

Mobile presentations with interactive chat for m-Learning

David Wafula Wanyonyi

A thesis submitted in fulfillment of the requirements for the
degree of Magister Scientiae in the Department of Computer
Science, University of the Western Cape.



Supervised by Dr William D. Tucker

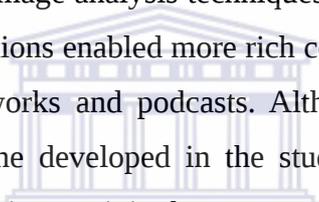
15th November 2010



UNIVERSITY *of the*
WESTERN CAPE

Abstract

Using presentations in an m-Learning environment enables delivery of rich content to a mobile phone learner. This study investigated how to prepare and stream presentations from a desktop computer to a mobile phone in near-realtime. It also addressed communication between users using interactive text chat in the same environment. Our analysis of text/podcast-based m-Learning applications revealed limited interactivity and lack of diversity in content streamed. To address this, we developed a mobile-based application that uses a task-timer model to synchronize with a server every n units of time to enable near-realtime streaming of presentation slides between mobile and desktop users. The application included text-based instant messenger. Laboratory experiments investigated the use OpenOffice and PowerPoint presentations and techniques used to convert these presentations into mobile phone compatible formats. Experiments were carried out with smart mobile phones running on a third generation cellular network. We employed transaction-logging techniques in addition to automated image analysis techniques to observe and record data. Analysis of the results revealed using presentations enabled more rich content than text-based models such as short message service-based frameworks and podcasts. Although m-Learning is not yet widely adopted, applications such as the one developed in the study offer high hopes for m-Learning because of the use of rich content and interactivity between users.

UNIVERSITY of the
WESTERN CAPE

Keywords

m-Learning

e-Learning

Image transformation

Interpolation

Image-scaling

Mobile presentations

Podcasting

Short message service

Desktop-mobile interaction

Mobile logging

Extensible messaging presence protocol

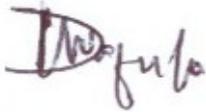


Declaration

I declare that *Mobile presentations with interactive chat for m-Learning* is my own work, that it has not been submitted for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by complete references.

Full name David Wafula Wanyonyi

Date. 2010/11/15



Signed.....



Acknowledgements

I would like to thank the staff and members of the Free Software Innovation Unit (FSIU) at the University of Western Cape (UWC) and e-Learning Support and Innovation Unit (eLSI) at the University of Witwatersrand for providing a useful testing environment and for offering invaluable insight into the Chisimba framework.

Without the financial assistance provided by Telkom, Cisco and THRIP through a grant from the Telkom Centre of Excellence (CoE) programme, I would have had difficulty in completing my research.

My supervisor, Dr William Tucker, from the Department of Computer Science at UWC, was unfailingly encouraging with advice throughout my research.

Finally, I would like to thank my family for their patience, support and understanding.

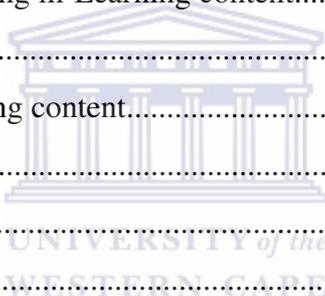


Table of contents

Abstract	iii
Keywords	iv
Declaration.....	v
Acknowledgements	vi
Table of contents	vii
List of figures	xi
List of tables	xiii
Glossary	xv
1 Introduction.....	1
1.1 m-Learning applications.....	1
1.2 Presentations and chat for m-Learning.....	2
1.3 Research questions.....	4
1.4 Research methodology.....	5
1.5 Research tools.....	6
1.6 Thesis outline.....	6
2 Related work.....	7
2.1 m-Learning systems.....	7
2.1.1 Text-based models.....	7
2.1.2 Podcasting and Image based m-Learning applications.....	10
2.1.3 Rich internet applications for m-Learning.....	14
2.1.4 Mobile based learning management systems	16
2.2 m-Learning data transmission.....	17
2.2.1 Short message service.....	17
2.2.2 General Packet Radio Service.....	19
2.2.3 Wireless Applications Protocol.....	21
2.2.4 Third generation networks.....	24
2.2.5 Bluetooth	25
2.3 m-Learning data presentation.....	26
2.3.1 Textual presentation.....	27
2.4 Customizing images for mobile display.....	29
2.4.1 Image manipulation.....	29



2.4.2 Interpolation.....	30
2.4.3 Image adaptation.....	31
2.5 Summary.....	32
3 Methodology.....	33
3.1 Challenges of existing m-Learning systems.....	33
3.1.1 Text-based applications.....	33
3.1.2 Podcasting.....	35
3.1.3 Rich internet applications for m-Learning.....	36
3.1.4 Summary of challenges.....	36
3.2 Research questions.....	37
3.3 Experimental design.....	38
3.3.1 Software to help answer research questions.....	38
3.3.2 Experimental environment.....	39
3.3.3 Formatting and transporting m-Learning content.....	39
3.3.4 Air time cost	43
3.3.5 Presentation of m-Learning content.....	44
3.4 Summary.....	45
4 System design.....	47
4.1 User requirements.....	47
4.2 Target audience.....	47
4.3 System structure.....	48
4.3.1 Choice of development platform.....	48
4.3.2 Mobile Client.....	52
4.3.3 Communication model.....	58
4.3.4 Server.....	58
4.4 Database.....	64
4.5 Open source libraries.....	65
4.6 Communication Flow.....	67
4.7 Summary.....	67
5 Data analysis and discussion.....	69
5.1 Practical aspects of experimentation.....	69
5.2 Scaling down presentations and analyzing quality.....	70



5.2.1 Clear Slides	71
5.2.2 Poor Slides.....	73
5.2.3 Trends.....	75
5.2.4 Performance cost of scaling down slides.....	79
5.3 Air time costs	82
5.3.1 Transporting presentations.....	83
5.3.2 Polling users.....	85
5.3.3 Polling chat messages.....	87
5.3.4 Average bytes of data transported in a session.....	89
5.4 Discussion.....	89
6 Conclusion and future work.....	91
6.1 Overview of the thesis.....	91
6.2 Scope and Limitations.....	93
6.3 Future work.....	93
References.....	95





UNIVERSITY *of the*
WESTERN CAPE

List of figures

Figure 2.1: Diagrammatic view of SMS-based m-Learning application.	9
Figure 2.2: Push - Subscribe model often used in distributing podcasts.....	11
Figure 2.3: Mobile Learning Engine (MLE) illustration.....	13
Figure 2.4: How GPRS interfaces with the Internet.....	20
Figure 2.5: Layered communication stack in a Wireless Application Protocol.....	22
Figure 2.6: The various uses of Wireless Application Protocol.....	23
Figure 2.7: AD-HOC m-Learning system model.....	27
Figure 3.1: How presentations are converted to images for presentation.....	40
Figure 3.2: Desktop viewer for preview converted slides as images.....	41
Figure 4.1: High level overview of mChisimba framework.....	48
Figure 4.2: The structural layout of mChisimba.....	52
Figure 4.3: The overall structure of mChisimba.....	53
Figure 4.4: mChisimba login class diagram.....	56
Figure 4.5: Task timer model.....	57
Figure 4.6: mChisimba server components.....	59
Figure 4.7: An Information Query (IQ) packet.....	60
Figure 4.8: The server plugin packet interceptor.....	61
Figure 4.9: Server class diagram.	62
Figure 4.10: The gateway plugin.....	63
Figure 4.11: The Gateway class diagram.	64
Figure 4.12: mChisimba server plugin and the database management system.....	65
Figure 4.13: Database access class diagram.	66
Figure 5.1: A diagrammatic categorization of mean image values.....	71
Figure 5.2: Snap shot of a bright slide as seen from desktop client.....	72
Figure 5.3: Distribution of images values of a bright slide.....	72
Figure 5.4: A snapshot of bright slide when rendered on a mobile screen.....	73
Figure 5.5: A snapshot of a dark slide on desktop client.....	74
Figure 5.6: A darker image plotted.....	74
Figure 5.7: A poorly rendered slide.	75
Figure 5.8: Researcher rating scores as related to the mean presentation image value.....	77

Figure 5.9: All shaded bar show presentations with high mean value, except for three.....78

Figure 5.10: Duration it takes to scale down an image slide for presentation.....80

Figure 5.11: A chart view of the scale times for a second presentation.81

Figure 5.12: Average scale time per presentation for all the presentations.....82

Figure 5.13: Amount of data transferred when using a presentation.....83

Figure 5.14: Amount of data transferred when using a second presentation.....84

Figure 5.15: Average number of bytes per presentation recorded at mobile client.....85

Figure 5.16: Amount data transported when polling users.....87

Figure 5.17: Second example result of amount of data transported to mobile phone.....87

Figure 5.18: Data recorded when simulating 7 users chatting.....88

Figure 5.19: Second data recorded when 7 simulated users are chatting.....88



List of tables

Table 2.1: Possible uses of podcasts compared to SMS-based applications for e-Learning.	12
Table 2.2: Short message submission using block mode.....	18
Table 2.3: Short message submission using text mode.....	18
Table 4.1: A comparison showing how J2ME compares to Flash Lite 3 in graphics.....	49
Table 4.2: Accessing local resources on a mobile device capabilities of J2ME and Flash Lite 3.....	50
Table 4.3: Comparison of secure communication capabilities of J2ME and Flash Lite 3.....	50
Table 4.4: J2ME versus Flash Lite 3 connectivity.....	51
Table 4.5: Miscellaneous comparison between J2ME and Flash Lite 3.	51
Table 5.1: Mean presentation image value together with researcher rating.	76
Table 5.2: A table showing two presentations with a very high mean value.	78
Table 5.3: Amount of data in bytes transported when polling users.....	86
Table 5.4: Vodacom South Africa data rates.....	89





UNIVERSITY *of the*
WESTERN CAPE

Glossary

2G	Second-generation wireless networks that used digital technology and enabled data services for mobile phones, most significantly short message services.
2.5G	Circuit-switched second generation cellular networks that included packet-switched technology during the transition to 3G networks.
3G	Third Generation mobile phone networks.
3GPP	Third Generation Partnership Project.
API	Application Programming Interface.
Chisimba	A framework for rapidly developing distributed web-based applications in PHP.
CLDC	Connected Limited Device Configuration. Defines the base set of application programming interfaces and a virtual machine for resource-constrained devices like mobile phones, pagers, and mainstream personal digital assistants.
DHCP	Dynamic Host Configuration Protocol.
EDGE	Enhanced Data GSM Environment. A technology that allows improved data transportation rates over GSM, and is part of the 3G family.
GGSN	Gateway GPRS Support Node.
GPRS	General Packet Radio Service. A 2G-based service for cellular communication, used in GSM, and in 3G networks. It is a packet oriented service, with transfer rates between 56-114 kilobits per second.
GCF	Generic Connection Framework.
GSM	Global System for Mobile communications.
GUI	Graphical User Interface.
HDML	Handheld Device Markup Language.
HTML	HyperText Markup Language.
HTTP	Hyper Text Transfer Protocol.
IDE	Integrate Development Environment.

IETF	Internet Engineering Task Force.
IP	Internet Protocol.
IQ	Information Query.
ISM	Industrial, Scientific, Medical.
ITM	International Telecommunications Union.
ITU	International Telecommunications Union.
JABWT	Java API for Bluetooth Wireless Technology.
JavaFx	A Java-based platform that allows developers to build rich applications that run on the desktop, the web and mobile devices using scripting technology.
J2ME	Java2 Micro Edition.
JSR	Java Specification Request.
KEWL	Knowledge Environment for Web-based Learning. A web-based learning environment built on Chisimba, and can be easily extended to rapidly develop distributed applications.
LMS	Learning Management System.
LWUIT	Lightweight User Interface Toolkit.
mChisimba	A mobile based application that runs on GPRS/3G mobile phones and allows mobile users to connect to desktop Chisimba Realtime Tools.
MIDP	Mobile Information Device Profile.
MILO	Mobile Interactive Learning Object.
MLE	Mobile Learning Engine.
MMAPI	Mobile Multimedia API.
MMP	Multimedia Messaging Platform.
MMS	Multimedia Messaging Service .
MP3	Moving Picture Experts Group -1 Audio Layer 3 .
MS	Mobile Station.

MSA	Mobile Service Architecture.
MXit	A mobile based instant messaging client that connects users in a chat room and allows interaction via text chat and can connect using networks like Yahoo, ICQ, AIM and Google Talk.
AOU	Arabian Open University.
PDA	Personal Digital Assistant.
PDU	Packet Data Unit.
RIA	Rich Internet Application.
RTSP	Real Time Streaming Protocol.
RTP	Realtime Transport Protocol .
RSS	Really Simple Syndication. A technology that allows content, especially in audio format, to be pushed to users whenever updates occur. RSS is also widely used to push text-based content.
SATO	Self Assessment Tool.
SDP	Session Description Protocol.
SMIL	Synchronized Multimedia Integration Language.
SMS	Short Messaging Service.
SGSN	Serving GRPS Support Node.
SSL	Secure socket layer
TCP	Transmission Control Protocol.
TE	Terminal Equipment .
UDP	User Datagram Protocol.
UMTS	Universal Mobile Telecommunications System.
URL	Uniform Resource Locator .
UTF	Unicode Transformation Format
WAE	Wireless Application Environment.
WAN	Wide Area Network.
WAP	Wireless Application Protocol.

WML	Wireless Markup Language .
WTA	Wireless Telephone Application.
XML	Extensible Markup Language. A standard for creating, storing and exchanging structured data. mChisimba uses XML for exchanging information between gateway plugin and server plugin
XMPP	Extensible Messaging and Presence Protocol. An open Extensible Markup Language protocol for near-realtime messaging, presence, and request-response services. XMPP is the main protocol used between the gateway and server for all of their communication structures in the application we developed.



1 Introduction

This thesis addresses the streaming of presentations to a mobile phone and use of interactive text chat in the same environment. The use of presentations enables delivery of rich content to mobile phone learners in an environment where mobile phones are used to augment desktop e-Learning. e-Learning systems, when compared to traditional learning approaches, can be advantageous in terms of convenience, independence, adaptation, and interaction [1]. This is evident in distance education where students are able to access learning content regardless of the distance and location. This is also true for long-life learning and on-the-job-work-force training [1]. In a purely desktop-based e-Learning model, users need a computer and a network connection in order to experience e-Learning effectively, and such equipment is not always readily available, particularly in developing regions. A mobile phone, on the other hand, is readily available in most parts of the world. With constant advancements in technology, most mobile devices in many countries support third generation network services, or 3G [2]. 3G systems have enough bandwidth to support realtime mobile-based communication. For example, Universal Mobile Telecommunications System (UMTS) enables transmission of 384 kilobits/second for mobile systems and 2 megabytes/second for stationary systems [3]. As a result, it is possible to port some functionality available in desktop e-Learning systems to a mobile-based application, as a way of diversifying the infrastructure that can be used in e-Learning. The next section gives a brief introduction to such mobile applications.

1.1 *m-Learning applications*

In general, e-Learning applications are built with a variety of features designed to make learning more convenient. The impact of such features, however, is only realized when the technology used enables transportation of rich content and enables all parties interactively build their understanding in realtime. A mobile-based e-Learning application can supplement this by helping the learner/instructor to conveniently access some of the e-Learning features from virtually anywhere, without the need of setting up a desktop computer to achieve the same ends. Such an application, however, should present content to the user without greatly compromising quality, in spite of the hardware and other resource constraints of a mobile phone. The application should also have a user interface that is familiar (to desktop applications) and easy to use. Applications exhibiting such characteristics are referred to as m-Learning applications. m-Learning can be defined as learning that takes place with the help of electronic portable devices [4]. Due to physical portability of the devices used, m-Learning allows an interesting interaction between mobile computing and e-Learning. It enables the possibility of accessing e-Learning resources wherever one is, with even a

greater possibility of using rich media and experiencing a convenient environment for learning [4]. Features available in a specific m-Learning application largely depend on how data is presented and transported from source to target device. In other words, different presentation and transportation protocols have a direct impact on the type of functionality that can be built into an m-Learning application. Different m-Learning technologies can be used for different learning activities. For example, Short Messaging Service (SMS) can be convenient for vocabulary learning, text blogs can be used for training in writing, and podcasts can be used for content dissemination [3]. Presentations are suitable for rich content delivery, while video can provide rich multimedia content for the learner. Various m-Learning applications have been built, each using specific presentation and transportation technology. Most of the implemented solutions work in their respective environments, although they all face challenges that are worth looking at. Most of the challenges are associated with content formatting and how data is transported between instructor and learner. User interaction is a major challenge facing these applications. In order to understand the reason why such challenges exist, it is worth examining issues involved in presenting and transporting m-Learning information from source to destination.

1.2 Presentations and chat for m-Learning.

When the source data destined for mobile presentation is textual in nature, the most common approach is to extract the text and reformat it using a technique like Wireless Markup Language (WML) [5]. This means that source data needs to be first transformed into formats supported with technologies like eXtensible Markup Language (XML) [5]. This can be accomplished with a highly multi-threaded server-based application container for realtime communication, like Openfire (<http://www.igniterealtime.org/projects/openfire/index.jsp>). Although this can be an appropriate approach, it immediately introduces a number of challenges. First, this strategy is inconvenient when large amounts of content are involved that can result in a huge number of WML pages. The strategy might also not work well if the content is non-textual, for example graphics and audio/video. Possible alternatives include using techniques that support graphics and rich media content. Delivering content using presentations (OpenOffice, PowerPoint) could be an ideal way of implementing this. However, such presentations are usually created from a desktop computer, using complex software that cannot run on a mobile phone. In order to display them on a mobile phone, the presentations will need to be converted into a format supported on the device. One such approach would be to convert each of the slides in the presentation into a series of images, then transporting these images to a mobile phone. Presenting content as images is more convenient, as the technology for manipulating graphics is abundant. Images too can easily represent highly

technical content. Techniques supporting images/graphics involve *image transformations*, which can be broadly defined as operations that can be applied to source graphics in order to obtain target graphics that meet the requirements of the target display screen [6]. Transmission of images (graphical content) is possible in 3G mobile-phone networks, which support sufficient data transfer rates as mentioned above. Presentation of the images on a mobile phone in high quality format is also possible with advance in mobile phone screen technology. To introduce interactivity, a mobile application could include instant text messaging client that connects to a presence and messaging server using Extensible Messaging and Presence Protocol (XMPP). This approach has advantages over use of techniques like SMS. While SMS is provided by all mobile companies, it has limitations. A single SMS message can only support up to 160 characters of text. Content transported via SMS is only limited to text, and as such use of graphical content is not possible. SMS-based learning applications are also limited to short-answer and question-answer scenarios. It is also convenient to use podcasts to transmit content. Podcasts could be appropriately split into small file sizes that fit most mobile phone storage requirements. However podcasts still present a problem in e-Learning; a podcast cannot satisfactorily describe highly technical content that requires visual presentation. Podcasts are passive; there is no direct interaction between the learner and instructor. When podcasts are split into small files they interrupt learning sessions. SMS and podcasts face one common challenge: they cannot satisfactorily describe technical content that requires graphical representation. Such content is best presented and transported in a graphical format. The next section introduces research questions, based on the environment we have just covered.

1.3 Research questions

It would be desirable for an m-Learning application to transmit rich e-Learning content to a mobile phone using a technique that supports immediate interaction between instructor and student using all the four common content formats: text, graphics, audio and video. It would be even more advantageous if such a technique supports near-realtime interaction. Being able to do so can enhance interaction between an instructor and the learner during the learning process. The main aim of this research was to explore m-Learning applications that support the use of presentations to stream e-Learning content to mobile-phone users as well as enable constant interaction between the learner and the instructor, and whenever possible, in realtime. Our research question was:

“How do we provide presentations and interactive chat to support m-Learning?”

Our research question was meant to help us study how to provide presentations on a mobile phone. Our main concern was how to prepare such presentations before we transport them to the mobile phone device for display on the tiny screen of the mobile device. We were also interested in how the users would interact in the mobile environments. We started answering this question by first exploring the techniques used for converting presentations into a format that is easy to manipulate without greatly compromising the quality, while the presentations are still on a desktop computer. We then investigated how to scale down such presentations and transport them to mobile device for display. We developed a proof-of-concept application that formats presentations created on desktop computers and transports and displays them on a mobile phone. We used the software to carry out a series of experiments which we described in Section 3.3. All the experiments were conducted in a laboratory. We developed secondary research questions to concentrate on specific sub-domains of the research. Our first second research question covers the actual process of formatting the slides for mobile display:

“How do we scale down presentations, thus reducing the file size, for mobile device display?”

We answered this question by converting slides in a presentation into a series of images, which were then scaled down to a format and size appropriate for mobile display. The second secondary research question is about the quality of the slides a mobile phone:

“How do we measure the quality of presentation slides when displayed on mobile phone?”

Because an average mobile phone has inherent limitations, the main obvious one being the tiny screen size, it was important to investigate the quality of slides presented on such screens. We used image histograms to measure the quality of slides formatted for mobile phone display. Our third

research question covered the air-time costs:

“How do we minimize airtime costs when sending presentations and conducting interactive chat on a mobile phone?”

This question was answered by implementing an algorithm that allows the mobile client to poll the data from the server only when there has been an update from the instructor's screen. We then recorded the amount of data that is received at the mobile client and used this to calculate the air-time cost implication. The polling process takes place in three separate threads: the first thread checks the status of slides from the instructor and updates the mobile client accordingly. The second thread updates the user list. The third and most important thread with respect to the above research question was the chat polling. This thread refreshes the chat transcript as long as there are new messages. The interactive chat was implemented as a typical instant messaging client, connected to a presence and messaging server. We used XMPP as one of the protocols to achieve this, however, this was only limited to the desktop clients. Mobile-based clients used a combination of Hyper Text Transfer Protocol (HTTP) and XMPP. HTTP was used to poll the server gateway. Although we used a fixed-interval polling model, the actual polling process does not introduce any significant air time cost element. The rest of this chapter discusses the steps we followed in the experiments. The algorithm used for this is presented in Chapter 4.

1.4 Research methodology

The research methodology was primarily experimental. We used 21 presentations available at the repository managed by the library at the University of Witwatersrand. The experiments consisted of converting each of these presentations into formats suitable for mobile viewing, and then transporting them to mobile phone devices, in near-realtime. The presentations were then displayed on a presentations viewer on users' mobile phones/desktops. In order to check that the quality of the presentations is maintained in the process, we employed image analysis techniques to analyze the quality of slides generated when the presentation slides are transformed into mobile-presentable format. We used image histograms for this process. We used transaction logging techniques to record the performance and latency costs. We measured the duration it took to convert slides as well the amount data transported through a mobile during a learning session. We analyzed the data with open-source tools and libraries suited for this purpose. One of the areas we were concerned with as calculating the air-time cost of accessing presentations from a mobile phone. These calculations were done based on South Africa's Vodacom cellular network, using the data bundle costs available at the time of writing this thesis.

1.5 Research tools

The primary research tool was the software developed for the experiments. We used the software to simulate typical m-Learning sessions in which an instructor uses OpenOffice/PowerPoint presentations to deliver lessons to both desktop and mobile phone users via an interactive whiteboard. The software converts the presentations into formats appropriate for desktop and mobile phone display, then transports the content to the users. We integrated other open-source and third-party libraries that were used in implementing appropriate transport protocols, namely XMPP, and for designing desktop-like graphical user interfaces for mobile clients, namely LWUIT (<https://lwuit.dev.java.net/>). Other software libraries were used for transaction logging , Log4J (<http://logging.apache.org/log4j/1.2/>), as well as data analysis, ImageJ (<http://rsbweb.nih.gov/ij/>).

1.6 Thesis outline

Chapter 2 discusses related work, and details the various approaches that have been used by other researchers in implementing m-Learning applications. The chapter focuses on existing text-based m-Learning applications, podcast m-Learning applications and rich Internet applications for m-Learning. We discussed how the applications transmit and present content to the users. The chapter examines the contexts in which these applications are used and concludes with a highlight of constraints of text-based techniques and podcasting in m-Learning environments.

Chapter 3 revisits the applications discussed in Chapter 2. In this chapter, we place emphasis on the shortcomings of each of the applications, and the challenges they face as a result of the technologies they use. Based on the challenges, we present the motivation behind this research. The main research question and the corresponding secondary research questions are introduced in this chapter. The rest of the chapter explains how the research question was answered. It covers the research design, and gives a detailed reporting of how the experiments were carried out.

Chapter 4 covers the design and implementation of the software we developed for the experiments. The chapter starts by examining the user requirements and target audience of the m-Learning applications. The rest of the chapter reports the design structures of each the three components that make up the software: desktop client, server and mobile phone client.

Chapter 5 presents and analyzes the data collected during the experiments in appropriate formats, including graphs and pie charts. This analysis is used for drawing conclusions in Chapter 6.

Chapter 6 summarizes the steps followed when conducting this research and the findings that emerged. The chapter then highlights challenges faced and what needs to be done as future work.

2 Related work

Many studies have been carried out with the intent of porting e-Learning functionality available in most desktop applications to mobile phones. As a result, a number of applications have been built that implement several technologies that can be used to achieve this. This chapter discusses a number of such applications. It also discusses the existing technologies that have been used to transmit and present instructional content when the target device is a mobile phone. Section 2.1 discusses different types of m-Learning systems based on their transmission protocols. Section 2.2 explores the applications based on how they present content to the user. Section 2.3 covers the general protocols used by these applications. Section 2.4 discusses image manipulation. Section 2.5 summarizes the key areas covered in this chapter and points towards the next chapter.

2.1 *m-Learning systems*

m-Learning can be defined as learning that takes place with the help of portable devices [4]. The devices can range from palm tops, mobile phones, personal digital assistants to light netbooks. m-Learning offers an interesting interaction between mobile computing and e-Learning and enables the possibility of accessing e-Learning resources wherever one is with even a greater possibility of using rich media and experiencing a convenient environment for learning [4]. Features available in a specific m-Learning application largely depend on how data is transported from source to the target device, and also how the information is formatted and presented. In other words, different transmission protocols and different presentation modes have a direct impact on the type of functionality that can be built into an m-Learning application, and consequently its usability. In the following sections, we discuss transmission and presentation technologies, and the various applications that have been built in an attempt to implement m-Learning.

2.1.1 Text-based models

The most commonly used text-based mode of transporting m-Learning content is SMS [7]. SMS is one of the value-added services provided by Global System for Mobile Communications (GSM) networks in addition to voice. Most SMS messages are created by typing using the mobile keypads, which is not always a convenient thing to do. SMS allows users to exchange alphanumeric messages, of up to 160 characters¹, with other users instantly. When using SMS, the delivery of the message is guaranteed even when the target phone is unavailable. The network will store the messages in cases where the target phone is offline and deliver it once the target phone becomes

¹ 160 characters are based on 7-bit characters. Some sources cite 140 8-bit characters

available over the network. SMS supports international roaming with very low latency, a fact made possible by the interconnection of GSM network. Two types of SMS are available: *cell broadcast*, and *point-to-point* [7]. In cell broadcast, all active mobile phones that have subscribed to a service receive SMS messages that are transported. It is a one-way service, and there is no confirmation of the receipt of the messages sent. Point-to-point service allows messages to be sent from one mobile phone to another directly, or from a personal computer to a mobile phone, or vice versa. An SMS center is used to maintain and manage the transmission process. Point-to-point service supports confirmation of receipt of messages. Because delivered SMS are automatically stored locally, it can be particularly useful for quick future reference from detailed information, for example a class booking, test scores and short course notes.

Several m-Learning applications have been built with SMS functionality. Early applications implemented this by providing user interfaces that consisted of true/false, multiple-choice options and ranking-matching questions [8]. Others improved on these systems by introducing question-answer models and query-based models where the user would either query for information and get instant replies or provide instant replies to supplied queries [8]. One such system has been developed and published by Wen et al. [8] According to them, some existing SMS applications were mainly developed for administrative purposes, like delivering messages to students as reminders or alerts for some learning activities. Wen et al. present a system which is used to query information and knowledge by use of SMS messages. These messages convey content related to learning processes like glossary of items, course summaries, examination preparation notes, student guidance, answers to exercises and second language learning tips, in a mobile environment. The system consists of a GSM module, a dialogue control module, a querying processing module and a knowledge base. Users send SMS queries to the GSM module, which in turn forwards the query to the dialog control module. This generates suitable query tasks for query the processing module. The query processing module searches and matches information from a knowledge base or the Internet to produce suitable answering messages for the users. This is illustrated in Figure 2.1. This system works on a one-to-one basis; each user sends a request to the server individually and awaits for the results. The communication between the system and the human user is via a predefined protocol. For example, a student might send a string like “Get CN-3.3 COMP611” [8], where these words are following a pre-specified protocol. The string is then parsed, and used to query a database. Then appropriate results are sent back to the student.

