

Transporting data between wireless applications using a messaging system—MMS

Miraj-E-Mostafa*[†]

Nokia Corporation, PO Box 88, FIN-33721 Tampere, Finland

Summary

There is a demand for using messaging systems to transport data between wireless applications in mobile communications. As multimedia messaging service (MMS) is a promising messaging system to meet the demand, it is under enhancement in related standardization bodies to make it effective in transporting application data. The enhancement should be compatible with the existing MMS services for its smooth and safe deployment. This paper proposes a solution to make the enhancement backward compatible, so that a new MMS service of transporting application data has no impact on the existing MMS services. The solution also provides forward compatibility, so that an MMS service of transporting application data does not hinder the introduction of any application in the future, allowing instantaneous deployment of a wireless application. The standardization bodies did not define any format for data communication between a wireless application and MMS in a terminal. A common format in this regard would help mass availability of wireless applications for different purposes independent of manufacturing a mobile terminal. Based on an analysis described in this paper, eXtensible Markup Language (XML) is the suitable format for the purpose. This paper also presents examples of XML file to indicate what data should be communicated in this regard. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: wireless application; mobile communication; messaging system; multimedia messaging service (MMS); standardization

1. Introduction

A messaging system is typically used for the messaging that directly involves a person. As for example, messaging between persons (person-to-person messaging), messaging from a content provider to a person to provide contents (content-to-person messaging), messaging from a person to a service provider to get a service (person-to-service messaging). References [1–3] provide examples of these uses, where a person

is the direct creator and/or the consumer of a message. Beside these uses, a messaging system can be also potentially used for transporting data between applications (application-to-application messaging), where an application is the creator and the consumer of a message. As for example, when two users are playing chess remotely against each other using a chess application in their mobile terminals, the chess application can use a messaging system to transport the information about each move made by the user on the

*Correspondence to: Miraj E. Mostafa, Nokia Corporation, PO Box 88, FIN-33721 Tampere, Finland

[†]E-mail: miraj.mostafa@nokia.com

chessboard to the peer application. More examples of application-to-application messaging are available in Section 2.2. As messaging is flourishing in mobile communication, there is a demand to use available messaging systems for transporting data between wireless applications. This new use of a messaging system is addressed as a new feature of the messaging system in this paper. Though wireless application is the focus here, the word application is mostly used in this paper instead for brevity.

Internet brought a new dimension in messaging by both incorporating multimedia and providing means to have a message in electronic form end-to-end, making a messaging system usable in transporting application data. It introduced widely used messaging systems like e-mail and Instant Messaging. Nowadays, there are solutions for e-mail in mobile communications, but the dominant solutions so far are mostly proprietary. So, it is difficult to have a new feature consistently over different non-interworking e-mail services in mobile communications. Instant Messaging has been developed in the Internet mainly based on proprietary solutions. Solutions for Instant Messaging in mobile communication are either not deployed much or under development. Moreover, there are issues about interworking between the proprietary solutions of Instant Messaging used in the Internet and the solutions of Instant Messaging in mobile communication. Though mobile communication started with pure voice communication, it introduced popular messaging system like short messaging service (SMS) [4]. SMS is also not suitable for transporting application data, as an SMS message is very limited in terms of size and type of content.

As the above-mentioned conventional messaging systems have been designed not keeping application-to-application messaging in mind, those are not useful for transporting application data without enhancements. Most of these messaging systems have been in use for considerable amount of time, and there is no significant development work going on to enhance any of them. Rather, a relatively new messaging system, having the scope for further development, would be suitable to have the enhancement to transport application data. Multimedia messaging service (MMS) has been recently defined to overcome the main limitations of SMS and e-mail. Enhancement of MMS is still going on. Moreover, MMS is able to transport data for various applications, as it can carry data of both wide variety and significant size. So, it has been lately enhanced with

the new feature of transporting application data as content of an MMS message. The enhancement is mainly based on defining different application-specific headers in MMS. Section 2 provides an overview of MMS to the need of this paper, explains more why MMS is suitable for transporting application data, and describes the application-specific headers and other related enhancements in the MMS specifications.

Defining the application-specific headers in MMS is important, but may not be enough for successful and safe deployment of the new feature. MMS is already deployed in various markets. It is important that the new feature is backward compatible, so that its deployment has no impact on the existing MMS services. It is possible that an existing MMS-capable terminal, not supporting the new feature, behaves unexpectedly receiving an MMS message containing application data, for example, causing bad user experience or violating legal constraints (e.g., digital rights, copyrights) of the data. The new feature should be also forward compatible, so that a new application can be deployed any time in the future beside other available applications without any problem. A solution for handling both backward and forward compatibility of the new feature is introduced in Section 3 of this paper.

The format of data communication across the interface between an application and MMS in a terminal is not defined. It is also not described what data should be communicated across the interface. The lack of such definitions may not be a big problem if an application is integrated to a terminal during its manufacturing stage. It might be tricky for a downloaded application to use MMS as its transport, if there is no common agreement about the format of data communication and the specific data to be communicated across the interface. A downloadable application makes availability of an application independent of manufacturing a terminal. So, it provides true scope for growing applications in mobile communications for various purposes, making such common agreement important. This paper makes an analysis, described in Section 4, in search for a format meeting the demands of the interface, proposing eXtensible Markup Language (XML) as the suitable format in this regard. Examples of XML file are depicted in the same section to indicate what data should be communicated across the interface. The same section also provides few more uses of the data format in MMS. Section 5 concludes by summarizing the outcome of this paper.

