Innovative Mobile Learning: Techniques and Technologies

Hokyoung Ryu
Massey University, New Zealand

David Parsons
Massey University, New Zealand

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE
Hershey · New York

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

and in the United Kingdom by
Information Science Reference (an imprint of IGI Global)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanbookstore.com

Library of Congress Cataloging-in-Publication Data

Innovative mobile learning : techniques and technologies / Hokyoung Ryu and David Parsons, editor.

p. cm.

Includes bibliographical references and index.

Summary: "This book includes the challenges and practical experience of the design of M-Learning environments, covering current developments in M-learning experiences in both academia and industry"--Provided by publisher.

ISBN 978-1-60566-062-2 (hardcover) -- ISBN 978-1-60566-063-9 (ebook)

1. Mobile communication systems in education. I. Ryu, Hokyoung. II. Parsons, David, 1959 Oct. 13-

LB1044.84.I56 2009

658.3'1240402854678--dc22

2008010308

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter XVI
Assessing the Benefits of AJAX in Mobile Learning Systems Design

Feng Xie
Massey University, New Zealand

David Parsons
Massey University, New Zealand

ABSTRACT

Mobile technologies are rapidly changing our lives with increasing numbers of services supported by mobile devices, including Web-based learning applications, providing opportunities for people to study anytime and anywhere. However, using Web-based mobile applications to present learning resources is a challenge for developers because the performance of the mobile Internet over GPRS networks is often unacceptably slow. A new Web development model, Ajax, may help to address this problem. Ajax (asynchronous JavaScript and XML), is an approach to Web application development that uses client-side scripting to reduce traffic between client and server and provide a seamless user application experience. In this chapter, we address the question of whether mobile Ajax provides measurable performance advantages over non-Ajax mobile learning applications. An empirical study was undertaken to measure mobile learning application performance over a GPRS network, comparing an Ajax application and an active server pages (ASP) application with identical functionality. Our results suggest that mobile Ajax can reduce the bandwidth requirement by around 70 percent, and cut the server's response time in half. In addition, these performance improvements were noticed by users in our small group usability test.

INTRODUCTION

We live in an information society where learning becomes ever more important, and not all of this learning can take place in a static environment. The mobile revolution is changing our lives and can also facilitate new learning processes, but learners need educational services with a fast response speed and good user interaction if mobile learning systems are to be readily adopted.

An increasing number of people have been using mobile Internet access through wireless networks (Church, Smyth, Cotter, & Bradley, 2007), but due to the limitations of the mobile communications infrastructure and hardware this can still be very problematic. There are a number of reasons, for example: small device screens, high network latency, low bandwidth and interaction complexity (Chakravorty & Pratt, 2002). Such weaknesses can impede the mobile learning process.

The poor performance of commonly used wireless networks such as GSM (global system for mobile communications) and its associated digital packet switched data service, GPRS (general packet radio service) is a major problem for mobile learning systems. There are various reasons for this poor performance, for example high and variable latency, fluctuating bandwidth, occasional link 'blackouts' (Chakravorty, Cartwright & Pratt, 2002), packet loss, and link outages. Sometimes, even simple requests can lead to long delays (Stuckmann, Ehlers & Wouters, 2002). However, despite these problems we should recognise that one distinct feature of mobile learning over other learning activities is *mobility* (Leung & Chan, 2003). The GPRS mobile phone network is the most commonly used network in the world, with the widest coverage, considerably larger globally than 3G (third generation) wireless networks. Using this network for mobile learning can provide services anytime almost anywhere around the world, with extensive international roaming. In addition, although 3G was intended to resolve technological fragmentation in the wireless communications market, this has not happened in practice and there are several competing 3G technologies. Cost is also an important factor, with a large number of low cost GSM/GPRS devices on the market, and in many territories the GSM/GPRS service fee is cheaper than 3G. Therefore, despite their limitations, we will need to continue to work with GPRS systems for some time to come.

Mobile Learning Technical Challenges

The combination of wireless telecommunications and mobile computing is resulting in a transformation of the educational landscape (Alexander, 2004). The growth and rapid evolution of wireless technology have created new opportunities for the 'anytime and anywhere' learning paradigm (Seong, 2006) that is mobile learning.

Various researchers have defined mobile learning (m-learning) in different ways. Pinkert et al. (2003) define it as e-learning that uses mobile devices and wireless transmission. Polsani (2003) defines it as a form of education whose site of production, circulation, and consumption is the network. Traxler (2005) defines m-learning as any educational provision where the sole or dominant technologies are handheld or palmtop devices. Sharples (2005) defines it as a process of coming to know, by which learners in cooperation with their peers and teachers, construct transiently stable interpretations of their world.

