LIVESMOBILE: Extending the LIVES system to support SMS and MMS for mobile learning

by

Kuljeet Singh

B.E., Army Institute of Technology, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

January 2011

© Kuljeet Singh, 2011
Abstract

The past decade has seen a proliferation of cellular devices actively being used in day to day lives by people in developing nations. Education however still remains a challenge as children are either deprived of basic education or are victims of favoritism due to deep-rooted inequalities linked to family wealth, gender, ethnicity, language and location. Less educated adults face problems due to lack of skills or qualifications required to get decent jobs. The proper use of educational material can tackle some of these problems and in general improve the general quality of life amongst this section of the population. Learning through Interactive Voice Education System (LIVES) is a system developed at the University of British Columbia (UBC) which aims at providing education to people in developing countries by using voice-based lessons delivered using a pre-existing mobile phone infrastructure. LIVES offers a two-way communication channel between the user and the system and thus has the ability to offer a customized learning experience based on the users’ preferences. In the LIVES system, this customization is offered by providing the user with a list of options through the use of voice dialogues within calls at which the user chooses a particular option. In this the-
sis, we propose a way to integrate Short Message Service (SMS) and Multimedia Message Service (MMS) into the LIVES system to enhance the learning experience of the end-user. By using SMS, we propose that the user can customize his learning experience in ways which are hard to achieve through voice. Moreover, the addition of MMS to the LIVES system provides an additional mechanism for content delivery to the student. We propose that the addition of MMS capability to the LIVES system can tackle some of the challenges faced during content delivery by using voice as a delivery mechanism.
# Table of Contents

Abstract ................................................................. ii  
Table of Contents ..................................................... iv  
List of Tables ........................................................... vii  
List of Figures ........................................................... viii  
Glossary ................................................................. x  
Acknowledgments ....................................................... xiv  

1 Introduction ......................................................... 1  
  1.1 Motivation ....................................................... 1  
  1.2 Thesis Contributions ......................................... 4  
  1.3 Thesis Organization ............................................ 6  

2 Background and Related Work ................................. 7  
  2.1 Mobile Learning ................................................ 7  
  2.1.1 Benefits of Mobile Learning .............................. 9
5.1 Simulation Setup .................................................. 53
  5.1.1 Testing SMS Performance .................................. 53
  5.1.2 Testing MMS Performance ................................. 56
5.2 Simulation Parameters ........................................... 58
5.3 Simulation Results .............................................. 59
5.4 Critical Evaluation .............................................. 64

6 Conclusions and Future Work ................................. 69
  6.1 Conclusions .................................................. 69
  6.2 Future Work ................................................ 70

Bibliography .................................................... 73
List of Tables

Table 3.1  List of Features Supported Through SMS .......................... 23
Table 4.1  Kannel: Core Group ..................................................... 42
Table 4.2  Kannel: SMS Center (SMSC) Group ................................. 42
Table 4.3  Kannel: SMSBox Group ............................................... 43
Table 4.4  Kannel: SMS-Service Group ......................................... 44
Table 4.5  Mbuni: Mbuni Group ................................................... 46
Table 4.6  Mbuni: Multimedia Message Switching Center (MMSC) Group 46
Table 5.1  Kannel: SMS Loss Ratio .............................................. 62
Table 5.2  Average Monthly Phone and SMS Usage for U.S. Wireless Subscribers, 2008 ......................................................... 66
Table 5.3  Number of Messages Required to Complete Tasks Within LIVES-MOBILE ......................................................... 67
List of Figures

Figure 3.1 LIVES System Design ............................................. 19
Figure 3.2 MMSC Reference Architecture ................................. 25
Figure 3.3 MMS Messaging Passing Architecture ....................... 28
Figure 3.4 LIVESMOBILE System Design ................................. 31

Figure 4.1 Kannel Architecture ................................................ 35
Figure 4.2 Sample Solr Schema Used for LIVES ......................... 39
Figure 4.3 Call Originator for LIVESMOBILE ............................ 41
Figure 4.4 SMS Gateway Status ............................................. 49
Figure 4.5 SMS Gateway Administration ................................. 49
Figure 4.6 Search Server Configuration .................................. 50
Figure 4.7 SMS Message Dispatch ......................................... 50
Figure 4.8 MMS Message Assignment to Learning Units ............... 51

Figure 5.1 SMS Gateway Load Testing @1 SMS Message Per Second Using Virtual SMSC .......................... 60
Figure 5.2 SMS Gateway Load Testing @80 SMS Messages Per Second Using Virtual SMSC ......................... 61
Figure 5.3   SMS Gateway Load Testing @20 SMS Messages Per Minute
Using a Global System for Mobile Communication (GSM)
Modem ......................................................... 62

Figure 5.4   SMS Gateway Load Testing @40 SMS Messages Per Minute
Using a GSM Modem ........................................... 63

Figure 5.5   MMS Gateway Load Testing @60 MMS Messages Per Minute
Using Virtual MMSC ........................................... 64

Figure 5.6   MMS Gateway Load Testing @1 MMS Message Per Minute
Using Virtual MMSC ........................................... 65
## Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td><strong>LIVES</strong></td>
<td>Learning through Interactive Voice Education System (<a href="http://lives.cs.ubc.ca">http://lives.cs.ubc.ca</a>)</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>PDAS</td>
<td>Personal Digital Assistants</td>
</tr>
<tr>
<td>UMPCPS</td>
<td>Ultra-Mobile Personal Computers</td>
</tr>
<tr>
<td>MP3</td>
<td>Moving Picture Experts Group (MPEG)-2 Audio Layer 3</td>
</tr>
<tr>
<td>MP4</td>
<td>MPEG-4 Part 14</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Experts Group</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
<tr>
<td>MMS</td>
<td>Multimedia Message Service</td>
</tr>
<tr>
<td>LMS</td>
<td>Learning Management System</td>
</tr>
<tr>
<td>LCMS</td>
<td>Learning Content Management System</td>
</tr>
<tr>
<td>VOIP</td>
<td>Voice Over Internet Protocol</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communication</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>TDMA</td>
<td>Time Division Multiple Access</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>3GPP</td>
<td>Third Generation Partnership Project</td>
</tr>
<tr>
<td>SIM</td>
<td>Subscriber Identity Module</td>
</tr>
<tr>
<td>MT</td>
<td>Mobile Terminal</td>
</tr>
<tr>
<td>MO</td>
<td>Mobile Originated</td>
</tr>
<tr>
<td>PPP</td>
<td>Point-to-Point Protocol</td>
</tr>
<tr>
<td>MM</td>
<td>Multimedia Message</td>
</tr>
<tr>
<td>SMSC</td>
<td>SMS Center</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper text transmission protocol</td>
</tr>
</tbody>
</table>
MMSC  Multimedia Message Switching Center
VAS  Value Added Service
IP  Internet Protocol
MIME  Multipurpose Internet Mail Extensions
SMIL  Synchronized Multimedia Integration Language
XML  eXtensible Markup Language
OMA  Open Mobile Alliance
SMTP  Simple Mail Transfer Protocol
VASP  Value Added Service Provider
TCP/IP  Transmission Control Protocol/Internet Protocol
GPRS  General Packet Radio Service
HSPA  High-Speed Packet Access
URL  Uniform Resource Locator
WSP  Wireless Session Protocol
WDP  Wireless Datagram Protocol
EAIF  External Application Interface
SOAP  Simple Object Access Protocol
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLR</td>
<td>Home Location Register</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
</tbody>
</table>
Acknowledgments

First, I would like to thank my supervisor, Dr. Son T. Vuong, for his guidance, inspiration and encouragement. I am also grateful to Kurt Eiselt for being my second reader and for his useful comments which have improved this thesis.

Thanks to all the members of the NIC lab, who it was very pleasant to work with. In particular, I want to thank Jonatan Schroeder and Mohammed Alam for providing me with intensive details about the LIVES system. I would also like to thank Chen Zhang for being patient with the project and making some valuable contributions. Also, many thanks to my lab mates - Wei Li, Ricky Chen, Stanley Chu and Andrew Tjia.

Finally, I would like to thank my parents, my sister and my brother for their endless love and support.

KULJEET SINGH

The University of British Columbia

January 2011
Chapter 1

Introduction

*If you talk to a man in a language he understands, that goes to his head. If you talk to him in his language, that goes to his heart.*

— Nelson Mandela

1.1 Motivation

The past few decades have seen a proliferation in the use of cellular phones by people all over the world. By the end of 2009, there were an estimated 4.6 billion mobile cellular subscriptions, corresponding to 67 per 100 inhabitants globally. It is estimated that by the end of 2010, the number of mobile cellular subscriptions globally will reach five billion [23, 37]. This growth can primarily be attributed to the availability of advanced services and handsets in developed countries and increased take-up of mobile health services and mobile banking in the developing world.

It is worthy to note that the people who are using these mobile devices have
varying levels of skills and education. Most people using these devices in developed countries are educated and have a basic level of literacy, and thus are able to identify, understand, interpret, create, communicate, compute and use printed and written materials associated with diverse contexts [34]. In the developing world, due to lack of educational infrastructure, a large percentage of the population only possess a subset of these skills.

Learning through Interactive Voice Education System (LIVES) is a system developed at the University of British Columbia (UBC). The primary goal of LIVES is to spread education in the developing world by providing people with voice-based lessons which may be beneficial to them while carrying out their daily activities. The goal of LIVES is to provide just-in-time learning to its users which can be helpful to them in their daily activities or professional work areas.