2. Positioning MMS as a Transport for Application Data

2.1. Overview of MMS and Reasons for Using it for Transporting Application Data

MMS delivers multimedia content end-to-end in terms of a message in store-and-forward fashion. MMS can be used for person-to-person, content-to-person, and person-to-service messaging. An analysis is made over MMS in Reference [3]; it is also described in detail in References [5–7]. The first version of the complete MMS specifications was available in 2001. The work is still going on to enhance MMS with incremental features. The standardization and specifications of MMS are described in References [1,8]. Important features and functionalities of MMS are listed in References [1,3,8], and it is compared with SMS and e-mail in Reference [1].

A simplified MMS architecture is depicted in Figure 1. An MMS user agent (UA), also known as an MMS client, provides means for composing, sending, retrieving, viewing, and other controlling functions to users. Storage, address resolution, routing, billing, content adaptation, routing reports, and interworking are important functions of an R/S. It also interacts with other entities (e.g., other R/Ss, VASP, other messaging systems) as shown in Figure 1.

Different types of messages are used in MMS for controlling and transporting purposes. Transport message is called multimedia message (MM), which is carried as the payload of a lower-layer Protocol Data Unit (PDU). An MM consists of information elements (also known as headers) and multimedia elements, which contain multimedia content and

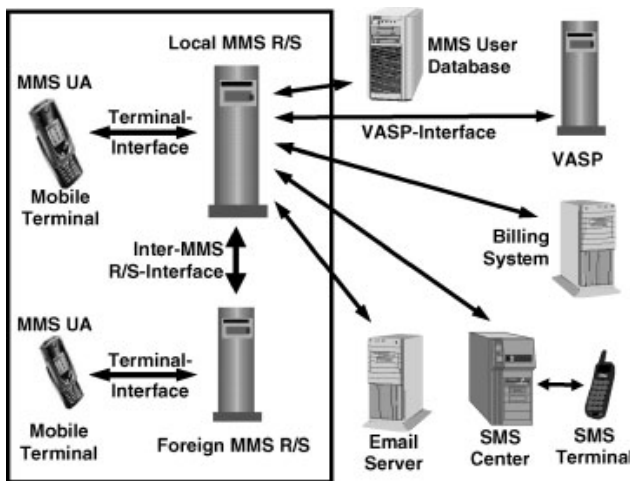


Fig. 1. MMS architecture, depicting different important involved entities and interfaces.

content-specific headers. Multipurpose Internet Mail Extension (MIME) format combines different elements to form a message. An MM is binary encoded before its transmission to use the radio interface efficiently [9]. An example MM is shown within a PDU in Figure 2. Information elements are mostly addressed as headers in this paper for brevity. The contents of the MM in the figure are related. For related contents, MMS typically uses Synchronized Multimedia Integration Language (SMIL) as presentation content, also known as scene description. The presentation content provides rendering instruction of a message by indicating spatial layout and temporal relation among the contents of a message. The flow of different types of messages used in MMS for the end-to-end content delivery is described in References [1,2].

Peer applications, residing in different entities, may need to communicate for different purposes, requiring suitable transport to carry data. The type and size of the data could vary in wide range depending on an application and its purpose. If an application is within a mobile terminal, there are a few choices for a suitable transport, as there are not many carriers that can carry various kinds of data in mobile communication. The structure of an MM is defined to carry metadata and virtually any content in terms of headers and multimedia elements respectively.

Though it works in store-and-forward fashion, MMS is mostly used for delivering content immediately after its submission. Compared to other

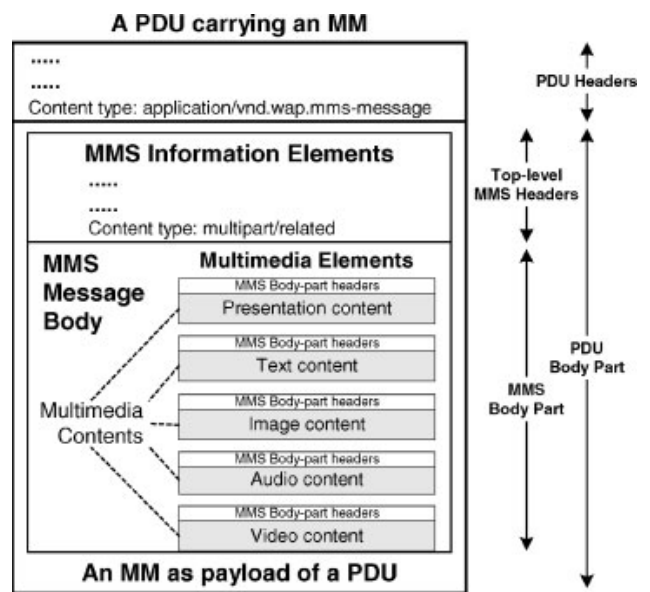


Fig. 2. Structure of an example multimedia message (MM) as the payload of a PDU.

transports, content adaptation and additional hops in multi-operator case are two possible causes of additional delay in MMS for delivering content submitted for immediate delivery. An MMS R/S does not adapt a message containing application data. The transmission in the additional hops between MMS R/Ss, not involving radio interface, does not cause significant delay. So, causes of additional delay in MMS do not have much impact.

