap between the content classes. This is performed by the MMS Proxy-Relay[3]. The technique is used to some extent today. MMS Proxy-Relays that support content adaptation can offer this service.

To be able to perform the adaptation, the server needs some information from the recipient. It needs to know the capabilities of the MMS Client, and what content classes it conforms to. This information is encapsulated in a UAProf (User Agent Profile)[3]. This basically means that the server notifies the recipient that it has a message waiting, the recipient attaches its UAProf in the answer, the server performs the necessary adaptations and then sends the message to the client.

For an MMS Proxy-Relay to be conformant to content adaptation according to OMA, the following requirements need to be fulfilled[3].

- It must support UAProf.
- It must be able to perform minor adaptation.
- If it is able to perform major adaptation, this feature must be possible to disable.
- The original content (eg the content before adaptation) should be available to the end user.

As the content classes increase in number, it is obvious that the desire for content adaptation will become bigger. This indicates that MMS Proxy-Relays will have to conform to content adaptation.

OMA has proposed the following 11 rules for MMS Proxy-Relays that handle content adaptation[3].

1. **Major** To reduce size, GIF89a content is converted into other supported image formats. The animated part is removed.
2. **Minor** Image resolution is reduced to 160*120 px.
3. **Minor** Message size is reduced to $\leq$ 30 kB, by reducing resolution and quality for image and audio content.
4. **Minor** Message size is reduced to $\leq$ 100 kB, by reducing resolution, frame-rate and quality for image, audio and video content.
5. **Major** MIDI level 1 or SP-MIDI media type is removed.
6. **Major** One or more video frames are converted to supported image formats, and the video part is removed. The resolution is scaled to 160*120 px.
7. **Major** One or more video frames are converted to supported image formats, and the video part is removed.
8. **Major** Image resolution is reduced to 160*120 px.
9. **Major** Message size is reduced to $\leq$ 30 kB, by reducing resolution and quality for image, audio and video content.

10. **Minor** Image resolution is reduced to 640*480 px.

11. **Minor** Message size is reduced to $\leq$ 300 kB, by reducing resolution, frame-rate and quality for image, audio and video content.

To adapt a content, one or several of these rules should be applied.

3.7.3 Adding interaction to the service

As mentioned, MMS can, roughly speaking, be seen as an improved variant of SMS. It uses the same approach of creating a message the way you want the recipient to see it. MMS provides a bit more features, since you can use things like slide duration to give the recipient a better flow when reading the message. But it seems like you would want to demand more of the service.

Some researchers propose that MMS should be added a larger support for interaction. An example of this approach is the iMMS (interactive Multimedia Messaging Services)[8]. iMMS is, as the name tells you, an interactive variant of the MMS.

The proposed approach would provide easier interaction for the user. Suppose that a user receives an advertising MM offering her the possibility to purchase some ring-tones. With the tools offered by today's MMS standard, she would have to select to reply on the MM and probably add a certain code representing the ring-tone she wants to purchase. She would then have to send the message before the order is complete. This ordering procedure involves a number of steps, which is not very good from a usability perspective. There is a risk that the user discards the offer, because of the time consuming procedure to order the ring-tone.

If iMMS was used instead, the ordering procedure could be simplified. The MMS application would then provide tools that reduces the number of steps needed to perform a certain task. For instance, in the ring-tone ordering example, the MM could contain a button with an associated action. This action could tell the MMS application directly that it should compose an MM, add the selected code to it, and send the message to the correct recipient. This way the ordering procedure would be reduced to pushing one button. Naturally, this would be a big improvement.

iMMS could also provide new possibilities for the usage of MMS. The service could, for instance, be used for home surveillance purposes[11]. The home surveillance system could contain detectors like infrared sensors, gas detectors and cameras. The system could then, when triggered, send messages to the owner containing pictures and data from the detectors. The user could then investigate the data received, to see whether something is wrong. If everything is normal, the system could be reset by pushing a button in the MM.

Implementing iMMS

To be able to introduce iMMS to the market, two things have to be added to the MMS standard: Client side responsiveness and local resource interaction[8].

Client side responsiveness can be seen as a set of controls in MMS for the user to interact with. It could be for instance, text input, buttons and selection lists. To support one of these controls in MMS, three attributes need to be defined[8]:

- **Visual presentation** Defines how a specific control should be displayed on the screen.

– **Action** Defines the function of the control, and what should happen when the action occurs.

– **Relationship to other controls** This is an optional attribute that defines internal or external relationships between different controls on the screen.

The current presentation format, SMIL, does only support layout and presentation on the screen. It has no support for interaction. This indicates that introducing iMMS would require introducing an interactive presentation. For WAP applications today, the XHTML Mobile Profile is the standardized presentation format[8]. One way to solve the introduction of iMMS could hence be to adopt this profile as a presentation language in MMS. Of course, there would still have to be support for SMIL presentation. The optimal way would be to combine these two, to get the best from two worlds.

The MM structure would have to be altered to comply with iMMS. Shen et al have proposed the structure displayed in figure 3.4[7]. There, the data in the message is separated from the different presentation parts. This makes it possible to render the message even if the client can not handle the presentation format. This is needed to be backwards compatible to old MMS standards.

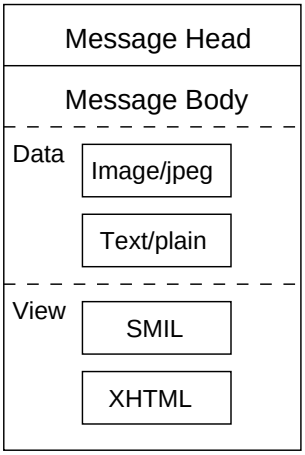

Figure 3.4: Proposed message structure for iMMS

The presentation parts of the message contain information about how to display the message, and what actions that should be associated to the controls. For clients supporting iMMS, the XHTML part should be used to present the message. The SMIL part is used to control the appearance for clients non-compliant with iMMS.

iMMS vs WAP browser

When adding interaction to MMS, one might think that this is trespassing into the WAP browser area, but this is probably not the case. iMMS and WAP browser are two services that would have different areas of use. Some of the proposed features for iMMS, like for instance ordering ring-tones, could of course be achieved using the browser. iMMS would however provide some solutions to a couple of the drawbacks for the browser, such as the following[7].

- **Availability** - The WAP browser can only be accessed when connected to a network. As anyone that has a mobile phone knows, network stability is not always as good as one might desire.
- **Usability** - The WAP browser is not always particularly usable. For each link that is clicked on a page, the user needs to wait for the return of the sent request. This is sometimes very time consuming.
- **Network efficiency** - When using the WAP browser, an entire HTML file is sent every time a refresh is done. This is performed even if there is only a slight change done since the last update. This naturally leads to delays that may decrease the user's experience.