LIVES offers a two-way communication channel between the student and the system. Unlike some of the previous voice-based approaches which use radio broadcasting or dial-in voice kiosks hosting voice sites, our system services the users request in real time as per the option chosen by the user. The user can however only choose the options presented to him during the voice call while it is in progress. There are two main factors motivating this thesis. The first is to explore the role of Short Message Service (SMS) as a customization and content delivery mechanism to allow the users to retrieve information based on their learning needs. The LIVES system does not have a channel through which the user can request information not provided through the voice dialogues. We developed an SMS server which lets the users retrieve information specific to their
learning needs and which also acts as a specialized just-in-time content delivery mechanism providing messages with small bits of information. We acknowledge that the SMS feature will be beneficial to the segment of the population with a basic level of language reading and writing skills. With the SMS feature, local community leaders can enrol new students into LIVES and register them in courses by sending an SMS message from the new users cellphone.

The second motivating factor behind this thesis is the use of voice as the only content delivery mechanism in the LIVES system. Voice has its limitations when trying to represent complex bits of information primarily due to the lack of a visual component. In this thesis, we explore the use of Multimedia Message Service (MMS) as an additional content delivery mechanism. We developed an MMS server which is capable of sending multimedia messages relevant to the course to its enroled users. We propose that the addition of MMS as an additional content delivery mechanism would allow the distribution of rich content which can also target the population without reading and writing skills.

The LIVES system has been tested in Tamil Nadu, India in 2010. We repeatedly received requests to implement additional content delivery and customization mechanisms which would allow the system to send and receive images and allow the user to control his/her own learning experience. We implemented a prototype in an attempt to address this specific need.
1.2 Thesis Contributions

This thesis proposes the following extensions to the LIVES system: (1) The implementation of an SMS mechanism by which the LIVES system can receive user’s requests which may contain queries for additional learning content; queries for real time information like weather updates; or requests to add new user profiles or make changes to existing profiles; (2) The implementation of an SMS mechanism by which the LIVES system can respond to user’s queries received through SMS requesting for additional information or with results of previous queries; and (3) The implementation of an MMS mechanism by which the LIVES system can send MMS messages with rich content to the users. A prototype implementation containing the proposed features has been implemented and evaluated. The following are the main contributions of this thesis:

1. We have provided a way by which the user can send SMS messages to the system. This has been done by integrating an open source SMS gateway with the LIVES system. The system can get information from the user in the form of text messages and can get a better idea of the users’ requirements as opposed to providing the user with a fixed set of options to choose from during a voice call. In the prototype implementation, the user can query the system for topics of interest and the system responds back with a list of items which matched the query. The user can ask the system to schedule a voice call containing that particular topic of interest by sending a text message. The system can also provide weather updates on request by the
user. New users can register to use LIVES by sending a text message.

2. We have provided a way by which the system can send MMS messages to the user. This has been done by integrating an open source MMS gateway with the system. The system can send additional information related to a particular course like facts and figures or instructional videos or presentations which enhance the student’s learning experience. The same can be used to send complex questions which cannot be asked through traditional voice-based calls and which require the user to work on a piece of paper to reach the solution. Compared to voice-based systems which are limited by their linear manner of presenting information, MMS makes it possible to display learning material (including questions) as a whole and does not suffer from such limitations. This capability can further extend LIVES in the future by providing a way to send graphs or reports to the user regarding the progress he or she has made so far in the course. In the prototype implementation, an instructor can upload new MMS messages to the system and assign them as learning content to pre-existing courses. The system sends these MMS messages to the students registered for those courses during the available times as indicated by their user profiles.

3. We have implemented a prototype which integrates the SMS and MMS functionality with the LIVES system. The original LIVES system uses Drupal as a front-end for the Learning Content Management System (LCMS) [19]. The LCMS for the prototype has also been implemented using Drupal. The
LIVESMOBILE LCMS exists as a separate installable module and is not part of the main LIVES module. The design of the LCMS however has been kept very similar to that of LIVES to facilitate future integration. The SMS and MMS servers and the Call Originator software constitute the LIVESMOBILE Learning Management System (LMS) which are responsible for sending and receiving short text messages, placing calls and delivering multimedia messages to students.

4. The prototype implementation was evaluated and performance measurements were made which are presented in Chapter 5 of this thesis.

1.3 Thesis Organization

This thesis consists of six chapters. Chapter 2 provides the background to mobile learning, as well as a detailed description of related work. In Chapter 3, we present the system design. We first present the design of the LIVES system onto which our system has been built. We then present the design of the LIVESMOBILE prototype and how it integrates with the existing LIVES system. Chapter 4 presents details regarding the prototype implementation. Chapter 5 presents the evaluation setup and evaluation results. We conclude the thesis and discuss potential avenues for future research in Chapter 6.
Chapter 2

Background and Related Work

This chapter provides background information on Mobile Learning and its benefits to the Learner and the Institution. Then we provide a survey of the related work classified according to the core technologies used by systems to promote education.

2.1 Mobile Learning

Mobile learning can be defined as "Any sort of learning that happens when the learner is not at a fixed, predetermined location, or learning that happens when the learner takes advantage of the learning opportunities offered by mobile technologies" [27]. In spite of the definition cited above, the concept of mobile education or mobile learning is still emerging and still unclear. There are different stakeholders and factors at work in this process of conceptualizing mobile education and the outcome is uncertain [31].
There are definitions and conceptualizations of mobile education and learning that define it purely in terms of its technologies and its hardware, namely that it is learning delivered or supported solely or mainly by handheld and mobile technologies such as Personal Digital Assistants (PDAS), smartphones, or netbooks. These definitions, however, are constraining, techno-centric, and tied to the current technological instantiations. Mobile devices and technologies are pervasive and ubiquitous in many modern societies, and are changing the nature of knowledge and discourse in these societies [31]. This in turn, alters both the nature of learning and the manner in which it is achieved. Learning which used to be delivered "just-in-case", can now be delivered "just-in-time", "just-enough" and "just-for-me". Finding information rather than possessing it or knowing it becomes the defining characteristic of learning generally and of mobile learning specifically.

Mobile Learning has the following core set of defining features. Mobile learning can take place at any time, at any location, in any environment including the traditional learning environment such as classrooms as well as workplaces, at home, in community locations and in transit. Mobile technologies may include mobile phones, smartphones, PDAS, Moving Picture Experts Group (MPEG)-2 Audio Layer 3 (MP3)/MPEG-4 Part 14 (MP4) players, handheld gaming devices, Ultra-Mobile Personal Computers (UMPCS), netbooks, handheld Global Positioning System (GPS) or voting devices, and specialist portable devices used in science labs, engineering workshops or for environmental or agricultural studies. Mobile learning involves connectivity for downloading, uploading and/or online working via wireless networks, mobile phone networks or both, and linking to institutional
systems e.g. virtual learning environments (VLEs) and Management Information Systems (MIS).

2.1.1 Benefits of Mobile Learning

Mobile Learning Network (MOLENET) is the largest UK based organization dedicated to the cause of Mobile Learning. It has conducted over 104 projects involving 40,000 learners and over 7,000 staff members for 3 years from 2007 to 2010 and has surveyed the effects of implementing Mobile Learning projects on a wide spectrum of User types with varying requirements [32]. A large number of benefits of mobile learning for learners, teachers, and institutions has been identified. An obvious benefit is that technology-supported learning can take place in many different locations. Mobile learning "provides learners with choice over and ownership over their learning", and "with good planning in place, mobile technologies can encourage creativity and innovation by both learners and teachers". It can also provide a safe, private, and non-judgmental environment for learners to try out ideas and make mistakes in order to progress.

At an institutional level, benefits of implementing Mobile Learning projects include improved learner attendance, retention and achievement, plus improved communication, staff motivation and increased ability of the institution to meet learner expectations [40].
2.2 Related Work

Mobile Learning as a concept has been around since the 1970s [20, 27]. There have been a large number of projects which have tried to use mobile technologies to implement an educational system. This section describes the most popular and well-known projects which have targeted mobile devices in order to effectively distribute educational content. The core features of each implementation are briefly discussed and a comparison is drawn out between each and the LIVES core system with the LIVESMOBILE prototype.

2.2.1 Web-Based Solutions

These Mobile Learning projects use the World Wide Web (WWW) as a primary medium of content delivery. Some of the important and successful projects in this domain include the following:

**OnPoint Digital**

The OnPoint Learning and Performance suite is an LMS/LCMS system which facilitates the creation of rich-media content, provides a platform to deliver that content to learners, and enables the management of learner performance and skill tracking on an ongoing basis [13]. It uses a web interface for content authoring and course creation as well as for learners/users to access the content. The users of the system need to have a web enabled device which can access Hyper Text Markup Language (HTML) pages over a wireless or mobile network. The ability to distribute learning content over the web allows such a system to provide rich
multimedia content directly to its end-users. This functionality however comes at a price to the end-user in terms of upgrading existing hardware or purchasing expensive devices to communicate over the Internet with the content server. In contrast, the LIVES system and the LIVESMOBILE prototype leverage the already available infrastructure and do not require any additional expense on behalf of the end-user [43]. This approach is better suited for people in developing countries where there is lower penetration of mobile Web capable phones.