Retrieval delay, time to retrieve a message by an MMS UA from an MMS R/S without rendering, usually contributes most in the end-to-end delay for delivering a message in MMS. Based on an experiment in a typical GPRS network, retrieval delay for a message of about 1, 30, 65, 100, and 300 kilobytes are about 2 to 4, 4 to 8, 10 to 12, 12 to 15, and 27 to 30 s, respectively. Other events (e.g., message submission, processing in MMS R/S) cause additional delay, and may contribute similar delay like retrieval delay does in the worst case. The delay is expected to be less in a 3G network. So, MMS could serve the purpose of transporting light time-sensitive data. The additional delay in MMS for transporting heavy data is due to longer transmission time, and possibly, the use of inferior QoS in MMS compared to the same used in real-time transport [2]. MMS is already standardized to interwork with streaming [10] to minimize such additional delay, making it suitable for even transporting heavy time-sensitive application data. Moreover, there is proposal to further enhance the interworking between MMS and streaming [2]. Many of the possible cases of application-to-application communication, as outlined in Section 2.2, do not involve time-sensitive data.

As the development of MMS is still going on in standardization bodies, it is relatively easy to introduce a new feature in MMS. As most of the major terminal vendors have indicated, the penetration of the MMS-capable terminals is expected to grow in the future. Considering all these, MMS has potential as a transport of application data.

2.2. Standardized Works to Use MMS for Transporting Application Data and Examples

Different MMS standardization bodies have recently started working to enrich MMS with the feature of transporting application data. Figure 3 depicts different layers of communication in transporting application data using MMS, showing the vertical interaction within an MMS-capable terminal and an MMS R/S to achieve communication between two applications

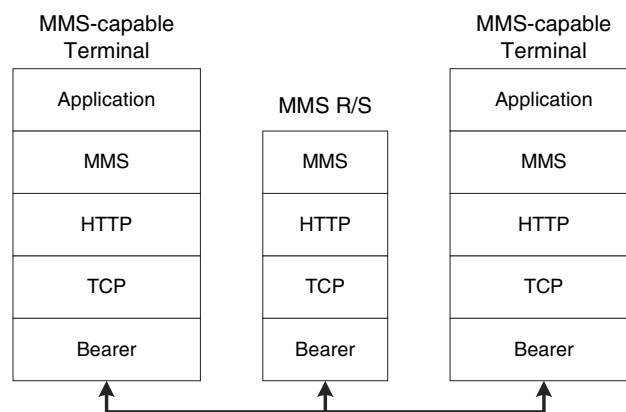


Fig. 3. Different layers of communication for transporting application data between peer applications in MMS-capable terminals through MMS R/S.

residing in two different MMS-capable terminals. The figure assumes a simplified scenario, where the terminals belong to the same MMS R/S. Only HTTP is shown there, as it is the mostly used protocol across the terminal-interface. Lower layer protocols, below MMS in the figure, are MMS-specific, and are beyond the scope of this paper.

MMS architecture has been lately extended to include applications [11]. Application is defined as a system element that may interact with an MMS UA or a VASP to transport its data to its peer application, interacting with another MMS UA or a VASP, using MMS. The extension of MMS architecture is depicted in Figure 4, where unrelated entities and interfaces are not shown. Though an application can interact with either an MMS UA or a VASP to transport its data, MMS UA is mostly mentioned in this regard in this paper to cover both for brevity. Though only one application is shown within a mobile terminal and VASP in Figures 3 and 4, there can be multiple applications, as shown in Figure 5. The application-interface between an application and an MMS UA is described in Section 4.

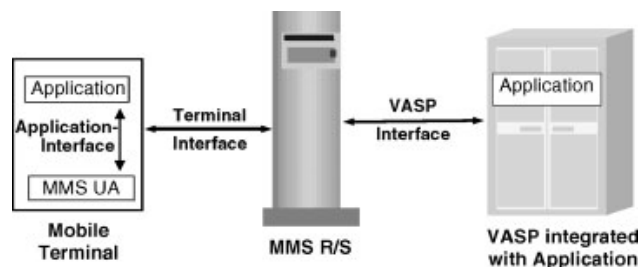


Fig. 4. Introduction of an application and related interfaces in the MMS architecture.

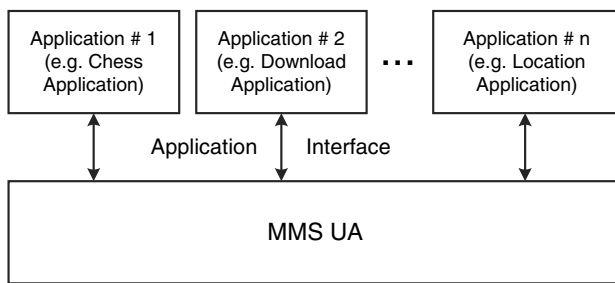


Fig. 5. Interactions of multiple applications with an MMS UA in a terminal.