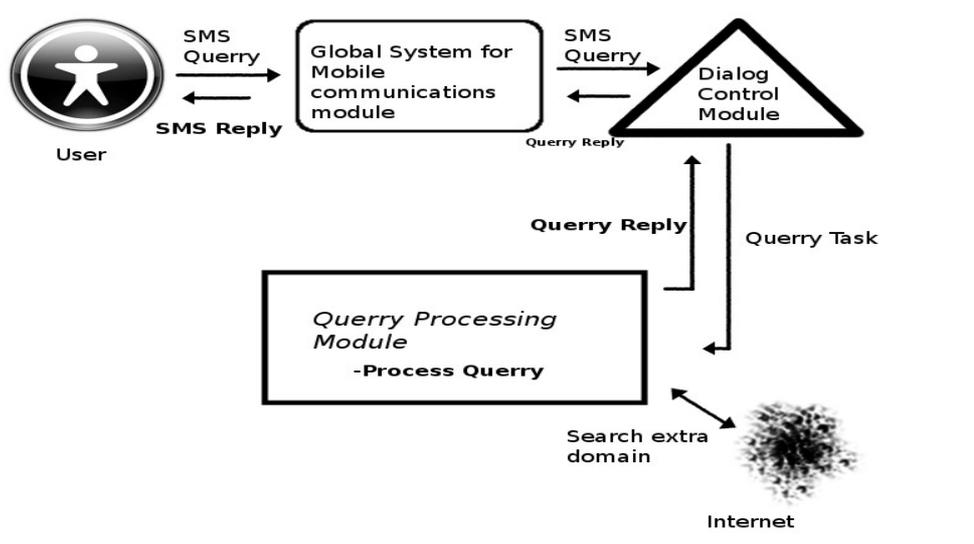


Figure 2.1: Diagrammatic view of SMS-based m-Learning application.

A user sends an SMS query to a GSM module, which forwards it to a dialog control module in the appropriate format. The dialog control module then generates a query task for the query processing module. This searches for information in a knowledge base and the Internet if necessary then, returns an appropriate reply to the user.

The system developed by Wen et al. is only suitable in environments where the information being queried can be provided in text format. Audio, video and graphical content cannot be parsed. The system also relies on the availability of the information in the knowledge base, lack of which would provide no answers to a user. Although the system provides timely responses to queries, it does not provide any direct contact between an instructor and a student.

Another SMS-based m-Learning system, developed by Balasundaram and Ramadoss, uses the question-answering model, in which instructors disseminate questions to students via SMS texts [9]. When students answer questions from instructors, they might reply using natural language, or some answers might not be unique. As an example given in the paper, an instructor might ask a free form question: 'How many bits a byte has?'. Typical answers would include more accurate ones like '8 bits one byte', 'one byte has 8 bits'. Other answers may be like 'I think it is 8 bits for a byte' or 'Is it 8 bits per byte?'. As illustrated by these answers, it is challenging to accurately computationally process some of the answers. The SMS-based m-Learning system uses a simple matching algorithm to extract key words from an answer matched against expected key words. Depending on the number of occurrences of the key words, a 'significant level' is computed and is used to declare a correct or wrong answer. This system is only limited only to a question-answering model which can only support short-sentence answers.

Another popular mobile text-based application, MXit (<http://www.MXit.co.za>), is a South Africa-based instant messenger that has been used in education by school children [10]. It connects

users in a chat room and allows interaction via text chat. Users can also connect using popular IM networks like Yahoo, ICQ, AIM and Google Talk. According to Tangkuampien, the deployment for use in educational environment was achieved by building a 'bot' that could send educational information to any user who sent it blank message using Mxit [10]. Information sent back could be examination tips. A 'bot', in this context, refers to a computer program that can participate in an online chat. Tangkuampien reported that the MXit bot recorded a sharp increase in usage by students two weeks preceding the final examinations. Interviewing some of the students after the examination revealed that they felt the bot service was useful to them, without which they would not have been able to answer some of the examination questions. The paper reported one major challenge MXit users faced. It is reported that users experienced delays when using the system. It was reported that some users could spend up to 50 percent of the time waiting for a response after sending a chat message.

The systems presented by Wen et al., Balasundaram and Ramadoss, and Tangkuampien do not support graphical content. Most of the text-based applications discussed were developed to satisfy needs of a specific domain of e-Learning. They are not able to fully satisfy today's education-hungry population, who want to access far more diverse content than just text data, any time, anywhere.

2.1.2 Podcasting and Image m-Learning applications

On mobile phones, the most commonly used technology for distributing e-Learning content in audio/video format is via podcasts. A podcast is a file in audio and/or video format that can be distributed over the Internet using Really Simple Syndication (RSS) technology, for listening/viewing on mobile devices and personal computers [11] A podcast is automatically downloaded onto the target environment using software capable of reading an RSS feed. RSS uses XML to present a strictly formatted content that can be read and interpreted by any standard RSS feed reader. This is based on a push technology such that a publish/subscribe model is used. This model is illustrated in the Figure 2.2.

The content distributor starts by releasing the recorded podcast onto a web server in the RSS environment. The environment is typically configured to model the publish-subscribe scenario shown in Figure 2.2. The subscribers' feed-reading software will automatically start downloading a new file once it is published, making it available to the user immediately. Table 2.1 that shows how podcasting compares to SMS based services. The data presented in Table 2.1 is based on the discussion to be presented in Section 2.1.1 on SMS and Section 2.1.2 on podcasting.

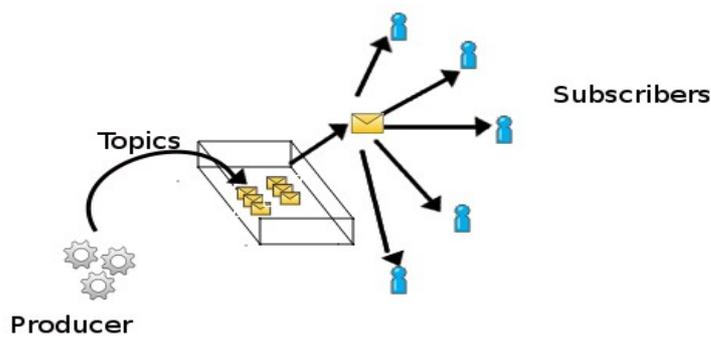


Figure 2.2: Push - Subscribe model often used in distributing podcasts.

In this model, there is only one sender and one or more receivers. There is no queuing to send or receive a message. Instead, interested clients subscribe to a topic in order to receive messages associated with a topic. The consumers are delivered messages without having to query for them.

Table 2.1 shows that podcasts improves m-Learning over text-based application by being usable in more diverse environments. A number of applications exist that use this technology to deliver e-Learning content to portable devices, including mobile phones. Some of these are discussed next.

An m-Learning project, involving use of podcasts, was implemented at the University of Ulster to investigate the delivery of supplementary e-Learning material, targeted amongst other devices, mobile phones [11]. Lectures were recorded and posted on a web site using RSS feeds. Mobile-based subscribers could then be aware of any updates, based on the RSS model. Due to the technical nature of the content involved, the project was faced with one major challenge: presentation of highly technical and complex material that audio could not satisfactorily describe.

Another major challenge faced by the project was the duration of the lectures. Hour-length sized recordings were initially broken up into manageable portions. However this process required more efforts in terms of editing the recorded content and would disrupt the flow of the lecture.

Uses/Application Type	Podcast-based m-Learning applications	Text-based m-Learning applications
Record live lectures	✓	
Access missed lectures	✓	
Good for assessments	✓	✓
Useful for life-long learning	✓	
Can be used for quick short tutorials	✓	✓
Good for short sequences of definitions and summaries	✓	✓
Used for archival purposes	✓	
Can be very useful for the visually challenged students	✓	

Table 2.1: Possible uses of podcasts compared to SMS-based applications for e-Learning.

In this table we see that podcasts are suitable for a wider variety of environments than text-based systems.

In another project implemented by Lightbody et al. a study was carried out to explore the effect of podcasts on students who lived at a distance from their learning environments [11]. The researchers used short, three to five minute podcasts that were structured as informal talkback-radio-style segments. The researchers developed a podcasting system where the recordings were in Moving Picture Experts Group -1 Audio Layer 3 (MP3) format and were uploaded onto a Learning Management System (LMS). These segments were accessed as RSS feeds. An LMS platform provides an online learning environment that enables management, delivery and tracking of learning material in both traditional and online classrooms. Students were given the option of accessing the podcasts via their mobile devices, or direct download from the site and uploading them to their mobile devices, e.g from laptop to a mobile phone. The study observes, among other things, that podcasting provides a flexible option for the delivery of e-Learning content, for recorded audio allows the user to listen without requiring visual fixation. However the researchers also observe that a number of participants were not satisfied with podcasting as means of delivering learning material, and favored an environment in which they could have direct interaction and collaboration with the instructor and/or other sources of information in near realtime [11].

In general, podcasting is a good option for disseminating e-Learning content to mobile users due

the ease with which one can listen to recorded audio. However, in modern educational environments, audio alone cannot satisfactorily describe the content, especially graphical content. Also as noted by Lightbody et al, most users prefer an environment where they can engage the sources of information in a near realtime collaboration and interaction modes.

Doctors, nurses and medical students work in a highly mobile environment. Although most hospitals provide access to stationary clinical workstations, their working stations do not always coincide with these sources of information. As a result, it is desirable to provide a source of information that is independent of location and time: a mobile source. Doctors, nurses and medical students can benefit from such sources through bed-side teaching or problem based learning. Mobile Learning Engine (MLE) is a multimedia application that was developed to address these scenarios [12]. MLE was developed using J2ME. A core ability of the MLE is the presentation of learning objects enhanced with formatted continuous text, integrated images, clickable hyperlinks, control bars for audio and video playback and a graphical question management system with an ability to mark hot spots within a picture. An MLE screen shot is shown in Figure 2.3.

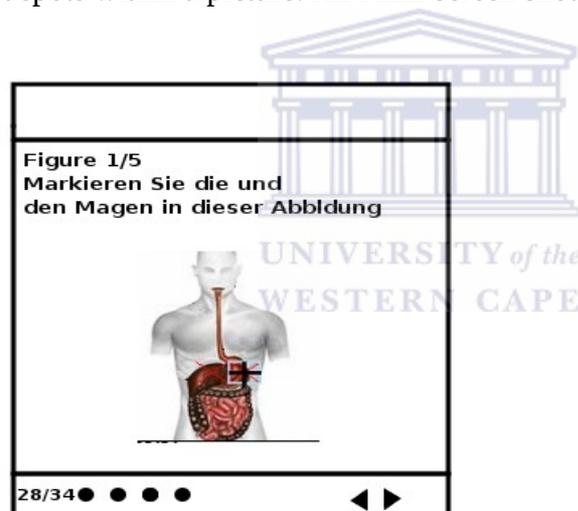


Figure 2.3: Mobile Learning Engine (MLE) illustration.

MLE presents learning objects on a mobile screen that are enhanced with formatted continuous text, integrated images, clickable hyperlinks, control bars for audio and video playback and a graphical question management system with an ability to mark hot spots within a picture [12]

MLE operates on the principle of Mobile Interactive Learning Objects (MILOs) [12]. MILOs are learning objects for e-Learning that are structured carefully due to the target environment in the which they are used: the mobile phone. MILOs created in MLE are XML based, offering the instructor the flexibility of easily changing the content. MLE offers a combination of text, audio,

multimedia and graphics. This makes them desirable for m-Learning, as they improve on the services offered by text-based m-Learning applications and mobile-based podcasts. However, none of these features are offered in an environment where the users can freely and easily interact with the instructor in near realtime, a feature that would enhance the learning.

2.1.3 Rich internet applications for m-Learning

Rich Internet Applications (RIAs) refer to applications that are Internet-based yet they have characteristics of desktop applications. These applications are typically deployed in a web browser that supports standards required to run RIAs, via browser plugins or independently in their own virtual machines. RIAs have certain advantages over traditional web applications in that they are more interactive and responsive. Unnecessary page reload, multimedia support and drag and drop are some of the basic characteristics of these applications. The term 'rich' usually implies these applications support complex graphical user interfaces using advanced interaction patterns. The term 'Internet Applications' highlights the fact that such applications implement intensive use of communication networks. With the advancement in mobile technology, most mobile phone devices are able to support RIAs either via mobile browsers or virtual machines available on the devices. When implementing RIA for mobile applications, a developer is faced with a decision of choosing fat-client applications or thin-client applications. Fat-client applications need first to be downloaded and installed on client device. The application then runs locally on the device and is capable of using the device's processing and storage capabilities. Fat clients typically are implemented to run on the Java ME platform. Thin clients do not require any downloads, and if they do, it would be only lightweight downloads. Such clients can simply be accessed via the phone's web browser. Either way, RIAs are typically executed within a particular runtime environment, either pre-installed on the target device by the manufacturer or installed as a browser plug-in. RIA user interface layouts are usually implemented using scripting languages, with the business logic and back-end services running on a remote server. It is also often the case that RIA are implemented as native applications on the phone, in which case such applications benefit from the fact that they potentially run faster as they have direct access to the device's resources. Such applications access directly the underlying native features using the operating system's Application Programming Interface (API). Most common RIA frameworks for mobile phones include Java/JavaFX and Adobe Flash. This section covers the two technologies.

Flash Lite is a mobile version of Flash software developed by Adobe (<http://www.adobe.com>). It is intended to allow mobile phone users to view and use multimedia content and applications

developed using Adobe's Flash desktop tools. This technology is implemented at the client side, and developers can create applications using ActionScript. Flash Lite can be used for creating flash cards that are useful for reviewing foreign language vocabulary and grammar. This is a good choice for learners who wish to learn new words. In addition, Flash Lite can be used with science lessons, for example creating flow charts and also for solving mathematical problems. <http://freedom-moat.bile.jp> is one example of a site providing flash based rich format contexts on mobile phones.

At the Institute for Medical Informatics, Statistics, and Documentation, Medical University of Graz, Austria, a project titled Self Assessment Tool (SATO) was implemented to provide a self-assessment tool for civil engineering students [13]. The tool allowed students to learn about calculation of normal stresses for a typical cross-section and material and supported self-directed learning. The initial SATO implementation was for a desktop computer, but later a prototype version was ported to a mobile phone using Flash Lite. Flash Lite was chosen as the mobile platform at the time because it provided an environment to build good animations that were the core of interactive learning objects used in the application. Although Holzinger and Ebner do not give any detailed information of what specific features were implemented with Flash Lite, they conclude by saying the actual prototypes were built only to run on Personal Digital Assistants (PDAs), since Flash Lite was not available on smart mobile phones in Austria at the time of the publication [13].

Another multimedia based technology, JavaFX, developed by Sun Microsystems, is a platform that allows developers to build rich applications that run on the desktop, the web and mobile devices. Using JavaFX, developers can easily build robust, impressive applications that can be deployed anywhere, most importantly the millions of Java-enabled mobile devices. The JavaFX Tech Test Train (<http://www.javafx.com/learn/training.jsp?>) is an e-Learning and m-Learning JavaFX application that can be used for JavaFX training and certification. The application is built as a game and is fun to use. Taraghi et al. describe a mobile-based application named FeedBoard that could be used in personal learning environment [14]. Because FeedBoard is implemented in JavaFX, it runs on the desktop as well. The paper describes the prototype FeedBoard as being capable of downloading specified RSS feeds, parsing and displaying the results using different views. Users of FeedBoard are notified every time a new feed is available. FeedBoard is implemented as a widget. A widget is a small client side application that provides a graphical user interface and typically can be embedded in a webpage, or a desktop. Widgets in learning environments allow simpler but distributed transfer of knowledge. Different widgets can be built to provide some of the features available in large LMS. FeedBoard is a widget for personal learning environments.

2.1.4 Mobile based learning management systems

Moodle (<http://moodle.org/>), a well known LMS, uses a relatively advanced software designed for planning and managing learning activities on-line and off-line. It was deemed desirable to have the ability to access these features from anywhere anytime. Houser and Thornton developed Poodle, a mini LMS course management tool designed to read quizzes from the Moodle system [15]. The quizzes are read from the Moodle server, then they are reformatted to fit a tiny mobile screen, each question presented as a small web page of its own. Students can then answer the questions and Poodle sends the results back to the instructor. Poodle was also used for polling. During a lecture, an instructor could decide to pose a question. The question was posted on a website that was accessible by Poodle. Students with a mobile phone could then use Poodle to participate in a poll, with results sent back to the instructor for analysis. Another feature mentioned by Houser and Thornton was the ability to edit a Wiki using Poodle. A special Wiki server was developed for this purpose, and this allowed instructors and students to collaboratively edit content. Houser and Thornton report that the use of Wiki was not a success, as most students confused it with a Forum and started posting replies or comments to an article as separate text. As a result, a Forum was eventually implemented to satisfy these needs. Houser and Thornton report general success of the Poodle project [15]. However Poodle supported a rather small number of features to be effectively used in learning environments.

Researchers at the Arabian Open University (AOU) designed a Wireless Application Protocol (WAP) based application that was integrated into Moodle and used for m-Learning [16]. To make this work, a mobile telecommunications service provided mobile phones and the WAP service subscriptions. Interested students could subscribe and access e-Learning tools via their mobile phones. One of the services provided included an e-discussion application that allowed students and a tutor to access e-discussion forums via their WAP enabled phones. The tutor could perform administrative functions like managing forums and work groups as well as carrying on discussions. This included creation of new threads, viewing, replying and deleting messages. Four types of forums were available:

- 1) News Forum: Mainly used for announcements such as examination time tables.
- 2) Course Forum: used for disseminating course contents to registered students
- 3) Tutor group forum: used for tutor-student interaction. Participants could create new threads, read messages and reply to posts.
- 4) Dialog forum: used for discussions between a student and the tutor but in privacy mode,

where others can't see it.

Alsadi and Abushawar report several technical problems that were encountered when using WAP technology [16]:

- 1) Most of the phones experienced short battery life span
- 2) Some students were having difficulty in logging into the WAP-based forums.
- 3) There were slow transmissions, and failure to send information to target devices.

Although Alsadi and Abushawar did not explain in detail the cause of these problems, they conclude by stating that WAP can be used as a successful online technology for education if more research is done on the technology in order to refine it [16]. Alsadi and Abushawar suggested that WAP is suitable for creating of mobile learning course material, if it is optimized and carefully used in designing applications that can be used realtime long-distance learning, as demonstrated by AOU.

2.2 *m-Learning data transmission*

In this section, we discuss the data transmission services that can be used in m-Learning and the protocols used to accomplish this. Section 2.2.1 discusses SMS, the most widely spread packet oriented mobile data service, available to Second Generation (2G) cellular communication systems using GSM. Section 2.2.2 discusses wireless application protocol as one of the implementations of data services on 2G networks. Section 2.2.3 discusses the third generation networks and the data services they offer.

2.2.1 Short message service

As introduced earlier, SMS is a service that was developed as part of GSM Phase 2 specification. SMS allows a cellular terminal to send and/or receive short messages in alphanumeric format that are up to 160 characters in length. The European Telecommunications Standards Institute (ETSI) is the body that specified the protocol for short message submission as part of the GSM standard [7]. Three interface modes are defined for the transfer of SMS short messages between a Mobile Station (MS) and Terminal Equipment (TE). This is done via an asynchronous interface. The first, the *block mode*, includes error detection mechanisms and is therefore suitable for use in environments where the probability of an error occurring is high if the link between the application and the phone is not reliable. With this block mode, the message is sent as a binary string and includes a header and the short message packet data units. The Block mode is described in Table 2.2. Block mode includes the following functions:

- submission of a short message
- deletion of messages from the phone
- listing of messages in the phone
- transfer of all messages or one from the phone to the application
- configuration of the application such that it is notified every time a new short message is received.

The second, *text mode*, and allows an application to pass messages in plain text to the phone, which in turn constructs appropriate text packet data units. Text mode is limited in functionality when compared to block mode, and it does not support features like notification of incoming messages. The Table 2.3 shows an example text mode.

Element	Meaning	Length in bytes
Message type	Insert SMS type: the value defined in the specification is 0X07	1
Insert type	1. Store in phone 2. Send or 3. Store and send	1
Recipient destination address	Address of the recipient as defined by GSM standard 04.01	01/12/09
SMS text packet data unit	As defined by GSM 03.40	Max 140

Table 2.2: Short message submission using block mode

Command	Meaning
AT+CMGS="27727763724"<CR> This is a text message<CR> ^Z	Send a message to 27727763724
+CMGS=3 OK	Message accepted by the phone with a reference number 3.

Table 2.3: Short message submission using text mode

The third mode, Packet Data Unit (PDU), is similar to text mode with the exception that the application is responsible for building short message text packet data units. This mode allows more sophisticated packet data units to be constructed. When a message is send as a PDU string, it

contains not only the message, but also meta-data about the sender, his/her SMS service center, the time stamp etc. This is formatted in hexa-decimal *octet* or decimal *semi-octets*. The following string is an example of what is received on a [Nokia 6110](#) when sending the message containing "hellohello" from [www.mtn.co.za](#).

```
07 917283010010F5 040BC87238880900F10000993092516195800AE8329BFD4697D9EC37
```

The following example shows how to send the message "hellohello" in PDU mode from a Nokia 6110.

```
AT+CMGF=0 //Set PDU mode
AT+CSMS=0 //Check if modem supports SMS commands
AT+CMGS=23 //Send message, 23 octets (excluding the two initial zeros)
>0011000B916407281553F80000AA0AE8329BFD4697D9EC37<ctrl-z>
```

Examples of applications that use SMS as their mode of transporting m-Learning content were discussed in Section 2.1. The next section discusses the data oriented protocols that allow transmission of more content than SMS, also operating on GSM networks.

2.2.2 General Packet Radio Service

General Packet Radio Service (GPRS) is a value-added service that allows GSM networks to provide packet-based communication [17]. GPRS network has an interface that allows it to be connected to an Internet protocol-based packet data network. As a result, GPRS supports Internet Protocols (IP) as well as traffic for terminal equipment in a Wide Area Network (WAN), through a GSM wireless connection. Because GSM is a 2G network, GSM cellular system combined with GPRS is known as 2.5G. In a conventional GSM setup, data transmission rates are typically between 56-114 kilobits per second (kbits/s). GPRS billing can be based on per megabyte of data transferred, as compared to traditional circuit switching which bills per minute/second of connection time, independent of whether the user is using the service or is in idle state. The advantage of using data-based billing is that a user can be online for a long period of time but will be only billed based on the volume of data transported. A GPRS user who wants to exchange data packets with an IP network gets assigned an IP address. The IP address is supplied by the GPRS operator. This is accomplished by using Dynamic Host Configuration Protocol (DHCP) . The mapping of IP addresses to GSM addresses is performed by Gateway GPRS Support Node (GGSN). This kind of configuration allows GPRS to act as a wireless extension to the Internet. It allows the mobile user to have a direct connection to the Internet. This is illustrated in the Figure 2.4. Most

mobile phones with built-in browsers use IP version 4. Transmission of data using IP has certain challenges, especially when this takes place over GPRS. The main challenge is data loss, and this might lead to degradation of quality. The main causes of this include fading, hand off and other radio effects associated with GPRS. To address this problem, GPRS supports protocols that minimize data losses. One such supported protocol is Transmission Control Protocol (TCP) [18]. TCP requests retransmission of lost packets, but at the same time prolongs transmission times as well as increases the amount of data sent over the network. This can be costly on a GPRS channel.

User Datagram Protocol (UDP), on the other hand, is a minimal message-oriented transport layer protocol. UDP is documented in Internet Task Force (IETF) Request For Comments (RFC) 768 [19]. It does not ensure delivery of packets, nor does it provide correct ordering. UDP is supported under GPRS, and often makes a better choice since it generates less overhead than TCP as there is no need for acknowledging packet delivery. As a result additional protocols have been developed to take advantage of UDP's quick data delivery.

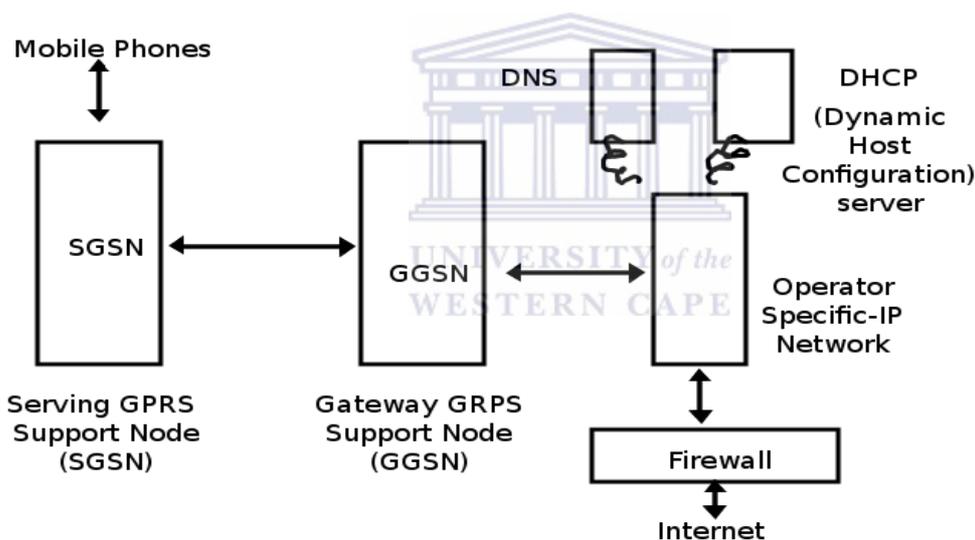


Figure 2.4: How GPRS interfaces with the Internet

A GPRS user who wants to exchange data packets with IP network gets assigned an IP address. The IP address is supplied by the GPRS operator. This is accomplished by using DHCP. The mapping of IP addresses to GSM addresses is performed by a GGSN. This then maps to a Serving GRPS Support Node (SGSN) that has an interface to mobile phone infrastructure.

The Real Time Streaming Protocol (RTSP) is an application-level protocol for control over the delivery of data with realtime properties [20]. RTSP provides an extensible framework to enable controlled, on-demand delivery of realtime data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data

delivery sessions, provide a means for choosing delivery channels such as UDP, multi-cast UDP and TCP, and provide a means for choosing delivery mechanisms based upon Realtime Transport Protocol (RTP) [21]. RTSP supports UDP, and it is available for GPRS. From a developer's point of view, RTSP has been implemented via the Mobile Multimedia API (MMAPI) of J2ME, when Java is the target platform [22]. A developer can use this protocol by creating a media player on the mobile device. When the client successfully connects to a server, this API is used to play multimedia content streamed from the server.

Because of successful support of IP by GPRS, a dedicated data-oriented protocol was possible: WAP. The WAP standard is based on Internet standards (HTML, XML and TCP/IP). In addition, WAP can run over GPRS, if it is available.

2.2.3 Wireless Applications Protocol

The WAP was first published in a specification drafted by WAP Forum, which then consisted of main cellular companies: Ericsson, Motorola, Nokia and Unwired Planet (<http://www.openmobilealliance.org/Technical/wapindex.aspx>). The membership has since grown. The objectives of the WAP forum were:

- to make Internet and other digital data services available on mobile phones
- to create a standard wireless protocol that can work across different wireless networks globally
- to enable developers and users to create content and scalable applications that can be supported by a wide range of devices and wireless networks
- embrace and extend the existing standards in order to improve wireless technology

With these objectives in mind, a WAP 1.0 specification was published. Over time there has been revisions and the current specification is WAP 2.0. Before the introduction of other more powerful and sophisticated wireless protocols, WAP was a very crucial technology as it allowed users to have access to desktop-like applications running on mobile phones. Since a mobile device is highly portable, users could use these applications anytime anywhere. WAP is a data-oriented protocol. It allows users to securely access data-oriented services from anywhere and any time using mobile phones and communicators, where GPRS is available. WAP operates on the concept of a stack, just like the Internet stack, as shown in Figure 2.5.

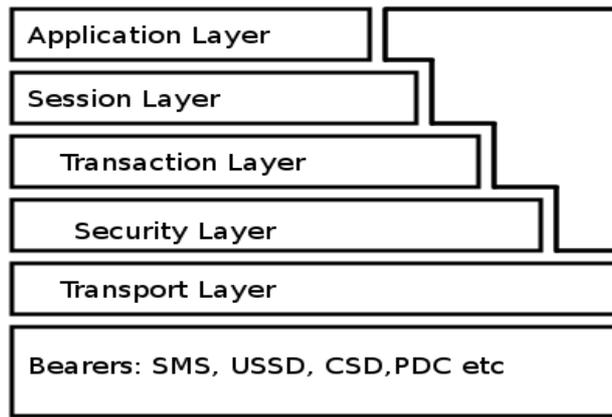


Figure 2.5: Layered communication stack in a Wireless Application Protocol

Just like the Internet stack, WAP operates on a layered stack. The stack is composed of an application layer (mini browser would be at this layer), session layer, transaction layer, a security layer and transport layer.

Two main essential features of WAP are: an end-to-end application protocol and an application environment that is based on a browser. A typical configuration consists of WAP client and a WAP server. A WAP client initiates connection to the server by sending an appropriate request that has the server Uniform Resource Locator (URL), to a WAP gateway using the WAP protocol. The WAP gateway then forwards this request as a conventional HTTP request with the URL. When the server receives the request, it acts accordingly. The server might send a file back to the gateway in form of HTTP replies with appropriate headers, or it might launch an application, grab the output from the application and send it back to the WAP gateway using HTTP. The WAP gateway would then verify the replies, encode them into an appropriate binary format and forward the response to the mobile device using WAP.

There are many applications that can be used with WAP (see Figure 2-6). A few examples of such applications as listed by Erlandson and Ocklind, in their technical paper [23]:

- **Internet browsing:** WAP can be used to build application that can connect to the Internet and allow users to browse information. Such applications, typically, would be WAP browsers. A WAP browser is however limited in terms capabilities, especially tiny screen and memory constraints. These limitations are imposed by the mobile phone, which in itself is has limited resources.
- **“Serviceman application”:** a serviceman with a WAP-enabled mobile phone can benefit from having a WAP application installed on the phone that enables him/her to access

company records remotely, for example, to check the availability of spare part. This same application could contain functionality that allows the serviceman to order such a spare part.