**UpSide Learning**

UpSide Learning has used the recent popularity of Smartphones such as iPhone, Blackberry and Android devices to create a learning suite comprising quiz applications, HTML courses, mBooks for iPhone, custom videos and Augmented Reality applications on Android [26]. The LIVES system is primarily focused on spreading education in developing countries where the popularity of Smartphones or PDA! (PDA!)s is very restricted. Most of the users targeted by the LIVES system have low annual incomes and can only afford one cellular device per family. In such a case, only the technologies available through the mobile service provider and supported by the mobile terminal can be used. The LIVES system does not impose any new requirement on behalf of the service provider or the end-user and is thus an effective tool to distribute content in such scenarios.

**MoULe (Mobile and Ubiquitous Learning)**

MoULe is an online environment for collaborative learning. It integrates smart phones and portable devices to enable educational activities based on exploration
of a geographical place [18]. The system includes specific functionalities to search and access information spaces, to communicate and to annotate places according to their geographical coordinates. It associates Learning Resources to specific geographical sites. The system can be used through computers or through mobile devices. This form of learning is context aware learning, assuming the context is the users current location. In the LIVES system, all students who are enrolled into a particular course are believed to have the same context. Courses are tailored to the needs of a specific user population. We derive the context of the user based on the users geographical location, profession or interest in a particular course. The implementation of the LIVESMOBILE prototype lets the users search for new learning material based on changes to their context.

2.2.2 Voice-Based Solutions

These projects use Voice as their primary mechanism for content delivery. Some features of these projects which cannot be supported through voice require the Web as a mode of transport for educational content. Some of the popular and well known systems in this category are summarized below:

Learnosity

The core focus of the Learnosity Voice Learning Suite is to enhance learning of a new language through the use of spoken words and visual tools [38]. The Learnosity Connect project uses a voice-based framework to develop language skills within students. Students are matched in pairs by a computer and instruc-
tions for their conversation are set by the teacher. The students then use their cellular devices or Skype on a computer to connect with their conversation partner [41]. Typical conversations involve role-playing scenarios in a language they are trying to learn. The recorded conversations are stored for later assessment and feedback. This system allows students to put their learning to test and get useful feedback on their performance by leveraging the existing mobile infrastructure. In contrast, LIVES uses a voice-based framework for providing learning content to a target audience in a language they understand. In the case of LIVES content is created by an instructor and is distributed to the users as part of a course, more like the classroom scenario in which an instructor is giving a lecture to the students. This way, we can address a large section of the user population and distribute learning content in an efficient manner. The LIVESMOBILE prototype implementation allows the distribution of MMS content in addition to the voice content provided by LIVES to provide a more holistic learning experience.

**VoiKiosk**

Another project which uses Voice as a primary content delivery mechanism is the VoiKiosk system [29]. VoiKiosk uses the concept of a VoiceSite which is similar in meaning to a web site but is accessed through phone calls rather than a browser. The information in this case can be listened to rather than being read or seen. Each VoiceSite is responsible for an entire village in a rural population. Kiosk operators and villagers are allowed to create and listen to content. In this case, the end-user can control his or her interaction with the system by choosing
from a set of voice-based options. This system has many distributed voice sites which have to be managed individually and there is no central way to control the distribution of content. The LIVES system contains a central LCMS which allows instructors to manage the learning content from a central location for all the courses for which they have privileges. In addition, The LIVESMOBILE prototype allows the instructor to perform administrative tasks without the need to have access to a personal computer connected to the Internet by using SMS.

### 2.2.3 SMS-Based Solutions

These systems use SMS as their primary mechanism of content delivery. The use of SMS is very limited as a content delivery mechanism since it can only support 160 characters in one message [6]. So, these forms of Mobile Learning systems can address learning problems which do not have large chunks of data to send to the users.

**Grameen Foundation: AppLab**

In Uganda, Applab is establishing a network of Community Knowledge Workers, trusted village residents who are equipped with and trained on the use of mobile phones to share agricultural information with fellow rural, mostly poor farmers. The intermediaries inform farmers on market prices, improved agricultural techniques and weather forecast among other critical areas [35]. This system uses SMS technologies to allow users to send a text message with a keyword to pull out relevant information on a number of subjects related to the keyword. The sys-
tem primarily uses SMS as a content delivery mechanism. In the LIVESMOBILE prototype, SMS is used primarily as a querying mechanism. The content is delivered through voice or through MMS. We do not use SMS as a content delivery mechanism due to limitations on the size of SMS messages.

**FrontlineSMS**

FrontlineSMS is a free, open source software that turns a laptop and a mobile phone into a central communication hub. The program enables users to send and receive text messages with groups of people through mobile phones. Some of the popular projects which have been implemented using FrontlineSMS include violence against Children SMS helpline in Benin, Peacekeeping at Burundi elections, Texting for Life in Pakistan to provide support to people who have been displaced due to disaster or conflict [8, 17, 25]. In Iraq, it has been used by the country’s first independent news agency, FrontlineSMS:Medic is used to gather health data and assist in patient follow-up [22]. Even though these project do not directly contribute towards Mobile learning, they have used SMS as the underlying mobile technology to spread awareness among people in situations where other forms of communication were limited or unavailable. LIVESMOBILE also follows a similar approach and supports both Pull and Push models for SMS based communication. Instructors are able to create custom SMS messages and send them to students enroled in a particular course or to individual students. Students can send SMS messages to the system which can be stored and appropriate responses can be generated. LIVES has been deployed for testing in India and uses voice based lessons
for content delivery. A version of LIVES with the LIVESMOBILE prototype has yet to be deployed.
Chapter 3

System Design

In this chapter, we first present the design of the LIVES system. The first section gives an overview of the whole architecture of the LIVES system discussing each key component. We then present the design of the LIVESMOBILE system and illustrate the approach followed to integrate both the systems and depict the interaction between the key components of both the systems.

3.1 LIVES System Design

LIVES is an educational software system that delivers learning components via voice to its users (students) over the telephone network (or cellular network) and assesses user performance via stored numeric feedback from the users [42]. The LIVES system consists of two parts:

1. **LEARNING MANAGEMENT SYSTEM (LMS):** The LMS effectively and seamlessly delivers educational material over the telephone (and cellular)
network to multiple users at the same time. This component deals with communication, load balancing and security of the system.

2. LEARNING CONTENT MANAGEMENT SYSTEM (LCMS): The LCMS provides an interface for the management of the learning content. It comprises a set of technologies which provide a user interface which allows instructors to upload new material, create new courses, enrol students in a course and view progress reports of students enrolled in courses.

The main components of the LIVES system as described in Figure 3.1 are as follows:

1. **Main Server**: This is the main Asterisk server. It makes phone calls to all the users (students) [33]. This is achieved through Voice Over Internet Protocol (VOIP) providers in the prototype implementation but can be extended to use phone lines to make calls locally.

2. **Call Manager (AGI) Server**: This server deals with the interaction with the users (students). It defines how the voice clips should be played; how the user profiles should be managed; how the user will be prompted with options; and how the users feedback will be recorded. The LIVES Main Server (Asterisk) instructs the Call Manager server on how it should make the call.

3. **User Profile Database and Voice Database**: The User Profile Database keeps all the information related to the users (students) including available
time, student name, location, and phone number. User progress in course material is also tracked in the database. The voice database stores the modularized lesson and quiz voice clips.

4. Call Originator Software: This software instructs the LIVES Main Server to make calls to the students and play the lesson(s)/quizzes from the Voice Server according to the user information (rules) fetched from the User Profile Server.

Figure 3.1: LIVES System Design
3.2 LIVESMOBILE System Design

The LIVESMOBILE prototype is an extension to the LIVES system. The LMS primarily consists of an SMS server to send and receive SMS messages, the MMS server to send multimedia messages, a full-text search server to index and search through learning content from the database, and a modified Call Originator Software from LIVES used to dispatch Multimedia Message (MM) messages. The LCMS consists of a Drupal module which enables Learning Content Management.

3.2.1 SMS Server

The SMS server is responsible for implementing the core SMS functionality for our prototype. The first part of this section gives background information on SMS followed by the LIVESMOBILE SMS server design.

Background

SMS allows for sending and receiving messages between mobile phones. SMS was introduced in 1985 as part of the Global System for Mobile Communication (GSM) standard [28, 36]. Now, it is available in wireless technologies which use Code Division Multiple Access (CDMA) and Time Division Multiple Access (TDMA). The GSM and SMS standards were originally developed by European Telecommunications Standards Institute (ETSI) [21]. Now, the Third Generation Partnership Project (3GPP) is responsible for the development and maintenance of the GSM and SMS standards. Each SMS message can contain at most 140 bytes of data in the form of:
• 160 7-bit characters. This is suitable for encoding Latin characters like English alphabets.

• 70 16-bit unicode characters. SMS messages containing non-Latin characters like Chinese characters use 16-bit character encoding.

Besides text, SMS messages can also carry binary data. It is possible to send pictures, ringtones, operator logos, wallpapers, animations, business cards and Wireless Application Protocol (WAP) configurations to a mobile phone with SMS messages. SMS has become a massive commercial industry, worth over 81 billion dollars globally as of 2006 [1].

SMS services are content services initiated by an SMS message to a certain (usually short) phone number, which then answers with requested content if available. The LIVESMOBILE SMS service can be thought of as being a similar kind of service in which the user sends an SMS message to the SMS server requesting for information.