As for example, a chess application within a terminal may offer its user to play chess with a user of a different terminal having the same application. If both the terminals are also MMS-capable, the application may use an MM to convey the information about each move made by a user on the chessboard to the other user. Other interactive games (e.g., golf, snooker) in a mobile terminal can also use MMS as the transport similarly. A location service could also be based on this new MMS feature, so that a location server, acting as an application over a VASP, can provide location-specific information to a terminal. Moreover, the feature is also useful for downloading any content (e.g., a media object, a new game, a new plug-in of an existing game, an application, media codec) from a VASP. Furthermore, a configuration application could use the feature to get a new or an updated configuration of a terminal (e.g., connectivity information for any network service like MMS, browsing) from a network. These are just few examples, and the feature can be used for many other purposes.

Beside extending the MMS architecture as shown above, the MMS standardization bodies also defined three new top-level MMS headers [9,10]. The headers are described below, while example uses of the headers are available in Section 4.2. The first header identifies the recipient application, so that the recipient MMS UA can easily recognize from the top-level headers that the content of the message is for a specific application, allowing quick handover of the content to the right application. The second header identifies the originating application, so that the recipient application can easily send a response back to the originating application, as an originating and a recipient application can have different identifications.

The third header is defined flexibly so that it could be used for different possible application-specific purposes on demand basis. A user of a specific application in a terminal can communicate with

multiple users of the same application in different terminals (e.g., a user is playing chess with multiple users at the same time). It is also possible that a user of an application in a terminal is communicating with another user of the same application in another terminal in multiple channels at the same time (e.g., two users are playing multiple chess games simultaneously). In these cases, the identifications of the applications may not be enough, as the same application has multiple channels for communication. The third header can be used to distinguish a channel of communication within an application. The third header could also potentially be used for security purposes to protect any possible misuse of the feature (e.g., spamming, sending unsafe data). As for example, the third header could carry any key (e.g., password) to authenticate the originating application, so that the recipient application can ignore the data coming from any invalid or unauthorized application.

3. Compatibility Problem and Solution

It is not expected that an MMS UA understands application data within a multimedia message. Rather, only the recipient application is supposed to have the right knowledge about handling (e.g., processing, presenting, storing) the data. If the recipient MMS UA does not support the feature of transporting application data, it may not recognize the new application-specific MMS headers. If such an MMS UA receives a multimedia message containing data for an application, it is likely that the MMS UA unexpectedly mishandles the data, as handling of unsupported content within an MM by an MMS UA is mostly implementation-specific. This may cause bad user experience. As for example, if the MMS UA presents the data to the user, she may not be happy to see it appearing out of the blue, as the data may not mean anything to her. Moreover, it is also likely that the recipient MMS UA violates any legal constraint (e.g., copyright, digital rights) while presenting, processing, or storing the data. If there is an application-specific way to describe legal constraint of application data, an MMS UA may not understand it. As for example, it may allow distribution of the restricted content using any proximity (e.g., Infrared, Bluetooth, Wi-Fi) or radio interface. User experience and legal constraint are just two examples, while in reality, the consequence may be involved with any unforeseen aspect.

An MMS UA may not support the new feature for various reasons. According to the MMS

specifications, the support for the new feature of transporting application data is an optional requirement for an implementation of MMS. So, an MMS UA in a new terminal may not support the feature. More importantly, MMS-capable terminals are already available in the market. These existing MMS UAs are not likely to understand the newly defined headers. So, backward compatibility of an MMS UA, supporting the feature, with an MMS UA not supporting the feature is an important issue to be solved to avoid the consequences mentioned above, allowing safe and smooth deployment of the new feature. The development of new applications is a continuous process. Thus, it is possible that a recipient MMS UA supports the feature, but the recipient terminal is not equipped with a specific application, which is the recipient application for the data of a received multimedia message. Mishandling of the content is also possible in this case, as described above. So, forward compatibility with future applications is also involved with the problem beside backward compatibility.

Due to the compatibility problem described above, it is a risk to deliver application data, requiring specific treatment, to an MMS UA that does neither know nor understand the requirement. Moreover, such data is most probably useless for a recipient not having the right application to handle it. It is not efficient to deliver such data across the expensive radio interface. Such unnecessary delivery can be avoided, if the recipient MMS R/S either removes the data in question from a message or rejects the whole message before its retrieval by the recipient MMS UA. The latter way would be more practical, as there is not much point to deliver an empty message to a terminal. In either way, the recipient MMS R/S needs to understand if the recipient of a multimedia message, containing application data, supports the recipient application before deciding if to make the multimedia message available for the retrieval. It can be achieved by making a list of supported applications by each recipient available in the associated MMS R/S in terms of the capabilities of the recipient.