As the points specify, for some tasks iMMS would be a better choice than the WAP browser. iMMS is however something that probably would be application driven. The user needs to retain an interactive message to be able to use the service.

3.8 Conclusions

This chapter has presented a deeper look at the MMS. The MMS is a messaging service that is growing at a rapid rate. It is not only the P2P messaging that is increasing. A2P messaging is the part of the service that has the highest compound annual growing rate.

An MM consists of two parts, the message header and the message body. The message header contains the header fields of the message, and the body contains the contents attached to the message together with a presentation part. The presentation type used in OMAs standards for MMS is SMIL.

MMs are divided into content classes, that specify the content types that are supported. Version 1.3 of OMAs standards contains six content classes, and at least two of these need to be supported. The standards only guarantee interoperability within content classes, not between them.

MMS needs to deal with some issues to continue being a success. Some of the problems that it will be facing are the following.

- Transmission delays.
- The amount of value added from earlier services.
- Interoperability between content classes.

The discussion has shown that a combination of solutions probably will be needed to tackle current and future problems. To solve transmission delay issues, P2P and A2P messages could be separated by using different channels. This would decrease the risk for the entertainment messages to be delayed. As a result of this, A2P messaging may continue to gain ground.

Content adaptation at the MMS Proxy-Relays is a way to solve interoperability issues between content classes. This seems to be of utmost importance. If users experience problems with receiving and watching messages, this will most certainly turn the users away from the service.

iMMS seems to be a way of adding value to the service. With this functionality, more flexibility will be provided in the creation and interaction with messages. For instance, pushing a button could give the MMS application instructions to send a reply with a

specified content. This would probably increase the user's experience of the service, since the user would not be forced to go through the normal routine with selecting to reply on an MM, adding content to it, and selecting to send it. Several steps will be omitted, which would be a good thing from a usability perspective.

iMMS would probably only be used for A2P messages. It is not likely that future MMS clients will be designed to create this kinds of messages. Creating interactive messages would involve specifying the actions and controls that should be present. No one would probably be interested in doing these kinds of advanced operations on a small mobile terminal.

iMMS would not be a way to replace the WAP browser either. They would offer different services to the users. iMMS would provide the possibility to perform off-line operations using the interactive interface, where as the WAP browser provides the possibility for the user to browse the information she wants from the net.

Chapter 4

Implementation

This chapter provides information about the implementation part of the work. But, since the implementation was based on a couple of Obigo's products, some information is confidential and therefore not explained in detail.

4.1 Preliminaries

The implementation part of this master's thesis was divided into two stages.

1. Pre-study
2. Main task

The pre-study stage involved the following tasks.

- Gather desired functionality from future users.
- Determine a user interface based on usability and configurability.
- Investigate the possibility to present SMIL messages using a browser.
- Determine code base, aspects regarding maintenance and continuing development.

The main task stage involved the actual implementation of the system.

4.2 Determining functionality

To gather the desired functionality from future users, a survey was used. The survey presented the core plan with the system, to give the users a vision of the resulting product.

The survey resulted in a lot of suggestions. Depending on their characteristics, they were divided into the following categories.

1. **Functional requirements**
   - Support for selecting char-set on text added to an MM.
   - Support for adding content with different encoding.

- Support for selecting background color and text color.
- Support for copying an existing MM, and editing it.
- Support for composing an MM to an exact size.
- Support for editing all headers in the MM.
- Support for importing from and exporting to ProtellerPacketEditor (an internal program used at the company).
- Support for controlling the order of the body parts.
- Support for empty body parts.
- Support for deliberately sabotaging parts of the MM.
- Support for creating and editing other types of MMs than Multipart Related.
- Support for creating MMs according to different MMS standards (currently 1.0, 1.1, 1.2 and 1.3).
- Support for parsing a PDU to be able to find incorrect parts.
- Support for connecting the editor to printPDU (an internal program used at the company).
- Support for handling DRM (Digital Rights Management) protected content.

2. **Design requirements**

   - The structure of the MMS editor should enable it to be integrated in a complete web based messaging client in the future.
   - The structure of the MMS editor should make it easy to add new elements and attributes, to comply with new MMS standards.

3. **Other requirements**

   - A database with test content available.
   - A search function available, that enables searching among messages as well as within messages.

The given suggestions represent advanced functionality. The decision regarding which functionality that should be implemented was based on the suggestions as well as the available tools. As mentioned in the problem specification section, another part of the company had developed a variant of an MM editor that works as a standalone program. The intention was initially to investigate this editor to see if it could be used as a basis for the implementation. If this editor had the core functionality, there would be time to implement more advanced features. However, internal policy in combination with a high workload caused a lot of delays in the collaboration with the originators of the existing editor. Because of this, the decision was made to discard this editor. The risk of delays caused by awaiting information and answers from them was considered to be too high.

The MMS editor was instead to be implemented from scratch. This meant that there would not be time to implement the same amount of advanced features. The suggestions retained from the survey was then divided into the following priority groups.

- **Group 1**

- Support for selecting char-set on text added to a MM.
- Support for selecting background color and text color.
- Support for editing all headers in the MM.
- Support for copying an existing MM, and edit it.
- The structure of the MMS editor should make it easy to add new elements and attributes, to comply with new MMS standards.

– **Group 2**

- Support for creating and editing other types of MM than Multipart Related.
- Support for deliberately sabotaging parts of the MM.
- Support for composing an MM to an exact size.
- Support for creating MM according to different MMS standards (currently 1.0, 1.1, 1.2 and 1.3).
- Support for adding content with different encoding.
- Support for controlling the order of the body parts.
- Support for empty body parts.
- Support for importing from and exporting to ProtellerPacketEditor.
- Support for connecting the editor to printPDU.

– **Group 3**

- Support for handling DRM content.
- Support for parsing a PDU to be able to find incorrect parts.

– **Group 4: Other**

- A database with test content available.
- A search function available, that enables searching among messages as well as within messages.
- The structure of the MMS editor should enable it to be integrated in a complete web based messaging client in the future.

The first two items in priority group 4 were more or less applicable to the editor. The suggestion of a database with available test content was to be considered. The editor should preferably have support for adding content from a database residing at the MMST, but filling that database was not a part of this master's thesis. The search function desired was a functionality that is more suitable to place at the MMST instead of in the MMS editor, so it was not to be implemented. The last suggestion in group 4 was something that would be taken into consideration, but it would not be classified as a requirement.