When an SMS service needs to be used, the client Mobile Terminal (MT) sends an SMS message to a certain number, which points to a certain SMS Center (SMSC) responsible for that number (plus possibly many others). This SMSC then sends the message onward to the specified receiver using an SMSC specific protocol. Practically, every different kind of SMSC uses a different protocol, and an SMS gateway is used to handle connections with SMS centers and to relay them onward in a unified form.
An SMS gateway can also be used to relay SMS messages from one GSM network to another, if the networks do not roam messages normally.

**LIVESMOBILE SMS Server**

The SMS server is responsible for processing all incoming text messages through the users’ cell phones. In order for the system to recognize the user request, the incoming messages have to be structured in a predefined format which the system understands. The system is able to determine the type of requests being made by the users and can process it on their behalf. The SMS server also has the capability to send text messages to the users. If a certain user request requires additional information from the user, the system can let the user know by sending additional text messages back to the user requesting for additional information.

The prototype implementation allows the deployment of LIVESMOBILE with SMS support only. The MMS server depends on the SMS server for deployment.

Table 3.1 shows the list of features supported through SMS for users of the LIVESMOBILE prototype.

### 3.2.2 MMS Server

The MMS server is responsible for implementing the core MMS functionality for LIVES. The first part of this section gives background information on MMS followed by the LIVESMOBILE MMS server design.
Table 3.1: List of Features Supported Through SMS

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
<th>Syntax</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>help</td>
<td>provides a list of all the commands</td>
<td>help</td>
<td>displays a list of all the commands</td>
</tr>
<tr>
<td></td>
<td>provides help information on a particular topic</td>
<td>help [topic]</td>
<td>display help related to the topic.</td>
</tr>
<tr>
<td>find</td>
<td>search for available voice lessons</td>
<td>find [query]</td>
<td>returns top 3 search results</td>
</tr>
<tr>
<td>sched</td>
<td>schedule a lesson</td>
<td>sched [query] [num]</td>
<td>schedules search result number from query</td>
</tr>
<tr>
<td>weather</td>
<td>display weather information</td>
<td>weather [loc]</td>
<td>display weather information for a location</td>
</tr>
<tr>
<td>reg</td>
<td>register a new user</td>
<td>reg [Name], [UName], [Pwd], [StartTime], [EndTime]</td>
<td>create a new user in the system</td>
</tr>
</tbody>
</table>

Background

MMS is a standard way to send messages that include multimedia content to and from mobile phones. It extends the SMS capability which allows sending of messages containing 160 7-bit encoded characters. It is a non real-time delivery service which utilizes the store and forward model. The standard is developed by the Open Mobile Alliance (OMA) and during its development was a part of 3GPP and WAP groups [39]. By 2008, the worldwide MMS usage level had passed 1.3 billion active users [30] who generated 50 billion MMS messages[2] and produced annual revenues of 26 billion dollars [14].
The MMS technology is defined by two important standards: 3GPP (3GPP TS 23.140)[24], and OMA standard. These two standard bodies cooperate to collectively define the MMS protocols. The following protocols are part of the MMS protocol suite:

1. **MM1**: This protocol specifies the interaction between a mobile device and the Multimedia Message Switching Center (MMSC). It defines how mobile phones send and receive messages through the MMSC.

2. **MM3**: This protocol is used between the MMSC and other messaging systems. It defines the requirements for how an MMSC must be able to interoperate with other messaging systems such as Simple Mail Transfer Protocol (SMTP).

3. **MM4**: This protocol defines the interaction between two MMSC’s. It is an SMTP based protocol with additional headers defined.

4. **MM5**: This protocol defines the interaction between the MMSC and the Home Location Register (HLR). The HLR is a central database which contains details of each mobile phone subscriber that is authorized to use the GSM core network. The HLR’s store details of every Subscriber Identity Module (SIM) card issued by the mobile operator.

5. **MM7**: This protocol is used to allow Value Added Service Provider (VASP) applications to send and receive MMS messages via an MMSC. The MM7 implementation defined by the 3GPP standard is a SOAP-based protocol.
which involves the exchange of eXtensible Markup Language (XML) and Multipurpose Internet Mail Extensions (MIME) content over HTTP-POST.

Figure 3.2 depicts the MMSC reference architecture.

In addition to the above mentioned protocols, there are also the MM2, MM5, MM6, MM8, MM9, MM10, MM11, External Application Interface (EAIF) and other vendor specific proprietary variation of the MM7 protocol. Most of the protocols within the 3GPP specification are defined at an abstract level except for MM4 and MM7 for which complete implementation details have been provided.
The OMA group provides complete implementation details for interfaces where none are specified in the 3GPP standard, in particular the MM1 protocol.

MMS is designed to be transported largely over Internet Protocol (IP) rather than traditional GSM networks. It is also designed to inter-operate with other IP services such as email and WAP. MMS messages are typically transported over WAP and are encoded using WAP MIME formats. When a mobile phone receives an MMS, it first receives an MMS notification message through SMS (WAP Push). The MMS notification message contains header information about the MMS message and a Uniform Resource Locator (URL) pointer which the recipient must fetch in order to download the contents of the MMS message.

This URL pointer is a dynamically generated URL for the MMS message content which is stored on the MMSC. In a typical phone-to-phone MMS transaction, the process of sending and receiving the MMS message works like this:

1. The sending phone initiates a data connection that provides Transmission Control Protocol/Internet Protocol (TCP/IP) connectivity, usually over General Packet Radio Service (GPRS).

2. The sending phone performs an Hyper text transmission protocol (HTTP) POST to an MMSC by first encoding the message using the MM1 protocol. The encoded MMS message includes all of the content of the MMS message, as well as the header information, including a list of intended recipients for the message.

3. The MMSC receives the MMS message submission and validates the message
4. The MMSC stores the contents of the MMS message and makes it available as a dynamically generated URL link.

5. The MMSC generates an MMS notification message, which is sent via WAP push over SMS to the message recipient(s). The MMS notification message contains a URL pointer to the dynamically generated MMS content.

6. The recipient receives the MMS notification message. It then initiates a data connection that provides TCP/IP network connectivity (usually over GPRS).

7. The recipient phone performs an HTTP (or Wireless Session Protocol (WSP)) GET to retrieve the MMS message content URL from the MMSC.

Figure 3.3 depicts the main architecture of the MMS message passing mechanism.

The following are the main components described in Figure 3.3:

- **MMS Client:** A device through which the user receives or sends multimedia messages. This might be a phone or a PC-based MMS client. The Client sends messages to and receives messages from the MMSC using WAP/HTTP as transport. The MMSC is the switching center of the service provider. In the prototype implementation, Rogers was chosen as the service provider and a SIM card with an active data connection was used to test the MMS sending capability.
Figure 3.3: MMS Messaging Passing Architecture

- **MMS Gateway**: Switches messages between different MMS clients and between MMS and Email. The Gateway may also interface with other gateways to exchange messages destined for foreign networks. This is also more properly known as the MMSC.

- **MMS Server**: This component provides persistent storage of messages on the network. Typically users can access stored messages via a web interface.

- **Other MMS systems**: Other systems, such as Third Party MMS systems (e.g.
MMS Value Added Service (VAS) providers) can interface to the MMSC to receive and send MMS content. The Interface used is termed MM7. In the prototype implementation, this role is carried out by the LIVESMOBILE MMS module.

- **SMSC**: The MMSC utilizes WAP Push to send notifications to MMS Clients. These are typically sent using SMS as the bearer service, hence the need for a link to a Short Message Service Center.

In order to send MMS messages from the LIVESMOBILE prototype, we had the choice of either implementing an MMSC which can send the MMS messages directly to the phone or to implement a VASP which sends the MMS messages to an intermediate MMSC (from the service provider) which is responsible for sending the MMS messages to the end-users. The advantages and disadvantages of both are discussed below:

1. **Implementing an MMSC**
   
   (a) The MMSC is responsible for sending the MMS message directly to the phone. If the subscriber is located on a different cellular network, the MMSC should have appropriate routing information and necessary privileges to be able to interact with MMSC’s from other service providers.

   (b) The MMSC has to provide a public interface which can be accessed from all cellular devices affiliated to that particular MMSC. While
sending an MMS message, a local copy of the message needs to be created and a URL has to be sent to the receiver which points back to the local copy. This required an overhead in terms of maintaining local copies.

(c) Most cellular devices are pre-configured with settings (IP address, port number) specific to their service provider’s MMSC. These MMSC’s only allow MMS related traffic over the GPRS connections established to them. So, any message stored on our MMSC would be considered as non-MMS traffic and would not be accessible to the end-users unless they change their phone settings to point directly to our MMSC. Making a user point to our MMSC would disable them from receiving any MMS messages which they would usually get from their own MMSC.

2. **Implementing a VASP**

   (a) A VASP is responsible for sending MMS messages to an MMSC hosted by a service provider. The MMSC sends the messages directly to the subscriber if the subscriber is located on its own network or forwards the message to the relevant MMSC if the subscriber is on a different network.

   (b) There is no need to maintain a local copy and provide an external URL accessible over the mobile network. So, the VASP does not need to maintain an active GPRS connection at all time. The VASP only establishes a GPRS connection with the operator MMSC when it needs
to send an MMS message. However, a local copy can be maintained incase there is a need to resend the MM message.

(c) Since the VASP will be sending all traffic to an operator MMSC, the end-users do not have to change the settings on their phones. They will always retrieve LIVES MMS messages through their own operator’s MMSC.