Regardless, mobile learning is different from our traditional learning experience; it has its own problems and limitations. Mobile learners may feel uncomfortable because they cannot have face-to-face interaction with teachers or other students (Stodel, Thompson, & MacDonald, 2006). There are also limitations on what it can deliver. Berri, Benlamri & Atif (2006) describe it as mainly a time-constrained exercise with lightweight content-oriented instruction.

Traxler (2007) categorizes previous mobile learning studies into 6 aspects:

- Technology-driven
- Miniature but portable e-Learning
- Connected classroom learning
- Informal, personalized, situated mobile learning
- Mobile training / performance support
- Remote / rural / development mobile learning

From a design process point of view, these aspects fall into three major categories: pedagogical learning models, adapting to mobile environments, and technical challenges (Berri et al., 2006). Currently, most mobile learning studies focus on pedagogical learning models, or mobile environments. In contrast, this study focuses on technical challenges.

Despite extensive research into mobile learning systems, technical support for Web-based mobile learning has been explored in a relatively small number of studies. On example is a mobile learning system using Instant Messaging and wireless networks (GSM/GPRS and 802.11 Wi-Fi), developed by Kadirie (2007). This system is Web-based, and the kernel of the instant messaging server is written in Java. Learners use a mobile Web browser to login to this learning system to chat with other learners or view their messages. Nakahara, Hisamatsu, Yaegashi, and Yamauchi (2005) describe a Web-based collaborative learning site with a bulletin board system (BBS) that allows learners to interact, exchange information, engage in discussion, and collaborate on projects. Another technical paper is by Seong (2006). He developed a mobile learning course manager portal to demonstrate and exemplify the usability guidelines proposed. Kukka and Ojala (2006) developed a Java and XML based learning system for both desktop and mobile devices. Other technical examples include Cao, Tin, McGreal, et al. (2006), Lee and Lu (2003) and Theng (2007). These examples all use Web-based technologies plus a mobile Web browser or mobile application as the mobile learning client.

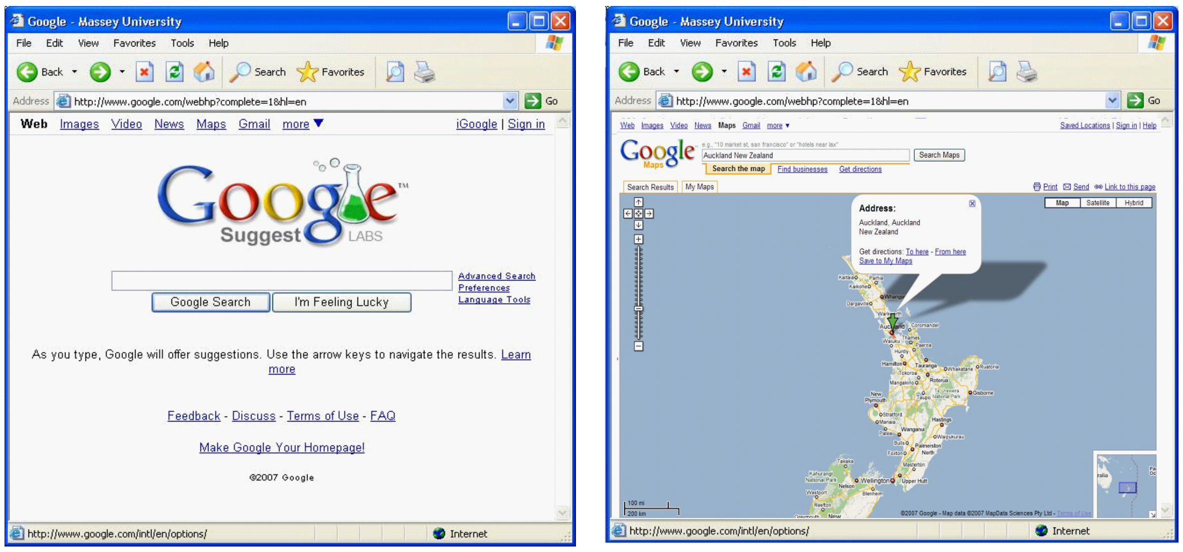
These Web-based mobile learning systems use a traditional synchronous client-server application architecture, typically using the GSM/GPRS network, but in most cases Web application performance over a high latency GPRS network is poor (Chakravorty et al., 2002; Chakravorty & Pratt, 2002; Stuckmann et al., 2002). One issue is that the TCP/IP Internet protocol needs to initialize before it starts to transfer actual data; this process takes over 7 seconds to enable the connection when the initial request is made (Chakravorty, Clark, & Pratt, 2003). In addition all wireless networks have high latency; GPRS link latency is 600ms-3,000ms for the downlink and 400ms-1,300ms on the uplink. Round-trip latencies are therefore are least 1,000ms (Chakravorty et al., 2002). In a lab environment, download times for CNN's Web site over GPRS was between 125 and 170 seconds (Chakravorty & Pratt, 2002). This latency does not improve with increased bandwidth, and deploying a 3G mobile phone network is not necessarily helpful (Hunaiti, Garaj, Balachandran & Cecelja, 2005), for example the TCP initialization time on GPRS is 5-7 seconds, but the TCP initialization time on 3GSM is up to 12-15 seconds (Nortel, 2007).