MMS already has means for capability negotiation. A recipient MMS UA can advertise its different capabilities (e.g., supported media types, media formats, image resolution, character sets, languages) to its MMS R/S each time it requests for the retrieval of a message. The MMS R/S can adapt/delete the message accordingly before the retrieval. The OMA has standardized the use of user agent profile (UAProf) [12] for capability negotiation in MMS [13]. In UAProf, it is possible to indicate about changed capabilities

dynamically using specific header (e.g., `x-wap-profile-diff` header if HTTP is used [12]). An MMS UA can also use `User-Agent` header [14] to advertise its capabilities. As a solution to the compatibility problem described above, the list of capabilities an MMS UA advertises can be extended with the following two application-specific capabilities:

1. if the MMS UA supports the new feature of transporting application data, and
2. if supported, what are the supported applications by the MMS UA.

This would allow an MMS R/S to reject an incoming message, containing data for an application, for a terminal not supporting the application. It solves the problem of mishandling data by an MMS UA at root, as an MMS UA would not receive the content intended for any unsupported application. Though either or both UAProf and `User-Agent` header could be used to indicate both the application-specific capabilities mentioned above, only UAProf can handle the case of downloading any new application to a terminal by indicating about the new application dynamically using `x-wap-profile-diff` header. It is already proposed to extend the MMS-specific UAProf schema based on this solution.

The solution is network-based, making it also practical considering the relative limitation a mobile terminal has in terms of available power and handling complexities. Moreover, the solution provides better control to a network operator or a service provider, as an MMS R/S would be easily configurable to control the delivery of a message that carries application data. As for example, the solution could be easily used by a network operator or a service provider to configure its MMS R/S to reject any message that carries data of any unknown or suspicious application.

4. The Application-Interface

An application is supposed to register itself with an MMS UA before it starts transporting its data using MMS. The registration process provides means for agreeing both the format of data communication and specific data to be communicated between the application and the MMS UA. It also allows the application to make its identification available to the MMS UA. An application can be integrated with an MMS-capable terminal during its manufacturing stage, or it can be downloaded at any later time. In the former

case, the registration process is part of the manufacturing process. In the latter case, the registration process takes place right after downloading and installing the application in a terminal. The MMS specifications acknowledge both the registration process and the interface between an application and an MMS UA (i.e., application-interface), but neither is defined there. An interface between two entities is usually defined in a specification to ensure interoperability between the entities. As the application-interface is within an entity, both the OMA and the 3GPP have decided not to define it for the time being. Moreover, the availability of related MMS specifications would be delayed, had they spent time for defining the interface. Java community process (JCP) has defined some application programming interfaces (API) for the interface [15]. But, the APIs are not for any general application, as those are limited for the Java™ applications only.

Though it may not be reasonable to define the details of the registration process in the MMS specifications, at least it is important to have common understanding about both the format of data communication and what data should be communicated across the application-interface. Without such common understanding, there is a risk that a downloaded application would not be able to use MMS as a transport in different terminals. An application-developer has to agree about the same with each terminal manufacturer individually to avoid the risk. This would hinder the use of MMS as transport for application data, and thus, mass availability of applications in mobile communication.

4.1. The Format of Data Communication Across the Application-Interface

The format of data communication across the application-interface is expected to be simple, so that it can be easily parsed. Simplicity is important to make it easy for an MMS UA to construct a multimedia message from an instance of the format and vice versa. Simple format would also ease and accelerate application development process. As the format is expected to carry only the components of a message that an application wants to deliver to its peer application, information about underlying transport details and transmission mechanism should not be carried by the format. Moreover, the format should be extensible, so that it can describe all the required MMS headers useful for an application. Both extensibility and simplicity are also important to make the

same format useful for other purposes of MMS. Few additional uses of a similar format in MMS are outlined at the end this section. Different potential formats are considered below for the data communication across the application-interface, starting with the formats that are already in use for MMS, based on the expectation described in this paragraph.

4.1.1. *MIME-based binary encoded format*

As described in Section 2, an MM is transmitted in MIME-based binary encoded format [9], where the encoding rules are very strict but extensive. If it is used across the application-interface, an application also needs to support the rigorous format. This needs one more cycle of unwelcome encoding and decoding of the format both in the originating and recipient side, requiring more time and processing power, which are usually critical issues in a mobile terminal. The situation can be severe, if there are multiple applications in a terminal using MMS as transport. A typical MIME-based binary encoded multimedia message contains some headers only useful for transporting and controlling purposes in the network or in the recipient MMS UA (e.g., MMS version, transaction identification, message type). This information has no use in an application. Using the format across the application-interface would make these unnecessary headers available in an application, requiring a screening process to filter out the unnecessary information. Rather, it is more effective, if an MMS UA provides only the necessary information to the application agreed during the registration process, as an MMS UA needs to decode and screen the information of a multimedia message in any case for other purposes. Moreover, the MIME-based binary format is defined for transporting an MMS message across networks. As a message could be transmitted over any restricted network, the format also needs to consider 7-bit transmission. The format across the application-interface does not need to consider such lower-level transport mechanism or transmission restriction aspects, as the interface is within an entity. Considering all these points, the MIME-based binary encoded format of an MMS message may not be suitable across the application-interface.

4.1.2. *Synchronized multimedia integration language (SMIL)*

As mentioned in section 2.1, SMIL is another format used in MMS for describing the rendering instruction

of the content of a message. It is based on XML, and thus, it can be easily parsed. SMIL is designed dedicatedly for presentation purposes, like Hyper Text Markup Language (HTML) is designed for browsing. All the elements and attributes in SMIL have defined meanings, and it is not extensible to describe headers used in MMS. So, SMIL is not also a suitable format for the application-interface.