(d) One drawback of this approach is that the operator MMSC usually has a billing mechanism in place and they can charge the subscriber and the LIVESMOBILE server for each MMS received or for the amount of data sent over their phone network. The charges vary from operator to operator.

Figure 3.4: LIVESMOBILE System Design
Figure 3.4 depicts the various components of LIVESMOBILE.

During the design of the LIVESMOBILE system, we decided to implement the MMS functionality by implementing a VASP. We decided that it would be best to not have the end-user change the settings on his cell phone to be able to interact with our system as that would require a certain amount of technical understanding on the part of the user. With our implementation, users of the system will be able to receive MMS messages from the LIVES server out of the box without having to make any changes to their phone configurations.
Chapter 4

System Implementation

This section discusses the implementation of the LIVESMOBILE prototype. The first part discusses an open source SMS gateway used to implement the SMS server for LIVESMOBILE. The subsequent part presents Mbuni, which is an open source MMS gateway used to provide the MMS capabilities to LIVESMOBILE. The second to last part discusses Apache Solr which has been used to implement the indexing mechanism for all of the learning content on LIVES. The last part discusses the Call Originator software used to place voice calls and dispatch MMS messages. It is all put together in the next section which provides complete implementation details of the prototype.

4.1 Kannel: Open Source SMS Gateway

In order for the LIVESMOBILE prototype to be able to send and receive text messages, there is an additional system requirement which is not needed for the LIVES
module. Typically, an SMS message can be sent using a modem attached to the SMS server or through the use of online bulk SMS gateways. The prototype implementation uses a GSM modem with a SIM card that has an active connection with a service provider to send and receive text messages. In order to communicate with the GSM modem, we have chosen to use Kannel v1.4.3 which is an open source WAP and SMS gateway [10]. A more recent version of Kannel v 1.5.0 had been released after the successful implementation of the prototype but hasn’t been tested with the system. This section describes the functional parts of Kannel and how it fits in with the prototype implementation.

4.1.1 Overview of Kannel

Kannel works as an SMS gateway talking with many different kinds of SMSC’s and relaying the message onwards to content providers in the form of HTTP requests. The content providers then answer this HTTP request and the answer is sent back to the MT with the appropriate SMSC connection using the SMSC specific protocol. Kannel achieves this by abstracting each SMSC protocol to a well known HTTP protocol.

In addition to serving Mobile Originated (MO) SMS messages, Kannel also works as an SMS push gateway - content providers can request Kannel to send SMS messages to the MT. Kannel then determines the correct SMSC to relay the SMS message to and sends it using the SMSC specific protocol. Using this approach, the content provider does not need to know about any specific SMSC protocol and can use the unified Kannel SMS sending interface. The LIVESMOBILE prototype
Figure 4.1: Kannel Architecture

uses this approach.

The gateway has interfaces to three external agents:

- SMS centers, using various protocols
- HTTP servers, to fetch WAP and SMS content and to push WAP content.
- WAP phones, implementing the WAP protocol stack and WAP push client.

For the purposes of our prototype implementation, we are primarily concerned with the SMS center interface and the HTTP interface.
The various functional parts of Kannel are presented below.

- **bearerbox:** The Kannel bearerbox is responsible for implementing the bearer level of WAP (the Wireless Datagram Protocol (WDP) layer.) It connects to the SMS centers. Kannel implements SMS as a WAP push bearer. The SMS gateway functionality has to interact with the WDP layer. It uses the single SMS center connection for sending SMS messages and as a WAP bearer.

  There is only one instance of the bearerbox process. This requirement is imposed because each SMS center can only be connected to by one client at any given time.

- **smsbox:** The smsbox implements the rest of the SMS gateway functionality. It receives text messages from the bearerbox, interprets them as service requests, invokes a URL corresponding to the kind of request and responds to them in a manner specified by the URL.

- **wapbox:** The wapbox implements the WAP protocol stack and WAP Push (An application level protocol).

  There can be multiple smsboxes and wapboxes. Having multiple smsboxes is beneficial to improve load balancing. These boxes need not be run on the same host as that of the bearerbox.

  Complete implementation details for Kannel have been provided in the next section.
4.2 Mbuni: Open Source MMS Gateway

Mbuni is an open source MMS gateway [3]. It includes both core network MMS switching as well as messaging gateway features. It can operate as an MMSC or as an MMS VAS gateway. For the purposes of the prototype implementation, we used mbuni as a VAS gateway to provide rich content to the end-users. Mbuni implements all major MMS interfaces, including phone-to-phone (MM1), phone-to-email (MM3), inter-MMSC (MM4) and MMS VAS (MM7).

This section describes the functional parts of Mbuni and how it fits in with the prototype implementation.

4.2.1 Overview of Mbuni

In the LIVESMOBILE prototype implementation, all MMS messages are originated from the system and terminate on the client handset. In the MMS architecture, the MMSC acts as a message-switching system within the core network, while the VAS gateway acts as a message dispatch and content management system. We have used Mbuni to implement a VASP which creates and send multimedia messages to users. The features offered by Mbuni while operating as a VASP are as follows:

- Support for Simple Object Access Protocol (SOAP) and EAIF connectivity with an operator MMSC
- Multiple connections to different MMSC of different type can be maintained
- MMS content can be loaded from file, URL or as the output of a program
• Composition of MM messages from Synchronized Multimedia Integration Language (SMIL). The gateway automatically fetches all components referenced in the SMIL and adds them to the message.

• A URL interface for MM dispatch.

There is only one functional part of Mbuni if it is used as a VASP which is discussed below:

• mmsbox: The mmsbox is a single multi-threaded program which is responsible for receiving incoming MM messages from operator MMSCs, dispatching requests to services, composing and sending responses and listening for and handling requests to send MM messages. When sending MM messages, it is responsible for encoding SMIL messages into binary messages capable of being displayed on the phone. It parses the SMIL and finds all the images, audio, video and text it references and adds them to the response MM. The SMIL message is modified to reference the content within the MM.

Complete implementation details for Mbuni have been provided in the next section.

4.3 Apache Solr: Search Server

Apache Solr is an enterprise search platform from the Apache Lucene project [11, 15]. Its major features include full-text search, hit highlighting, faceted search, dynamic clustering, database integration, and rich document handling. Solr is highly scalable, providing distributed search and index replication.
Solr is written in Java and runs as a stand-alone full-text search server within a servlet-container such as Tomcat [16]. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has Representational State Transfer (REST) like HTTP/XML and JavaScript Object Notation (JSON) Application Programming Interface (API)s that make it easy to use from virtually any programming language [11].

The LIVESMOBILE prototype relies on the database integration feature of Apache Solr. Its index is created via XML over HTTP. All queries are sent using HTTP GET and results are returned in XML format. It provides a schema element to customize the internal indexing mechanism to match the database design. Figure 4.2 shows a part of the schema file used to configure Solr. Using the indexing mechanism, we store all the relevant information associated with each Learning Object for fast search. Every time a new Learning Object is added, removed or updated from the database, we update the index to reflect the changes.

```
<field name="1DObject" type="string" indexed="true" stored="true" required="true"/>
<field name="ObjectName" type="text" indexed="true" stored="true"/>
<field name="Description" type="text" indexed="true" stored="true"/>
<field name="Transcript" type="text" indexed="true" stored="true"/>
<field name="Audience" type="text" indexed="true" stored="true"/>
<field name="LearningGoals" type="text" indexed="true" stored="true"/>
<field name="Keywords" type="text" indexed="true" stored="true"/>
<field name="Status" type="ignored"/>
<field name="StartTime" type="ignored"/>
<field name="ExpiryTime" type="ignored"/>
```

**Figure 4.2:** Sample Solr Schema Used for LIVES

The advantages of using Apache Solr is that the index can be created and up-
dated from multiple instances of the LIVES installation. Any time a student wishes to search for a Learning Object, we can query all instances simultaneously to return the results. The Solr server can be hosted on an independent machine thereby taking off load from the database server to serve user call related information without incurring an additional overhead.

4.4 Call Originator

The Call Originator software has been implemented in Java [9]. We used the call originator from LIVES and made additional changes to dispatch MMS messages with each call. The software is responsible for polling the LIVES database at regular configurable intervals and create a list of students that can be called. The list is generated by taking into account the current time, the time availability range of the student and the current status of the student. We create a list of messages for each student while the call is placed and dispatch the requests to send MMS messages. Figure 4.3 shows the interface for the Call Originator.

4.5 Implementation Details

The LIVESMOBILE prototype has been implemented in two stages. The LMS portion of the prototype consists of the SMS and MMS servers, the LIVES User profile and Voice servers, the Asterisk Call Manager and the Call Originator software. The LCMS portion of the prototype consists of the LIVESMOBILE Drupal module.
4.5.1 Kannel Implementation

Kannel consists of a core configuration file which is sub-divided into the following groups:

1. **Core**: This is the core configuration section for Kannel. It is used to configure the bearerbox and any additional parameters related to the other boxes. The important parameters configured through this section are listed in Table 4.1.