- **Notifications:** A WAP application can be used to notify users on various issues, for example when a user receives an email in his/her inbox. The application could be configured to receive a notification from the server. Upon receiving such a notification, it can notify the mobile user in an appropriate form, for example short ring tone or vibration.

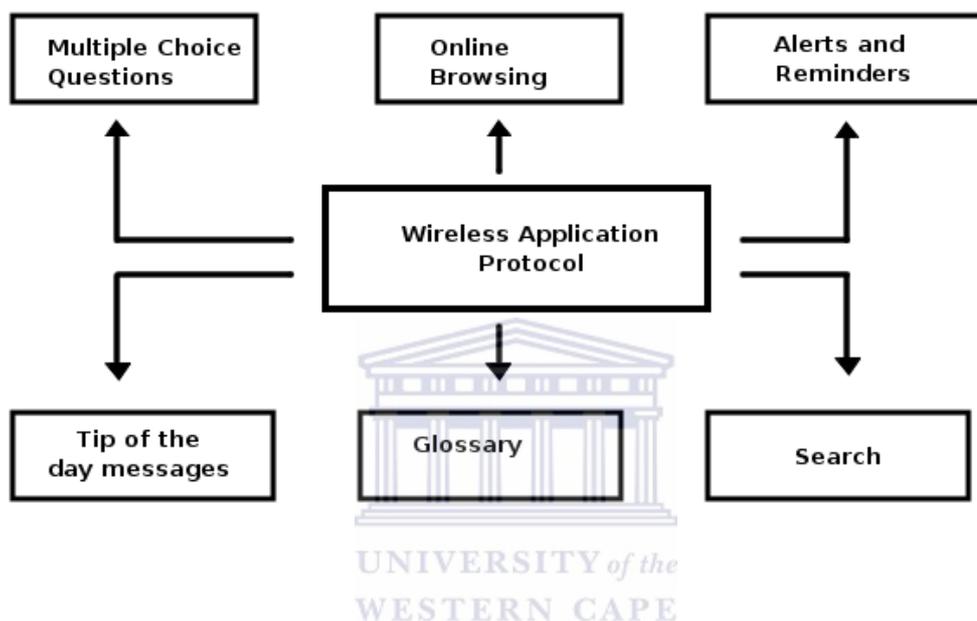


Figure 2.6: The various uses of Wireless Application Protocol.

From posting multiple choice questions to users, WAP can be used to browse the Internet, deliver alerts and reminders, display tip of the day messages and on line search.

WAP faces one major challenge. This challenge arises from the fact the WAP uses WML which inherits most of the syntax from Hyper Text Markup Language (HTML), but it lacks many features available in HTML. WML is based in XML, and as such its more strict than HTML. As a result, only a limited number of features can be incorporated in WAP applications. For example, WML does not support data synchronous streaming. Advances in wireless technology led to improvements in data transmission techniques. Newer, more powerful, faster and sophisticated networks were born, and these are discussed in the next section.

2.2.4 Third generation networks

The 2G and 2.5G networks discussed in the preceding section only allow data transfer rates of between 56-114 kilobits per second (kbits/s). These data transfer speeds can not satisfy the needs of applications that make use of services such as wireless voice and video calls, and other applications that require higher bandwidth to efficiently transmit rich data to end users. As a result, the International Telecommunications Union (ITU) defined the third generation (3G) mobile standards, referred to as ITM-2000, to facilitate growth, increase bandwidth and support more diverse applications [24]. The transition from 2G to 3G networks started with an improvement in the data transfer in GSM environments by introducing more sophisticated systems with higher data-transfers such as the Enhanced Data GSM Environment (EDGE).

3G networks are characterized by several features. One of the most important features is data rate. Although the ITU did not provide a clear definition of the data rate that users should expect from 3G networks, the proposals state that 3G networks should provide a minimum of 2 Megabits per second for local-area coverage and 384 Kilobits per second for wide area coverage [24]. These kind of speeds can work well for applications that require high quality streamed content. Such applications include:

- 1) Mobile TV: TV can be viewed on a mobile phone operating on a 3G network due to high data transfer rates offered.
- 2) Video conferencing: this is feasible due to the enhanced bandwidth, which can allow streaming of video content up the speeds of 2 Megabits per second
- 3) Telemedicine: this could be a service that can be used used on top of video conferencing to allow patients in remote places to get attention from a doctor

3G is also characterized by improved security features compared to 2G networks. This is achieved by allowing user equipment to authenticate against the network it is connecting to. This is accomplished by use of advanced encryption algorithms [25].

To successfully provide a mobile streaming service based on a 3G network, a common standard is required as there is no guarantee that all target mobile devices will be able to support all streaming formats used. This can be achieved by establishing standardized components that define features such as multimedia protocol stacks and codecs. The Third-Generation Partnership Project (3GPP; <http://www.3gpp.org>), is a body for mobile streaming standardization. 3GPP also addresses areas like video conferencing and services that provide multimedia messages using text, images, audio or short video clips/streams. Outside mobile streaming, 3GPP also covers all aspects of

mobile communication systems including network infrastructure, 3G terminals and 3G services. Some of the protocols defined in the 3GPP standards include:

- 1) RTSP and Session Description Protocol (SDP) for session setup and control,
- 2) Synchronized Multimedia Integration Language (SMIL)
- 3) HTTP and TCP for transmission of session layouts, images and text
- 4) RTP: for transporting realtime data such as voice and video.

These protocols are meant to be used in wide area networks. There are instances when devices connected to 3G networks communicate with each other, and it turns out the traffic between the devices is purely local. For example, vending machines sending information to suppliers or cars sending service requirements to a garage can benefit from a technology that is less expensive if the devices they communicate with are within local reach. Bluetooth is such technology, and is discussed in the next section.

2.2.5 Bluetooth

Bluetooth is a wireless technology that offers wireless connection between devices over a short distance [26]. Bluetooth is now widely available in mobile phones and Personal Digital Assistants (PDA). Bluetooth uses unlicensed Industrial, Scientific, Medical (ISM) frequency band, and as such any data transmissions between devices are not chargeable. Devices with Bluetooth can act in peer-to-peer fashion amongst themselves.

Java Specification Request 82 (JSR -82) defines an API that can be used by all Bluetooth devices that support J2ME [27]. This API is built using standard J2ME and the Generic Connection Framework (GCF) that is defined in the Connected Limited Device Configuration (CLDC). JSR-82 has concluded and is bundled inside the J2ME Wireless Toolkit. Bluetooth devices talk to each other to form a piconet, where one piconet is composed of one master device up to seven slave devices [27]. A master can have 7 active or 255 inactive slaves per piconet. Systems using Bluetooth thus can implement this by having short quick communication bursts between the devices.

One m-Learning system that uses Bluetooth technology was developed using the Java API for Bluetooth Wireless Technology (JABWT) [28]. Communication channels are secured using authentication and encryption. The Bluetooth m-Learning system allows an instructor to interact with students, access lecture resources and get statistics on a student's performance. A student using the system can actively participate in a class and answer an instructor's questions. The system is

designed on a client-server architecture, in which the client is a mobile phone, although it could be a PDA or even a laptop. The server side is a desktop computer or a laptop. Communication between the client and the server is via Bluetooth. An instructor wishing to interact with students will initialize the server from a desktop or a laptop, thereby advertising the e-Learning service. This is done by registering the e-Learning service in the service discovery database. A student with the Bluetooth enabled mobile phone can then run the client, which will automatically search and discover available e-Learning services, from which he/she can select and join.

Malliou et al. implemented a project named 'AD-HOC Project – e-Learning anywhere, anytime', aimed at developing a multimedia language learning tool capable of running, amongst other hardware, portable hand-held devices supporting wireless networks [29]. The project's objectives were “to encourage linguistic diversity throughout the European Union (EU) and support the ‘life-long learning’ by motivating and aiding individuals to learn languages any time and in any place”. Although at time of writing the project was a work-in-progress, part of the programming aspects included the implementation of device-to-device networking using Bluetooth. Bluetooth seemed a good option because of its low power consumption and extremely low or no costs during transmission of data including audio data. The user-device, typically a hand-held terminal like a mobile phone, connects to a hotspot by sending the service provider the terminal characteristics. The provider then transmits the required information based on the capabilities of the device. The end-user product of the AD_HOC system is educational content that is broken down into small, independent multimedia modules. This is achieved by using a Multimedia Messaging Platform (MMP) that provides a two way communication user-interface. The back-end consists of a web-based application running on a web-server, used for collecting responses from the users, directly from the web or from the mobile network. Figure 2.7 shows how the system was modeled.

Bluetooth technology is costless to the user and consumes minimal battery power. However, Bluetooth technology can only be used in short range distances, where the service providers have connection points available. This section has covered transmission protocol options available to mobile devices. The next sections discusses presentation schemes.

2.3 *m-Learning data presentation*

Before data meant for a desktop computer is displayed on a mobile phone, it needs to be formatted into appropriate format. This is because mobile phones do not have the same level of resources as desktop computers. Most mobile phones, as pointed out earlier, have constrained resources such the size of display screen and limitations in memory and computing power. This

section looks at some of the techniques used to format data destined for mobile phones. For the context of this thesis, we will limit the discussion to textual and graphical content.

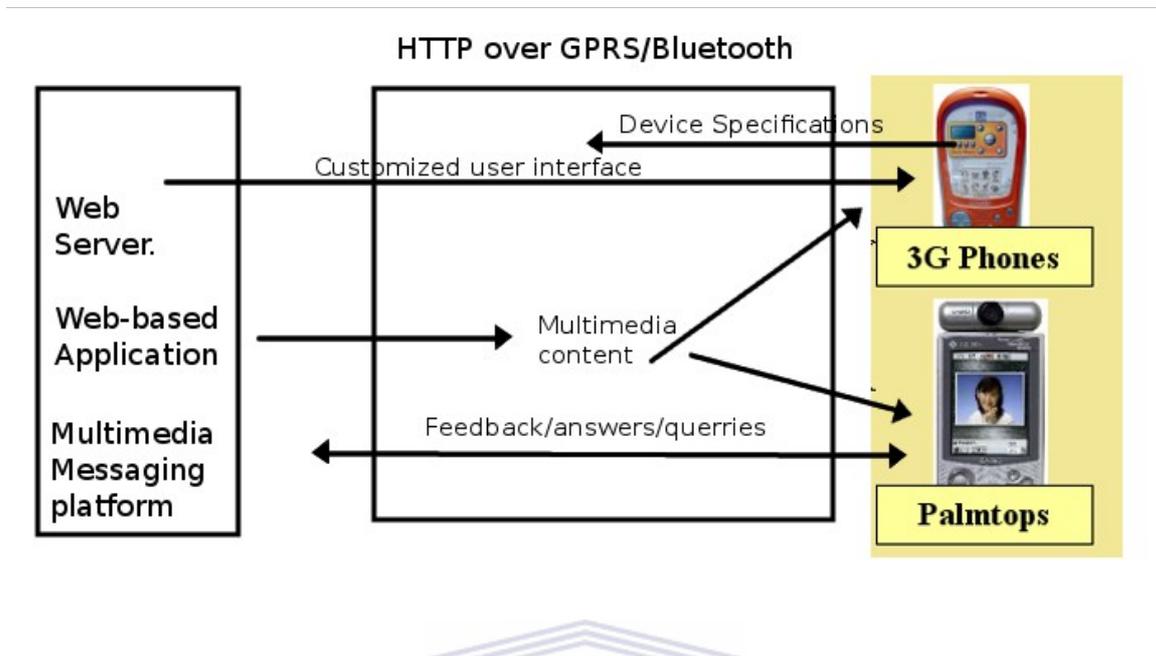


Figure 2.7: AD-HOC m-Learning system model.

Using blue tooth as the transmission protocol, a hand-held terminal connects to a hotspot by providing the service provider with the device's capabilities. The provider then sends back a customized user interface suitable for the device. The user can then access the education content, enhanced by a two-way communication multimedia messaging platform available in the system. Users' feedback are accessed via a back-end web-based application running on a web-server.

WESTERN CAPE

2.3.1 Textual presentation

One of the methods used for formatting textual data destined for mobile presentation is by using WML. WML is an XML-based markup language that implements the WAP specification and is intended for use on devices such as mobile phones. We refer to WML here based on the WAP WML specification 1, published at <http://www.openmobilealliance.org/tech/affiliates/wap/wap-191-wml-20000219-a.pdf>. WML was designed with knowledge that it is meant to be used on portable devices with resource constraints. These devices include mobile phones. The constraints include:

- Small display screen and resolutions: a small device like a mobile phone may have few lines of textual display, with each line containing 8 -12 characters.
- Limited user input facilities: a mobile phone typical has a tiny keypad with a few additional function specific keys. Sophisticated smart phones may have pointing devices, programmable buttons and virtual or miniature keyboards.

- Narrowband network connection: most mobile users in developing countries use 2G networks with only support up to 14 kilobits per second bandwidth.
- Limited memory and computational resources: mobile phones generally have less powerful CPUs and smaller memory size compared to desktop computers. Mobile phones also run on limited power supplies.

Applications created in WML are able to cope with these limitations when running on a mobile phone. The structure of WML includes four major functional areas:

- Text presentation and layout: WML supports text and images and provides a variety of formatting and layout commands. Formatting includes cases like specifying boldfaced text.
- Deck/cards organization: information formatted in WML is organized as cards and decks. Cards specify units of interaction, for example screen of text, text entry or choice menu. Cards are grouped together to form decks. From a user's point of view, a user navigates through a WAP-based application by shifting through a series of WML cards. A WML deck can be said to be similar to an HTML page.
- Inter-card navigation and linking: WML supports navigation between cards and decks. It also includes provisions for event handling in the device. Event handling can be used for navigation purposes or for script execution. WML also supports the use of anchors, similar to what is available in HTML
- String parameterization and state management: WML decks support parameterization using a state model. This means variables can be used in place of strings and substituted at runtime. Parameterization helps in the efficient use of network resources.

WML supports the use of URLs. Because of this, it interoperates with standard transport protocols like HTTP. The URL format follows the specifications of HTML. In fact, WML adopts the HTML standard of naming locations within a resource. The WML 1.3 specification includes the support of following URL schemes:

1. WML browsers must implement the URLs that are specified in Wireless Application Environment (WAE).
2. WML supports the use anchors in a document. A WML fragment is specified by the document URL, followed by a hash mark (#), followed by a fragment identifier.

Since WML is an XML language, it inherits the XML document character set. A character set refers

to a set of all logical characters that a document type may contain. Both WML and XML use full Unicode encoding. The most common set of Unicode encoding used is UTF-8. Documents not encoded using UTF-8 or UTF-16 must declare their encoding as specified in the XML specification. WML syntax is based on XML. All XML syntactical specification applies to WML. WML supports different data types. The main data types supported are text, links, specified by HREF tag, boolean data types, numbers and media. Media type is specified by using the *ContentType* attribute. WML can therefore be used to create applications that run on the mobile phones. Applications created in WML can run on low resources and as such makes WML ideal for presentation of mobile content.

2.4 Customizing images for mobile display.

Section 2.2.4 discussed 3G network, and explained how such networks support higher data transfer rates. This implies that it is possible to transmit images on such networks in a fairly reasonable time. Although this is good news for software developers of multimedia applications on mobile phones, there is still one glaring challenge: the screen size of a mobile phone. We don't expect the sizes of these screens to increase much largely due to the mobility requirements of mobile phone. It therefore makes sense to modify an image targeted for mobile phone display. Typically, this is achieved by employing image manipulation techniques. The next section discusses some principles used in image manipulation in line with the scope of this thesis..

2.4.1 Image manipulation

Many studies have been carried out involving image manipulation. One of the most common image manipulation techniques is *image transformations*, that can be broadly defined as operations that can be applied to source graphics in order to obtain target graphics that meet the requirements of the target display screen [6]. Such transformations can range from simple operations to very complex ones. If the transformation is targeted for a tiny mobile phone screen, it typically involves scaling down the graphic and reducing the number of colors used. This in turn reduces the size of the graphic but might alter the quality. A transformation might also take into account the dimensions of the original graphic before starting the process. There are two main types of such transformations. The first one is *uninformed transformation*, sometimes referred to as a blind transformation. In this case, very little or no information from the source graphic is taken into account when the transformation is performed. Although it might be fast, this technique might not always give results that meet target environments [6]. The second one, *informed transformation*, involves the analysis of source graphics before the actual transformation takes place. During the analysis phase, informed transformation carries out an analysis of the source taking into account

both syntactic and semantic features. The work done by Rist and Brandmeier implemented semantic classifiers that distinguish between portrait and non portrait images, and outdoor versus indoor images [6]. There is always a possibility of creating an image class that performs a specific transformation such that it produces acceptable results for general problems associated with instances of that class.

Rist and Brandmeier explain that it is a difficult task to make accurate an mapping between available image transformations and recognized features of an image, and appropriate parameter adjustments [6]. They propose a solution that involves the use of machine learning techniques. In machine learning training phase, a graphics design expert manually assigns the transformations to source images and this allows the system to recognize and generalize correspondences between image features and transformation parameters. Then after the training phase, the system is asked to select the appropriate transformations to use in a specific scenario. The system selects and gives the user the recommendations of the best transformations to use. Rist and Brandmeier note that there are situations where the system may be undecided on the recommendations, a time which manual intervention would be appropriate [6].

Another approach of handling images is by using *re-generation of graphics*. In re-generation, the source image is not modified. Rather, a thorough semantic analysis is done on the source image and the information obtained is used to generate a new image. The semantic representation of the image content can be extracted, then used to generate a variety of new graphical representations that meet the target device. The next sections examines some algorithms used in image manipulation techniques.

2.4.2 Interpolation

Unser and Blu loosely define interpolation as a “model-based recovery of continuous data from discrete data within a known range of abscissa”, a definition that fairly fits the work done in the thesis [30]. Unser and Blu. present hypothesis for interpolation:

- 1) the data provided is defined continuously
- 2) using the underlying continuous function at any abscissa, it is possible to compute its data value, provided there are data samples
- 3) when applying the underlying continuous function at the sampling points, one should arrive at the same results at the data themselves

Interpolation is widely used in the medical field where often the goal is to modify the sampling

rates of pixels in an operation referred to as rescaling [30]. According to Unser and Blu, rescaling is used often when using image acquisition devices like a scanner, in which case often a scanner is equipped with non-homogeneous resolution. Rescaling is often used to change the aspect ratio of the pixels in such a way that the final product can easily be manipulated (visualized, rotated). Another operation, referred to as reslicing, is an interpolation technique used to present the volume in a set of images oriented perpendicular to slices [30]. These special orientations are usually referred to as axial, coronal and sagittal, and at most require rescaling for proper display. Reslicing is relevant when an image acquired by a scanner has a volume with a higher within-slice than across-slice resolution. Interpolation is also used in advanced visualizations, such as volume rendering. In such cases, often texture is applied to the facets that compose the rendered object. In biomedical rendering, texture is displayed based on a map consisting of true data samples. In such cases, there are geometric operations involved, for example perspective projection. As such, it is always necessary to resample the map, an operation that involves interpolation. Volumetric rendering also requires computation of gradients, an operation best handled by including interpolation.

The more simple and extensive use of interpolation is found in cases where actual image manipulation is required, as opposed to volumes. Using a medical example, a physician might be interested in inspecting an image at a coarse scale as well as fine scale. To satisfy this need, interpolation techniques like zooming-in and out are essential [31]. Other operations that could come in handy in such cases include sub-pixel translation, also known as panning, or rotation [31]. Other interpolation-based operations include polar-to-Cartesian scan conversion function used to transform acquired polar-coordinate data vectors from ultrasound transducer into the Cartesian raster image appropriate for most display monitors [31]. The next sections present related work done by other researchers covering manipulation and customization of images in order to display them on tiny phone screens.

2.4.3 Image adaptation

Image adaptation has been studied by many researchers. Chen et al. briefly discuss such adaptations in their related work section [31]. They highlight a proxy-based architecture used to perform an on-demand data-type-specific content adaptation on images. The adaptation includes image distillation, which actually involves compressing the image, a technique that reduces the size of an image hence the transmission time, file size reduction, color reduction and format conversion [31]. Chen et al. also highlight another study in which an image-transcoding system based on the content and classification of image purpose and type is presented. In the study, the images are first

classified into image-type and image-purpose through analysis of the image characteristics, the related text and the context in which the image is available, namely the website serving the image. Then using the analysis results, the image is modified by the trans-coding system along the dimensions of spatial size, fidelity, and color and in some cases substituting the images with text. Finally, Chen et al. highlight a more recent new approach of using region of interest-based adaptation, in which case an image is manipulated at each region of interest separately, instead of manipulating the whole image. This allowed the delivery of important region to the target display device, especially when the device is small. This work takes user perception into consideration. Chen et al. themselves present a solution for adapting image content based on user attention to fit different display screen sizes. Chen et al.'s study involves taking into consideration most contextual constraints for adaptation, but considers screen size more critically. Their approach involves scaling, compressing, cropping images as well as helping locate conceptually important regions when user is browsing regions.

In another study, Jiang et al. present a method of automatically transforming static images to dynamic video clips that fits the requirements of a mobile devices. They employ techniques that include face detection and attention detection, and extract important regions in an image where users might be interested in [32]. Then they use a procedure of video resolution adaptation that conforms to different screen size and aspect ratio and use an algorithm to simulate camera motions to introduce motion effects. In the end, they encode a video clip consisting of the images for playing on mobile platforms. Jian et al.'s solution enables users to view photographs dynamically using mobile devices by glancing over the whole picture as well as being able to see the picture details.

2.5 Summary

This chapter has discussed some existing technologies and applications that have been developed to implement m-Learning. The discussion ranged from SMS based applications, podcasting, graphics to introducing technologies like GPRS, WAP, 3G and Bluetooth. The chapter ended by looking at two main techniques used for formatting text and graphics that are destined for mobile phones. This has given us a broader understanding of some considerations one has to keep in mind when implementing m-Learning solutions. In the next chapter, we revisit some of the applications discussed here, and look at the challenges facing them. It is on this principle that we state clearly the purpose of our research, and how we intend to address some of the problems.

3 Methodology

The first sections of this Chapter cover the main challenges facing some of the existing m-Learning applications that use text and podcasts. Sections that follow discuss the research design used in the thesis to investigate how to address some of the challenges. Section 3.1 revisits the applications and technologies discussed in Chapter 2, with emphasis on the challenges they face. Based on these challenges, the primary research question is raised in Section 3.2, together with secondary research questions. Section 3.3 covers the steps we followed in designing our research as a way of answering the research question. It explains the *experimental design*, the research design used in the study. The methods used to collect data in during the experiments are also explained in this section. We end with a summary in Section 3.4. The description of the software used in the experiments is discussed in Chapter 4.

3.1 Challenges of existing m-Learning systems

Existing applications and technologies used for m-Learning discussed in Chapter 2 have challenges associated with the way they present and transport e-Learning content to a mobile phone. We look at the specific challenges in the next sections. Section 3.1.1 covers text-based applications, Section 3.1.2 covers podcasting, and Section 3.1.3 looks at Rich Internet Applications.

3.1.1 Text-based applications

As introduced in Chapter 2, the most widely used text based technology in m-Learning is SMS. This is because SMS is one of the basic services that is available in every phone, and it is provided by every cellular company. m-Learning applications using SMS are however limited only to a number of features. One of the most obvious shortcoming of using SMS in m-Learning is the fact that SMS supports short text messages of only up to 160 characters. Wen et al. presented an m-Learning application that uses a query-answer model that can be used by students to query a knowledge base [8]. A student initiates the process by sending an SMS to the application's query processing engine. The query processing engine parses the query, and if valid, searches and matches information from knowledge base or Internet to produce suitable results which are returned to the user as SMS replies. Examining this model, we notice challenges. Starting with the student query, a student can submit a query in an unpredictable way, possibly using natural language. This query is interpreted by the *system query processing engine*, as there is no human-to-human interaction. As such, the *system* needs to employ a fairly advanced query analyzing algorithm to be able to satisfactorily process and send relevant answers back to the student. The obvious

shortcoming is communication within the application as is limited to text-based content only. A student wishing to learn more complex educational content that requires visualization for better understanding might not benefit from this application. It must also be noted that there is no direct realtime interaction between the student and the instructor; an instructor wishing to disseminate information to the students has to upload the content into the knowledge base first, in text format, for later querying.

Balasundaram and Ramadoss presented an SMS based application for m-Learning [9]. Almost like the system published by Wen et al., the application employs a query-answer model in which the lecturer sends questions/tests to the students using SMS and they reply via the same mode. The system is responsible for processing the answers. Exhibiting similar characteristics like in Wen et al., the system has to employ a fairly complex system for analyzing the student responses, as most of the time the student responses are unpredictable due to use of natural language. The communication is also limited to text-based content, implying more diverse content containing graphics, audio and video cannot be used. There is no evidence of realtime interaction between learners and instructors too.

Tanguampien explains how MXit was used by school children in preparing for their examination [10]. A 'bot' was used to send back information when a user sent a blank message to the bot. The educational information sent back is randomly generated. The explanation behind introducing the bot was that sometimes users experienced long delays between sending a message and getting replies for another user in a chat room, and the idle time could be used to 'ask' the bot for tips. Because the user does not have control over what information should be send back, interaction is the main challenge facing this approach. Because MXit does not use SMS as the mode of transporting content, it does not suffer from the message length or cost limitations of SMS.

Wen et al and Balasundaram and Ramadoss inevitably gave us examples that show how m-Learning applications using SMS for presenting and transporting content are limited to text-only content. It is also clear that such systems are limited in terms of how much information can be send at any given time as SMS supports only up to 160 characters. The other main challenge facing the SMS-based m-Learning applications is they are only best used in query-answer models or models which can only support short texts. One of the aims of this thesis was to present a proof-of-concept application that allows an m-Learning application to be used in more diverse domains and as well as support more diverse content than just text.

3.1.2 Podcasting

Podcasts are the most common audio-based method of disseminating e-Learning content. As introduced in Chapter 2, the podcasts are published on a web server and are automatically downloaded by a subscriber's client software running on a portable device, like a mobile phone. This is accomplished by using the RSS technology, which employs a publish-subscribe model.

At the University of Ulster, an m-Learning project was established to supplement delivery of e-Learning material using mobile phones [11]. Using the RSS architecture, live lectures were recorded and posted onto a website. Students' feed-enabled mobile phones then automatically downloaded this content. Like many podcast-based applications, the project was faced with a challenge of delivering highly technical courses. Audio can not satisfactorily describe such content. Another major challenge was the length of the lectures. Long recordings resulted in large files which could not be downloaded onto most phones easily without incurring the cost of the downloads. The devices are also have limited space. Recordings could be broken up into small many files, but this can again disrupted the flow of the lectures. There was no direct interaction between the instructor and the learner in this model.

Hall also carried out a study to establish the effect of podcasts on long distance students [33]. Recorded lectures were put on the university's website and the students were allowed to download them onto their mobile devices. The study revealed that a number of participants were not satisfied with podcasting as means of delivering learning material, and favored an environment in which they could have direct interaction and collaboration with the instructor and/or other sources of information in near realtime [33].

In current educational environments, podcasts can play a good role a means of disseminating educational content. Yet as illustrated by these few examples, audio alone cannot satisfactorily describe the content, especially graphical material. Another major challenge facing audio-podcasts the length of recorded audio. Most commonly used mobile phones are limited in storage capacity. This means recording and downloading lengthy lectures can be both expensive and time-consuming for such mobile users. Although a quick work-around would be to split long files into smaller short clips, this would disrupt the flow of the lecture, and its does not make the *whole* recording any shorter. Also as noted, most users preferred an environment where they could engage the sources of information in a near realtime collaboration and interaction modes.

3.1.3 Rich internet applications for m-Learning

Section 2.1.3 introduced Rich Internet Applications, or RIAs, for m-Learning. Two main technologies were discussed: Java/JavaFx and Flash Lite. The section also discussed applications that are using these technologies. Java is the dominant technology with respect to its installation and developers base. Java (<http://www.java.com>) was introduced to developers with a 'Write Once Run Anywhere' slogan. While this has been true to most desktop applications written in Java, it does not hold for mobile phones. m-Learning applications are not exempt from this fact; screen size, color depth, keyboard values and memory vary from one mobile device to another. As a result, developers have to provide slightly different versions of their applications based on different choice of configuration profiles.

Flash Lite is suitable for creating animations, games, front-end users and device-specific content. Flash Lite does not offer developers full-fledged API for developing rich applications, compared to Java ME platform [34] Flash Lite suffers from poor graphics performance, partially due to the demands for processing vector graphics. As a result, m-Learning applications developed using Flash Lite will inherit this characteristic. This holds true for the SATO tool developed at the Institute for Medical Informatics, Statistics, and Documentation, Medical University of Graz, Austria [14].

Although we highlight the main shortcoming of Java-based RIAs as the inability to create truly mobile-device independent application, the application developed in our study is not entirely immune to this problem as well. However, we employed recent advances in Java technology that greatly improve on the platform independence. We however steered clear of the Flash Lite specifically due to lack of full-fledged API for developing rich applications, compared to Java ME [34]. Having looked at the challenges affecting SMS-based m-Learning application and Podcasts, we summarize the challenges in the next section.