The requirements from the company's point of view were the following.

– The MMS Editor shall be available for users at the Obigo MMST.

– It shall be possible to create correct and incorrect MMs with the editor.

– Created MMs shall be possible to save at the Obigo MMST.

– Created MMs shall be possible to send via the Obigo MMST.

The main priority was to implement the functionality defined by the company requirements mentioned above. If time would be left afterwards, functionality would be added to the editor according to the priority given above.

4.3 Browser support for presenting SMIL

Some users expressed a wish for the editor to be able to preview an MM that is being created. To determine the possibilities to achieve this using a web browser, an information search was performed. The basis for this search was Wusteman's article "XML to the desktop"[15]. Since this article was published two years ago, an investigation was performed to verify the results. The information search indicated that there currently are two browsers that support SMIL presentations, Internet Explorer and X-smiles.

Internet Explorer Internet Explorer version 5.5 or higher supports SMIL presentations [12] and version 6.0 or higher supports SMIL 2.0 [12]. The browser supports the MIME types used in MM as well as timing and synchronization. However, the way the different parts in the SMIL should be structured to be displayed in the browser is not the same as in an MM. Figure 4.1 displays the layout of a SMIL MM used in mobile messaging and figure 4.2 displays the core layout of a SMIL to be presented in a web browser.

```
<smil>
    <body>
        <par dur="2s">
            <text src="text.txt" region="Text">
            <img src="image.gif" region="Image">
            <audio src="sound.amr">
        </par>
        <par dur="10s">
            <img src="flower.jpg" region="Image">
        </par>
        <par dur="5s">
            <text src="text2.txt" region="Text">
            <audio src="beep.amr">
        </par>
    </body>
</smil>
```

Figure 4.1: SMIL structure for MMS

In SMIL MMS, each slide is represented by a parallel module[3]. The contents within a parallel module will be displayed at the same time. Basically, the SMIL presentation contains a parallel module for each slide in the message.

The SMIL for browser presentation, on the other hand, uses another approach. There, all contents that are supposed to be displayed within a certain layout area has to be specified within a sequential module[12]. The different sequential modules that define the different layout areas also have to be included within a parallel module.

```
<body>
    <par>
        <seq repeatCount="1">
            <text src="text.txt" dur="2s">
            <text src="text2.txt" begin="10s" dur="5s">
        </seq>
        <seq repeatCount="1">
            <img src="image.gif" dur="2s">
            <img src="flower.jpg" region="Image" dur="10s">
        </seq>
        <seq repeatCount="1">
            <audio src="sound.amr" dur="2s">
            <audio src="beep.amr" begin="10s" dur="5s">
        </seq>
    </par>
</body>
```

Figure 4.2: SMIL structure for browser

Consequently, the browser is not capable of handling an MM in its original form. To be able to use the browser presentation, a parser is needed to fetch and order the information from the message. The size of this task depends on the demands on the presentation. If the task is to only display the different slides in the MM, with correct duration, it will not be too time consuming.

X-smiles X-smiles is a Java based XML browser. The browser includes a SMIL player that supports nearly all SMIL 2.0 features[9]. The SMIL player can be used in three ways:

- Integrated in the X-smiles browser.
- Run as a standalone player.
- Used as a library package.

Conclusions The investigation indicated that X-smiles was a bit more flexible to use than Internet Explorer. However, Internet Explorer is widely used. The browser is installed in nearly all computers in use. X-smiles, on the other hand, requires downloading and adjustments of settings to suite the MMS editor. So, since they support the same features, Internet Explorer might be the better choice.

4.4 Determining user interface

To determine the design of the user interface, the gathered functional requirements was used as a basis, together with the knowledge of the future users. This system was not to be used only by advanced users. The future users could be divided into the following two main groups.

– MMS application developers.

– MMS application testers.

Obviously, an MMS application developer is a person possessing quite a lot of knowledge about MMS. The developers can be categorized as advanced users. The MMS testers, on the other hand, might have varying levels of knowledge. Some testers at the company are students. This means that they might not have been working much with MMS, and are therefore considered to be unaccustomed to it. To be able to execute test cases, they do not need to know the internal structure of an MM.

The conclusion that could be drawn was that the user interface had to be designed to suite the needs of both advanced and novice users to MMS. It must be possible to create an MM that uses default values on the headers, as well as it has to be possible to create a specially adjusted MM. The design of the user interface was therefore a conflict of interests. The system should be user friendly, but at the same time suited for advanced users.

To make design decisions, a review group was put together. The group contained ten employees at Obigo, that represented a broad spectrum of users: test, application design, development and GUI design. The review group was given propositions of the interface, and was able to give feedback on it.

4.5 System architecture

When designing the system architecture, several considerations had to be taken into account. The MMST used by Obigo was an Apache Web Server containing among other things an MMS Proxy-Relay module and an Obigo MMS client[13]. The MMS editor should be integrated with these two modules, that were implemented in C.

The system was implemented in a client-server manner. That is, the MMS editor acted like a client, that connected to the MMST using specific request arguments.

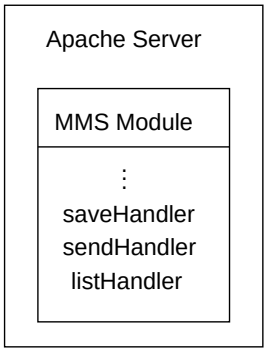

Figure 4.3: Handlers at the Apache server

4.5.1 Server side components

The part of the system that resided at the server was implemented in C. This language was selected, to simplify using the original components the server contains. C also

provides good possibilities to allocate and use memory spaces, which is necessary when building the messages.

The server part is a C-file included in the Apache library, that handles everything that originates from the MMS editor. To process the requests from the editor, handlers were added within the MMS module of the Apache server. This is illustrated in figure 4.3. The request is passed along to the correct handler, which performs the desired operations. The following actions can be performed by the added handlers:

- Save an MM at the MMST.
- Send an MM to the recipients specified in the MM.
- Request a file list. This is used to be able to determine where in the file structure the MM should be stored.

The MMS module distinguishes the requests from each other based on the uri field.

4.5.2 Client side components

The MMS editor has been implemented as a JApplet in Java. This choice was based on the following factors.

- **GUI support** Java has good support for GUI design issues.
- **OO Design** Java is object oriented, which suits this type of project well.
- **Ease of use** The applet will be able to run directly via the MMST web interface. Hence, the users do not need to bother about installations or settings.
- **XML support** Java has good support for handling XML.