2. **SMSC**: This section specifies configuration parameters for detecting the SMSC. In the prototype, a modem was used as the SMSC. The values in this group tell Kannel how to detect and use the modem. The parameters are...
### Table 4.1: Kannel: Core Group

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>admin-port</strong></td>
<td>15000</td>
<td>The web administration port for Kannel. This port is used by the LCMS to administer Kannel</td>
</tr>
<tr>
<td><strong>smsbox-port</strong></td>
<td>15001</td>
<td>The port at which the smsbox connects to the bearerbox. All incoming and outbound SMS messages are passed through this port</td>
</tr>
<tr>
<td><strong>unified-prefix</strong></td>
<td>&quot;+1,001,00;+,00&quot;</td>
<td>The prefix of all phone numbers accepted through the gateway.</td>
</tr>
<tr>
<td><strong>store-type</strong></td>
<td>file</td>
<td>Any undelivered or queued SMS messages will be stored to disk.</td>
</tr>
</tbody>
</table>

listed in Table 4.2

### Table 4.2: Kannel: SMSC Group

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>smsc</strong></td>
<td>at</td>
<td>Specifies that the SMSC is a modem which recognized AT commands.</td>
</tr>
<tr>
<td><strong>modem-type</strong></td>
<td>auto</td>
<td>Specifies that Kannel should try to auto detect the modem.</td>
</tr>
<tr>
<td><strong>device</strong></td>
<td>/dev/ttyACM0</td>
<td>Specifies the location at which the modem can be reached in the system.</td>
</tr>
<tr>
<td><strong>denied-smsc-id</strong></td>
<td>fake</td>
<td>Specifies that any messages originating or termination at the fake smsc will be routed through this particular SMSC.</td>
</tr>
</tbody>
</table>
3. **SMSBox**: This group configures the smsbox. It specifies the interaction behavior between the core and the box. Also, the LIVESMOBILE LMS uses the values from this section to send SMS messages. The configuration parameters for this group are listed in Table 4.3.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sendsms-port</code></td>
<td>15013</td>
<td>The port hosting the SendSMS service. Any request to send SMS messages comes on this port.</td>
</tr>
<tr>
<td><code>sendsms-chars</code></td>
<td>0123456789+--</td>
<td>The list of allowed characters in the destination number of SMS messages.</td>
</tr>
</tbody>
</table>

4. **SMS-Service** This group configures all services defined within Kannel. In the prototype, Kannel does not have any intelligent message handling capabilities of its own. All incoming messages are forwarded to our VAS application. The VAS application decides how to process the incoming message based on it’s content. Important configuration variables of this group are presented in Table 4.4.

We hosted the Kannel bearerbox and smsbox processes on the main server which hosted the LIVES database. We configured the bearerbox process to forward all incoming SMS messages from the GSM SMSC to the smsbox. The smsbox was configured to post the contents of the SMS message along with sender and receiver number and the receiving time to our web portal. We hosted the web portal at
Table 4.4: Kannel: SMS-Service Group

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyword-regex</td>
<td>.*</td>
<td>Specifies that all incoming messages coming in should be handled by this particular service.</td>
</tr>
<tr>
<td>max-messages</td>
<td>0</td>
<td>Specifies that Kannel should not immediately reply to incoming messages. It should rather invoke one of the other specified URLs and pass the message.</td>
</tr>
<tr>
<td>post-url</td>
<td><a href="http://localhost/sendSMS.php?from=%25p&amp;to=%25P&amp;time=%25t&amp;text=%25a">http://localhost/sendSMS.php?from=%p&amp;to=%P&amp;time=%t&amp;text=%a</a></td>
<td>The URL to invoke on receiving an SMS. The values pertaining to %p, %P, %t and %a are filled in by Kannel before invoking the URL.</td>
</tr>
<tr>
<td>accepted-smsc</td>
<td>gsm;fake</td>
<td>A semicolon separated list of SMSCs. Only messages originating from these SMSCs will be processed by this service.</td>
</tr>
<tr>
<td>catch-all</td>
<td>true</td>
<td>Process all incoming messages irrespective of their content.</td>
</tr>
</tbody>
</table>

http://localhost/kannel/sendSMS.php. All replies sent to the users were generated through our web portal. The smsbox was configured to suppress auto-replying to messages. The web portal implements the entire SMS querying functionality. The web portal has a central configuration file from which it picks up login credentials for the MySQL database and for the Apache Solr search portal [12]. Requests to create new users or enrol users to new courses are handled by the web portal and changes are made to the LIVES database. The call originator software also
has access to the LIVES database. If a new student is available and is enrolled in courses he hasn’t heard, then the call originator places a call to the new students.

Also, the LIVESMOBILE LCMS has a configuration section where we specify the path to the SendSMS interface. When an instructor drags and drops an SMS Object onto a Learning Unit or a Student, the LCMS posts the contents of the SMS and Calling Channel information to the SMS server through the SendSMS interface.

### 4.5.2 Mbuni Implementation

The Mbuni mmsbox uses a single configuration file which is divided into the following sections:

1. **Core**: This is the core configuration section for Mbuni. It is used to configure the mmsbox. The important parameters configured through this section are the log file locations and the verbosity level of logging desired in the logs.

2. **mbuni**: This group is used to configure the web interface exposed by the mmsbox. Some of the important parameters configured through this section are listed in Table 4.5.

3. **mmsc**: This group is used to configure the parameters related to the modem connected to Mbuni. It specifies how to establish a Point-to-Point Protocol (PPP) connection with the operator MMSC and how to access the Kannel SMSC. The configuration used for this group are listed in Table 4.6
Table 4.5: Mbuni: Mbuni Group

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>storage-directory</td>
<td>/var/spool/mbuni</td>
<td>Specifies the location on the disk where the incoming and outgoing queue for mmsbox will be stored.</td>
</tr>
<tr>
<td>max-send-threads</td>
<td>5</td>
<td>The number of threads to run for sending MM messages.</td>
</tr>
<tr>
<td>sendmms-port</td>
<td>10001</td>
<td>The port number at which any properly formatted incoming request would result in the sending of MM messages.</td>
</tr>
</tbody>
</table>

Table 4.6: Mbuni: MMSC Group

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>gsm-modem</td>
<td>Specifies that the MMSC being used is of type GSM modem.</td>
</tr>
<tr>
<td>type</td>
<td>custom</td>
<td>Specifies that a custom initialization string would be used to configure Mbuni.</td>
</tr>
<tr>
<td>custom-settings</td>
<td>smsc-on='localhost:15000/start-smsc'; smsc-off='localhost:15000/stop-smsc'; gprs-on='wvdial'; gprs-pid=cat /var/run/ppp0.pid—head -1; port=8181; mmsc-url='mms.gprs.rogers.com'; proxy=10.128.1.69:80;</td>
<td>Contains a list of semicolon separated values used to configure Mbuni.</td>
</tr>
</tbody>
</table>
The Mbuni mmsbox process was hosted on the main server hosting the LIVES database. We used Mbuni as a VAS gateway. When an instructor assigned an MMS object to a course, the association is made in the MySQL database. The call originator software is responsible for dispatching MMS send requests to Mbuni. When a call is placed to a student, the database is checked for any unsent MMS messages to the student. The call originator creates a list of all unsent MMS messages and send the messages to the student while the call is in progress. The Call and the MMS functionality are decoupled. The MMS will get delivered even if the user hangs up the call. Once the MMS messages have been dispatched, they are marked as 'Sent' to that particular user and are not sent again in the subsequent calls. In order to send the MMS messages, the LCMS posts to the Mbuni SendMMS interface. In our implementation, requests to dispatch MMS messages were sent to localhost at the port 10001 at which Mbuni was configured to listen for requests.

4.5.3 LCMS

The LCMS has been implemented in Drupal. Through the LCMS an instructor is able to create new SMS messages and dispatch them to all the students enrolled in a course or to individual students. This feature can be used to make important course related announcements or issue Public Safety Announcements or Warnings to the user base. The LCMS also allows the Instructor to upload new MMS messages. In the prototype, we used a third-party tool to create MMS messages [4]. The instructor can assign MMS messages to courses by dragging them into the list of available courses. The messages get dispatched when the Call Originator
places a call to the student.

This section presents some of the screenshots from the LIVESMOBILE LCMS. Figure 4.4 shows the screen where the Administrator can see the status of the SMS server. Figure 4.5 shows the screen where the Administrator can configure the SMS server. The operations allowed are Start, Shutdown, Suspend, Isolate, Resume and Restarting the main gateway and Stopping and Starting the SMSC. Figure 4.6 shows the configuration screen for Apache Solr. The Administrator can Create, Update, Delete and Query the search index. Figure 4.7 shows the screen where an Instructor can dispatch SMS messages to students. The SMS message is displayed on the left and a list of available courses with students enrolled is displayed on the right. In order to send the SMS message, the instructor can drag the message from the left and drop it onto a particular course on the right or on a particular student if the message is address to an individual. Figure 4.8 shows the screen for assigning MMS messages to students. A similar drag and drop approach is followed here as well. The list on the right however only displays Learning Units and their children which are also of the same type. In the case of the SMS Dispatch screen, the sublist contains all users registered in a Learning Unit.
**Figure 4.4:** SMS Gateway Status

**Figure 4.5:** SMS Gateway Administration
Figure 4.6: Search Server Configuration

Figure 4.7: SMS Message Dispatch
Figure 4.8: MMS Message Assignment to Learning Units
Chapter 5

System Evaluation

In this section, we evaluate the performance of our proposed system through extensive experiments. Our system utilizes SMS as a content request mechanism and MMS as a content delivery mechanism. In order to test the system, we required a sample usage pattern as would be observed during interaction with real users. In order to ensure the smooth functioning of the system, we first wanted to test the overall capacity of the system to handle large volumes of messages. The first part of the evaluation framework tests the system performance under a large load. We performed these tests to determine the maximum number of simultaneous messages which can be processed by our system. We then specify the criteria used to set a reasonable expectation of the system performance and we compare the results obtained through these tests with our expectations in the last section.
5.1 Simulation Setup

In order to test the performance of the LIVESMOBILE system, we used a singular server hosting the LMS and the LCMS components of LIVES. In addition, the SMS and the MMS servers from the LIVESMOBILE LMS were also hosted on the same machine. In order to test the SMS server, we attached a GSM modem to the machine over a USB 2.0 connection. In order to test the MMS server performance, another High-Speed Packet Access (HSPA) modem was connected to the same machine to send outgoing MMS messages using a USB 2.0 connection. The SMS and MMS evaluation framework was implemented in Java and tested the throughput of the SMS and MMS servers while measuring system performance in terms of CPU load and memory utilization. In order to test the SMS performance, we used a fake SMS server which could generate SMS messages and send them to the SMS server and receive responses addressed back to it. In order to test the MMS performance, we implemented a fake MMS server which could generate requests to send MMS messages to the gateway at regular time intervals.