3.1.4 Summary of challenges.

We summarize the challenges of existing m-Learning application as identified within the scope of our research:

1. **Format of content:** The applications that use SMS and podcasting do not support use rich content: formatted text and complex graphics. Both and text (SMS and chat text) and podcasts are thus not suitable for *visually* describing highly technical content.
2. **Realtime interactivity:** The applications discussed do not support any realtime interaction between instructor and learners.

In the next section we present the main research question. This follows with other two secondary research questions derived from the primary question.

3.2 Research questions

We present our research question within the scope of the main challenges of m-Learning applications summarized in Section 3.1.4:

“How do we provide presentations and interactive chat to support m-Learning?”

The research question helped us study how to prepare and transport (stream) presentations to a mobile phone. We identified presentations as an excellent medium for transporting rich content to a mobile phone. In order to display the presentations correctly on the mobile phone, we had to format them on the desktop computer first. We accomplished this by first exploring the techniques used for converting presentations into a format that is easy to manipulate without greatly compromising the quality. We then investigated how to scale down such presentations and transport them to mobile device for display. We developed a proof-of-concept application that formats presentations created on desktop computers and transports and displays them on a mobile phone. We used the software to carry out a series of experiment. The experiments are explained in Section 3.3. All the experiments were conducted in a laboratory. We developed secondary research questions to concentrate on specific sub-domains of the research. Our first secondary research question covers the actual process of scaling down the slides for mobile display:

“How do we scale down presentations, thus reducing the file size, for mobile device display”.

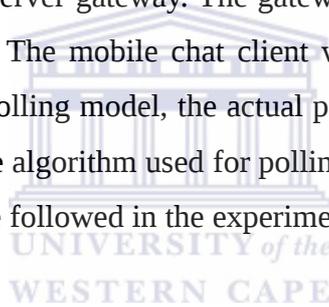
We answered this question by converting slides in a presentation into a series of images, which were then scaled down to a format and size appropriate for mobile display. The next secondary research question is about the quality of the slides a mobile phone:

“How do we measure the quality of presentation slides when displayed on mobile phone?”.

Because an average mobile phone has inherent limitations, the main obvious one being the tiny screen size, it is important to investigate the quality of slides presented on such screens. To answer this research question, we picked each of the converted images and extracted pixel information which was used to generate image histograms. We used image histograms to measure the quality of slides before transporting them for mobile phone display. Our third research question covered the air-time costs:

“How do we minimize airtime costs when sending presentations and conducting interactive chat on a mobile phone?”

This question was answered by implementing an algorithm that allows the mobile client to poll the data from the server only when there has been an update from the instructor's screen. We then recorded the amount of data that is received at the mobile client and used this to calculate the air-time cost implication. The polling process takes place in three separate threads: the first thread checks the status of slides from the instructor and updates the mobile client accordingly. The second thread updates the user list. The third and most important thread with respect to the above research question was the chat polling. This thread refreshes the chat transcript as long as there are new messages. The interactive chat was implemented as a typical instant messaging client, connected to a presence and messaging server. We used XMPP as one of the protocols to achieve this, however, this was only limited to the desktop clients. Mobile-based clients used a combination of HTTP and XMPP. HTTP was used to poll the server gateway. The gateway used XMPP to communicate with the messaging and presence server. The mobile chat client was limited to simple text messages. Although we used a fixed-interval polling model, the actual polling process does not introduce any significant air time cost element. The algorithm used for polling is presented in Chapter 4. . The rest of this chapter discusses the steps we followed in the experiments.



3.3 Experimental design

Laboratory experiments were used in this research because the author aimed at developing a proof-of-concept software application to demonstrate that using presentations in mLearning enabled use of rich content. The cost of carrying out such experiments were relatively low, without running the risk of other factors like reliability, safety and security. This section is presented as follows: Section 3.3.1 introduces the software that was the center of the experiments. Section 3.3.2 highlights the experimental environment. Section 3.3.3 explains the experiments we carried out to enable us scale down, format, determine the quality of the slides and transport presentations to a mobile phone. Section 3.3.4 highlights how we calculated airtime cost.

3.3.1 Software to help answer research questions

We developed software which was used as a proof-of-concept for forming and transporting presentations to a mobile phone. Specifically, the software:

1. Enables an instructor, using a desktop computer, to exchange rich educational content with

learners who are using mobile phones. The information is exchanged as presentations, either as Microsoft's Powerpoint or Oracle's Openoffice Impress.

2. Provides a near-realtime interaction between instructors and learners during a learning session, using instant messaging text chat.

3.3.2 Experimental environment

Our experimental environment was based in a laboratory. It consisted of a SunFire server (www.oracle.com) running Ubuntu 8.10 (www.ubuntu.com) that was connected to the Internet. On the server we installed Openfire 3.6.4 (www.igniterealtime.org) as our XMPP server. We implemented our server component as plug-in to the Openfire. We used Nokia N82 mobile phone connected to Vodacom South Africa (www.vodacom.com) for our mobile phone. Nokia N82 is J2ME enabled. This setup enabled us to emulate typical conditions instructors and learners would face in a South African context. We used Mysql 5.1 (www.mysql.com) for database functionality, also installed on the server.

3.3.3 Formatting and transporting m-Learning content

In these experiments, we were interested in how to format and transport rich e-Learning content targeted for mobile phone presentation. We started our experiments by downloading freely available existing presentations from a repository (<http://presentations.wits.ac.za>) managed by the library at the University of Witwatersrand, Johannesburg. The repository allows users to upload presentations either as Microsoft's Powerpoint or Oracle's Openoffice Impress and share it others via the Internet. Uploaded presentations are freely available to anyone. We downloaded 21 random presentations that ranged in sizes, from 17.7 KB to 28.4 MB. In order to simulate the steps an instructor would take when setting up a classroom, we uploaded the presentations using the desktop client of our experimental software. A presentation uploaded from a desktop client is transported to a server, for further processing. The processing is explained in the next paragraphs.

We searched for a suitable method to implement a way to transport content from the instructor's computer to the server. Our analysis of the typical virtual classroom sessions revealed it was best to use functionality that provides near-realtime communication. This information was obtained by reading existing literature on computer network communication models. We settled on use of XMPP, a protocol that is used for providing messaging and presence in realtime. This protocol was implemented between the desktop client and the server within the experimental software. During the course of the experiments, we found out, however, that we could not use XMPP to implement

the communication between the mobile client and the server, at least during the time we were conducting experiments. The main reason was lack of appropriate mobile-based XMPP library with small enough foot print to meet our requirements.

One of requirements for successful deployment of the m-Learning application was a small footprint for the mobile client, because of the memory constraints of the device mentioned in earlier chapters. Initially in our experiments, we ran the mobile client on a mobile emulator, which enabled us to debug the application quickly. After the software became stable, we started experimenting with an actual phone. The mobile client was uploaded on our University's website, from which we downloaded to a desktop computer before transferring it to the mobile phone via Bluetooth. Embedding an XMPP library into the mobile client would have considerably increased the footprint size, thus we discarded this idea. We then decided to use J2ME's built-in HTTP libraries, which meant we keep the footprint of our mobile application minimal. As a result, communication between mobile client and the server is HTTP-based.

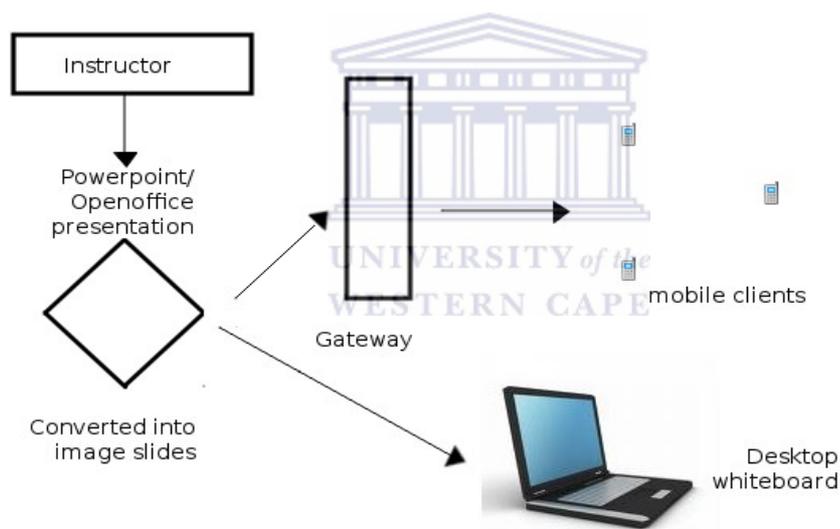


Figure 3.1: How presentations are converted to images for presentation.

PowerPoint/OpenOffice presentations are uploaded to the server by the instructor using the desktop client. The server converts the presentation into a series of images for presentation.

The core part of these experiments involved processing the presentations on the server by converting them into a format that would easily be transported to a mobile phone. Figure 3.1 illustrates how slides in a presentation are converted into a series of images on the server, before being transported to mobile clients. We employed transaction-logging techniques in addition to

observations to record our findings. We used JODConverter (<http://code.google.com/p/jodconverter/>), a Java library that converts office documents into different formats, using OpenOffice.org (<http://www.openoffice.org>). JODConverter supports many formats, and we were interested in converting OpenOffice/PowerPoint documents into HTML. Using JODConvert to convert office presentations into HTML involves taking 'snapshots' of slides in the presentations, and writing the snapshots as images to a disk file. This was an ideal process for us, because it would be easier to manipulate the generated images and transport them for mobile display. Note that we were not interested in the time it takes to convert a presentation into a series of images, because this process takes place before a learning session starts, and thus the *duration* it takes to convert a document into image slides does not directly impact the performance of the m-Learning application. We thus concentrated on the generated images, after the conversion.

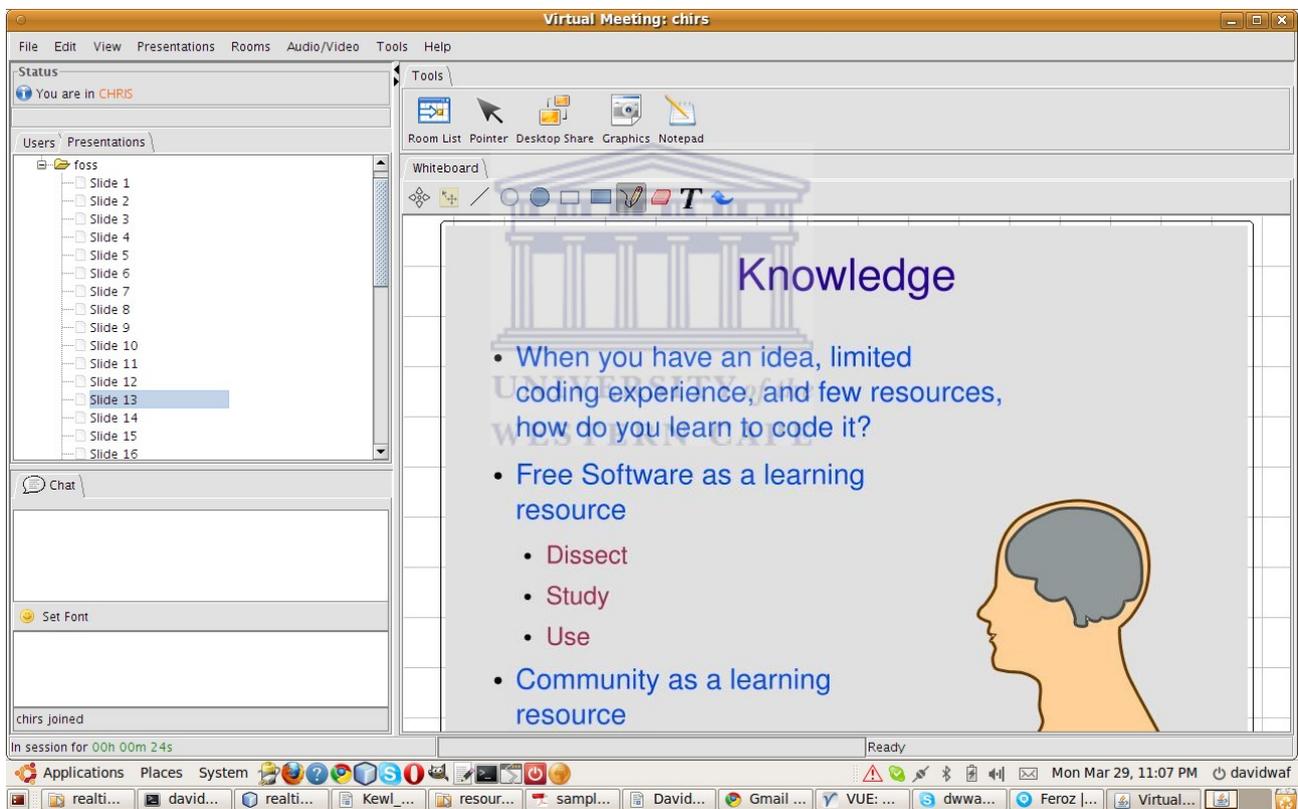


Figure 3.2: Desktop viewer for preview converted slides as images

A preview of a slide from a presentation that has been uploaded and converted on the server. This preview is actually an image representing a 'snapshot' of a slide.

The first step was to analyze the converted images, to determine the accuracy of the conversion. The easiest way to do this was to use a viewer we had developed, (see Figure 3.2) which enabled an instructor to preview results of the converted presentation. We used the viewer to manually spot any irregularities that may have occurred during the conversion through observation. This was an

important step because the human visual system is the most complex and efficient method of analyzing image *quality*. We however complemented this process by using automated image analysis techniques as well. We chose image histograms for this process. Histograms are frequency distributions, and an image histogram describes the frequency of the intensity values that occur in an image. We used histograms to plot the total number of pixels at each grayscale level in an image and thus determine overall intensity in an image; a histogram skewed to the left implies a darker image; to the right a bright image. We logged down the values for each of the slides in the 21 presentations. We then obtained a mean value for each presentation. Then, we ran each of the presentations from a desktop computer and viewed each of them from a mobile phone, and rated them. The score value was allocated by looking at how the presentation appears on the mobile screen, whether it rendered 'brightly' or 'darkly'. This enabled us to establish any relationship between a mean image value and how it renders on a mobile screen. A detailed discussion on the analysis of this is covered in Chapter 5. We used ImageJ 1.4.2, an image processing software written in Java, developed by Wayne Rasband at National Institutes for Health (<http://rsbweb.nih.gov/ij/>) to implement histogram analysis. ImageJ is open source and runs on virtually all platforms, and has a very well developed plugin interface, thus it allowed us to easily plug it into our code.

The motivation behind using histograms was because we were interested in quantifying the quality of the generated images. By *quality*, we refer to how *bright* an image is, or contrast, which by implication means a brighter image should be more visible on a tiny mobile screen. There are many methods for analyzing image quality; signal-to-noise ratio, resolution, detective quantum efficiency, just to name the basics. Each of these methods have their own shortcomings. We rejected the idea of using any of these methods on the account that we did not really need any complex advanced image analysis; most of slides created in a presentation are already high quality, even before conversion. We needed a basic way of checking the brightness of the images; to quantitatively augment the human visual system. A *good* quality image implies that the content residing on the slide has a higher probability of being successfully presented on a mobile phone. The success of this process also implied that we were one step closer to addressing one of the shortcomings identified other m-Learning applications discussed at the beginning of the chapter: ability to use rich content. The image histogram for each image was logged down, using Log4J (<http://logging.apache.org/log4j/1.2/>) . According to Log4J website:

“On an AMD Duron clocked at 800Mhz running JDK 1.3.1, it costs about 5 nanoseconds to determine if a logging statement should be logged or not. Actual logging is also quite fast, ranging

from 21 microseconds using the SimpleLayout, 37 microseconds using the TTCCLayout”.

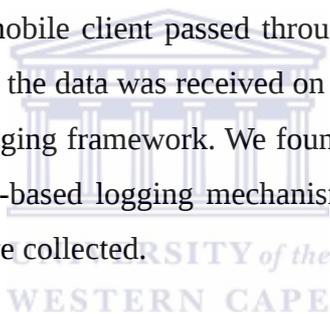
This made Log4J an excellent tool for logging, because of the negligible logging overhead. We ran a series of experiments, deliberately using 'dark' slides, noting down their mean image value, then observing how they render on a mobile phone. We then repeated the same process with 'bright' images, and noted how they render on a mobile screen. We present and discuss the results in Chapter 5. After observing the quality of the generated images, we started to process them. The best way we could process the generated images for mobile phone display was by down-scaling them. When down-scaling an image, we were interested in the overhead we pay when doing this process, since this process takes place in realtime when the mobile client is connected to the server. Using Log4J Java library, we recorded down our findings. Every time the mobile phone client connected to the server, it provided the screen resolution details of the mobile. This value was then used to appropriately scale each of the images. We made use of image interpolation techniques to scale down images. The simplest way to explain the interpolation concept in this context is: how do you calculate the value to assign a new pixel when an image is scaled or transformed. One of the quickest ways to implement interpolation is to create new pixels with values based on nearest existing pixels in the source image. With this approach, if there is no nearest pixel from source image, the pixel is assigned value of zero. This is referred to as nearest neighbor interpolation. Nearest interpolation is normally very quick, but it tends to produce choppy results in the destination image. Another method is to calculate value of the new pixel based on the linear average of surrounding pixels. The process is referred to as linear interpolation. It is more time consuming, but it produces smoother images. One often chooses either of the methods, depending on the needs. In our case, use of linear interpolation was more appropriate since we were interested in producing smooth images for mobile display. We employed the Java's built-in image manipulation algorithms to accomplish this process.

Log4J allows developers to programmatically configure the logging source. We used this feature to defined a custom file for logging the times it takes to scale an image. The log values were recorded in milliseconds. In order to get consistent results, we repeatedly carried out of the 21 presentations we had downloaded.

3.3.4 Air time cost

The calculations on air time cost were performed based on the amount of data transported through the network to the phone during a learning session. To obtain the data transferred, we conducted experiments in three phases. During phase 1 we recorded the amount of data transported when using

presentations. We measured this for 7 randomly selected presentations out of the 21 uploaded. We present the results in Chapter 5. Phase 2 involved recording the amount of data transmitted during the polling of users in the virtual room. We wanted to establish if the polling process introduced any significant increases on the amount of data transferred. We simulated with 7 users in a virtual room. The third phase involved simulating the 7 users with randomly generated chat text to emulate students chatting in a virtual room. We recorded down the amount of data transferred during the chat session. In order to obtain a wider range of results, we scheduled each of the experiments into a series of test Ids, and as such we present the data in Chapter 5 using the test ids. We used Nokia N82 for the mobile phone. Our air time cost calculations were based on the data rates published on Vodacom's website as of May 2010. We made use of Java's logging mechanisms, specifically tailored for mobile phones, to log the measurements. Microlog, an open-source logging library, is based on the well known Log4j API, but created from ground up with Java ME limitations in mind. We made use of this library, downloaded from <http://microlog.microsuite.org/>, by embedding it into the classpath of our mobile client application. We also implemented logs on the server. Any data transported from the server to the mobile client passed through the logging framework, to record down the amount in kilobytes. When the data was received on the mobile phone, we recorded down the packet size, using the mobile logging framework. We found out the values to be similar, and as a result, we chose to use the server-based logging mechanism to avoid taxing the mobile client. Chapter 5 presents the information we collected.



3.3.5 Presentation of m-Learning content

The mobile user interface was developed using LWUIT, which gave us the ability to deliver almost desktop-like user interface that were consistent and easy to map to mobile phone keys. Our experiments consisted of determining possible features that we could implement in our software to enable user interaction. We were able to accomplish this through introduction of features that enables a mobile user to:

1. Raise hand
2. Chat using text
3. Marking a point on a slide
4. Moving virtual pointer

The ability to perform any of these functions demonstrated possible interactivity characteristic of an m-Learning application.

3.4 Summary

This chapter discussed the challenges facing some of the existing text and podcast-based m-Learning applications and the need to address these challenges. From these challenges, we learned that most m-Learning applications that use text and podcasts do not support use of rich content. Text based applications do not support a wide range of formats for content, other than simple plain text. Podcasts are not sufficient for describing highly technical content. We therefore discussed how we used an application we developed to enable transport rich content by using presentations. We described how we used existing tools to format the presentation into a form suitable for mobile phone display. We took advantage of the higher data transfer rates offered by 3G networks to transport formatted presentations, as scaled-down series of images, to a mobile phone. We used J2ME as the platform for developing the mobile phone client because of the many advantages it offered over another candidate platform, Flash Lite 3. J2ME allowed us to use LWUIT to develop almost desktop-like graphical user interfaces for the mobile client. We presented the observation and transaction-logging methods used to collect data for analysis for Chapter 5. The results of the experiments are presented in Chapter 5 and discussed. Chapter 4 extensively discusses the software we developed.





UNIVERSITY *of the*
WESTERN CAPE

4 System design

System design entails a process of breaking down a large envisioned system into small components and describing how each of these components (should) interact with each other. Designing a system thus should include the correct way of decomposing functionality into smaller bits such that these pieces can conveniently interact with each other to make a complete working system. The end product of the design activity is existence of a document that describes the design – the design document. The software developed in the study enables instructors to share presentations with learners who are using mobile phones. It consists of three main components: mobile client, desktop client and server. These components interact to create a logical virtual room into which users join in order to view the presentation and interact via text chat. The *mobile client* consists of a graphical interface that consists of a presentation viewer, for displaying the slides, and the chat-room used for text chatting. The presentations viewer includes a set of basic navigational controls for browsing through slides and a 'laser' pointer for marking various parts on a presentation. The *desktop client* component is more complex and its system design discussion is out of scope of the research, except for the interactive whiteboard, which includes an embedded presentations viewer. The desktop client enables an instructor to create a virtual room and upload presentations for the class. The *server* is implemented as two plugins within a Servlet container. They are discussed in the chapter. This chapter discusses the structure of the three main components. Section 4.1 covers the user requirements, Section 4.2 explains the target audience, Section 4.3 gives a detailed structural design of the system, Section 4.4 explains how data persistence was implemented using a database management system. Section 4.5 briefly covers the open source libraries used and Section 4.6 gives a summary of the communication flow followed by conclusion in Section 4.7.

4.1 User requirements

The software in the study is designed to fulfill two user requirements in an m-Learning environment that are derived from the issues raised in Section 3.3.1. First, it allows instructors and learners to exchange rich learning material in textual, graphical format using Oracle's Openoffice Impress and Microsoft's Powepoint presentations. Secondly, it allows learners and instructors to interact via text chat near-realtime mode. In the next section, we briefly discuss the target audience.

4.2 Target audience

The mobile client is targeted at learners who would find it more convenient to 'follow' a presentation from their mobile phones under circumstances that would not allow them to attend

the presentation in person. The desktop client consists of a set of applications that are usable in other non-academic environments: remote demonstrations, live on-line presentations, remote demonstrations and remote desktop assistance.

4.3 System structure

As introduced earlier, the software consists of three main components: *mobile client*, *desktop client* and *server*. An instructor wishing to stream a presentation to learners must first prepare a virtual room, then invite participants into the room. Such a session is initiated via the desktop client, with an instructor, and participants can join via either desktop or mobile clients. Users are authenticated, validated, coordinated and managed through the server. The server is also responsible for interfacing the mobile client and desktop clients, The server additionally consists of the database for storing users' information and session data. All sessions are recorded on the server side. Figure 4.1 below show this high-level relationships between these components

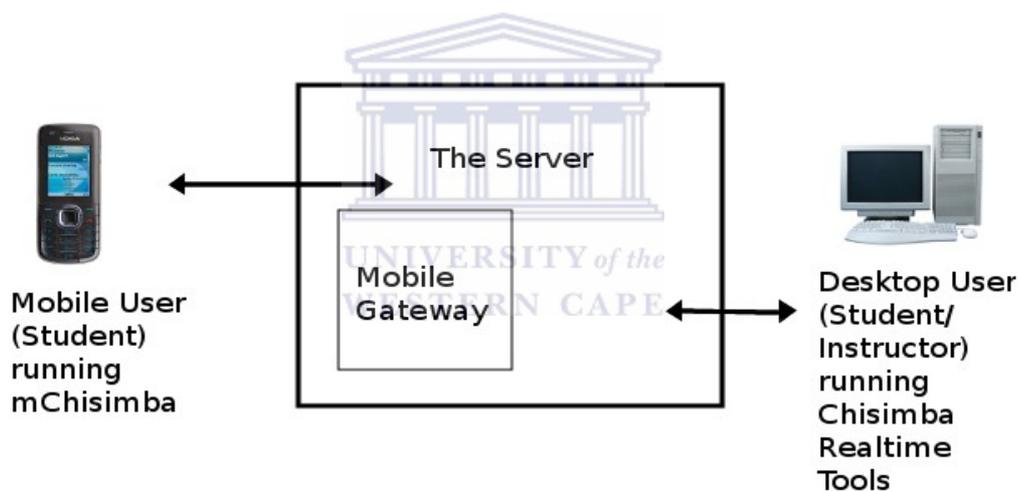


Figure 4.1: High level overview of mChisimba framework.

A student runs mChisimba (the mobile client) from a mobile phone, which then connects to a desktop computer via the server component. The server has gateway used for trans-coding packets between mobile phone and server. The session is normally initiated and controlled by an instructor from the desktop client.

The rest of this section discusses the design of the mobile client - mChisimba - and the external components it interacts with. Section 4.3.1 justifies the choice of development tools. Section 4.3.2 discusses the mobile client, and Section 4.3.3 discuss the server components.

4.3.1 Choice of development platform.

We decided to use J2ME as the platform for implementing the mobile client, after considering

technological challenges of Flash Lite, another possible candidate [34]. We based our decision on the results of the comparison between these two technologies that were published in a technical document on 'Mobility Tech Tips', by Sun Microsystems [34]. This comparison is between Flash Lite 3 specifications and Java ME Mobile Service Architecture (MSA). The document states that the figures released in 2008 show that Flash Lite is available in approximately 300 million mobile devices available on the market. This number is relatively small compared the the 1.2 billion devices that run J2ME. The implication of this is obvious: J2ME is more widely supported in mobile phones compared to Flash Lite. We reproduce the results of the comparison next.

The author of the publication presents the first comparison, *graphics*. Flash Lite 3 is based on desktop Flash 8, and it supports all the standard rasterized graphic formats. This is shown in the Table 4.1. The table shows that Flash Lite 3 does not support 3D graphics, a feature that is well supported in J2ME.

Graphics	J2ME	Flash Lite 3
GIF		✓
JPEG	✓	✓
PNG	✓	✓
SVG	✓	✓
3D	✓	
FLA		✓

Table 4.1: A comparison showing how J2ME compares to Flash Lite 3 in graphics.

Both J2ME and Flash Lite 3 support most commonly used graphic formats, but unlike J2ME, Flash Lite 3 does not support 3D graphics

Second comparison was on *access to local resources*. J2ME seems to be a clear winner when it comes to accessing local resources on a mobile device. The J2ME MSA specification allows a developer to have a far much greater access to local resources, such a microphone and camera, compared to Flash Lite 3. This is summarized in the Table 4.2.

Local Resource Access	J2ME	Flash Lite 3
Persistent data	✓	✓
File I/O	✓	
Calendar access	✓	
Contact list access	✓	
Photo capture	✓	
Recording video	✓	
Microphone access	✓	
GPS access	✓	
SIM card access	✓	
Accelerometer access		
Phone call initiation	✓	✓

Table 4.2: Accessing local resources on a mobile device capabilities of J2ME and Flash Lite 3

As seen from the table, J2ME has a far greater access to local resources compared to a Flash Lite 3.

Third comparison covered secure communication. Both J2ME and Flash Lite 3 support secure communication to Secure Socket Layer (SSL) enabled web-servers. The J2ME implementations however can also use digital certificates for secure communication. Table 4.3 shows how the two compares.

Security	J2ME	Flash Lite 3
HTTPS	✓	✓
Symmetric encryption	✓	
Asymmetric encryption	✓	
Digital certificates	✓	

Table 4.3: Comparison of secure communication capabilities of J2ME and Flash Lite 3.

In addition to HTTPS, J2ME supports the use of digital certificates for secure communication and is more secure compared to Flash Lite 3.

Fourth comparison covered connectivity. J2ME also has a far more wide support for connectivity, compared to Flash Lite 3. J2ME supports wider protocols when connecting to external resources. The mobile applications supporting J2ME MSA specification can also act as servers and thus act on requests using TCP and UDP transport protocols. This can be beneficial as it allows a mobile device to communicate in a peer-to-peer mode without having to use a proxy. The PushRegistry available in J2ME also allows applications to automatically run to receive data.

Mobile devices running J2ME platform also benefit from the ability to use the local serial and infrared ports available on a mobile device. Table 4.4 gives a summary of this comparison.

Network connectivity	J2ME MSA	Flash Lite 3
TCP Client mode	✓	✓
TCP Server mode	✓	
UDP client mode	✓	
UDP server mode	✓	
Wake up on incoming request	✓	
HTTP	✓	✓
Bluetooth	✓	
OBEX	✓	
SIP	✓	
SMS	✓	
MMS	✓	
Serial/COMM	✓	
Infrared	✓	

Table 4.4: J2ME versus Flash Lite 3 connectivity.

J2ME generally has more connectivity options compared to Flash Lite 3.

The other general comparisons presented in the article is reproduced in Table 4.5.

Miscellaneous	J2ME MSA	Flash Lite 3
Floating point math	✓	✓
XML parsing	✓	✓
Web services	✓	
Mobile payments	✓	
threads	✓	

Table 4.5: Miscellaneous comparison between J2ME and Flash Lite 3.