The client was implemented as a signed applet. This was required, since applets generally are considered to be insecure and are placed in a "sandbox". Signing the applet renders the possibility to access the user's hard drive. This was desirable, since users may want to use content from their local disk when creating messages.

Since the MMS editor was implemented in Java, it consisted of compiled Java classes compressed into a jar-file. However, it is fully possible to retrieve the original source code by decompiling the classes.

To protect the source code of the editor, an obfuscator was used. The purpose of an obfuscator is to make it hard to reverse engineer the code. All classes, methods, interfaces and variables are renamed using shorter names, that are meaningless to the context. This also leads to a smaller size of the rendered solution.

4.6 Communication protocol

Since the server and client parts of the solution should communicate via the Apache server, a protocol was needed. The decision was made to use XML for this purpose. The XML protocol that has been created for this purpose is displayed in Appendix B.

The editor generates an XML representation of the MMs that are to be saved or sent. The XML encoded data is then transferred to the server, which decodes the data. For the decoding process, libxml2 was used. This library contains both SAX and DOM

interfaces. For this system, though, the DOM interface was used. Using the DOM interface requires a bit more memory, but since the system will run on the server, this was not a problem.

For every received message, the parser creates a document tree representing it. The tree can then easily be traversed to fetch the information that is needed.

4.7 Base64 encoding/decoding

To be able to transmit rich data, eg images, sounds and videos, between the client and the server, the data needed to be encoded into a transmittable format. For this project, the base64 algorithm has been used. Base64 is a binary to text conversion, which converts an arbitrary sequence of bytes to printable ASCII characters. After the transmission, the data sequence is decoded into the original format again. The procedure is illustrated in figure 4.4.

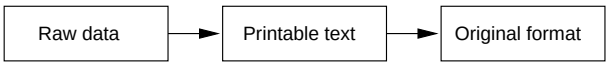

Figure 4.4: The idea of base64 encoding and decoding

The length of the resulting sequence is longer than the original by a ratio of 4:3. The basic idea with base64 is to divide the original data into subsequences of three bytes. These subsequences are then split into four different parts, each containing six bits, as illustrated in figure 4.5.

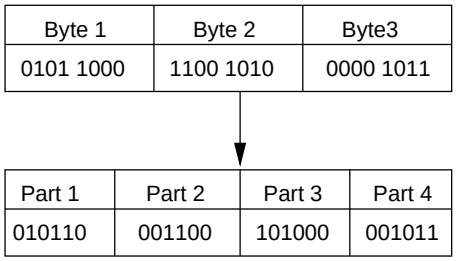

Figure 4.5: Base64 sequence encoding

To be able to represent all possible values for one of these parts, $2^6 = 64$ different characters are needed. Hence the name of the algorithm. The mapping between values and ASCII characters are displayed in table 4.1. Every value is then replaced with its corresponding ASCII character, and the encoding is done.

The decoding procedure is performed in the reversed order of the encoding. That is, the characters in the encoded string are replaced with their corresponding value. This procedure recreates the original binary data.

4.8 Creating SMIL code

The MMS Client residing at the MMST contains functions for creating SMIL code. However, the MMS Client is, of course, designed to create correct messages. This fact

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Table 4.1: Base64 mapping between values and characters

lead to the decision not to make use of this functionality in the implementation. In MMS testing, you do not only want to perform the tests using correct data. It is also interesting to be able to see how the MMS application reacts on incorrect data. Creating the SMIL code within the MMS editor enabled this possibility.

4.9 Save functionality

The save functionality enables the possibility for the editor to store MMs to the server.

The call sequence for the save process is displayed in figure 4.6. The process is initiated when the user selects the option save. The first call is then done to the server. The call handler at the server then executes a script that traverses the file structure and renders a representation of it in a file. The server then returns a simple HTTP response to the client. The client then fetches the updated file using an HTTP Get request. After receiving the response, the client generates a structure representing the file system at the server. This gives the user the possibility to select where in the structure the MM should be saved.

After the user has entered the desired name and location for the MM, the XML generation is initiated. It encodes the message using the XML protocol. Any content that is of multimedia type is encoded using base64 before it is appended to the XML code. The generated data is then sent to the server by a HTTP Post request. The uri in the request specifies that the request should be passed along to the save handler at the server.

When the request arrives to the save handler, the XML code is parsed. Any incorrectness regarding the mandatory parts causes the server to abort the creation and return an error message to the client. During the parsing, a message buffer is created. If the parsing is done successfully, the message buffer is then written to a file created at the user specified location. When finished, the server returns an HTTP OK response to the client.

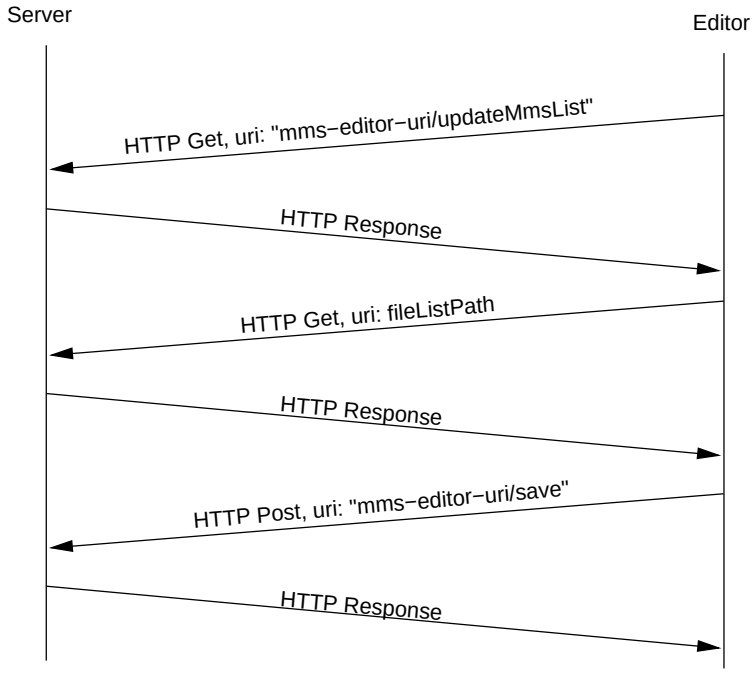

Figure 4.6: Call sequence when saving an MM

4.10 Send functionality

The send functionality for the editor means that the MM is sent to the specified recipients using the Proxy-Relay part at the editor.