5.1.1 Testing SMS Performance

The Kannel bearerbox process was hosted on the server using a specially crafted configuration file. The bearerbox would route all incoming messages addressed to the shortcode number '200' coming in from our 'fakesmsc' to our smsbox. The smsbox would accept messages originating from the fakesmsc or from the GSM modem. We assigned the origin number to be '100' for all messages which should be routed back to the fakesmsc. If the message was supposed to be routed
to a cellphone, the origin number was set to the cellphone number of the receiver. The smsbox process was also hosted on the same server. Before running each test, we ensured that there were no messages pending inside Kannel’s internal queue from previous runs.

The test mix consisted of sending query messages by choosing randomly from the following set of messages:

- 'help'
- 'find vancouver'
- 'weather seattle'
- 'help reg'
- 'sched neem 1 now'

We conducted the following tests and got results which are summarized in the next section.

1. We sent SMS messages from our message mix @ 60 messages per minute, or 1 message per second using fakesmsc as the sender and receiver

   (a) We checked the receipt of the messages in smsbox and whether they were processed properly. Since all messages were query requests, we checked whether responses were being sent back to the requesting agent.
(b) We checked the SMS store status at the end of each run and within each log file to check if any messages were getting dumped into the store. This would happen when the smsbox is busy processing previous requests and the bearerbox has received new incoming messages.

(c) We measured the bearerbox and smsbox process cpu and memory utilization at 10 second intervals to measure the system load for each test mix.

(d) We repeated the mix with a rate of @80 messages per second to determine system load at high message arrival rates and stored our results.

2. We sent SMS messages from our message mix @ 20 messages per minute, or 1 message per 3 seconds using fakesmsc as the sender and a cellphone as a receiver.

(a) We checked the receipt of the messages in smsbox and whether they were processed properly. Since all messages were query requests, were responses sent back to the user? We checked if the SMS responses were routed to the user cellphone.

(b) We checked the SMS store status at the end of each run and within each log file to check if any messages were getting dumped into the store. This would happen when the smsbox is busy processing previous requests and the bearerbox has received new incoming messages.

(c) We measured the bearerbox and smsbox process cpu and memory utilization at every 10 seconds to measure the system load for each test
mix.

(d) We repeated the mix with rates of @40 messages per minute and found some interesting results which are presented in the results section.

5.1.2 Testing MMS Performance

Testing the MMS performance required a different setup. Unlike SMS messages which are usually very short and are limited in size to 140 bytes, MMS messages have varying sizes and may range from a few kilobytes to a few megabytes. It is hard to decide a formal quality metric for judging MMS performance due to the following factors: (1) As previously mentioned, the size of an MMS message can vary by a large amount. As a result, the MMS server utilizes varying amounts of system resources and time to encode the messages before dispatch. In order to provide an estimate on the performance of the server, we decided to select an MM message which contained a collection of pictures and text snippets in the form of a presentation. (2) The quality of the MMS messages received on the client handset depends on factors which were outside the scope of the project. The factors influencing the quality of the received MMS depends on the particular device being used by the receiver as well as the encoding process used by the service provider before notifying the receiver about the new MMS. We noted that if the make and model of the client handset is registered with the service provider, a higher quality message can be expected. If however the receiver is using a different handset from the one registered or if no handset has been registered, then the quality varies unpredictably.
In our test mix, we chose an MM message which contained a presentation with 5 slides containing pictures and text. In total, the MM message contained 9 items, out of which there were 4 images and 4 text files, all of which were embedded in 1 SMIL file. The total size of the message was 42.2 kilobytes.

The Mbuni mmsbox process was hosted on the server using a specially crafted configuration file. We chose a new location for the mmsbox queue storage so as to not affect the state of the queue for the prototype. Before running each test, we ensured that there were no messages pending inside the queue from previous runs to ensure a clean start.

We conducted the following tests and got results which are summarized in the next section.

1. We sent our requests to dispatch our sample MM message @ 60 requests per minute, or 1 request per second using our fakemmsc

   (a) We checked the receipt of the message request in mmsbox and whether they were all encoded and dispatched properly. We noted that our server is only capable of encoding and sending one MM message at any given instance of time. All of the remaining requests were queued for later delivery.

   (b) We checked the MMS queue status at the end of each run and within each log file to check how many messages were being added to the queue.
(c) We measured the mmsbox process cpu and memory utilization at 10 second intervals to measure the system load for each run.

2. We sent requests to dispatch our sample MM message @ 1 request per minute using our fakemmsc. We tested the percentage CPU and memory utilization while encoding our MM message.

5.2 Simulation Parameters

The SMS and MMS servers were hosted on a laptop which met the following specifications:

- Intel Pentium 2.00 GHz Dual Core Processor.
- 3 GB RAM
- 160 GB Harddrive space @ 5400 rpm
- USB 2.0 Interface for Modems
- Ubuntu 10.04.

We wrote simulation tests to measure SMS gateway performance by first using a fake SMSC which was a program that sent SMS messages with varying content to the SMS server at different rates. We then tested the performance of the GSM modem, by using the phone as an SMSC. While we still generated the requests to the SMS server through the same program, the responses were routed through the GSM modem to the receiving cell phone to measure the GSM modem throughput.
We wrote the simulation tests to measure MMS gateway performance by using a fake MMSC which was a program that sent MMS dispatch requests containing the same MM message to the MMS server at different rates. While running the tests, the sending modem and the receiving cellphone were located in an area where cellphone signal strength for both varied between 80-100 percent.

5.3 Simulation Results

The simulation results are presented here.

Figure 5.1 shows results obtained when sending 60 SMS messages per minute (1 message per second) originating from a fake SMSC directed through the LIVES-MOBILE SMS Server and routed back to the fake SMSC. The results indicate that the bearerbox and smsbox CPU utilization is negligible during the test run. The memory utilization has not been indicated and is negligible.

Figure 5.2 shows results obtained when sending 80 SMS messages per second using the same setup. The smsbox utilizes approximately 25 percent of the CPU during the test run. The bearerbox has a lower utilization at approximately 10 percent. No messages were lost during the test run. There were periodic addition of SMS messages to the internal queue. On regular monitoring, we noted that the smsbox flushed the queue and dispatched all SMS messages before the termination of the test run. The results indicate that a significant amount of CPU is utilized while handling large volumes of SMS messages. If such a high load is expected, it is advisable to host the SMS server on a dedicated machine. The memory utilization remained negligible during the test run. This can be attributed to the fact that
Figure 5.1: SMS Gateway Load Testing @ 1 SMS Message Per Second Using Virtual SMSC

the server only deals with SMS messages which are 160 bytes in size each and are a very small percentage of the available memory.

Figure 5.3 shows results obtained when sending 20 SMS messages per minute originating from a fake SMSC but routed through the GSM modem to a cellphone. The results indicate that there is no significant change in the percentage CPU and memory utilization. All the messages originating from the fake SMSC were received on the cellphone without any losses. This indicates that the modem is capable of sending 20 SMS messages per minute without delivery reports.

Figure 5.4 shows results obtained when sending 40 SMS messages per minute using the same setup. All of the messages originating from the fake SMSC were not
Table 5.1 depicts the SMS loss ratio while running this test. It indicates that even though the SMS server is capable of handling incoming messages at a large rate, the modem has a rate limitation and is only capable of sending a few SMS messages per minute. We found that it takes approximately 2.5 seconds for the GSM modem to send a single SMS message without delivery reports. We experienced a loss ratio of approximately 40 percent while sending messages at this rate. In order to service SMS messages at a large rate, we suggest the use of bulk SMS gateways which are capable of delivering messages in the range of 50 messages in one batch at high speeds [7].
Figure 5.3: SMS Gateway Load Testing @20 SMS Messages Per Minute Using a GSM Modem

Table 5.1: Kannel: SMS Loss Ratio

<table>
<thead>
<tr>
<th>Test Attribute</th>
<th>Observed Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message(s) Sent</td>
<td>40 40 40</td>
</tr>
<tr>
<td>Message(s) Received</td>
<td>23 23 24</td>
</tr>
<tr>
<td>Message Loss Rate</td>
<td>42.5% 42.5% 40%</td>
</tr>
<tr>
<td>Time taken per message</td>
<td>2.60 s 2.60 s 2.5 s</td>
</tr>
</tbody>
</table>

Figure 5.5 shows results obtained when sending requests to dispatch 60 MMS messages per minute originating from our fake MMSC directed through the LIVES-MOBILE MMS Server and routed to a users cellphone. As we can note, the percentage CPU utilization keeps reducing towards the end of the test. This behavior is attributed to the fact that the gateway queues all the requests for MMS dispatch.
This indicates that the gateway is only capable of processing one MM message at any given time. We try to determine the CPU utilization for sending our test MM message in the next test.