The Miscellaneous comparison between J2ME and Flash Lite 3 shows J2ME outperforms Flash Lite 3.

As these comparisons show, each platform has its strengths and weaknesses. Flash Lite 3 is particularly good for handling multimedia and would make it the first choice for gamers. J2ME is the platform to use for developing mobile applications that require any form of asynchronous communication, accessing device resources, capture audio/video, render 3D graphics, use location-based services or communicate with Bluetooth devices. It is with these reasons that we chose to

implement our solutions using J2ME, due to its excellent networking capabilities.

4.3.2 Mobile Client

The mobile client, mChisimba, is built based on the J2ME platform and consists of two main layers: the viewer layer and the transport layer. The transport layer is responsible for transporting the packets between the server and the client. The viewer layer is responsible for interpreting incoming packets and displaying the output in appropriate format. This typically involves displaying presentations appropriately. The viewer layer also responds to user input and generates appropriate packets which are sent to the transport layer for forwarding to the server. Figure 4.2 shows how the two layers of the mobile client are structured.

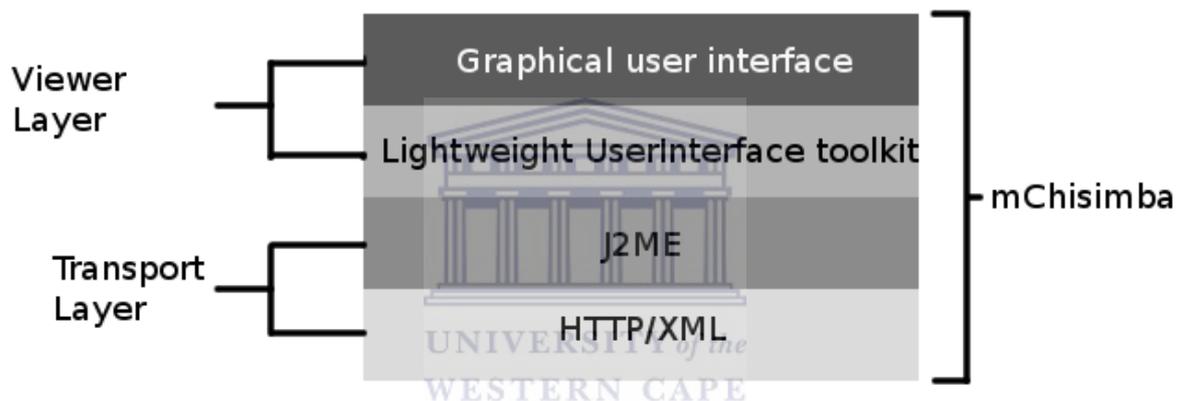


Figure 4.2: The structural layout of mChisimba

There are two main layers: transport layer and viewer layer. The viewer layer is used for interpreting incoming packets and displaying the output in appropriate format. The viewer layer also responds to human input and generates appropriate packets for the transport layer. The transport layer is used for sending the packets to the server via HTTP and for polling packets from the server at preset interval

The layered structure in Figure 4.2 is implemented as a set of classes, each with a specific function in the system. From Figure 4.2, the graphical user interface layer is built using LWUIT. LWUIT allows developers to create a GUI that is similar in many aspects to desktop applications. The user-interface layer consists of three main tabs: presentations viewer tab, the text chat and the user-list

tab. Figure 4.3 shows how these components are structured: The components comprise of individual classes, whose relationship is shown in the class diagram in Figure 4.4. The application entry point is *mChisimba* class. This class contains methods like *startApp()* and *destroyApp()*, for starting and stopping the application. *mChisimba* class also contains method *showMainForm()* that displays the main form that contains a list with three options: 'RoomList', 'Join Room Manually' and 'Settings'. Every time a user selects 'Room List' or 'Join Manually' options, *mChisimbaPersistence* is invoked to retrieve login details. These details are initially set when the user fills in the required option. The relationship between these classes is shown in Figure 4.4.

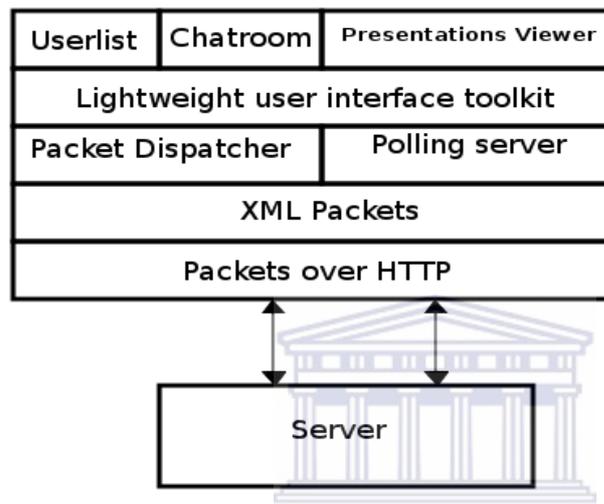


Figure 4.3: The overall structure of *mChisimba*.

The user interacts with the virtual room which consists of user-list, chat-room and the presentation viewer. These are built on top of LWUIT, which in turn interacts with the packet dispatcher and the component used for polling the server

Next we describe how a user interacts with mobile client and how the functionality maps to class-level implementation. Persistence is implemented through a J2ME file structure. This is done at class level in *mChisimbaPersistence* class. Access to *mChisimbaPersistence* is via the 'Settings' options on the main screen. Selecting 'Room List' option invokes the *RoomList* class, which, gets the login details from *mChisimbaPersistence* and connects to the server to query the rooms accessible by this user. When login details are retrieved from the settings, they are dispatched to the server, over HTTP. The actual login is handled by the *Login* class. HTTP is used as the protocol for sending and receiving packets. The packets are tiny well formed XML documents. If authentication is successful during the login process, the server sends a list of available scheduled sessions (also referred to as rooms), from the database, which are then presented to the *mChisimba* user screen from which the user can select and join. When the list of the rooms are displayed, the user can then select the desired room and join. Selecting 'Join Manual' option leads to a screen that prompts the

user to type in the name of the room s/he wishes to join, and then attempts to connect to the room by sending this information to the server.



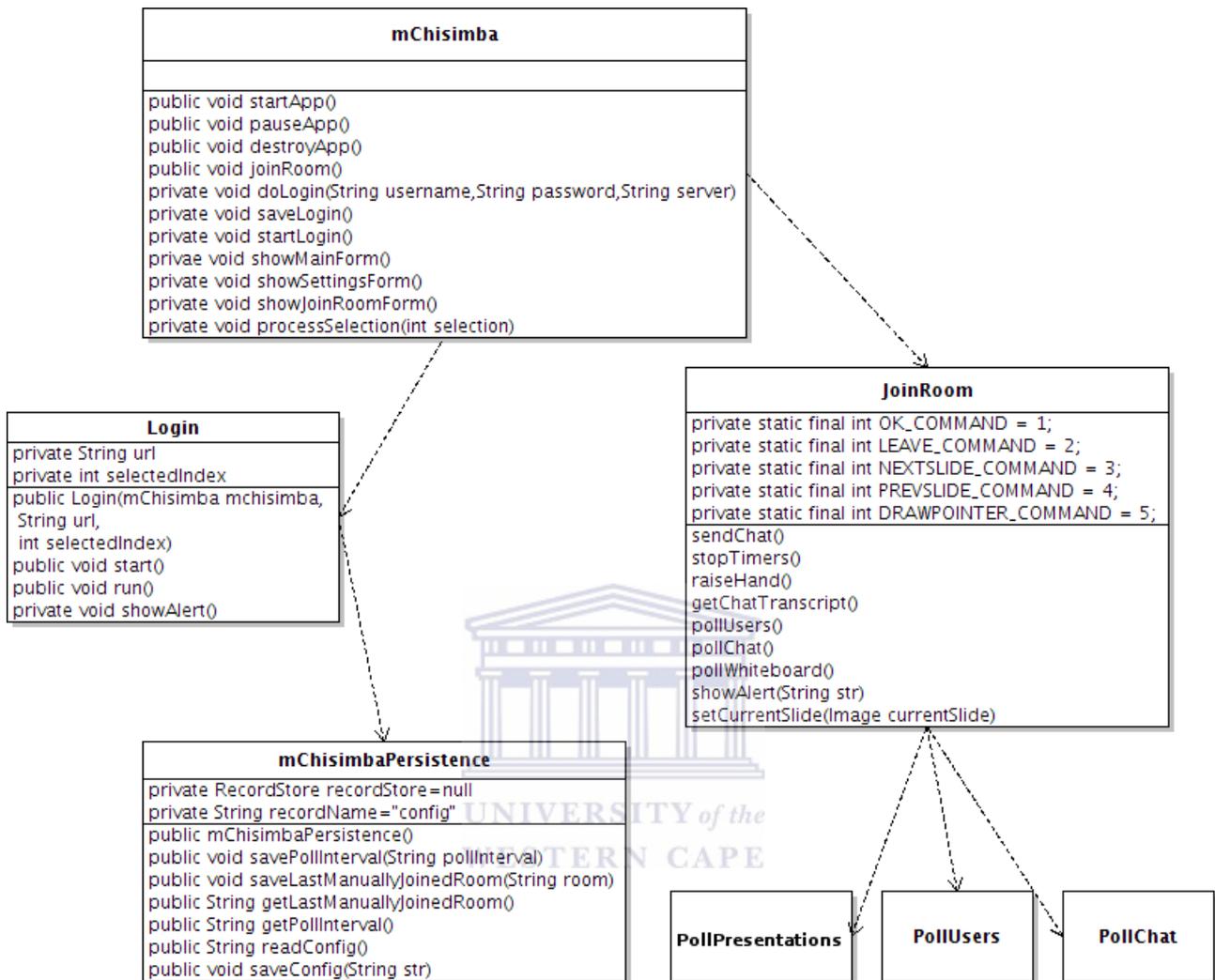


Figure 4.4: mChisimba login class diagram.

mChisimba is the main class, which uses *Login* class to login to the server. Login details are retrieved from *mChisimbaPersistence* class. Upon successful login, *JoinRoom* class is invoked.

Upon joining a session, a function handled by *JoinRoom* class, the user is presented with a virtual room whose view consists of a screen with three tabs: userlist, chatroom and presentations. Please see Figure 4.4 for a class diagram. Each of these entities has a timer that refreshes the screen every time the transport layer polls the server. The frequency of the polling is determined by the value set in the settings section. The polling functionality is implemented in *PollUsers*, *PollPresentations* and *PollChat* classes. The three classes extend *Runnable* class, which allows us to use Task Timers, hence the task-timer model we adopted. The task-timer model enables us to synchronize with the server frequently because it has methods that allow us to schedule tasks and execute the tasks

repeatedly at fixed rate in an efficient way. Figure 4.5 shows the class diagram of the three classes.

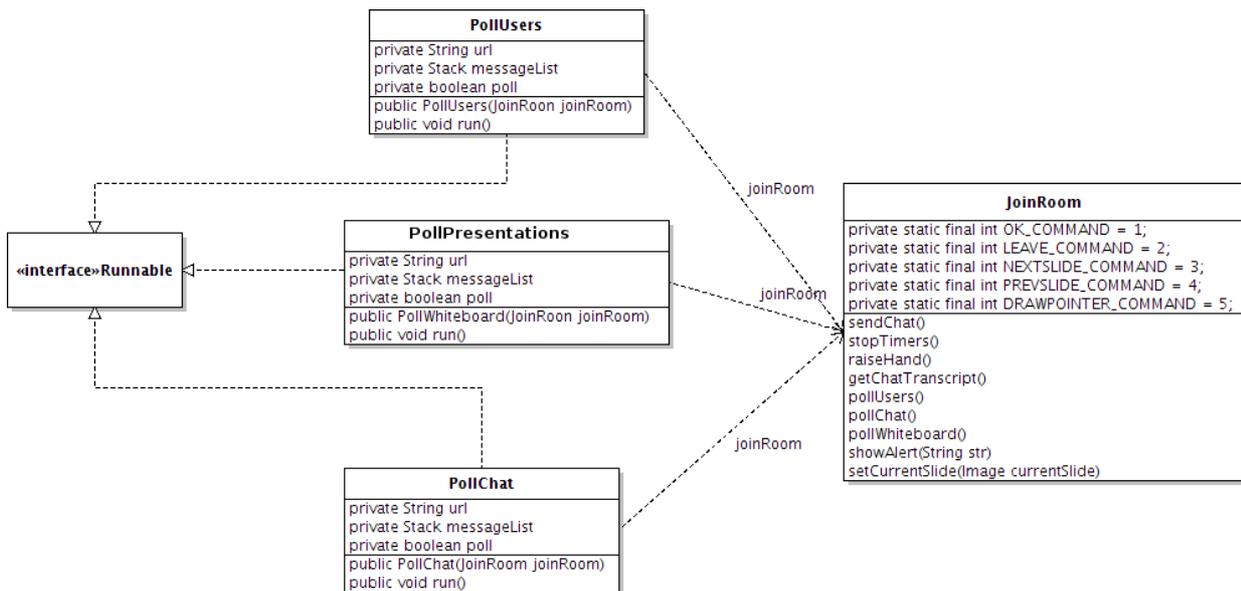


Figure 4.5: Task timer model.

PollUsers, PollWhiteboard and PollChat classes extend Runnable class which enables as adopt a task-timer model that enables us to schedule tasks and execute them repeatedly

Each of these classes has run() method, which initiates the tasks execution process. The classes also contain the url variable that is used to connect to the server via HTTP. The boolean poll variable is essential because it allows us to poll the server more efficiently, saving on bandwidth. A false value in poll indicates there has been no change in the content thus the system pauses until this value becomes true. The other core classes within the mChisimba framework include mChisimbaPersistence, which is used to store login and configuration information on the phone. The login details stored are username, password. The configuration details are server URL, poll-interval value in seconds and the last room joined. As a result, mChisimbaPersistence class contains methods like saveConfig() and readConfig() that are used to read and store configuration information. The SendChat class extends Thread and is used for sending text chats to the server for broadcasting. BandWidthMonitor is the class that is used for logging the amount of traffic flowing between the mobile device and the server. The logging is done on the server. SendCommand is a general command utility class, used extensively in the whiteboard to achieve interactivity. It is used to send commands like 'update pointer', 'update text position', whenever the user makes changes on the whiteboard. These commands are often accompanied by appropriate parameters. The next section explains the communication model between mobile client and server that we adapted.

4.3.3 Communication model

We adapted fixed-interval polling as a communication model between mobile client and server. Another possible candidate communication model was push-register. This section briefly explains why the push-register model was discarded in favor of fixed-interval-polling model during development. 'Push' technology allows an application to receive information in asynchronous mode as it becomes available without having to 'look' for it [35]. Push mechanism can only be achieved by using inbound connections. Inbound connections only work with socket-based protocols like TCP and UDP at the time of writing this thesis. However, TCP/UDP connections in data-oriented mobile phone networks cannot work outside local area networks because of security reasons. As such, in order to use TCP/UDP, the user must make special arrangements with cell phone provider to allow the phone in question to stream traffic through firewalls. This makes it impractical to use in real m-Learning environment because mobile phones are normally assigned IP addresses dynamically and this configuration will have to be done every time a new address is assigned. HTTP on the other hand is supported by cellular networks without need for any special configuration. Thus the easiest connection type one can use is HTTP. But since the HTTP connection is a client type of network connection, it does not support inbound connections. HTTP servers do not initiate connection to the clients, rather, the client must initialize a connection. Based on these observations, we adopted the HTTP-based task timer model. Using the task timer model, it is possible to synchronize with the server after every n seconds/microseconds. This is achieved by sending HTTP requests to the server at preset intervals. The server then responds with appropriate response. Using the task-timer model, the client is able to synchronize with the server after n units of time. The researcher adapted this model when implementing the mobile client, and as such the chat-room, the user-list and whiteboard poll the server periodically. The server in turn sends back updated snap-shot of the chat-room, user-list and presentation from the instructor's client (desktop client). To minimize traffic between the server and the mobile client, the server responds only when the instructor's presentation has changed. If there are no changes since the last poll, a client request is simply ignored.

4.3.4 Server

The server is implemented as two components, both of which are plugins for a Jabber-compliant realtime collaboration server, Openfire. Openfire uses the XMPP as the communication protocol. XMPP is an open Extensible Markup Language protocol for near-realtime messaging, presence, and request-response services. The first server component, the *server plugin*, processes client

requests, except requests from mobile clients. The second component acts as a gateway for communication between the mobile client and the server. This second component, *the gateway*, transforms XMPP packets from the server into HTTP format for sending to the mobile client as well as transforming incoming HTTP packets from mobile client into XMPP packets for server interpretation. Figure 4.6 shows the structural relationship of the server components inside a server container (openfire):

Server plugin.

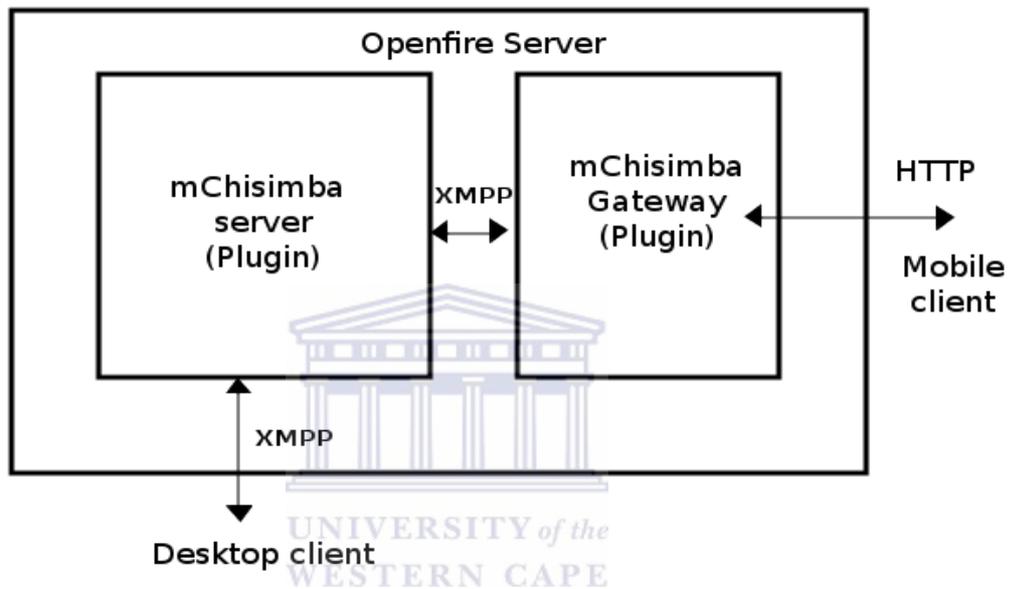


Figure 4.6: mChisimba server components.

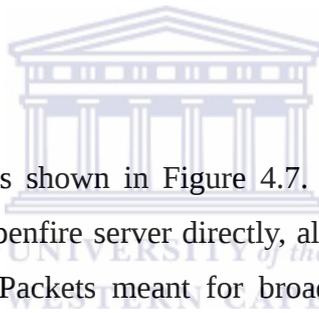
The server is made of two main components implemented as plugins inside Openfire server. The first plugin, server-plugin, is used for interpreting client packets, except for HTTP requests from mobile client. The second component, the gateway, transforms packets from server, meant for mobile client, into HTTP packets that are send to mChisimba mobile client, as well as transforming incoming HTTP packets from the mobile client into valid XMPP packets before being forwarded to mChisimba server.

The server plugin has a packet processor for handling Information Query (IQ) packets that come from either the gateway plugin or the desktop client. Packets from the mobile client are not directly send to the server plugin, rather to the gateway plugin, which is discussed in the next subsection .

```
<iq id="q2s0S-9" type="get">
  <query xmlns="iq:avoir-realtime:realtime">
    <mode>pointer</mode>
    <content>
      <pointer>1</pointer>
      <x>208</x>
      <y>61</y>
    </content>
  </query>
</iq>
```

Figure 4.7: An Information Query (IQ) packet.

A typical IQ packet from gateway plugin to server plugin. The gateway and the server plugins communicate by exchanging these types of packets over XMPP. Desktop clients too send these type of packets to the server plugin directly.



The general format of IQ packets is shown in Figure 4.7. With the exception of the *Message* packets which are handled by the Openfire server directly, all other incoming and outgoing packet pass through a **packet processor**. Packets meant for broadcast to all room participants, when intercepted by the packet processor, are first stored in a database then broadcast to all other participants connected to the session, using the **broadcast service**. Other packets meant for querying the server for information are forwarded to the **database service** for information retrieval/storage, then are returned via same route or forwarded to the broadcast service. Such packets, for example, are send when a client queries for available scheduled sessions. The Figure 4.8 shows the path the packets follow when they encounter the packet interceptor.

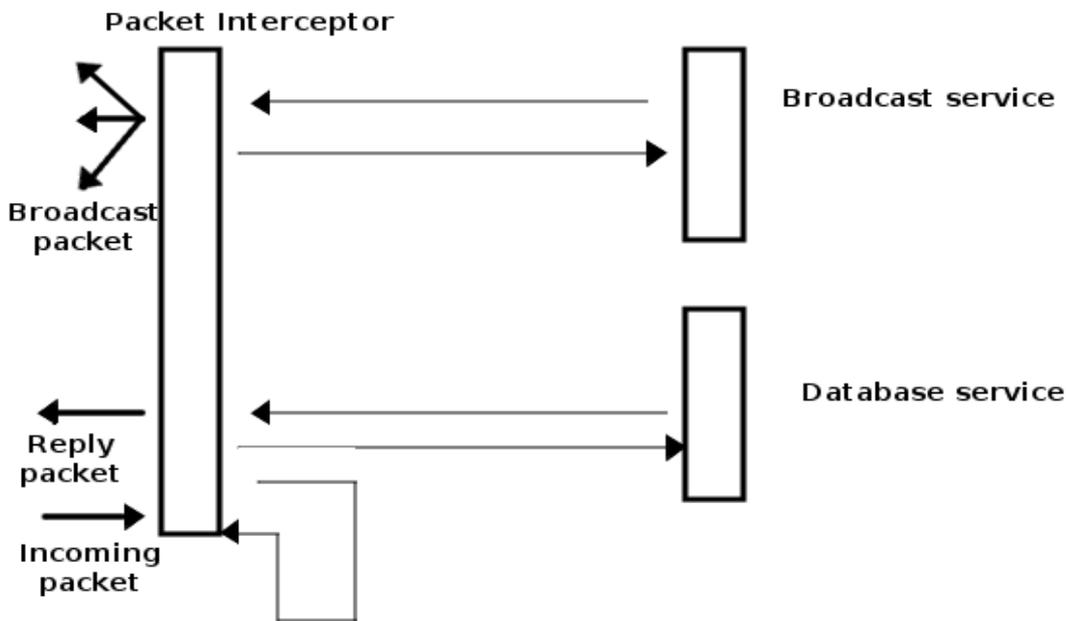


Figure 4.8: The server plugin packet interceptor

The server plugin consists of a packet interceptor, which (1) forwards packets to database service for querying information from the database (2) forwards packets to the broadcast service for broadcast and (3) returns the packet along same route.

The two core-services, the broadcast service and the database service, are implemented as a set of classes, that interact to provide their respective functions, as shown in Figure 4.9 class diagram. All incoming packets first pass through the *AvoirRealtimePlugin* class, which acts as the main class for the plugin, as well as the packet interceptor. The *AvoirRealtimePlugin* class then determines what to do with the packets, after decoding them using *XmlUtils* class. Packets requiring retrieving information from database are forwarded to *RoomResourceManager* class which consists of methods for accessing the database. Packets related to presentations are handled by the *SlideShowProcessor* class, while poll/survey/question-related packets are handled by the *QuestionProcessor* class. All other packets are processed by the *DefaultPacketProcessor* class.

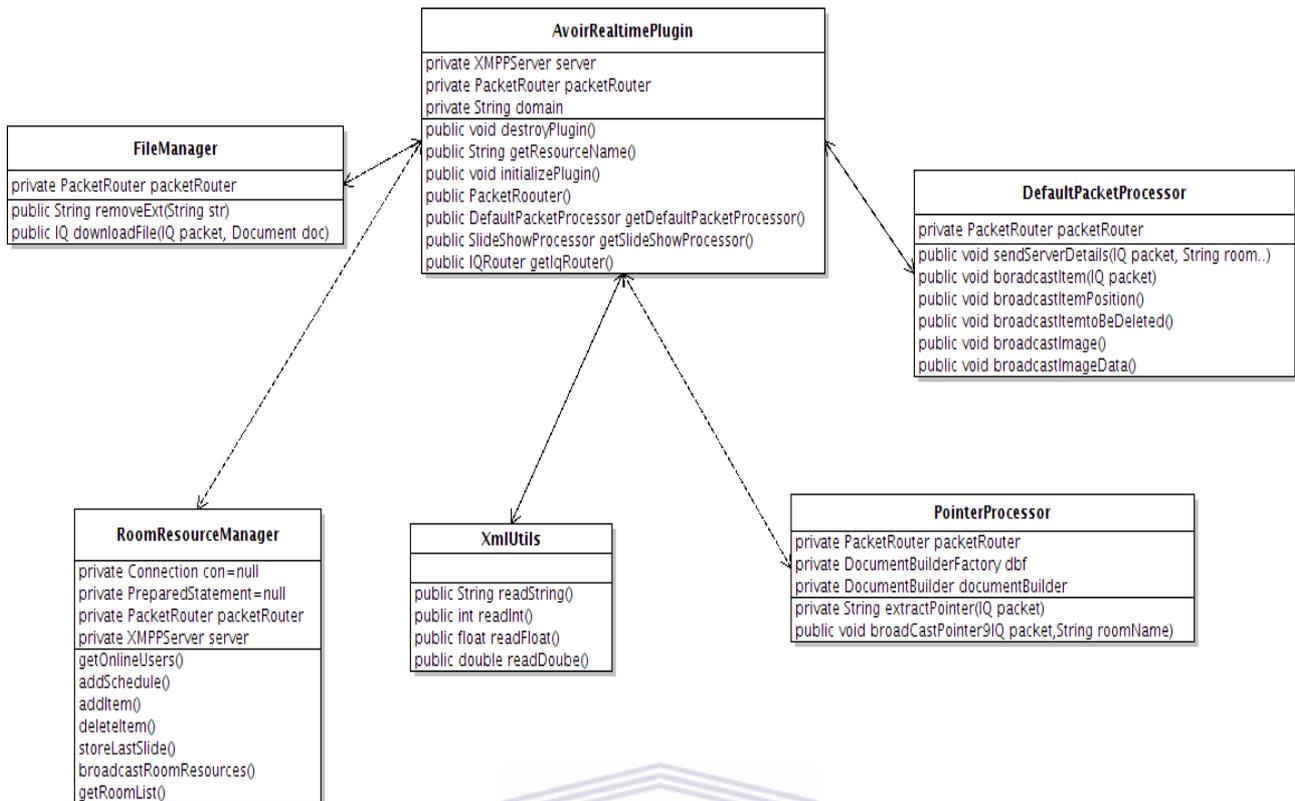


Figure 4.9: Server class diagram.

The server plugin is implemented as a set of classes which provide the two major services: database service and broadcast service. All incoming packets first pass through the AvoirRealttimePlugin class, which acts as the main class. The AvoirRealttimePlugin class then determines what to do with the packets, after decoding them using XmlUtils class. Packets requiring retrieving information from database are forwarded to RoomResourceManager class which consists of methods for accessing the database. Packets related to presentation are handled by the SlideShowProcessor class. All other packets are processed by the DefaultPacketProcessor class.

Gateway plugin

The gateway is the 'middleman' in the communication framework between server and mobile client since it understands both protocols from both the server and mobile clients. A request originating from a mobile client to the gateway typically consists of an HTTP packet with two major elements: *type* and *content*. When the gateway receives this packet, it determines the *type*. Based on the result, the content is extracted from the packet and repacked as a 'proper' XMPP IQ packet that is forwarded to the server plugin for processing. This is accomplished with the Smack API (www.igniterealtime.org). The gateway plugin intercepts packets from the mobile client, transforming them into valid XMPP packets before forwarding them to the server-plugin. Reply packets from the server plugin meant for the mobile client are first forwarded to the gateway plugin, which transforms them into valid HTTP packets before being sent to the mobile client. This is illustrated in the Figure 4.10:

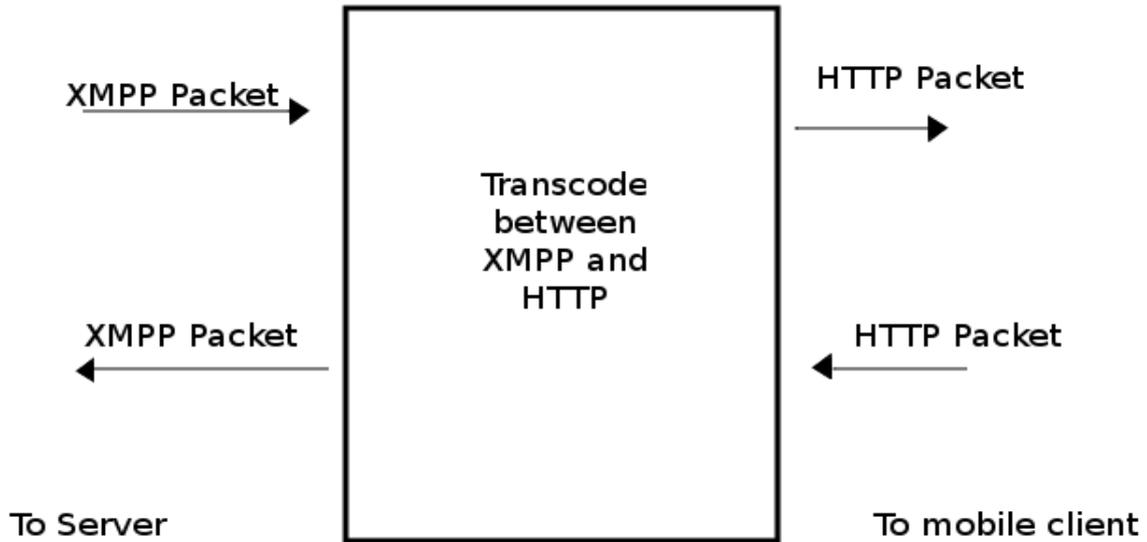


Figure 4.10: The gateway plugin.