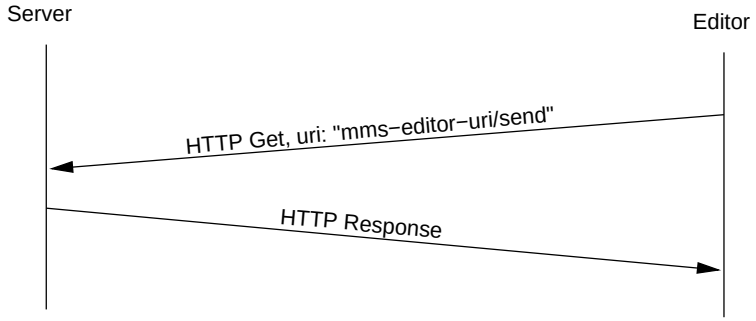

Figure 4.7: Call sequence when sending an MM

The call sequence for sending an MM is displayed in fig 4.7. As in the save call, the editor generates an XML representation of the message. The data is then forwarded to the server using an HTTP Post request. The uri in the request specifies that the call should be passed along to the send handler.

When the send handler receives the call, it reuses some functionality from save case. It parses the XML code and generates a message buffer. When the buffer is created, the addresses specified in the To, Cc and Bcc fields are fetched. Each found address is

mapped against the address book at the server. If a match is found, the entry information is fetched. A MM file is then created using the naming convention displayed in figure 4.8.

[telNr]_mmsEditor_name.mms[.udp.ipAdress]

Figure 4.8: Naming convention for MMs when sending

An entry in the server address book can be either of telephone number type or of IP address type. If an address entry is found that is of telephone number type, the number should be entered in the telNr part of the message name. The ending udp part should, of course, be omitted. A message name could be, for instance, 0701234567_mmsEditor_test1.mms.

If the entry, on the other hand, specifies an IP address, the last part of the naming convention should be used and the telNr part should be omitted. An example of a message name could, in this case, be mmsEditor_test1.mms.udp.10.11.1.15.

So, for each address that has a matching entry in the server address book, a message file is created in a temporary catalog and named according to the convention.

At the server, there exists one folder that is watched by a sending service. Whenever an MM file is detected in the folder, the service initiates a sending procedure. The procedure is performed in the following steps.

1. A notification is created for the message.
2. The message is moved to a storage folder.
3. The notification (which contains the path to the moved message) is sent to the recipient specified by the message name.

This service is made use of when the editor wants to send MMs. After the messages have been created and correctly named, they are moved into the watched folder. The reason why the messages are not created in the watched folder at once, is timing. If the files were created in the watched folder immediately, the server might not have time to write the contents to the files and close them before the sending service tries to send them.

After the messages have been moved into the watched folder, the server relies on the sending service to do its job. It therefore sends the HTTP Response message to the editor, indicating that execution was successful.

Chapter 5

Results

This chapter presents the result from the practical part of the work.

5.1 Implemented functionality

As mentioned in the implementation chapter, all the desired functionality was not possible to implement during the work with the master's thesis, due to the fact that the editor had to be created from scratch. Some changes were made to the priority groups containing the functionality suggestions. If the list of suggested functionality is used as a basis, the following functionality have been implemented.

- Support for editing the headers.
- Support for exporting to ProtellerPacketEditor.
- Support for deliberately sabotaging parts of the MM.
- Structure that enables adding new attributes easily.
- Support for creating MMs according to different standards.

These are the advanced functionalities that have been implemented. If the companies original requirements are investigated, the conclusion is that all of these requirements are fulfilled. That is, it is possible to do the following things:

- The MMS editor is accessible from the Obigo MMST.
- It is possible to create correct and incorrect MMs with the editor.
- It is possible to save created MMs at the Obigo MMST.
- It is possible to send created MMs via the Obigo MMST.

5.2 User interface

The user interface has been designed with consideration to the future users. The editor will have both advanced and novice users. Because of this, the interface provides the possibility to edit all parts of an MM, but the user is not required to do so. Default values are added to several fields.

The user interface contains a tabbed pane where each tab provides the possibility to edit a certain part of the message. The tabs that are present in this version of the editor are the following.

- Message body
- Message header
- Attachments
- SMIL presentation

The different panes will be presented in the coming subsections.

5.2.1 Message body

The message body pane gives the user the possibility to edit the in-line content of the message. The pane is divided into three sections as displayed in figure 5.1. The leftmost section displays the slide layout of the message. By selecting one of the content buttons, the user can edit that specific content. The section also contains navigation buttons that allows the user to move forward and backward through the presentation.

The topmost section to the right allows the user to edit the content that currently has been selected in the left section. It is possible to add, remove and replace the content. The user can also set start and end time for it.

The final section to the right allows the user to edit the slide that currently is selected. It is possible to add a new slide and to remove the current slide. The user can also set a duration for the slide, that is how long the slide should be played in the generated presentation.

5.2.2 Message header

The message header pane, displayed in figure 5.2, gives the user the possibility to edit the header part of the message. The fields that are possible to edit in this version of the editor are the following.

- **To** - The recipients that should be set in the To field.
- **Cc** - The recipients that should be set in the Cc field.
- **Bcc** - The recipients that should be set in the Bcc field.
- **From** - The address to set as the originator of the message.
- **Subject** - The subject to set on the message.
- **Delivery Report** - If a delivery report should be requested or not.
- **Read Report** - If a read report should be requested or not.

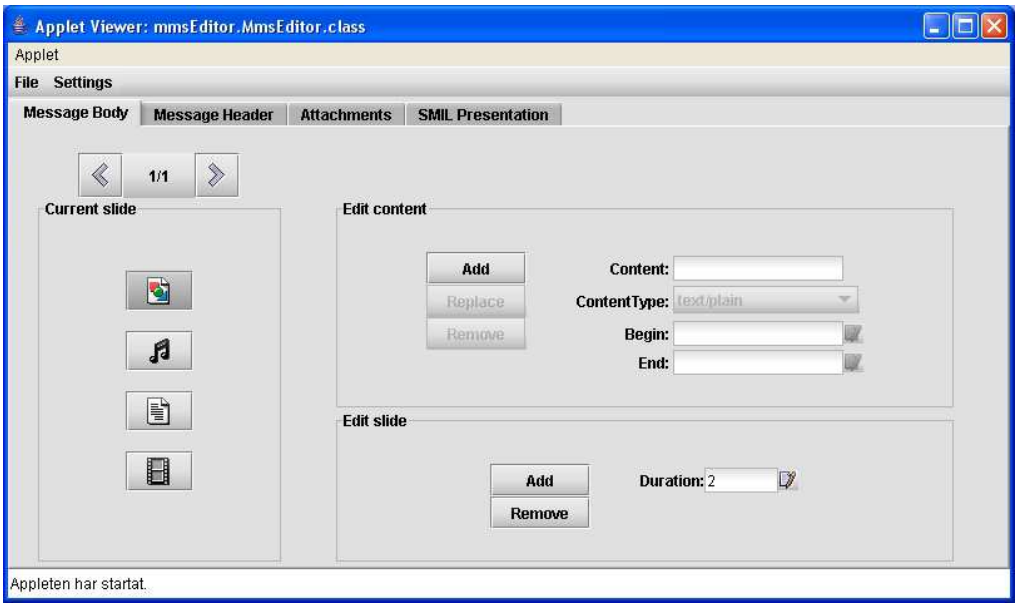

Figure 5.1: Screenshot of Message Body pane

- **MMS Version** - The MMS version to set on the message.
- **Priority** - The priority to set on the message.
- **Message Class** - The message class to set on the message.
- **Sender Visibility** - If the originator address should be visible or not.