This poor network performance is compounded by the synchronous request/response model of traditional Web applications. Therefore to improve mobile learning system performance we might use a higher performance network, where one is available, or perhaps apply some new approach to the architecture of the Web-based mobile learning system itself.

Ajax Web Applications

To address performance issues in Web-based mobile learning applications, we may look to recent developments in desktop browser technologies that might be implemented in the mobile environ-

Figure 1. The pioneers of the Ajax approach (Google Suggest & Google Maps)



As you type, Google will offer suggestions. Use the arrow keys to navigate the results. Learn more

ment. In 2005, Garrett introduced the concept of Ajax (Asynchronous JavaScript and XML) (Garrett, 2005). Since then, an increasing number of Web applications have used the Ajax approach (including learning systems—see chapter VIII of this book). Although Garrett's article was the first to use the term 'Ajax,' the technique he described was already being used by many Web applications. The pioneer developers of this technique were Google, with Google Suggest and Google Maps being two early examples of this approach (Figure 1), though some of the key technologies (such as the XMLHttpRequest object) were first developed by Microsoft.

Ajax was originally introduced as a desktop Web programming model, not a mobile environment solution, but because of its key characteristics (small transmission volumes, asynchronous communication and partial Web page updates), it can actually be very useful in the mobile environment, especially in high latency networks where Ajax can reduce the frequency and volume of data transfer.

The traditional Web application allows users to submit a request, which the server will process and respond to. Then the client browser will refresh the whole Web page, even if only a small part of the content is changed. In most cases, however, the new Web page will be very similar to the old one, which means that during these transmissions some duplicate content is transferred. This can waste both network resources and users' time.

Ajax uses a browser hosted 'Ajax engine' to handle both data transmission and partial updates to the Web page. The Ajax engine only requests new content from the server, reducing unnecessary data transmission. It can also partially update the current Web page when the response is received. Because the Ajax engine runs locally in the client browser, the Web application's response speed is faster and the user's experience is improved. In addition, client server communication in Ajax can be carried out asynchronously, enabling the user to continue interacting with the system even while the browser is waiting for data from the server. Compared with the traditional Web application model, Ajax can significantly increase a Web application's performance in the desktop browser environment (Smullen & Smullen, 2006, 2007; White, 2005).

Applying Ajax to a Mobile Learning Application

Previous studies have shown that Ajax reduces the data transmission volumes between the server and client device, and improves the user experience on the desktop. These are particularly relevant issues for Web based mobile learning systems, which tend to be content rich and require extensive client server interaction. Such systems can surely benefit from a strategy to reduce transmission volumes in the context of expensive and low speed connections, and provide a better user experience in browsers with limited screen real estate and navigation tools. With the increasing sophistication of mobile browsers, there are many mobile devices that can easily support Ajax applications. Therefore, we might ask the question, if we apply Ajax to a mobile learning environment, what might be the result? Does an Ajax mobile learning application provide measurable advantages over a non-Ajax mobile learning application?

In this chapter, we describe the development and analysis of a mobile learning system built using Ajax and tested in a GPRS mobile environment. We evaluate the system in terms of both measured performance against a non-Ajax system and usability testing, and assess what benefits we might gain from this approach.

We developed two mobile learning systems with identical functionality, one using Ajax and the other using a more traditional active server pages (ASP) architecture. These two Web applications have the same user interface and can both be accessed by commonly available mobile browsers. We measured the performance of these two applications over a GPRS network based on the data collected from the Web server's log files. We also carried out a small group usability test to assess the perceived benefits of Ajax over a non-Ajax system.