HTTP packets from the mobile client are first transformed into valid XMPP packets before being forwarded to the server plugin. XMPP packets from the server are also transformed into HTTP packets before being forwarded to the mobile client

The gateway plugin is implemented using three core classes. The *MChisimbaPlugin* is the main class and acts as the interface to the server plugin. Packets from the mobile client are forwarded to server plugin using this class, after being converted into XMPP format using *mChisimbaServlet* class. *mChisimbaServlet* is the direct interface to mobile clients since it implements HTTP protocol. It contains methods such as `doPost()` and `doGet()`, which implement logic for processing HTTP these methods enable us to read from a mobile device directly, as well as write to it. This class also contains `getScaledMethod()`, a crucial method used for image transformation. Source images from conversion of Openoffice/Powerpoint presentations are scaled down using this method. The method uses the Java API for Bilinear interpolation to achieve this. Other methods like `processConnect()` is used to pass connection parameters from the mobile client to the server plugin via the *MchisimbaPlugin* class. *RpacketListener* is used by *MchisimbaPlugin* to interpret packets coming from the server plugin. This is accomplished with *XmlUtils* class. The class diagrams for these classes are shown in Figure 4.11.

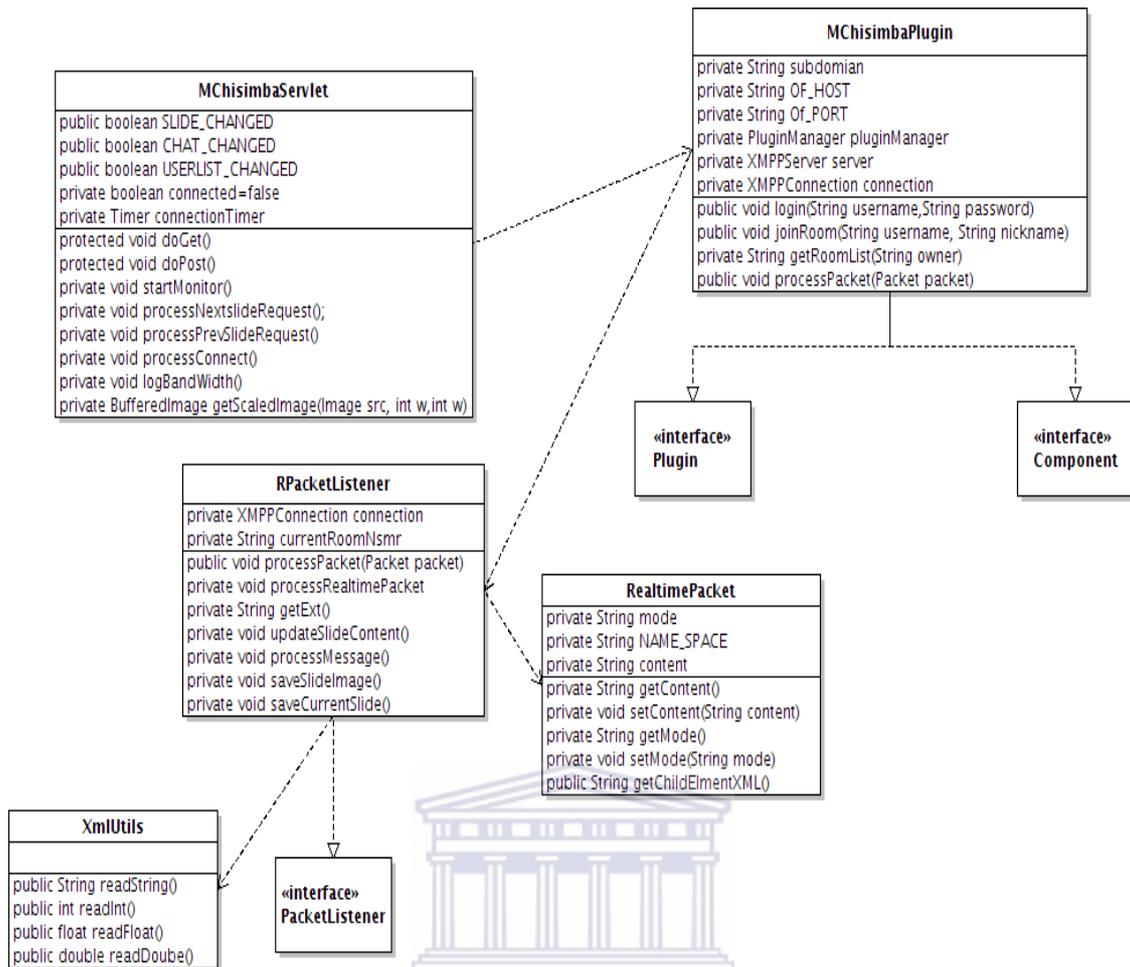


Figure 4.11: The Gateway class diagram.

The gateway plugin is implemented using three core classes: the *mChisimbaPlugin* class, which is the main class, the *RPacketListener*, which is responsible for interpreting packets and the *MchisimbaServlet* class, which is a class that directly interfaces with the mobile phone using HTTP

4.4 Database

MYSQL is used as the database management system and the server plugin is the only server component that directly accesses it, using the Openfire database access API. The server plugin component, however does not directly initialize connection to the database, as this is handled by the parent container, Openfire. All packets are automatically logged in the database. User profiles too are stored in the database. Figure 4.12 illustrates how the server plugin component is relates to the database inside the openfire server component.

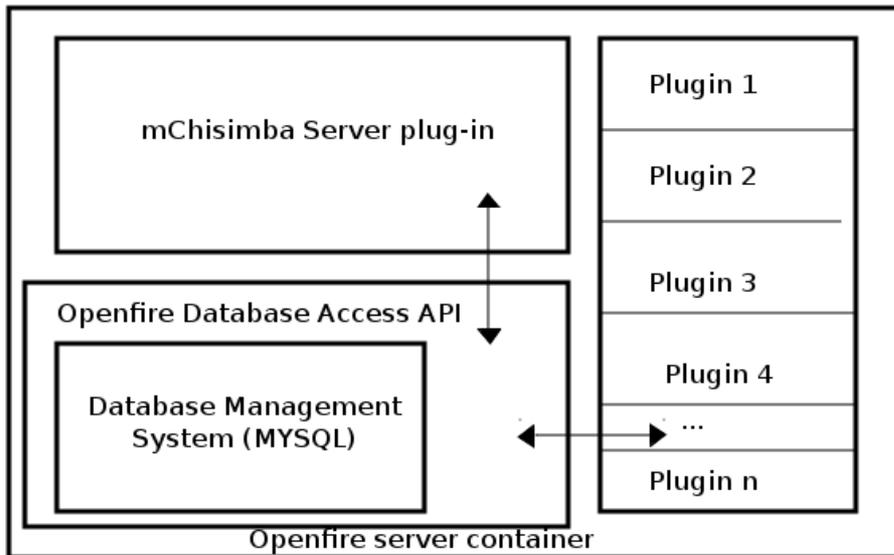


Figure 4.12: mChisimba server plugin and the database management system.

MYSQL is used as the database management system and the server plugin is the only server component that directly accesses it, using the Openfire database access API. The server plugin component, however does not directly initialize connection to the database, as this is handled by the parent container, Openfire. All packets are automatically logged in the database. User profiles too are stored in the database.

Database access is implemented using two classes. The *AvoirRealtimePlugin* is the interface to other components, it receives and sends packets. The *RoomResourceManager* class contains methods that use Openfire API to access the database directly. The class diagram for these two classes is shown in Figure 4.13. The next section gives a brief overview of other open source libraries used and the other protocols.

4.5 Open source libraries

mChisimba client and the server is built using a number of open source libraries. Openfire is used as a server container. Internally, Openfire uses a large number of open source components, notably the XMPP, the widely adopted open protocol for Instant Messaging. Smack is used by the gateway to communicate with the server. Smack is an open source XMPP (Jabber) client library for Instant Messaging and presence. The client uses LWUIT as the base framework for building the user interface. A GUI built using LWUIT is very similar to a desktop GUI. Another tool, called 'DJ Project' is used to embed Flash objects used to capture and stream audio/video from the instructor's desktop client to the server for broadcasting to clients. The tool contains a NativeSwing library that allows easy integration of some native components into Swing applications and provides some native utilities to enhance Swing's APIs. And as introduced in the previous section, MYSQL is used

as the database. Log4J is the logging library we primarily used for debugging and auto-logging information recorded.

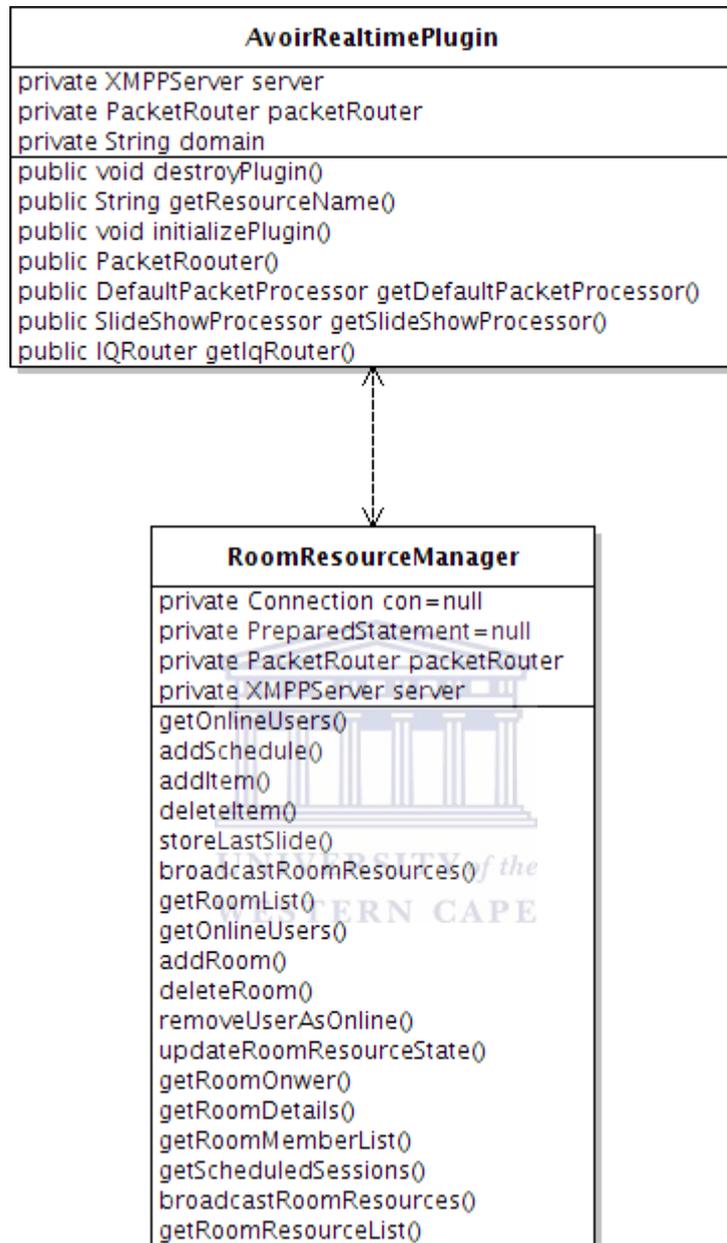


Figure 4.13: Database access class diagram.
 Access to database is provided by RoomResourceManager class which contains methods that use Openfire database API to directly access database tables

4.6 Communication Flow

A typical scenario when using mChisimba would involve the following steps:

An instructor uses desktop client to connect to the server and starts a classroom session. In this step, the desktop client sends the necessary XMPP packets to the server for handshake handling and authentication. The instructor is responsible for sending an invitation/notification to participants using tools available in the desktop client.

When a mobile participant receives an invitation, typically an email/SMS, s/he launches mChisimba (the mobile client) using a link sent in the email/SMS. mChisimba connects to the gateway and a handshake between the two takes place.

mChisimba then sends a packet containing the user's login details to the gateway that is in turn repackaged and forwarded to the server for actual authentication. The results from the server are sent back to the gateway, formatted and forwarded to mChisimba. If authentication was successful, the server will update the gateway with appropriate presence packets that are also broadcast to all other connected clients in the domain.

When the three communication components are in place, the desktop client is used by the instructor to stream presentations to the server plugin which in turn formats them and broadcasts the formatted presentations to all connected gateways. The server plugin also stores this content for a user-defined duration of time for future access. An mChisimba client can also access the recorded session at a later time.

4.7 Summary

This chapter discussed the structure of mChisimba client and the server. The client is implemented as a layered component consisting of two main components: the display layer, which makes up the user interface, and the transport layer, that interfaces with the server. These components are presented to the users in form of a GUI consisting of chat room and a presentations viewer. The chat room is used for text-based chat. The presentations viewer allows users to navigate through slides in a presentation, raise hands in a session, annotate over slides. The server consists of two main components: the *server plugin*, which is the main server implementation, and the *gateway plugin*, which is responsible for trans-coding the packets between the server plugin and the mobile client, since they use different communication protocols. The components were implemented in Java Standard Edition, version 1.6.



UNIVERSITY *of the*
WESTERN CAPE

5 Data analysis and discussion

This chapter discusses the data collected during the experiments. Our experiments were focused on answering the research questions raised in Chapter 3. We aimed at determining how to prepare presentations in order to transport and display them on a tiny mobile phone screen. Based on this, we divided our experiments into two main phases, and it is along these lines that we present the data collected. During the first phase, we were interested in finding out how to scale down the presentations to reduce the file size, suitable for transporting to a mobile device. We analyzed the quality of the scaled-down presentations and measured the time it takes to accomplish the scaling process. Section 5.2 covers this phase, explaining what experiments were done and what data was collected, and the analysis/discussion of the results. In the second phase we measured the air-time cost users incur when transporting the presentations to a mobile phone. We collected data based on the transportation of presentation slides, polling of users and when chat messages are exchanged between users. The discussion of the results from this phase are presented in Section 5.3. But first, let's revisit the environment under which we conducted the experiments and collected data.

5.1 Practical aspects of experimentation

All experiments were carried out inside a laboratory. The equipment used for the experiments consisted of computer hardware, mobile phone and software. Three hardware devices we used in the experiments: SunFire server (www.oracle.com), a desktop computer and a Nokia N82 mobile phone. The server was connected to the Internet with a public IP address. We used a wide range of software: Ubuntu 8.10 (www.ubuntu.com), the operating system installed on the server, Openfire 3.6.4 (www.igniterealtime.com), the XMPP server, installed on the Ubuntu operating system, the server component of the software we developed, deployed as a server plugin into the Openfire XMPP server, Mysql 5.1 (www.mysql.com), used as the database, Openoffice 3.2 (www.openoffice.org), that was used to convert slides in a presentation into a series of images, JODConverter (<http://code.google.com/p/jodconverter/>), a Java library that converts office documents into different formats, ImageJ 1.4.2, an image processing software written in Java, developed by Wayne Rasband at National Institutes for Health (<http://rsbweb.nih.gov/ij/>) to implement histogram analysis, Log4J (<http://logging.apache.org/log4j/1.2/>), Java-based logging library, and finally J2ME, which came pre-installed on the Nokia N82 phone. We used spreadsheets for generating some of the graphs presented in the next sections.

5.2 Scaling down presentations and analyzing quality

This was the first phase of the experiments. We started by downloading freely available existing presentations from a repository (<http://presentations.wits.ac.za>) managed by the library at the University of Witwatersrand, Johannesburg. The presentations consisted of Microsoft's Powerpoint and Oracle's Openoffice Impress. We downloaded 21 random presentations that ranged in sizes, from 17.7 KB to 28.4 MB. The first step towards scaling down each of these presentations was to upload them into the desktop-based software we developed using a desktop computer, in order to transport them to the server. The server, as mentioned in Section 5.1, had software installed that helped us do the down-scaling. Once each of the presentations was received by the server, they were moved to the JODConverter module. This module, using OpenOffice libraries, converted each of the slides in the presentation into a series of images, each image being a 'snapshot' of a slide. We were not interested in the time it takes to convert a presentation into a series of images, because this process takes place before a learning session starts, and thus the *duration* it takes to convert a document into image slides does not directly impact the performance of the m-Learning application. We thus concentrated on the generated images, after the conversion.

We analyzed the converted images, to determine the accuracy of the conversion, by using image histograms. An image histogram describes the frequency of the intensity values that occur in an image. We used histograms to plot the total number of pixels at each grayscale level in an image and thus determine overall intensity in an image; a histogram skewed to the left implies a darker image; to the right a bright image. In order to carry the analysis, we used Log4J to log down the histogram values for each of the images corresponding to slides in a presentation. To compute a histogram for an 8-bit image, we needed a set of 256 counters, each for an intensity value. Then we iterated through the image determining pixel value at each point, and incrementing the corresponding counter by one. At the end, the counters showed the number of pixels in the image that have that corresponding intensity value. However, histogram computation for images with more than 8-bit is a little more complicated, and the discussion of the algorithms involved is out of scope of this thesis. Detailed discussions on image histograms is provided by Burger and Burge [36].

The idea behind using image histograms was to quantitatively determine the quality of the image generated from taking a snapshot of a slide. In general, the image histogram values will fall somewhere between 0 and 255, since we are generating 256-color images. This range can be divided into 4 sections as shown in Figure 5.1.

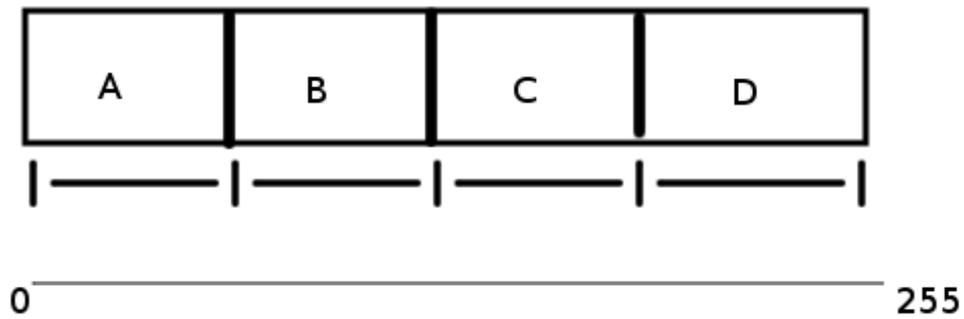


Figure 5.1: A diagrammatic categorization of mean image values.

A diagram showing the four main sections a mean image histogram values might lie. Section A represents the darkest image, Section D represents the brightest image..

Section A represents an image with very little light in it and Section D represents the brightest image. An image from a slide whose mean histogram value lies between Sections C and D is a bright image. While this does not automatically make it a good image, it implies the image (which we must remember is a snapshot of a slide), will be more visible on a tiny mobile screen. In the following sections, we report results from two scenarios that show how mean histogram values were used to predict how the slide rendered on a mobile phone. Section 5.2.1 gives an example result of a bright image, and shows how it appears on the mobile screen. Section 5.2.2 gives an example result of a darker image, and its mean histogram values, and how it renders on the mobile phone. A discussion on the trends from the experiment is given in Section 5.2.3.

5.2.1 Clear Slides

In this section we present an example result of qualities of slides that rendered more clearly on a mobile screen. The slides had a minimum mean image histogram value of 128, according to our recordings. Figure 5.2 shows a bright snapshot of a slide that was generated by the software we developed and Figure 5.3 shows the image histogram of this slide as generated by ImageJ.



Figure 5.2: Snap shot of a bright slide as seen from desktop client.
 Snapshot of a bright slide, rendered as an image with the desktop client of the software we developed

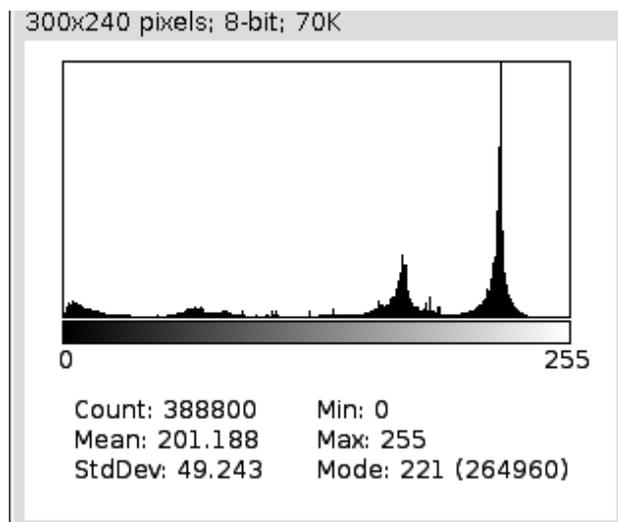
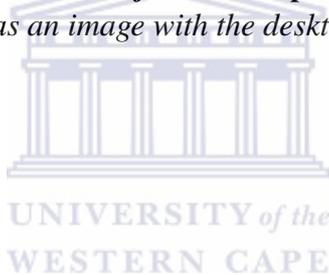


Figure 5.3: Distribution of images values of a bright slide.
 Image histogram of a slide as generated by the ImageJ plugin embedded in the software we developed.

From Figure 5.3, we can see that the image histogram is skewed to the right, which means the majority of pixels in the image have a high value that approaches 255. This implies that this is a bright image. However it is important to also notice the small bumps to the left, and somewhere between the middle and the right side of the graph. The bumps indicate that the image has some dark patches, but these are necessary to balance the brightness. While this is not the ultimate way of determining how 'bright' an image is, it is a indicative factor. In order to see if the shape of the histogram has any relationship with how the image renders on the screen, we provide a mobile screen shot for the above slide, shown in Figure 5.4. As we can see from Figure 5.4, the slide rendered 'correctly' and 'brightly' on the mobile screen since it had enough brightness to make it clearly visible on the mobile screen. The mean histogram value of this slide is 201.188 and this value is in Section D (the bright) of Figure 5.1.

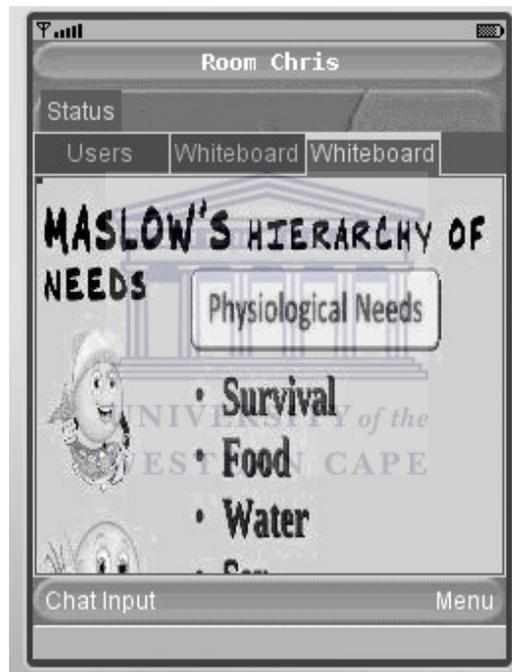


Figure 5.4: A snapshot of bright slide when rendered on a mobile screen.

This snap shot shows how 'clearly' and 'brightly' a bright slide renders on a mobile phone screen.

5.2.2 Poor Slides

In this section we present example result of a quality of a slide that rendered poorly on a mobile screen. Slides exhibiting similar characteristics had a maximum mean image histogram value of up to 64. Figure 5.5 shows a snapshot of an example slide that was generated by the software we

developed.

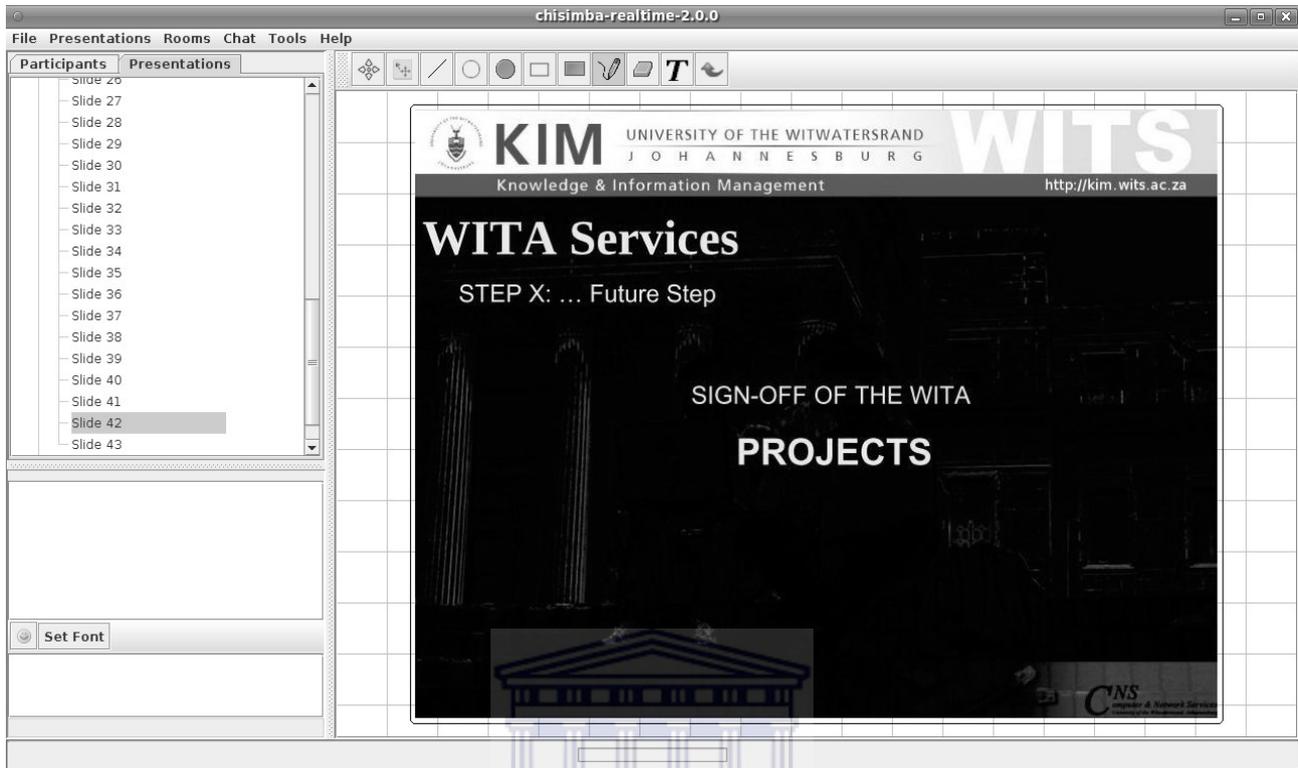


Figure 5.5: A snapshot of a dark slide on desktop client.

The slide shown in Figure 5.5 generated a histogram shown in Figure 5.6. Predictably, the histogram skewed to the left. This is because majority of the pixels in the have a value less that 128 and they tend to approach zero. This makes the image darker. This histogram has very few and small bumps to the right, and this makes it really dark. This is shown in Figure 5.7, when it renders on a mobile screen. As can be seen, this slide rendered poorly, not surprising since the mean histogram value of the slid is 40.907, which is below 64, and thus falls in Section A of Figure 5.1.

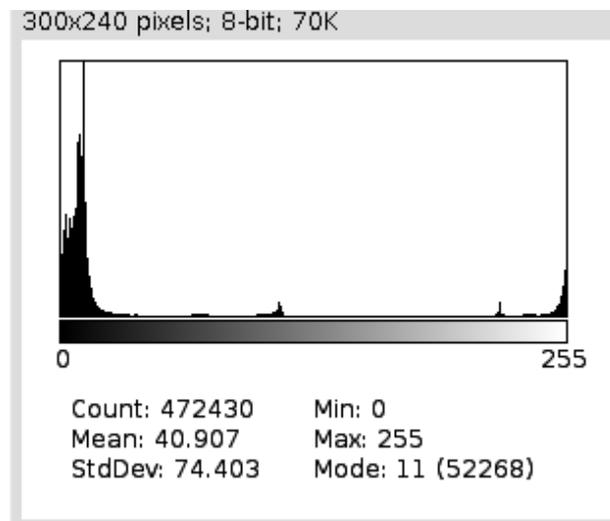


Figure 5.6: A darker image plotted.



Figure 5.7: A poorly rendered slide.

5.2.3 Trends

To be able to study the trends in image histograms of the slides for all the 21 presentations, we ran each of the presentations through the desktop client of the software we had developed and generated image histograms for individual slides. The desktop client includes a presentation viewer, capable of displaying the presentations. However, the viewer does not display the original presentation, rather, the converted version. Implying, the viewer actually displays the image snapshots of each slides, rather than the slides themselves. Now, using this approach, we were able to easily generate histograms from these images from user's side. We generated the histograms through an automated process, using an ImageJ-based function that automatically generates image histograms of all the slides in the current presentation through a single button click. It is these mean values for each of the presentations that are presented in Table 5.1.