The user is not required to fill in all of the fields. The fields Delivery Report, Read Report, MMS Version, Priority, Message Class and Sender Visibility are already filled in with default values. This means that it is possible to only add the recipients and originator to the MM and send it.

5.2.3 Attachments

The Attachments pane is displayed in figure 5.3. In this pane the attachments part of the message can be edited, that is the contents that are not supposed to be displayed in the SMIL presentation.

5.2.4 SMIL presentation

Figure 5.4 shows the SMIL presentation pane. It gives the user the possibility to edit the presentation part of the message. The following things can be altered.

- **Root-Layout** - The user can change the root layout, that is the layout of the two regions ”Image” and ”Text”.
- **Size** - The user can change the size of the root layout.
- **Ratio** - The user can change the ratio between the two regions.

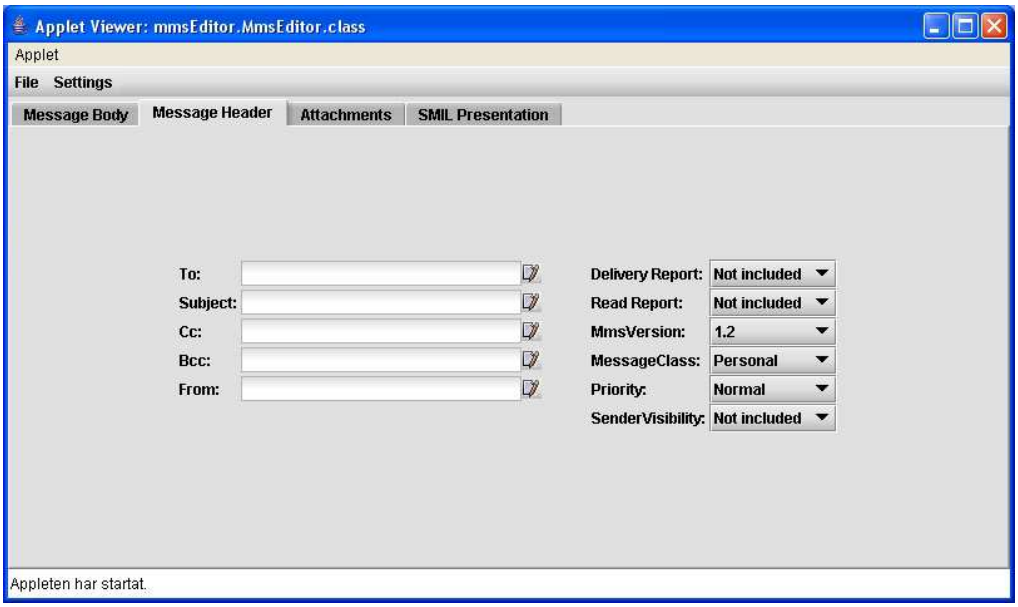

Figure 5.2: Screenshot of Message Header pane

The values set will be used for the entire presentation. It is not possible to use different layouts for different parts of the message.

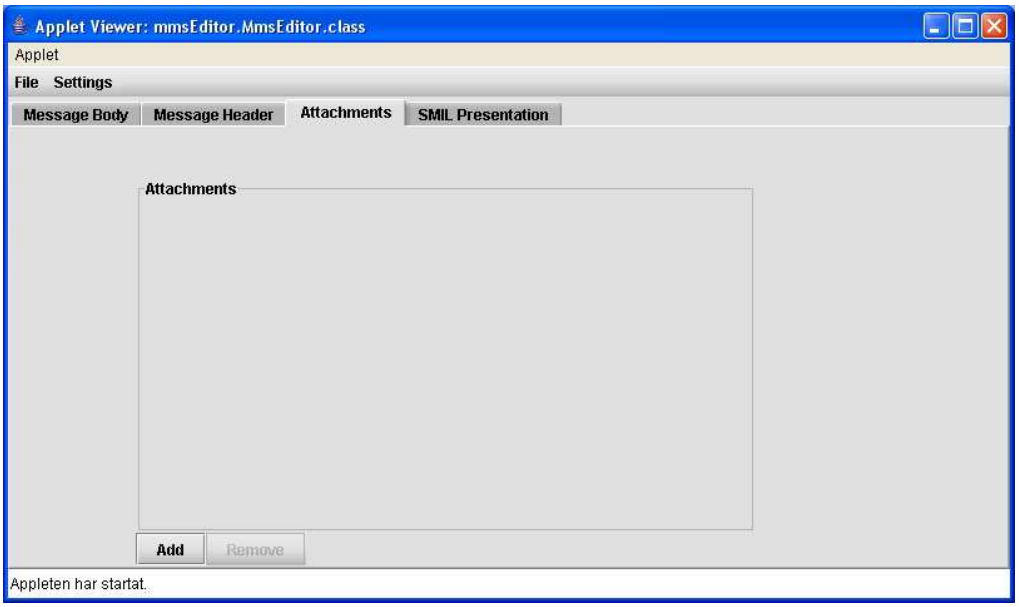

Figure 5.3: Screenshot of Message Attachments pane

Figure 5.4: Screenshot of SMIL presentation pane

Chapter 6

Conclusions

In this chapter, some conclusions about the project will be drawn.

6.1 Problems during the work

As in most software development projects, the time estimation of this project has been difficult. The planning was hard depending on many factors. Some factors that contributed to the difficulties were the following.

- Communication with another Teleca office was tedious and slow.
- Problems in determining the best approach to the implementation.
- Problems caused by poor documentation of the MMST.

These problems are explained in more detail in the following subsections.