Figure 5.6 shows results obtained when sending a request to dispatch 1 MMS messages per minute originating from our fake MMSC directed through the LIVESMOBILE MMS Server and routed to a users cellphone. We note that the amount of CPU and memory utilization required to encode our MM message containing pictures and text is high during the encoding phase of the message and becomes negligible when it dispatches the message. If the MMS server is hosted on a machine where many processes compete to share system resources, then message
5.4 Critical Evaluation

The previous two sections explained our setup for performance evaluation of the system. This section critically analyses the performance by first defining a framework for setting a reasonable expectation from the system. We then compare the performance of the system with our expectations to foresee the ability of the system to meet user demands.

In order to define a framework to set reasonable expectations, we need to get a good sense of the target user population. We define the following categories of delivery might be delayed.
Figure 5.6: MMS Gateway Load Testing @1 MMS Message Per Minute Using Virtual MMSC

users:

- Children between ages 13-17
- Young Adults between the age 18-24
- Adults between the age 25-34
- Older Adults between the age 35-44
- Other users

During the testing of the LIVES system in India, we noted that the age distribution of the users was between 25-45 years. This interval is covered by the
"Adults" and the "Older Adults" section of our framework. We make use of the wireless usage statistics presented in Table 5.2 for customers in the United States as a starting point for our evaluation [5]. We generalize this usage pattern to people in developing countries to establish a base line usage scenario. This scenario can be compared to actual survey results in developing countries when they are available.

Table 5.2: Average Monthly Phone and SMS Usage for U.S. Wireless Subscribers, 2008

<table>
<thead>
<tr>
<th>Age Group</th>
<th>Average Monthly Calls</th>
<th>Average Monthly Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Subscribers</td>
<td>204</td>
<td>357</td>
</tr>
<tr>
<td>Ages 12 and under</td>
<td>137</td>
<td>428</td>
</tr>
<tr>
<td>Ages 13 - 17</td>
<td>231</td>
<td>1742</td>
</tr>
<tr>
<td>Ages 18 - 24</td>
<td>265</td>
<td>790</td>
</tr>
<tr>
<td>Ages 25 - 34</td>
<td>239</td>
<td>331</td>
</tr>
<tr>
<td>Ages 35 - 44</td>
<td>223</td>
<td>236</td>
</tr>
<tr>
<td>Ages 45 - 54</td>
<td>193</td>
<td>128</td>
</tr>
<tr>
<td>Ages 55 - 64</td>
<td>145</td>
<td>38</td>
</tr>
<tr>
<td>Ages 65+</td>
<td>99</td>
<td>14</td>
</tr>
</tbody>
</table>

We note that the average number of text messages sent per month by the "Adults" category is 331 messages per month and that for the "Older Adults" category is 236 messages per month. This can be further simplified to an average of 11 and 7 messages per day per user from both groups respectively.

We use this as a starting point to begin our critical evaluation. An average user would have to send a variable number of messages to achieve different tasks.
by using our system. Table 5.3 lists down the number of messages required to complete various tasks within the system.

**Table 5.3:** Number of Messages Required to Complete Tasks Within LIVES-MOBILE

<table>
<thead>
<tr>
<th>Task to achieve</th>
<th>Number of messages required</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get weather information</td>
<td>1</td>
<td>Retrieve the weather information</td>
</tr>
<tr>
<td>Search for Learning Object</td>
<td>1</td>
<td>Each message returns a single page of results</td>
</tr>
<tr>
<td>Schedule a Learning Object</td>
<td>2</td>
<td>Includes searching and scheduling</td>
</tr>
<tr>
<td>Register a New User</td>
<td>1</td>
<td>Register a new user</td>
</tr>
</tbody>
</table>

If a user schedules one new lesson each day and retrieves weather information on the same day, the user needs to send at-least 3 messages. This is much lower than the average number of messages sent per day by our age groups. In order to evaluate our system, we assume that all the messages sent by the user per day were sent to our system. We take an average of message sending rates of both the age groups to get 9 messages per day.

During the testing of the LIVES system in India, we registered a user base of 200 individuals. If each individual sends 9 messages per day, we get a total system load of 1800 messages per day. Based on our performance evaluations, our SMS gateway is capable of processing at-most 23 messages per minute. Out of the 1800 messages per day, we assume that not all of the messages arrive at the same time. We allocate a time period of 3 hours during the day and 3 hours during the
evening for which we expect peak message loads. Assuming an even distribution of the messages, we expect a load of 300 messages an hour or 5 messages per minute. This limit is within the capability of our gateway. We acknowledge the rate limitation of the modem and state that the resulting behavior of the gateway will be different if there is an uneven distribution and more than 23 messages are received per minute. We suggest the use of modem banks or bulk SMS gateways to avoid losses related to this limitation.

To perform critical evaluations of our MMS server, we decide upon a reasonable expectation from the users. In the prototype, MMS messages are sent during voice calls placed to the users. Currently, MMS messages are treated as additional content messages which can be provided in addition to the voice lessons. We do not place a real-time content delivery restrictions on the content and thus set the expectation to deliver the message during the same day of the call while the content is still fresh in the users memory. Based on our performance evaluations, we noted that all requests to send MM messages are queued in the MMS Server if there exists a previous request which is being processed. We believe that this behavior is appropriate for our expectations and that sequential dispatching of MM messages would still ensure the delivery of the messages to the intended recipients.
Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we propose LIVESMOBILE, an extension to the LIVES system with support for SMS and MMS capabilities. In our system, the end-user can send and receive SMS messages to and from the system. The user can search for new lessons through SMS and get search results on his phone. The user can instruct the system to schedule voice calls based on the search results and thereby enhance his/her learning experience. The user can also get additional information such as weather forecasts and instructor announcements through SMS. In addition to SMS, the system can also send MMS messages to the end-users. The ability to process images, videos and animations adds new levels of functionality to the LIVES system. This enables us to transmit web content from the Internet to the users in the form of MMS messages. The users can continue to use their mobile equipments to re-
ceive SMS messages. However, in order to receive MMS messages, the user needs to have an MMS enabled phone. We implemented an LCMS module to manage the Learning Content. An LMS was also implemented which contained an SMS server, an MMS server and a search server to search for content. We implemented a modified version of the Call Originator software for LIVES which sends pending MMS messages to the users right after a call has been placed. The performance of the prototype was measured through extensive simulation experiments and was judged to be sufficient for a user load of 200 individuals sending an average of 300 messages per minute to the SMS server while requesting for content and scheduling lessons. We acknowledged the rate limitation on the GSM modems and suggested alternative approaches to use if a very high message rate is expected.

6.2 Future Work

There are five major directions which call for further investigation. First, in the prototype, we have a rate limitation on the number of SMS messages handled by the SMS server. The limitation is primarily because of the GSM modem. The use of modem banks or bulk SMS gateways can be evaluated to overcome this rate limitation and service more requests.

Secondly, the SMS based query mechanism uses a formal query language we designed. It may not be the most intuitive way to query for content from our servers. A possible direction worth exploring would be natural language based search. Also, currently English is the only language supported in the prototype whereas we can have voice lessons in any language. This gives rise to two possible
directions of future research. The first would be to explore the use of speech recognition software to generate textual transcripts of voice lessons and search through these transcripts for relevant content. The second could involve research into support for multiple languages as a query mechanism.

Third, the instructors need to have access to a personal computer connected to the LIVES system to be able to upload MMS messages. One possible avenue of research would be to explore the use of mobile phones to upload MMS messages to the server and distribute them to user groups based on their interests or subscriptions.

Fourth, we used a third party tool to create MMS messages for the purpose of demonstrating it in our prototype. In addition to allowing the instructor to upload MMS messages, there could be a provision in the LCMS to compose new SMIL messages. This would allow the instructor to upload content from his own machine and compose presentations or video messages at the LCMS and then assign it to courses. This particular feature would improve the usability of the system.

Finally, during our evaluation of the system, we measured the performance of the gateways and the application in terms of system resource utilization and responsiveness to high volumes of messages. As part of the future work, the usability of the application can be tested with a real audience to gather an understanding about the benefits of this implementation. During the testing of LIVES in India there have been frequent requests to incorporate additional features such as support for images and additional ways to customize the system. This particular prototype has been implemented specifically to address that need. We propose the
deployment of the system for a small user base of 50 to 200 students during the initial testing phase for LIVESMOBILE. In order to get the users acquainted with the SMS functionality, we suggest sending instructional SMS messages through the LCMS to all the users. The response messages obtained from the users can be stored and studied to get an idea about their skill level and understanding. We also suggest survey exercises to be performed after the trials to gather feedback from the users. The surveys should be designed to ask the users about the specific technical challenges they faced while working with the system and should ask the user to suggest features they would like to see in the system. We believe this exercise is an essential part of future development and can contribute new knowledge to the developers to help make the system more useful to the end user.
Bibliography


73


[18] MoULE (Mobile and Ubiquitous Learning). [moule.pa.itd.cnr.it](http://moule.pa.itd.cnr.it), unpublished manuscript. → pages [12]