For each presentation, the researcher assigned a clarity score value of between 1 and 4, depending on how 'brightly' the presentation rendered on a mobile screen in general. Please note

that the results of these scores are solely a researcher's observation, and no other users were involved in the process. These scores were then put in a column next to presentations' image mean values, each score corresponding to a presentation's mean image value. The clarity score value was arrived at when the researcher observed how the presentation rendered on a mobile phone. A value of 4 meant the presentation was very clear in general, a value of 1 meant the presentation rendered poorly. The researcher was interested in establishing the relationship between the mean presentation image histogram value and the clarity score.

Presentation	Mean Image Histogram Value of all the slides in the presentation	Researcher clarity Score on Mobile Phone (on Scale of 1-4). 1 is poor, 5 is best
1	168.86	3
2	119.21	2
3	111.27	1
4	170.02	2
5	164.46	2
6	195.33	3
7	237.97	4
8	207.32	1
9	216.58	4
10	161.69	4
11	87.34	4
12	109.13	1
13	243.84	4
14	195.33	3
15	146.26	4
16	162.35	4
17	174.56	4
18	189.51	4
19	205.91	1
20	186.92	1
21	225.33	1

Table 5.1: Mean presentation image value together with researcher rating.

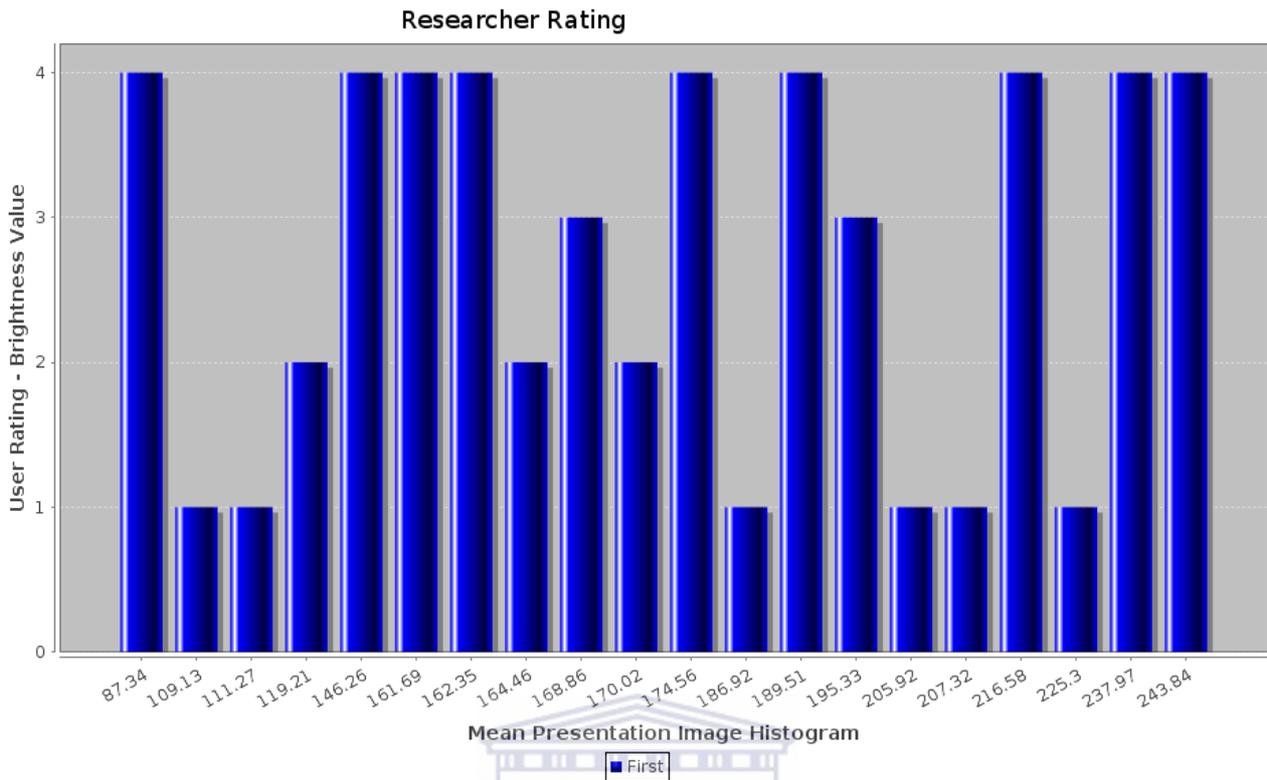


Figure 5.8: Researcher rating scores as related to the mean presentation image value

From Figure 5.8 we picked the presentations that scored the best clarity, and re-plotted them as shown in Figure 5.9. The dark shaded dark bars represent presentations with high scores. We found that majority of these presentations had a histogram value of more than 128.

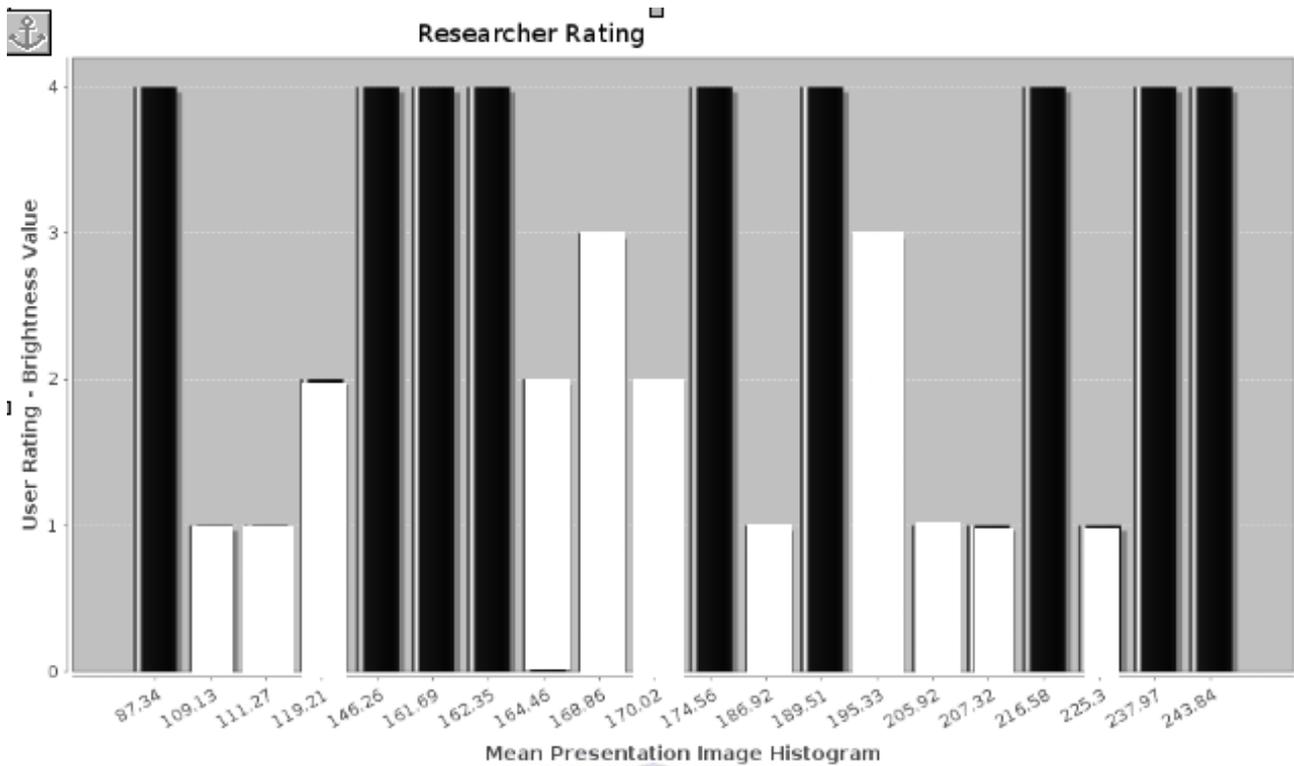


Figure 5.9: All shaded bar show presentations with high mean value, except for three.

We also noticed some interesting observations from Figure 5.9. Three presentations, each with a mean histogram value of more than 200, had different scores as shown in table 5.2. When we looked these presentations, we immediately determined that cause of this. The presentation that had clarity score of 4 had slides with text rendered font value 32. The other presentation had similar content format, but a lower font value of 18, hence the poor score.

Presentation	Histograms Score	Clarity Score
7	237.97 (font greater than 32)	4
8	207.32(font less than 32)	1
20	186.92 (font less than 32)	1

Table 5.2: A table showing two presentations with a very high mean value.

These presentations, however, have different score values. Presentation 8 and 20 had a low score because the slide content has small font.

5.2.4 Performance cost of scaling down slides

The actual process of scaling down a converted image slide before streaming it to the mobile phone is done in realtime, at the time when mobile user is connected to the server. This is because the scaling process is based on the screen dimensions of the mobile phone, which are only supplied to the server when the mobile phone client connects to the server. As such, it was important for us to determine the performance cost of the scaling process, and the effect it has on the performance of the m-Learning application. The scaling process takes place on the server, and we used Log4j logging library to record the observations. To obtain the results presented in Figure 5.10, we created a function in the code that reads images from a directory and uses Java's built-in interpolation algorithms to execute scaling operations. Specifically, we used bi-linear interpolation algorithms. For each scaling operation, we used Log4J to log down the duration it takes to complete the operation. The screen size of Nokia N82 phone used for the experiments is 240 x 320 pixels, so each the images were scaled to this size. Figure 5.10 shows example tabulation of the results of one such down-scaling operation.



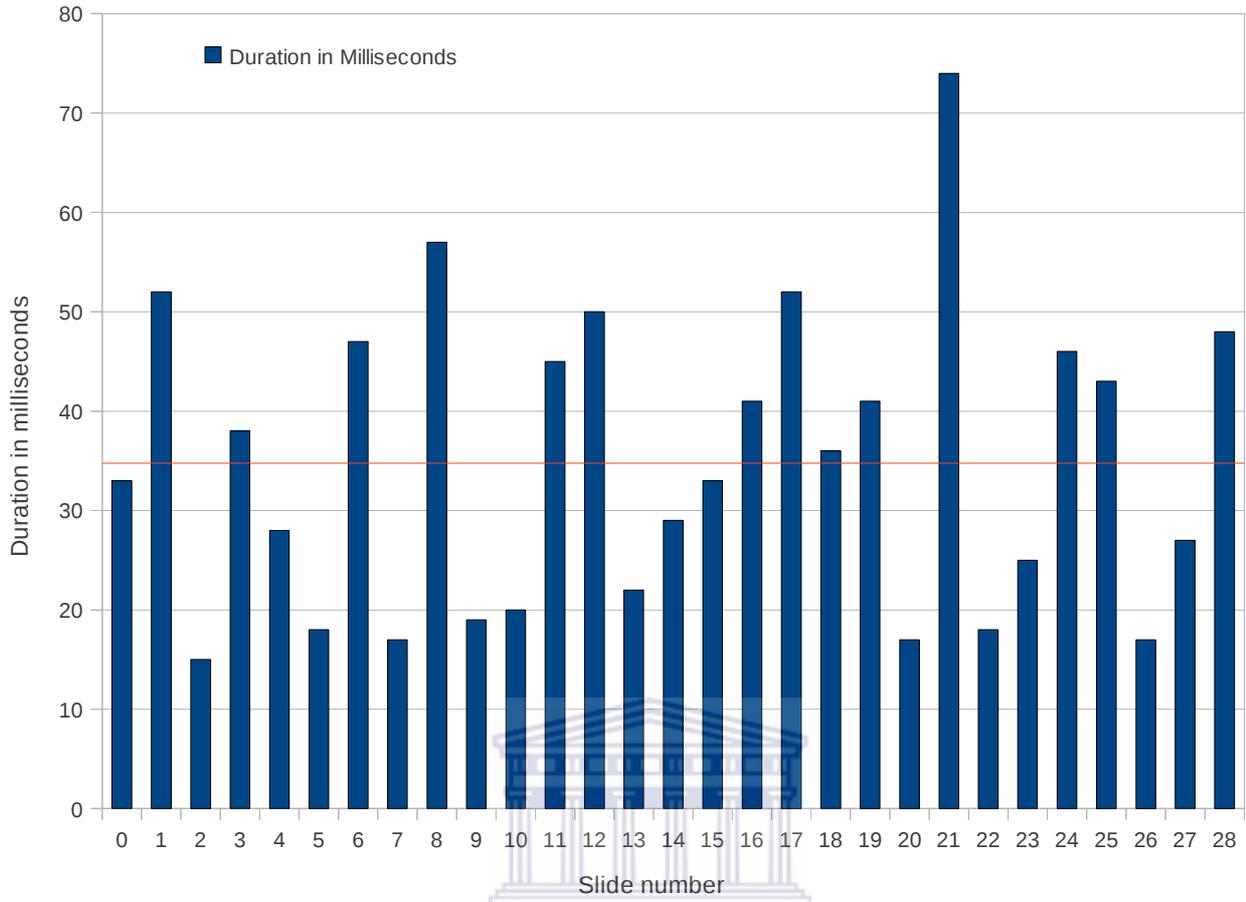


Figure 5.10: Duration it takes to scale down an image slide for presentation.

The scale times, on the vertical axis, are in milliseconds. The slide number are plotted on the horizontal axis.

The mean value line from Figure 5.10 shows that all the slides were scaled at a mean value of 35 milliseconds, fast enough to be appropriate for our m-Learning application. We also present Figure 5.11 to show example results of a second presentation, whose mean scale time is 48 milliseconds. Figure 5.12 shows the mean values for all the 21 presentations we tested with. The average value we obtained is 38.26. This value is well below 100 milliseconds.

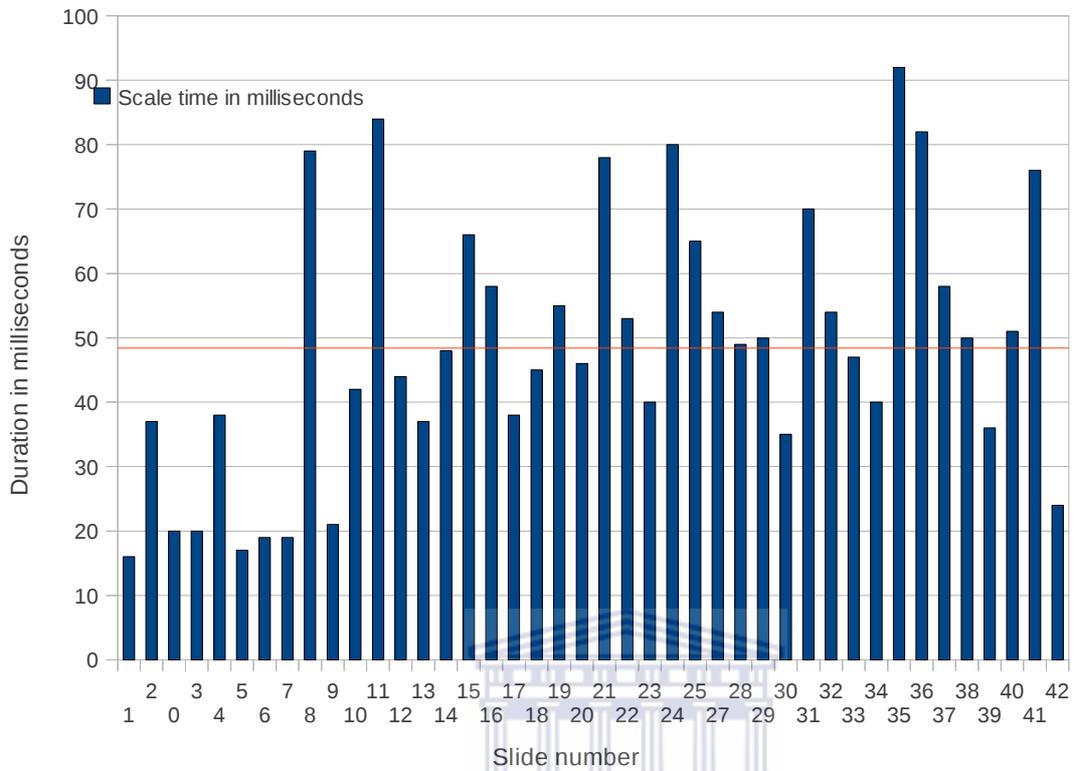


Figure 5.11: A chart view of the scale times for a second presentation.

The conversion duration is in milliseconds, plotted on vertical axis. The horizontal axis represents slide number.

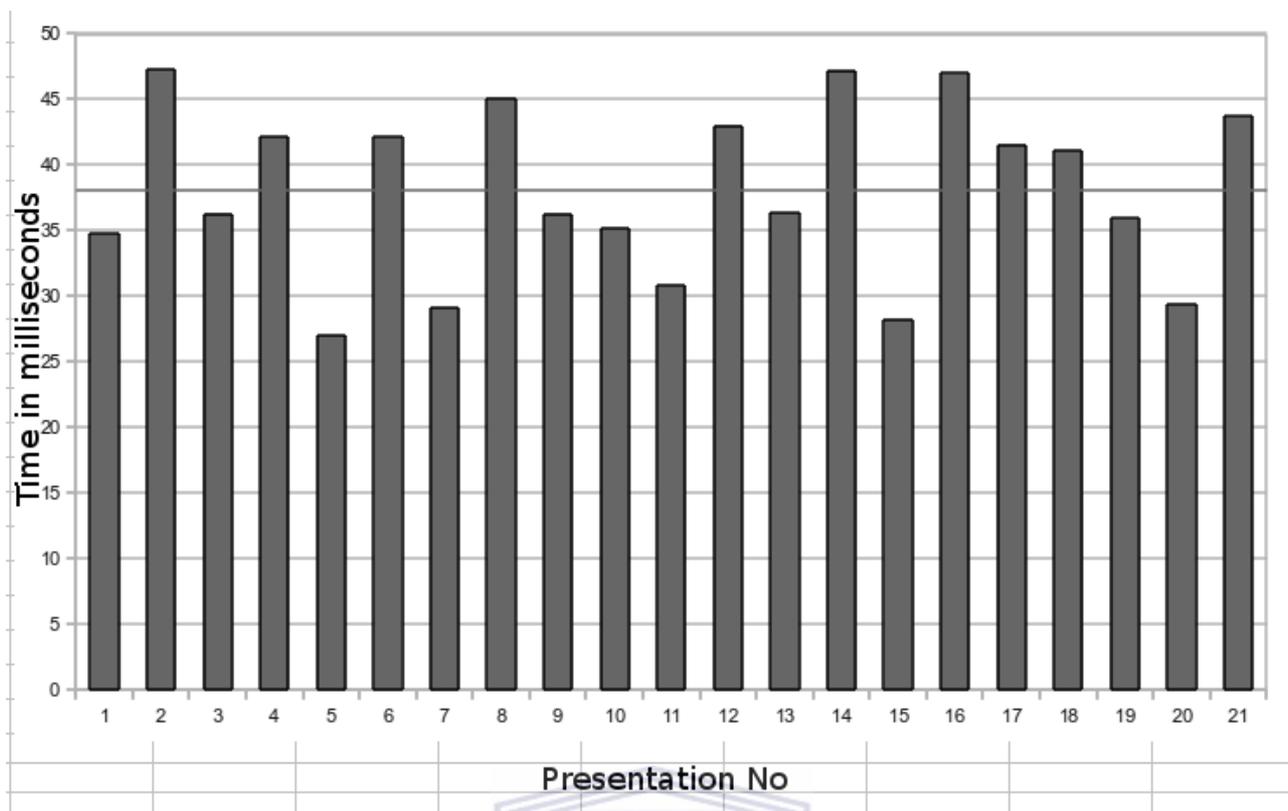


Figure 5.12: Average scale time per presentation for all the presentations.

The average time it takes to scale a slide image to fit mobile dimension specification is 38.05, less than 100 milliseconds. This makes this process ideal for realtime content formatting for mobile phone applications.

5.3 Air time costs

This section presents the results of the amount of data that is transported through the mobile phone during a typical presentation, and thus the air-time cost implication. The experiments were performed on the 21 presentations, using third generation cellular network Vodacom South Africa, using Nokia N82 phone. Vodacom bills per data transported, not by how long one has been online. As a result, these observations only apply to networks which bill per data transported and not based on how long one has been online.

The calculations on air time cost were performed based on the amount of data transported through the network to the phone during simulated learning sessions. To obtain the data transferred, we conducted experiments in three phases. During phase 1 we recorded the amount of data transported when using presentations. Phase 2 involved recording the amount of data transmitted during the polling of users in the virtual room. We wanted to establish if the polling process introduced any significant increases on the amount of data transferred. The third phase involved

simulating the 7 users with randomly generated chat text to emulate students chatting in a virtual room. We recorded down the amount of data transferred during the chat session. Our air time cost calculations were based on the data rates published on Vodacom's website as of May 2010. We made use of Java's logging mechanisms, specifically tailored for mobile phones, to log the measurements. Microlog, an open-source logging library, is based on the well known Log4j API, but created from ground up with Java ME limitations in mind. We made use of this library, downloaded from <http://microlog.microsuite.org/>, by embedding it into the classpath of our mobile client application. We also implemented logs on the server. When the data was received on the mobile phone, we logged down the packets size too, using the Log4J mobile logging framework. Each of the next sections present sample results of two of the presentations, and then gives the summary of the recordings of all the 21 presentations.

5.3.1 Transporting presentations

Figure 5.13 shows example results recorded down from the Phase 1 experiments, which involved recording the size of slides in a presentations when received on the mobile phone client. Note that the slides received on the mobile phone are actually down-scaled images resulting from the conversion process discussed earlier

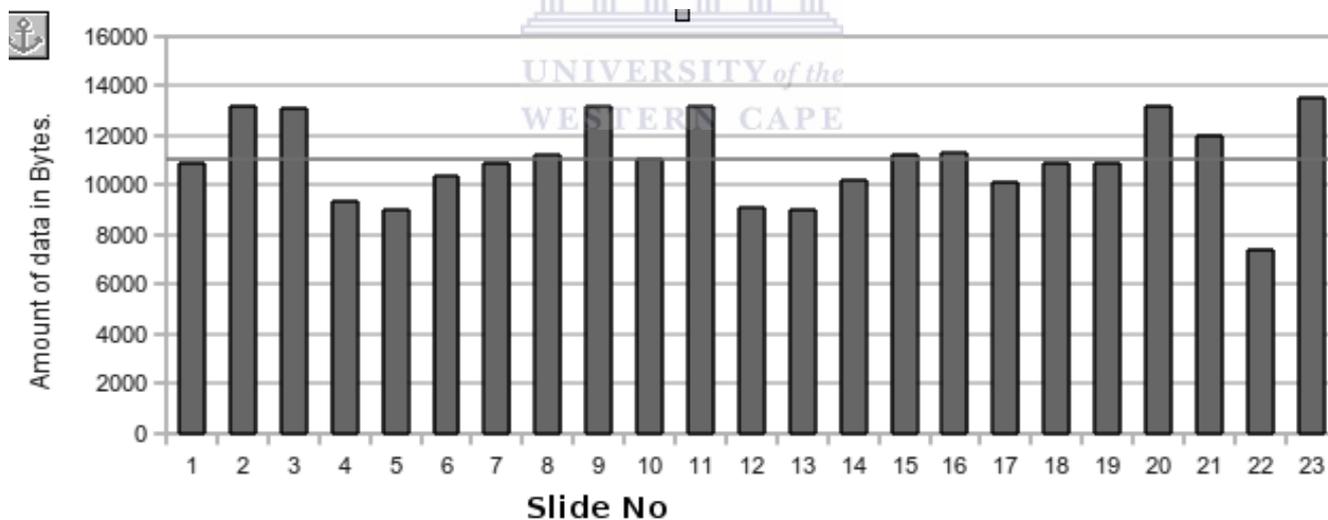


Figure 5.13: Amount of data transferred when using a presentation.

From the sample results in Figure 5.13, we are able to determine that an average of 11000 bytes per slide were transported through the mobile phone. 11000 bytes converts to 0.08392333984375 megabytes, rounded off to an average 0.084 megabytes per slide. Figure 5.14 presents with another example results of the amount of data recorded on the mobile client.

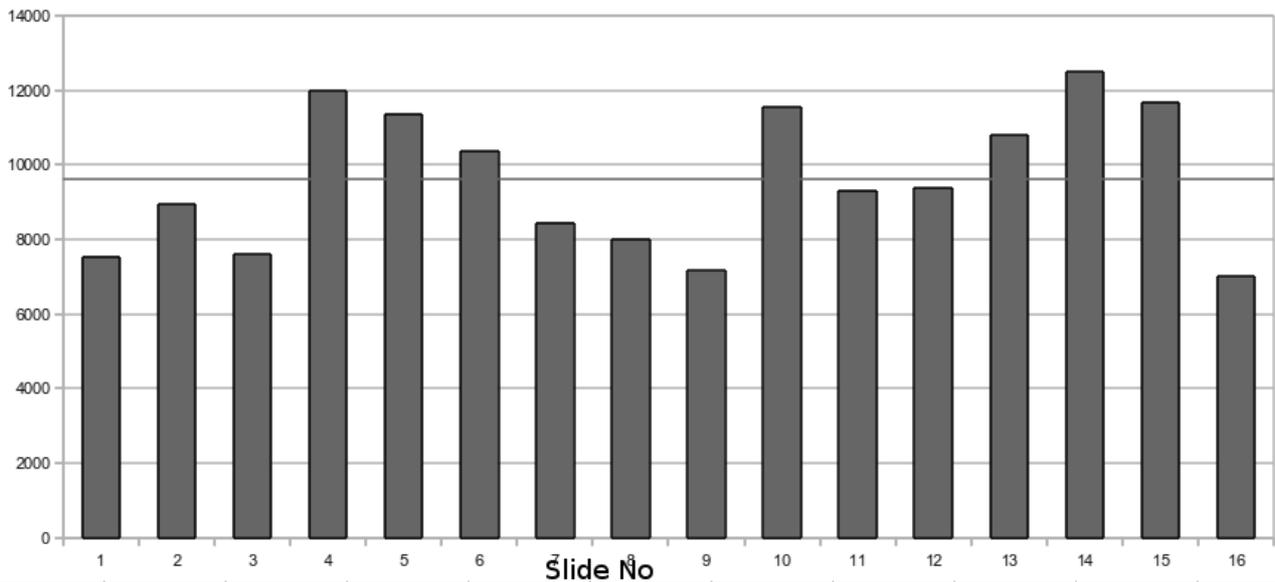
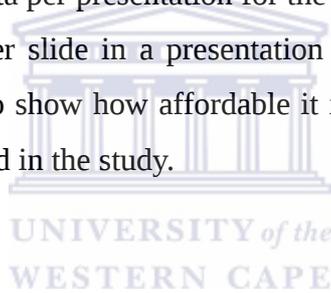


Figure 5.14: Amount of data transferred when using a second presentation

The average amount of data in bytes per slide from Figure 5.14 is 9600.375 bytes. This is equivalent to 0.0732450485229492 megabytes, rounded off to an average of 0.07 megabytes per slide. Figure 5.15 shows an average amount of data per presentation for the 21 presentations. From the mean plot line, the average amount of data per slide in a presentation is 9872.66 bytes. We will use these figures in the Discussions Section to show how affordable it is to use presentations when they are scaled using the techniques employed in the study.



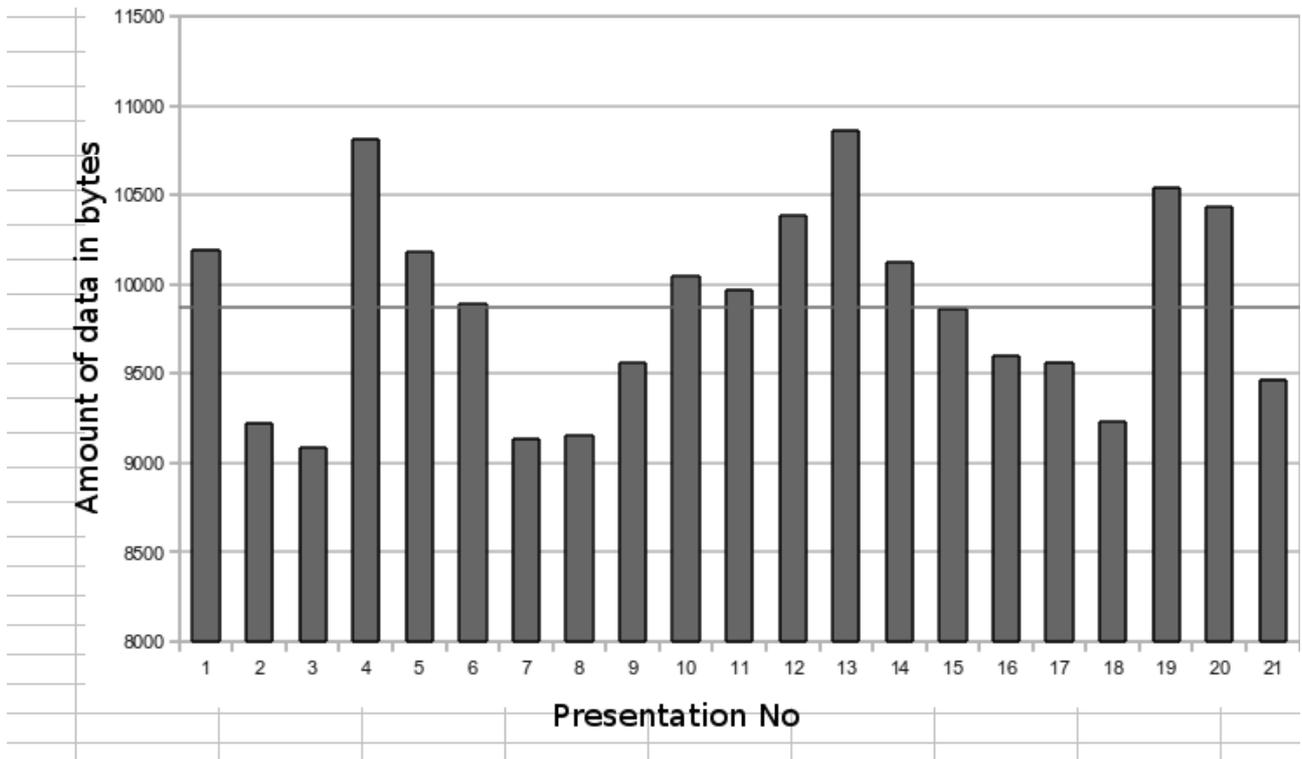


Figure 5.15: Average number of bytes per presentation recorded at mobile client.