6.1.1 Internal policy

As stated in the problem description, the main idea was to use an existing MMS editor in this project. The editor would be customized to fill the needs of this project. That is, support for the communication with an MMST would have to be added. Using the existing editor would also provide the possibility to add advanced functionality to it, since it already possessed the fundamental functionality.

However, internal policy between the offices made this process hard. A couple of weeks in the beginning of the project were spent on trying to get final decisions on whether or not they would allow their editor to be integrated in this project, and whether this would involve any expenses.

The other office basically decided to let us look at their editor, from a user perspective. That is, we were not allowed to access the source code. Instead, we could investigate the program, and see what changes that were needed. After deciding on the changes, they would make a decision on whether or not they would allow it.

As one might understand from the time it took to get this first decision, they would require at least a couple of weeks to make a decision regarding the changes on the editor. This way, the work with the thesis would have to be put on hold. This fact led

to the decision to discard using this editor, and instead create a new one from scratch. Naturally, this meant that the requirements list had to be shortened.

This is not meant as critique of the other Teleca office. Knowing the work load they are currently faced with, it is understandable that these kinds of decisions are not of top priority. However, one could wish that they, instead of saying that they should consider it, could have turned down the inquiry at once.

6.1.2 Poor documentation

Since the editor was to be integrated with the company MMST, some studying of its functionality was needed. The problem with this task was that there exists little or no documentation of it. There are also a number of people that at different times have been involved in the development of it. This way, several people had some knowledge of the functionality, but no one had an over-all perspective. This made it difficult to gain the knowledge needed to perform the integration.

6.1.3 Implementation decisions

The problems with deciding on implementation issues, comes partly from the previously mentioned fact that the MMST was poorly documented and that no one knew the entire system. The problem arised when trying to decide on the best approach to use in the integration between the editor and the server. There exist several different APIs that the editor could make use of. The top one would have been the best choice from a developers point of view. This way, the editor would not have to be aware of details from underlying layers. This was also the initial idea. However, it proved to be harder than what was initially believed to use this top layer.

So, the decision was made to use an underlying API instead. Of course, this way more control had to be given to the editor parts.

The first attempt was to use the secondary layer, which also provided some abstraction from underlying details. However, this approach was also discarded after a while. The functions in this API are designed to be used in mobile devices. Hence, they are constructed to be memory efficient. The consequence of this is that operations may be a bit more time consuming, since they try to make use of as little memory as possible. Since the editor will be residing on a server, it has access to a large amount of memory.

So, the final decision was to dig down into low level APIs, to provide the possibility to make use of large buffers while creating messages.

6.2 Restrictions and Limitations

Since the MMS editor was built from scratch, there was not time to add all the suggested functionality. So, the editor has some restrictions. The following are examples of them.

- The MMS editor can only create message of Multipart Related type.
- Only text with the char-set UTF-8 is handled.
- No previewing functionality has been implemented.

6.3 Future work

There exist much work that can be done based on this master's thesis. Since it has involved implementing an editor, there virtually is no limit on the functionality that could be added to it. However, if one would look at the editor from a realistic point of view, some additions would be more important than other. The following points are some of the most important.

- **Security**. Some security issues should need to be solved for the editor. Since it is integrated with an MMST, there exists a security gap where the editor calls are entered. At the moment, no check for valid users are done. Instead, everyone calling the server with the correct uri:s can access the functionality at the server. If the editor should become a commercial product, this is urgent to fix.
- **Previewing functionality**. Support for being able to preview messages in the editor could be added. As stated in the Implementation chapter, Internet Explorer has support for presenting SMIL. So, with some adjustments it should be possible to add this functionality. A previewer with support for individually set begin and end times for contents might be hard to implement, but a basic variant that only displays the contents during the entire slide duration time could be an option.
- **Char-sets for texts**. The editor should be added support for more char-sets than UTF-8.
- **DRM**. DRM is becoming more and more used today. Therefor, the editor should be added support for creating messages with DRM-protected content.
- **Size feedback**. Functionality should be added to give the user feedback on the current size of the message whilst creating it. This would enable the possibility to create messages to an exact size.
- **Open existing MMs**. Support for opening existing MM:s could be added to the editor. This could be achieved by using the XML-protocol backwards. That is, the server encodes a message using XML and sends it to the editor, which parses the information.

Chapter 7

Acknowledgments

I would like to thank my external supervisor at Obigo AB, Mikael Gustavsson, for his continuous support during the work with this master's thesis. I would also like to thank my internal supervisor at Umeå Universitet, Lars-Erik Janlert, for all his help during the writing of this paper.

The employees at Obigo AB in Umeå also receive my gratitude for all helpful ideas and for volunteering to evaluate the finished product. Thanks also for making this an enjoyable time.

I would also like to thank my family and friends for understanding my priorities during this period of time. Last but not least, I would like to thank Lars for his never-ending support and encouragement.

References

[1] Open Mobile Alliance. Open mobile alliance. Available at `www.openmobilealliance.org/`.

[2] Open Mobile Alliance. Multimedia messaging service, architecture overview, version 1.2. Technical specification, Open Mobile Alliance, OMA Office, 4275 Executive Square, Suite 240, La Jolla, Ca 92037, March 2005. Available online at `www.openmobilealliance.org`.

[3] Open Mobile Alliance. Multimedia messaging service, conformance document, version 1.2. Technical specification, Open Mobile Alliance, OMA Office, 4275 Executive Square, Suite 240, La Jolla, Ca 92037, March 2005. Available online at `www.openmobilealliance.org`.

[4] Open Mobile Alliance. Multimedia messaging service, conformance document, version 1.3. Technical specification, Open Mobile Alliance, OMA Office, 4275 Executive Square, Suite 240, La Jolla, Ca 92037, 2005.

[5] G. Le Bodic. *Multimedia Messaging Service: An Engineering Approach to MMS*. John Wiley And Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2003.

[6] J. A. Harmer. Mobile multimedia services. *BT Technology Journal*, 21(3):169–180, 2003.

[7] C. Guo J. Shen, P. Sun, Y. Yin, and S. Song. Delivering mobile enterprise applications on imms framework. *MDM*, May(5):289–293, 2005.

[8] J. Zhang J. Shen, P. Sun and S. Song. imms: Interactive multimedia messaging service. *IBM Journal of Research and Development*, 48(5):627–633, 2004.

[9] X-Smiles Organization. X-smiles.org. Available at `http://www.xsmiles.org/`.

[10] J. Ylinen S. Gratschew, J. Raitaniemi and P. Loula. A multimedia messaging platform for content delivering. Technical report, Tampere University of Technology, Pohjoisranta 11 P.O Box 300, FIN-28101 Pori, Finland, 2003. Available at `http://trc.pori.tut.fi/tots/P431.pdf`.