4.1.3. *Simple object access protocol (SOAP)*

SOAP, another XML-based format, is an envelope-like format defined mainly for exchanging data over a protocol like HTTP. SOAP is also used in MMS as the format across the VASP-interface [10]. SOAP defines a comprehensive range of encoding data, so that a SOAP message can be easily transported over lower-level protocol. There is no such need in the application-interface of encoding principle or transportation over lower-level protocol. So, SOAP may not also be a suitable format for the purpose here, though SOAP is extensible to define new elements and attributes.

Other XML-based formats are not separately considered here, as those may not be suitable as well due to the similar reasons mentioned in this or the previous section.

4.1.4. *extensible markup language (XML)*

XML itself is considered here, as it is different from some XML-based formats in some respects. XML is a general markup language for describing information. Unlike some other markup languages (e.g., SMIL, HTML), XML is not related to any specific application or purpose. It is possible within XML to define own elements and attributes to make information meaningful to any application, and thus, it can be easily extended. It is also easy to parse. Moreover, both headers and multimedia content can be easily described within an XML file, so that it is easy to construct a multimedia message from an XML file and vice versa. In this case, top-level headers with corresponding values could be included within an XML file, while the content can be referred from the file. So that, the content is easily identified and accessed from the storage when forming a multimedia message. XML could contain character set information. It allows the use of UTF-8, so that a separate encoding for non-US-ASCII header fields is not required. It also allows inclusion of content encoded in any format without conversion, as the content is only referred from an XML file. So, XML appears as a suitable format for the application-interface from different aspects.

4.1.5. *Binary XML*

Alternatively, binary XML format would be useful, where storage space in application or speed at the application-interface is a critical issue. As for example, if an application resides on a smart card within a terminal, both the storage on the smart card and the speed across the interface with the smart card could be limited. The binary XML format requires definition of the binary values to be used within an XML file. It is another advantage for XML, as it can be switched to binary XML to tackle the critical cases described above.

Beside using across the application-interface, XML could also be used as a common format to describe the components of a multimedia message for other purposes. As for example, a content provider can provide an XML file as a template for messages. The templates could be stored in a terminal or in a smart card at manufacturing or subscription stage respectively, so that a user can use it for creating a multimedia message. Moreover, an XML file could also be used to store a multimedia message to a device that does not fully support all the functionality of an MMS UA.

4.2. The Data to Be Communicated Across the Application-Interface

Since XML is identified as a suitable format for data communication across the application-interface in Section 4.1 above, now the question is what data should be communicated across the interface using the format. As headers and content within a message are different kinds of data, it is reasonable to address those under different elements in an XML file. As already mentioned before, all the MMS headers are not useful for an application. So, only a shortlist of MMS headers, which are useful for an application, should be communicated across the interface. Referring the content of a multimedia message from an XML file would serve the purpose, as described in Section 4.1.

If an application uses MMS as the transport for both sending and receiving data, the information flow across the application-interface is bi-directional. When an application wants to send its data using MMS, it generates an XML file that includes the components of a multimedia message (i.e., some headers with corresponding values and references to content), and sends the XML file to the MMS UA across the application-interface. At the same time, the application makes the content, which is referred from the XML file, available in a common storage, so that the MMS UA can construct the message based on the information available in the XML file. On the other

hand, when an MMS UA receives a multimedia message, indicating in the top-level MMS header that the message is for an application, the MMS UA extracts the content from the received multimedia message, and stores those in a common storage. The MMS UA then constructs an XML file, which includes the useful headers with corresponding values. The XML file also refers the content of the multimedia message. The MMS UA, then, makes the XML file available to the recipient application across the application-interface, so that the content is properly handled by the application. An example XML file, generated by an originating application, to inform an MMS UA about the components of a multimedia message is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<mmsmessage>
<header>
  <to>+358701111111</to>
  <subject>move#1</subject>
  <priority>normal</priority>
  <applic_id>com.abc.Chess</applic_id>
  <reply_applic_id>com.abc.Chess</reply_applic_id>
  <aux_applic_info>
    <rec_chnl>3</rec_chnl>
    <ori_chnl>2</ori_chnl>
  </aux_applic_info>
  <content_type>multipart/mixed</content_type>
</header>
<content>
  <number_of_parts>2</number_of_parts>
  <part>
    <type>text/plain</type>
    <filename>"move1.txt"</filename>
  </part>
  <part>
    <type>image/svg+xml</type>
    <filename>"aftermove1.svg"</filename>
  </part>
</content>
</mmsmessage>
```

The reference of both schema and Document Type Definition (DTD) are missing from the above example XML file. In practice, schema and/or DTD are expected to be defined to set constraints within the structure of an XML file used across the application-interface. An instance of an XML file, communicating data across the application-interface, would refer the defined schema and/or DTD.

It is assumed in the above example that a user is playing chess using a chess application in her terminal against other users having the same chess application in their respective terminals. It is also assumed that all the terminals are MMS-capable. After each move made by a user on the chessboard, the chess application sends the information about the move to its peer application using MMS as the transport. Each move is expressed by a text and a graphics (Scalable Vector Graphics—SVG) file, describing the move textually and showing the situation on the chessboard after the move graphically, respectively. The example XML file shown above is created by a chess application to send information about the first move to the MMS UA across the application-interface.

The root element of the file is `mmsmessage`, and it has two child elements `header` and `content` to describe headers and content within the multimedia message, respectively. Some of the headers within the `header` element are self-explanatory, and are not described further here. Elements `applic_id`, `reply_applic_id`, and `aux_applic_info` correspond to the newly defined application-specific headers, described in Section 2.2. Elements `applic_id` and `reply_applic_id` define the identification of the recipient and the originating application, respectively. Both the identifications are same in the example XML file above, though those can be different in practice. An application developer is expected to provide a unique identification for each application it develops. Rules for having globally unique identification for a Java™ application are defined in Reference [15]. Alternatively, a new MIME type can be registered for an application in IANA to get its unique identification. As for example, top-level media type “application” can be used to register an application. As the time required to register a new MIME-type can delay the deployment of even a simple application, the former approach can be time-efficient to meet instant market demand. The element `aux_applic_info` has two child elements, `rec_chnl` and `orig_chnl`, to identify the channel number within the recipient and the originating chess application respectively. The content of the message (a text and a vector graphics file in this case) are referred from the `content` element. A child element `part` is used for each content here. All other required headers in the message are generated and added to the message by the MMS UA. The multimedia message created, based on the above example XML file, by the MMS UA is shown below in text format.