5.3.2 Polling users

The next performance experiments, involving polling the server for the latest user list, were carried out with the purpose of determining the amount of data transported when the mobile client polls for users during an m-Learning session. We needed to include the results in calculating airtime cost of viewing the presentation, since the presentation viewer in the application we developed includes a userlist, necessary for text chatting. Since we did not use actual users for these experiments, we had to simulate 7 'robot' users. The 7 robot users were all connected at the same time during the tests. Table 5.3 shows sample results from one of the sessions when streaming a sample presentation. Note that the results in the third column are values of column 2 multiplied by 2, to cover the all round trip of sending request to the server and receiving a response.

Poll Id	Bytes transported	X2
1	38	76
2	41	82
3	43	86
4	36	72
5	39	78
6	33	66
7	40	80
8	35	70
9	29	58
10	43	86
11	34	68
12	32	64
13	40	80
14	32	64
15	38	76
Average	37.8	75.6

Table 5.3: Amount of data in bytes transported when polling users.

The table shows the results of amount of data transported in bytes when a mobile client polls the server for user list in an m-Learning session.

We plotted these results in a chart, shown in Figure 5.16. The mean plot line from the figure shows the average amount of data transported every time the mobile client polls the server is 74 bytes. Figure 5.17 presents a second sample results of polling users. The mean plot line from the figure shows the average amount data transported is 73.9375 bytes.

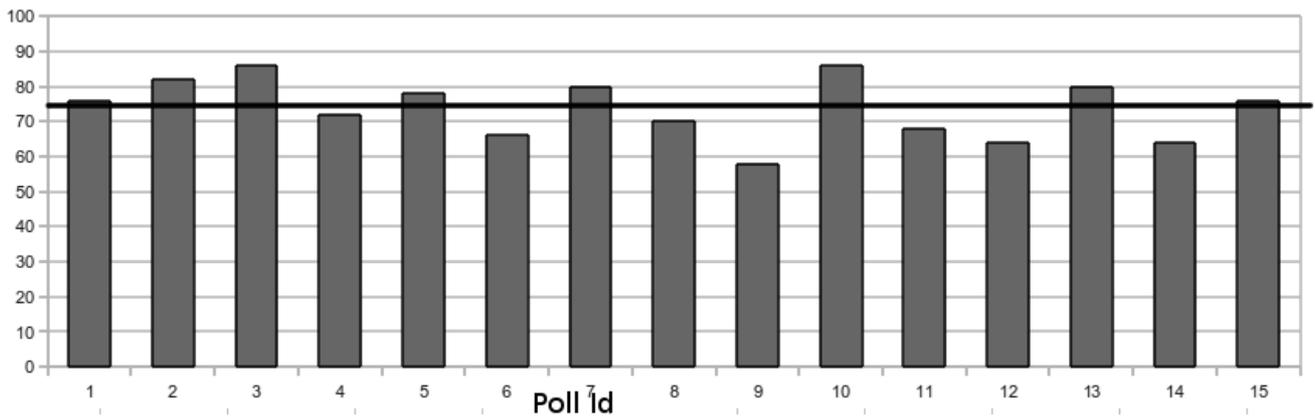


Figure 5.16: Amount data transported when polling users.

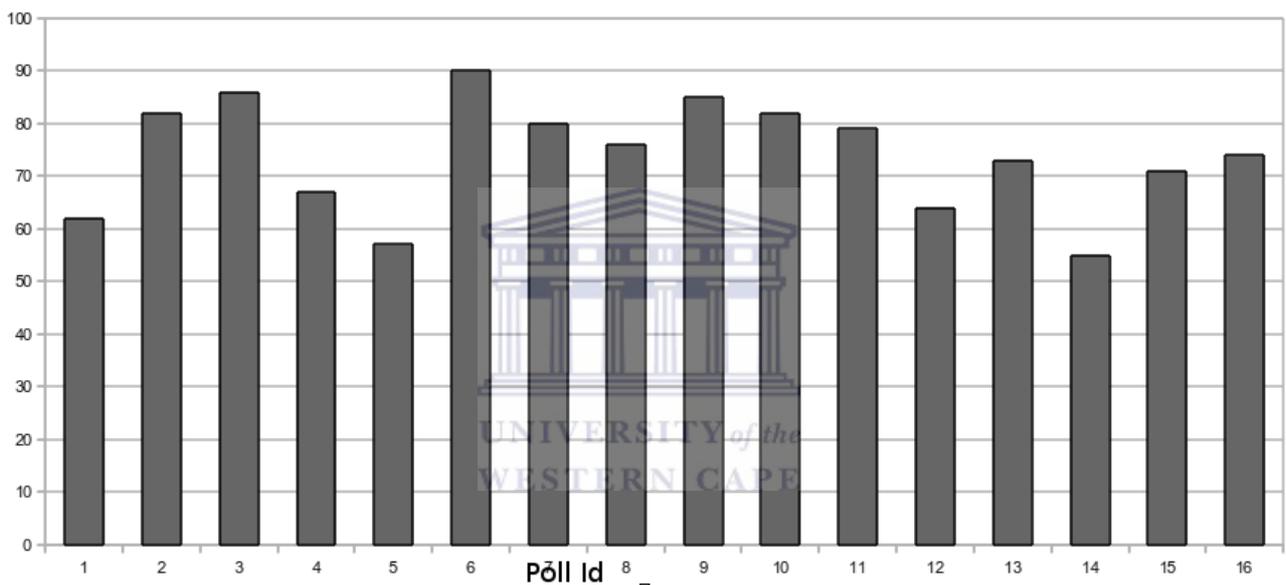


Figure 5.17: Second example result of amount of data transported to mobile phone.

Again, we shall use the observations recorded in this section to show how affordable it is to run an m-Learning application using based on our model.

5.3.3 Polling chat messages

The chat client was embedded into the presentation viewer on the mobile client to enable instant messaging between the mobile users and the instructor. Since all our experiments were confined to an academic laboratory, we did not use real users to carry out these experiments. Rather, we simulated the chatting process using 7 robotic users. All these users were connected at the same time during the experiments. Figure 5.18 and Figure 5.19 present sample results during a session when the chatting session was being simulated by making the 7 robotic users sent random text messages to each other.

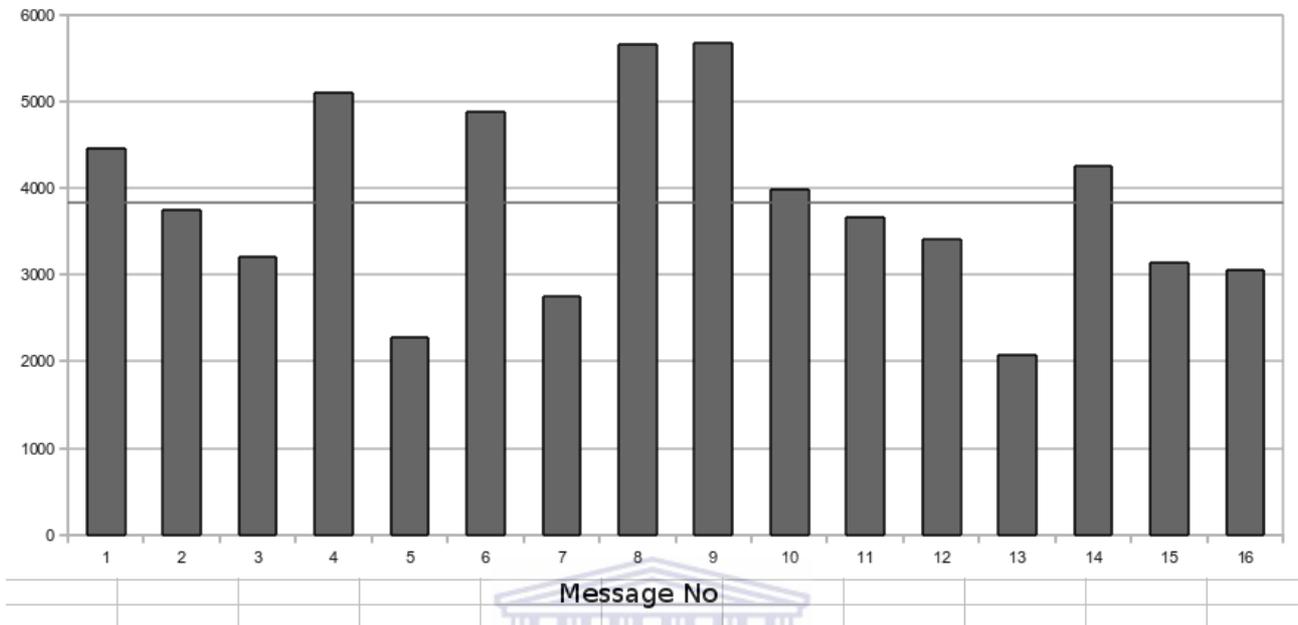


Figure 5.18: Data recorded when simulating 7 users chatting

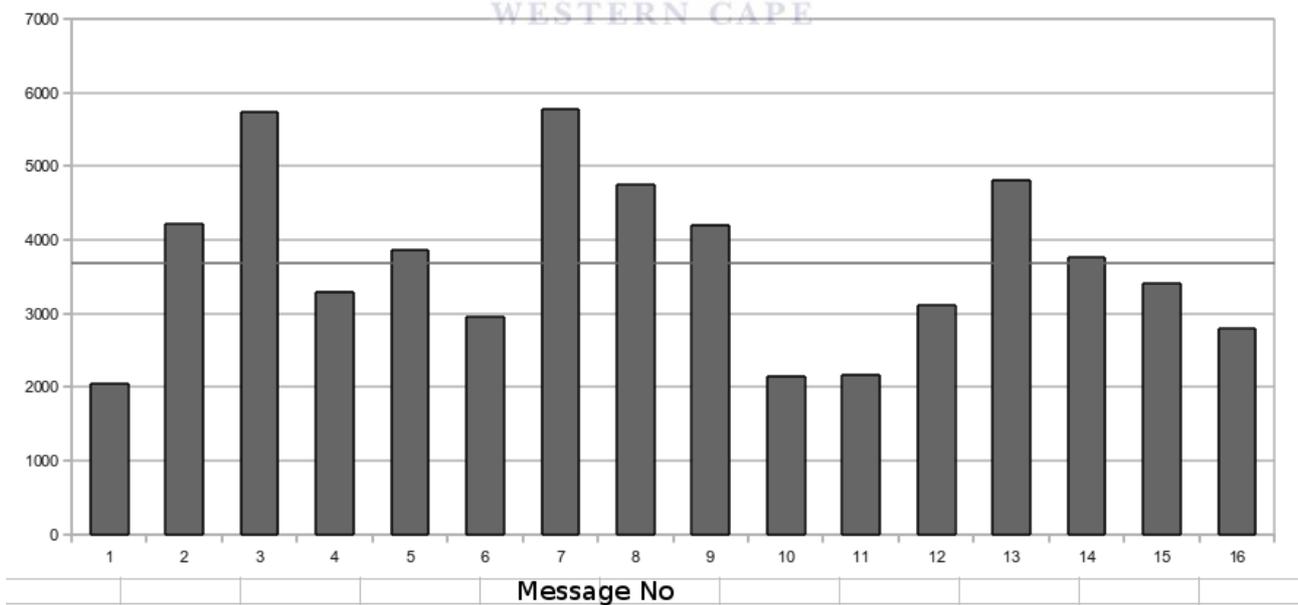


Figure 5.19: Second data recorded when 7 simulated users are chatting

From Figure 5.19, the average amount recorded is 3686.63 bytes, that translate to 0.028126754760742 megabytes, rounded off to 0.03 megabytes.

5.3.4 Average bytes of data transported in a session

Section 5.3.1, Section 5.3.2 and Section 5.3.3 presented us with example results average amount of data that is transported through mobile client. The most significant and comprehensive results are presented in Section 5.3.1, which measures the amount of data based on the slides transported. From our experiments with 21 presentations, we found that an average of 9872.66 bytes were transported per slide, converting to 0.08 megabytes per slide.

Bundle Option	MB/month	Monthly Subscription	In-bundle MB rate	Out of Bundle MB rate* (Bolt on)	Out of Bundle MB rate* (Prepaid/Top-Up)
MyMeg 8	8	R 9.25	R 1.16	R 2.00	R 2.00
MyMeg 30	30	R 28.00	R 0.93	R 2.00	R 2.00
MyMeg 110	110	R 88.00	R 0.80	R 2.00	R 2.00
MyMeg 175	175	R 119.00	R 0.68	R 2.00	R 2.00
MyMeg 300	300	R 139.00	R 0.46	R 1.20	R 2.00
MyMeg 600	600	R 189.00	R 0.32	R 1.20	R 2.00
MyGig 1.2	1229	R 289.00	R 0.24	R 1.20	R 2.00
MyGig 2.3	2355	R 389.00	R 0.17	R 1.20	R 2.00
MyGig 3	3072	R 589.00	R 0.19	R 1.00	R 2.00
MyGig 5	5120	R 989.00	R 0.19	R 1.00	R 2.00
MyGig 10	10240	R 1,989.00	R 0.19	R 0.50	R 2.00
MyGig 20	20480	R 3,899.00	R 0.19	R 0.45	R 2.00

Table 5.4: Vodacom South Africa data rates.

Based on Vodacom data rates presented in Table 5.4, we use MyMeg 300 in-bundle-data option which cost R0.46 per megabyte:

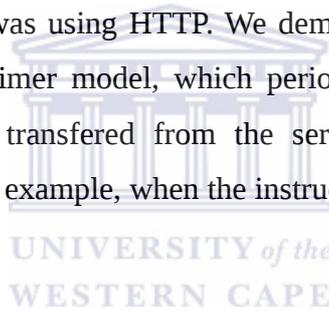
$$0.8 * 0.46 = R0.0368 \text{ cost per slide.}$$

While these figure are merely indicative, they demonstrate that it is quite affordable to implement an m-Learning solution that uses presentations to deliver rich content to users, provided proper techniques are used to format the presentations for mobile display.

5.4 Discussion

This chapter presented sample results of observations made when the application we developed was used to simulate sessions in which the presentations that were uploaded via desktop client to a

server. The presentations were then converted into a series of images, each image representing a snap-shot of a slide, then each of the images were scaled down to the specifications of the mobile-client, in realtime, before transporting them to the mobile device for display. We demonstrated that presentations (PowerPoint/OpenOffice) can be used as a medium for transporting rich e-Learning content to mobile users. We demonstrated that it is possible to stream a presentation to a mobile phone by converting slides in a presentation into individual images, which are then scaled down based on mobile phone screen dimensions before transmission. For best viewing on a mobile phone, the scaled down image should have a *minimum* mean image histogram value of 128. Our experiments also demonstrated that it is possible to take less than 100 milliseconds, on average, to scale down an image corresponding to a slide, a process fast enough to allow near-realtime transportation of slides to a mobile client. Experiments on air time costs demonstrated that the software developed uses minimal data when streaming content to users, making it an affordable option. The air time costing was based on South African Vodacom 3G network, which bills based on the amount of data streamed, and not how long a user has been online. All the communication between mobile client and server was using HTTP. We demonstrated that HTTP can be used by employing models such as a task-timer model, which periodically polls the server, and can be economical, because data is only transferred from the server to the mobile client when the instructor's presentation changes, for example, when the instructor advances a slide.



6 Conclusion and future work

In this chapter, we present an overview of the thesis, summarizing the experiments we carried out, the observations we made and the conclusion. We end the chapter by recommending future work.

6.1 Overview of the thesis

This thesis addresses how presentations can be prepared, transported and displayed on a mobile phone. It also addresses how an interactive text chat can be used in conjunction with a presentation in the same environment. The use of presentations in an m-Learning environment enables delivery of rich content from an instructor to a mobile phone learner. Use of instant messaging text chat enables near-realtime interactivity.

Chapter 2 discussed related work, and detailed the various approaches that have been used by other researchers in implementing m-Learning applications. The chapter focused on existing text-based m-Learning applications, podcast m-Learning applications and rich Internet applications for m-Learning. We discussed how the applications transmit and present content to the users. The chapter examined the contexts in which these applications were used and concluded by highlighting constraints of text-based techniques and podcasting in m-Learning environments.

Chapter 3 revisited the applications discussed in Chapter 2. The main focus in this Chapter was the shortcomings of each of the applications, and the challenges they face as a result of the technologies they use. Our analysis of text/podcast-based m-Learning applications revealed limited interactivity and lack of diversity in content streamed. Based on the challenges, we presented the motivation behind this research. The main research question and the corresponding secondary research questions were introduced in this chapter. Our main research question was:

“How do we provide presentations and interactive chat to support m-Learning?”

We answered this question by introducing a technique that enabled preparation of the presentations by scaling down presentations, formatting them according to a mobile phone screen specifications and transporting them to the device for display. We also introduced XMPP-based instant messaging that provided near-realtime text chatting functionality. The text chatting takes place in a virtual room, where all users receive chat broadcasts. We developed a proof-of-concept application that formats presentations created on desktop computers and transports and displays them on a mobile phone, by converting the slides in a presentation into a series of images. The software includes and embedded instant messaging text chat system. We used the software to carry out a series of experiments which we described in Section 3.3. All the experiments were conducted in a laboratory.

We developed secondary research questions to concentrate on specific sub-domains of the research. Our first secondary research question covered the actual process of preparing slides by formatting them on a desktop computer before transporting them to a mobile phone for mobile display:

“How do we scale down presentations, thus reducing the file size, for mobile device display?”

We answered this question by converting slides in a presentation into a series of images, which were then scaled down to a format and size appropriate for mobile display, before transporting them to the mobile device. We used laboratory experiments to investigate the use of Openoffice and PowerPoint presentations to achieve this aim. Software libraries that convert slides in these presentations into a series of images were used. The resulting images were then scaled down based on mobile phone specification and then transported to a mobile phone. Experiments were carried out with a smart mobile phone running on a third generation cellular network. We employed transaction-logging techniques in addition to automated image analysis techniques to observe and record data. The second secondary research question was about the quality of the slides on a mobile phone:

“How do we measure the quality of presentation slides when displayed on mobile phone?”

Because an average mobile phone has inherent limitations, the main obvious one being the tiny screen size, it was important to investigate the quality of slides presented on such screens. We used image histograms to measure the quality of slides formatted for mobile phone display. Our third research question covered the air-time costs:

“How do we minimize airtime costs when sending presentations and conducting interactive chat on a mobile phone?”

This question was answered by implementing an algorithm that allows the mobile client to pull the data from the server only when there has been an update from the instructor's screen. The polling process takes place in three separate threads: the first thread checks the status of slides from the instructor and updates the mobile client accordingly. The second thread updates the user list. The third and most important thread with respect to the above research question was the chat polling. This thread refreshes the chat transcript as long as there are new messages. Although we used a fixed-interval polling model, the actual polling process does not introduce any significant air time cost element.

Chapter 4 covered the design and implementation of the software we developed for the experiments. The chapter examined the user requirements and target audience of the m-Learning applications and continued to report the design structures of each of the three components that make

up the software: desktop client, server and mobile phone client.

Chapter 5 explained the experimental environment, presented and analyzed the data collected during the experiments in appropriate formats, including graphs and pie charts.

6.2 Scope and Limitations

We limited our study to use of OpenOffice/PowerPoint presentations and text chat as a medium for transporting content between instructors and learners in an m-Learning environment. The use of OpenOffice/PowerPoint formats enabled us to experience wider diversity with the content transported, compared to SMS and podcasting. This effectively enabled us to use of rich text and graphics in the content. We introduced interactivity by embedding an instant messaging text-based chat service. While it would have been desirable to introduce realtime interaction using audio and possibly video as the additional medium of interaction, the use of such technologies was beyond the scope of our research in the mobile phone context.

6.3 Future work

Our research builds a good platform for future research on how to format presentation in m-Learning applications for Computer Science students at postgraduate level. The following paragraphs highlight possible research avenues for future work.

The duration of the study could not allow us to research on possible ways of including realtime audio and video streaming as a medium of interactive communication in the m-Learning infrastructure we developed. Future researchers could focus on these technologies and include them in the framework we built, as possible alternatives for interaction.

The software we developed uses image-histograms to determine the quality of slides in a presentation after conversion. It would be desirable for future researchers to employ other techniques for accomplishing this process.

The current version the software does not address accessibility: it does not allow mobile users to zoom in or out of the screen when viewing a presentation. This feature is important as it would allow a user to zoom into a presentation to for better view of non-clear text and figures. Such feature would also be useful for users who are visually challenged. Future researchers could focus on this feature to improve on mobile phone presentation technology in m-Learning applications.

The software we developed connects to the server by sending messages in clear text even during username/password authentication. This amounts to a security risk if someone snoops on the

data being exchanged. We considered *bouncycastle*, an excellent encryption software that can be optimized to run on the cellphone. While our efforts were successful, the final client file size was too large for most mobile distribution. Future researchers could study on best way of implementing lighter encryption solutions for m-Learning applications.

The mobile client currently uses HTTP as the transmission protocol. This works fine in most networks because of the wide adaption of HTTP. However, this in itself introduces another challenge. Using HTTP meant that the developer could not achieve complete realtime streaming between mobile client and the server since HTTP is a request-response protocol. Future researchers can implement a communication protocol that can be widely adapted and support realtime communication in m-Learning environment.



References

- [1] D. Zhang, L. Zhao, L. Zhou, and J. Jay F. Nunamaker, "Can e-learning replace classroom learning?," *Communications of the ACM*, vol. 47, 2004, pp. 75 - 79.
- [2] J. Korhonen, *Introduction to 3G mobile communications*, Artech House Inc, 2003.
- [3] S. Wang and M. Higgins, "Mobile 2.0 Leads to a Transformation in mLearning," *Lecture Notes in Computer Science*, vol. 5169/2008, 2008, pp. 225-237.
- [4] "LiNE Zine - mLearning: Mobile, Wireless, In-Your-Pocket Learning,"
<http://www.linezine.com/2.1/features/cqmmwiyp.htm>, Accessed 2009-07-27 00:17:15.
- [5] C. Sperberg-McQueen, E. Maler, F. Yergeau, J. Paoli, and T. Bray, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation 04 February 2004
(<http://www.w3.org/TR/REC-xml>), 2009.
- [6] T. Rist and P. Brandmeier, "Customizing Graphics for Tiny Displays of Mobile Devices,"
Personal and Ubiquitous Computing, vol. 6, 2002, pp. 260 - 268.
- [7] C. Peersman, S. Cvetkovic, P. Griffiths, and H. Spear, "The Global System for Mobile Communications Short Message Service," *IEEE Personal Communications*, vol. 7, 2000, pp. 15-23.
- [8] D. Wen, J. Xiong, M. Ally, and F. Lin, "An SMS Based Querying System for Mobile Learning,"
mLearn 2006, the 5th World Conference on Mobile Learning, 2007.
- [9] B. Ramadoss and S. Balasundaram, "SMS for Question-Answering in the m-Learning Scenario," *Journal of Computer Science*, vol. 3, 2007, pp. 119-121.
- [10] J. Tangkuampien, "Kids, Education, and Cellular Handsets," *Interactions*, vol. 16, 2009, pp. 14 -16.
- [11] G. Lightbody, J. Hughes, M. Hutchison, and P. McCullagh, "The Use of Audio Podcasts to enhance the Delivery of a Computer Networks Course," *8th Annual Conference of the Subject Centre for Information & Computer Sciences*, 2007.
- [12] A. Holzinger, A. Nischelwitzer, and M. Meisenberger, "Mobile phones as a challenge for m-learning: examples for mobile interactive learning objects (MILOs)," *Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05)*, 2005, pp. 307-311.
- [13] A. Holzinger and M. Ebner, "Visualization, Animation and Simulation for Mobile Computers: Experiences from Prototypes," *Proceedings of the Central European Multimedia and Virtual Reality Conference*, 2005, pp. 37 - 41.
- [14] B. Taraghi, H. Mühlburger, M. Ebner, and W. Nagler, "Will Personal Learning Environments Become Ubiquitous through the Use of Widgets?," *Proceedings of I-KNOW '09 and I-SEMANTICS '09*, 2009, pp. 329-335.
- [15] C. Houser and P. Thornton, "Poodle: a course-management system for mobile phones," *IEEE International Workshop on Wireless and Mobile Technologies in Education, 2005. WMTE 2005*, 2005.
- [16] A. Alsadi and B. AbuShawar, "m-Learning: The Usage of WAP Technology in e-Learning," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 3, 2009, pp. 10-16.

- [17] H. Inanoglu, J. Reece, and M. Bilgic, "General Packet Radio Service (GPRS)," *The International Series in Engineering and Computer Science*, vol. 558 Part I, 2002, pp. 197 - 213.
- [18] J. Postel, "Transmission Control Protocol," *RFC-793*, 1981.
- [19] J. Postel, "User Datagram Protocol," *RFC-768*, 1980.
- [20] A. Rao, H. Schulzrinne, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," *RFC-2326*, 1998.
- [21] H. Schulzrinne, R. Frederick, S. Casner, and V. Jacobson, "RTP: Transport Protocol for Real-Time Applications," *RFC-3550*, 2003.
- [22] A. Rantalaihti, "Mobile Media API -JSR 135, Available at <http://jcp.org/en/jsr/detail?id=135> [Accessed September 5, 2009]," 2003.
- [23] C. Erlandson and P. Ocklind, "WAP - The wireless application protocol, Technical report, Ericson Review No 4, Available at <http://sysdoc.doors.ch/ERICSSON/1998041.pdf> [Accessed November 15, 2009]," 1998.
- [24] I. Elsen, F. Hartung, U. Horn, M. Kampmann, and L. Peters, "Streaming Technology in 3G Mobile Communication Systems," *Computer*, vol. 34, issue , 2001, pp. 46 - 52.
- [25] C. Blanchard, "Security for the Third Generation (3G) Mobile System," *Technical document, Network Systems & Security Technologies, Available at http://www.isrc.rhul.ac.uk/useca/OtherPublications/3G_UMTS%20Security.pdf* [Accessed September 19 2010], 2002.
- [26] D. Sönnerstam, *Bluetooth Specifications, Profiles of the Bluetooth System, Available at http://grouper.ieee.org/groups/802/15/Bluetooth/profile_10_b.pdf* [Accessed September 5, 2009], 1999.
- [27] JSR, *Java Specification Request 82: Java API for BlueTooth, Available at Java Community Process (<http://jcp.org/en/jsr/detail?id=82>)*, Java Community Process (<http://jcp.org/en/jsr/detail?id=82>), 2009.
- [28] Y. Zhang, S. Zhang, S. Vuong, and K. Malik, "Mobile learning with bluetooth-based E-learning system," *2nd International Conference on Mobile Technology, Applications and Systems*, 2005.
- [29] A. Miliarakis, E. Malliou, M. Stratakis, S. Stavros, and S. Sotiriou, *The AD-HOC Project: eLearning Anywhere, Anytime*.
- [30] M. Unser and T. Blu, "Image Interpolation and Resampling," *Handbook of Medical Imaging, Processing and Analysis*, 2000.
- [31] L. Chen, X. Xie, X. Fan, W. Ma, H. Zhang, and H. Zhou, "A visual attention model for adapting images on small displays," *Multimedia Systems*, vol. 9, 2003, pp. 353-364.
- [32] S. Jiang, H. Liu, Z. Zhao, Q. Huang, and W. Gao, "Generating Video Sequence from Photo Image for Mobile Screens by Content Analysis," *2007 IEEE International Conference on Multimedia and Expo*, 2007, pp. 1475 - 1478.
- [33] J. Hall, *Assessing Learning Management Systems, Feature Article for Chief Learning Officer, Available at http://pttmedia.com/newmedia_knowhow/KnowHow_Deploy/LMS/Docs/Assessing_LMS.doc*

[Accessed September 5, 2009], 2003.

- [34] B. Hopkins, "Comparing Mobile Platforms: Java ME and Adobe Flash Lite," *Mobility Tech Tips*, 2008.
- [35] D. Nourie and J. Knudsen, *Wireless Development Tutorial Part I*, Sun Developer Network, Available at <http://developers.sun.com/mobility/midp/articles/wtoolkit/> [Accessed October 17, 2009], Sun Developer Network, 2006.
- [36] B. Wilhem and Mark Burge, "Digital image processing," *an algorithmic introduction using Java*, Springer Science Business Media, 2008, pp. 37 -51.