[11] S. Pei S. Jun, Y. Rong and S. Song. Interactive multimedia messaging service platform. *MM '03*, November 2-8:464–465, 2003.

[12] W3 Schools. W3 schools. Available at `http://www.w3schools.com`.

[13] Teleca. Mmst technical overview. June 2005.

[14] X. J. Wang. Peer-to-peer messaging growing at 82.2 *Electronic News (North America)*, 50(40), April 2004.

[15] J. Wusteman. Xml to the desktop. *Library Hi Tech*, 21(2):238–245, 2003.

Appendix A

Abbreviations

- A2P = Application-To-Peer
- 3GPP = Third Generation Partnership Project
- 3GPP2 = Third Generation Partnership Project 2
- EMS = Enhances Messaging Service
- HTTP = Hypertext Transfer Protocol
- MM = Multimedia Message
- MMS = Multimedia Messaging Service
- MMSC = Multimedia Messaging Service Center
- OMA = Open Mobile Alliance
- P2P = Peer-To-Peer
- PDU = Protocol Data Unit
- SMIL = Synchronized Multimedia Integration Language
- SMS = Short Messaging Service
- UAProf = User Agent Profile
- WAP Forum = Wireless Application Protocol Forum
- XML = eXtensible Markup Language

Appendix B

XML protocol

This example shows the XML protocol that is used in the communication between the MMS editor and the MMSC.

```
<Mms>
    <Head>
        <To>0701234567</To>
        <Cc>0731122333</Cc>
        <Bcc>0701112233</Bcc>
        <From>1234</From>
        <Subject>just wanted to say hi!</Subject>
        <MmsVersion>
            <MmsVersionMajor>1</MmsVersionMajor>
            <MmsVersionMinor>2</MmsVersionMinor>
        </MmsVersion>
        <DeliveryReport>129</DeliveryReport>
        <ReadReport>129</ReadReport>
        <MessageClass>128</MessageClass>
        <Priority>128</Priority>
        <SenderVisibility>128</SenderVisibility>
    </Head>
    <Body>
        <NrBodyParts>3</NrBodyParts>
        <BodyPart>
            <ContentType>128</ContentType>
            <ContentLocation>text1.txt</ContentLocation>
            <Data>This is page 1...</Data>
        </BodyPart>
        <BodyPart>
            <ContentType>128</ContentType>
            <ContentLocation>text2.txt</ContentLocation>
            <Data>This is page 2...</Data>
        </BodyPart>
        <smil>
            <head>
                <layout>
```

```
                    <region id="Image" top="0" left="0" width="300" height="200" />
                    <region id="Text" top="200" left="0" width="300" height="200" />
                </layout>
            </head>
            <body>
                <par>
                    <text src="text1.txt" region="Text" dur="2s">
                </par>
                <par>
                    <text src="text2.txt" region="Text" dur="2s">
                </par>
            </body>
        </smil>
    </Body>
</Mms>
```

Appendix C

MMS Header Fields

These are some of the most common header fields available in OMA's MMS standard.

- **X-Mms-Adaptation-Allowed** Specifies whether content adaptation is allowed or not. that is used by the destination application.
- **Bcc** Specifies the Bcc recipient(s).
- **Cc** Specifies the Cc recipient(s).
- **X-Mms-Content-Class** Specifies the content class of the message.
- **X-Mms-Content-Location** Specifies the location of an MM that is to be retrieved.
- **Content-Type** Specifies the content type of the MM.
- **Date** Specifies the date and time the MM is sent.
- **X-Mms-Delivery-Report** Specifies whether the originating MMS Client wants a delivery report from the recipients.
- **X-Mms-Delivery-Time** Specifies the earliest allowed delivery time.
- **X-Mms-DRM-Content** Specifies whether the MM contains DRM protected content or not.
- **X-Mms-Distribution-Indicator** This field can be used for an MM that originated from a VASP. If the field is set to No, the message is not intended to be distributed further.
- **X-Mms-Element-Descriptor** Specifies the content reference to the message content.
- **X-Mms-Expiry** Specifies the time the MM will be available at the MMS Server.
- **From** Specifies the originating MMS Client. If the client is using anonymous sending, this field will be removed by the MMS Proxy-Relay.
- **X-Mms-Previously-Sent-By** Specifies the MMS Client that originally forwarded or sent the MM.

- **X-Mms-Previously-Sent-Date** Date and time the MM was originally sent.
- **X-Mms-Message-Class** Specifies the message class. If originating MMS Client does not provide the field, the MMS Proxy-Relay will set it to Personal.
- **Message-ID** The Message ID is a unique reference to an MM, that is used to match read and delivery reports, cancel PDUs and replies with the originating MM.
- **X-Mms-Message-Type** Specifies the PDU type.
- **X-Mms-Message-Size** Specifies the size of the MM.
- **X-MMS-Version** Specifies the MMS version that is used.
- **X-Mms-Priority** Specifies the priority of the message.
- **X-Mms-Read-Report** Specifies whether the originating MMS Client wants a read report from the recipient(s).
- **X-Mms-Recommended-Retrieval-Mode** If this field is present, the recommendation is to retrieve the MM manually.
- **X-Mms-Recommended-Retrieval-Mode-Text** Specifies the reason why manual retrieval of the MM is recommended.
- **X-Mms-Reply-Charging** If this field is set to Yes, it indicates that the recipient can reply to the message free of charge.
- **X-Mms-Reply-Charging-Deadline** Specifies the latest time the recipient can reply to the MM free of charge. After this time, the recipient will have to pay for the reply.
- **X-Mms-Reply-Charging-ID** Specifies the original MM the reply was created from.
- **X-Mms-Reply-Charging-Size** Specifies the maximum size the free of charge reply message is allowed to be.
- **X-Mms-Report-Allowed** Specifies whether sending of delivery reports is allowed by the recipient or not.
- **X-Mms-Retrieve-Status** Specifies MMS retrieve status value.
- **X-Mms-Retrieve-Text** Description of the MMS retrieve status.
- **X-Mms-Sender-Visibility** Specifies whether the address of the originating MMS Client to be displayed or not.
- **X-Mms-Status** Specifies the message status.
- **X-Mms-Store** Specifies whether the message should be stored to the user's MMBox or not.
- **X-Mms-Stored** Specifies whether the message was stored to the user's MMBox.
- **Subject** Specifies the subject of the MM.

- **To** Specifies the recipient address(es).
- **X-Mms-Transaction-Id** The Transaction ID is a unique number used to identify which MM that correspond to a certain notification.