```
X-Mms-Message-Type: m-send-req
X-Mms-Transaction-ID: abcdefghijk
X-Mms-MMS-Version: 1.3
From: +358601234567
To: +358701111111/Type=PLMN
Subject: move#1
X-Mms-Priority: Normal
X-Mms-Delivery-Report: Yes
X-Mms-Applic-ID: com.abc.Chess
X-Mms-Reply-Applic-ID: com.abc.Chess
X-Mms-Aux-Applic-Info: rec_chnl#3
X-Mms-Aux-Applic-Info: ori_chnl#2
Content-Type: multipart/mixed;
  boundary = ''54321''
```

-54321

```
Content-Type: text/plain;
  name = ''move1.txt''
Content-Transfer-Encoding: binary
```

...text content...

-54321

```
Content-Type: image/svg+xml;
  name = ''aftermove1.svg''
Content-Transfer-Encoding: binary
```

...graphics content...

-54321-

When the multimedia message is received by the recipient MMS UA, it recognizes from the top-level application-specific headers that the content of the message is for the chess application. So, it extracts and stores the content, and generates a similar XML file that includes useful headers and refers the content. In this case, the `from` element replaces the `to` element in the above XML file. The recipient MMS UA sends the XML file to the recipient chess application in the same terminal across the application-interface. Upon receiving an XML file, the chess application renders the new move on the chessboard for the user.

The contents of the XML file presented above are not related. In practice, it is possible that the contents are related, and a SMIL file describes the rendering instruction of the related contents. Another example XML file is shown below, where the main difference in the XML file compared with the previous one is the contents are related here.

```
<?xml version='1.0' encoding='UTF-8'?>
<mmsmessage>
  <header>
    <to>+358701111111</to>
    <subject>move#1</subject>
    <priority>normal</priority>
    <applic_id>com.abc.Chess</applic_id>
    <reply_applic_id>com.abc.Chess</reply_applic_id>
    <aux_applic_info>
      <rec_chnl>3</rec_chnl>
      <ori_chnl>2</ori_chnl>
    </aux_applic_info>
    <content_type>multipart/related</content_type>
    <rootcid>950100xyz@abc.com</rootcid>
  </header>
  <content>
    <number_of_parts>3</number_of_parts>
    <part>
      <type>application/smil</type>
      <filename>"move1.smil"</filename>
      <cid>950100xyz@abc.com</cid>
    </part>
    <part>
      <type>text/plain</type>
      <filename>"move1.txt"</filename>
      <cid>950101xyz@abc.com</cid>
    </part>
    <part>
      <type>image/svg+xml</type>
      <filename>"aftermove1.svg"</filename>
      <cid>950102xyz@abc.com</cid>
    </part>
  </content>
</mmsmessage>
```

In the above example XML file, `rootcid` element is used to indicate that the SMIL file is the first content in the MIME-based multimedia message. Each content is identified by the element `cid`, which is also used in the SMIL file to refer different contents. The multimedia message created by an MMS UA, based on the example XML file above, is shown below in text format.

```
X-Mms-Message-Type: m-send-req
X-Mms-Transaction-ID: abcdefghijk
X-Mms-MMS-Version: 1.3
From: +358601234567
To: +358701111111/Type=PLMN
Subject: move#1
```

```
X-Mms-Priority: Normal
X-Mms-Delivery-Report: Yes
X-Mms-Applic-ID: com.abc.Chess
X-Mms-Reply-Applic-ID: com.abc.Chess
X-Mms-Aux-Applic-Info: rec_chnl#3
X-Mms-Aux-Applic-Info: ori_chnl#2
Content-Type: multipart/related;
    boundary = '54321';
    start = '<950100xyz@abc.com>';
    type = 'application/smil'
```

-54321

```
Content-Type: application/smil;
    name = 'move1.smil'
Content-Transfer-Encoding: binary
Content-ID: <950100xyz@abc.com>
```

...smil content

-54321

```
Content-Type: text/plain;
    name = 'move1.txt'
Content-Transfer-Encoding: binary
Content-ID: <950101xyz@abc.com>
```

...text content...

-54321

```
Content-Type: image/svg+xml;
    name = 'aftermove1.svg'
Content-Transfer-Encoding: binary
Content-ID: <950102xyz@abc.com>
```

...graphics content

-54321-

5. Conclusion

The scope of MMS has been gradually extended from person-to-person, to content-to-person and person-to-service messaging. Very lately, standardization bodies have extended the scope of MMS further to cover application-to-application messaging as a new feature, so that an application can use MMS for transporting its data to its peer application. Section 2.2 outlines some example uses of the feature. The work in the standardization bodies, so far, is mostly about defining application-specific top-level MMS headers, as described in the same section.

Defining the headers is important, but may not be enough for smooth, secure, and mass deployment of the new MMS feature. It is also critical that the new MMS-capable terminals supporting the new feature are compatible with the existing MMS-capable terminals that do not support the feature. This paper provides a solution, described in Section 3, to make the new feature backward compatible. The same solution also makes the feature forward compatible with the future applications, so that an MMS-capable terminal supporting specific applications does not hinder the use of MMS as transport for any new application, allowing instantaneous deployment of a wireless application.

Another important missing item from the MMS specifications about the new feature is the definition of a format of data communication across the application-interface between an application and MMS in a terminal. An analysis over different possible formats in Section 4.1 reveals XML as the suitable format in this regard. A common format across the interface would allow mass development and deployment of wireless applications independent of manufacturing a mobile terminal. Examples of XML file are depicted in Section 4.2 to show what data should be communicated across the interface.

References

1. Mostafa M-E. Implementation solutions for the interworking between MMS and streaming. *International Journal for Communication Systems* 2003; **16**(10): 877–892.
2. Mostafa M-E. Improved implementation solution and general mobile network architecture for the interworking between MMS and streaming. *International Journal for Communication Systems* 2006; **19**(3): 335–357.
3. Mostafa M-E. MMS—the modern wireless solution for multimedia messaging. In Proc. *the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2002)*, Lisbon, Portugal, Sep 2002; vol. 5 of 5: 2466–2472.
4. Peersman G, Cvetkovic S, Griffiths P, Spear H. The global system for mobile communications short message service. *IEEE Personal Communications* 2000; **7**(3): 15–23.
5. Bodic GL. *Multimedia messaging service: an engineering approach to MMS*. Wiley: England, 2003.
6. Bodic GL. *Mobile messaging technologies and services SMS, EMS and MMS*. Wiley: England, 2003; pp. 197–334.
7. Ralph D, Graham P. *MMS: technologies, usage and business models*. Wiley: England, 2003.
8. Mostafa M-E. Interworking between MMS and streaming. In Proc. *IASTED International Conference on Communication Systems and Networks (CSN 2002)*, Malaga, Spain, Sep 2002; 376–383.
9. OMA-TS-MMS-ENC-V1_3-20050927-C. Multimedia messaging service encapsulation protocol. *Open Mobile Alliance*; candidate version 1.3, September 2005.

10. 3GPP TS 23.140. Multimedia Messaging Service (MMS)—functional description. *Technical Specification Group Core Networks and Terminals, 3rd Generation Partnership Project*; stage 2, release 6, version 6.10.0, June 2005.
11. OMA-AD-MMS-V1_3-20050617-C. MMS Architecture Overview. *Open Mobile Alliance*; candidate version 1.3, June 2005.
12. OMA-UAPProf-v2_0-20030520-C. User Agent Profile. *Open Mobile Alliance*; candidate version 2.0, May 2003.
13. OMA-TS-MMS-CTR-V1_3-20050927-C. Multimedia Messaging Service Client Transactions. *Open Mobile Alliance*; candidate version 1.3, September 2005.
14. Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, Berners-Lee T. Hypertext Transfer Protocol—HTTP/1.1, RFC 2616, *Internet Engineering Task Force*, June 1999.
15. JSR 205. Wireless Messaging API (WMA) for Java™ 2 Micro Edition. *JSR 205 Expert Group, Java Community Process*; version 1.0.2 (final release), May 2004.



Author's Biography

Miraj-E-Mostafa has been involved with different multimedia communication related research and development works for Nokia Corporation since the beginning of 1998. Presently, his main role is to look after different technologies involved in messaging and related services (e.g., Presence, Group Management) in mobile communications. He has been active in standardization works in 3GPP and OMA. Miraj got his Bachelor of Science in Electrical and Electronics Engineering and Master of Engineering in Telecommunications from Bangladesh University of Engineering and Technology and Asian Institute of Technology in 1995 and 1997, respectively. He is also pursuing his post-graduate studies in multimedia-based Mobile Communications in Tampere University of Technology.