To perform our experiments we needed to implement a representative mobile learning system that was structured enough to enable effective performance measurement. To meet these requirements we chose to implement a mobile quiz as the learning content of the applications. Quizzes are popular components of learning systems, which can help the learner to improve their personal knowledge and problem solving ability (Yokomoto, 2000). They force the learner to think about every question, answer them all carefully, and review all answers at the end. A good quiz-based learning system needs a responsive user interface to give rapid feedback. This requirement can be easily met in a desktop environment, but it is not so easy in a mobile Web-based system. Nevertheless there have been some examples of using quizzes in mobile learning systems (Black & Hawkes 2006; Bar et al., 2007; Seong, 2006). Therefore we concluded that a mobile quiz was both an appropriate and effective example to use for our experiments.

WEB APPLICATIONS, AJAX AND MOBILITY

Since the first Web page built by Tim Berners-Lee in 1991 (Lee, 1992; Watson, Rainer & Koh, 1991), a synchronous request and response cycle has been used for Web access, with a 'click, wait and refresh' action approach (Figure 2) where the user needs to click to send a request and wait until the browser refreshes with a new page (Wei, 2005).

In this Web interaction model the user must wait for a request to be converted into a data stream, sent by HTTP over the Internet, processed on the server, and have its response returned by HTTP to be displayed in the browser (Crane, Pascarello & James, 2005). The user's waiting

Figure 2. Classic Web application model: Full page refresh and synchronous communication (adapted from Wei, 2005)

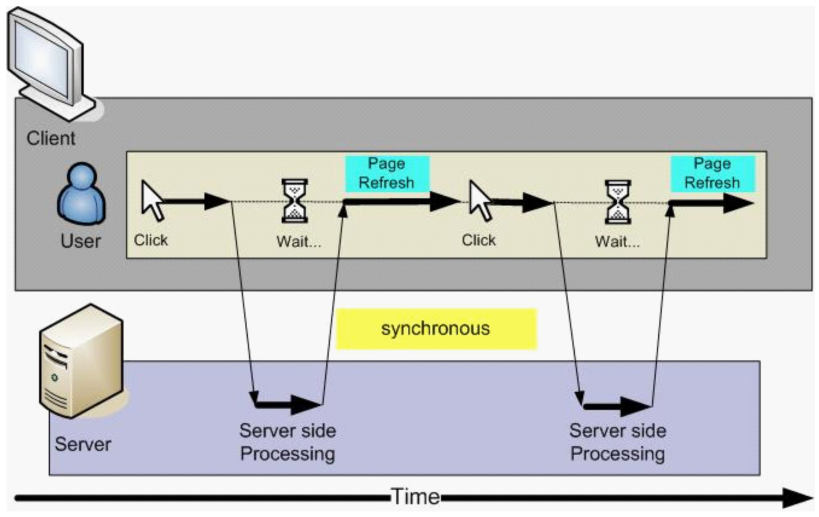
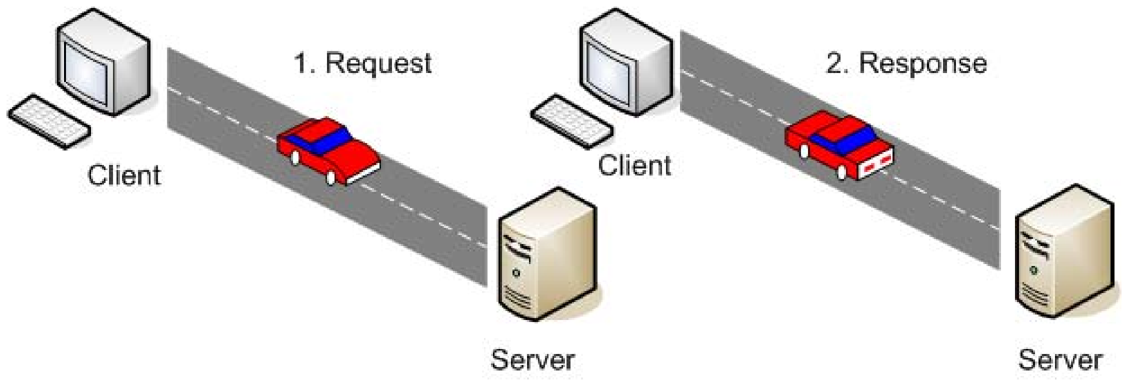


Figure 3. The synchronous request and response model



time includes network transition time, server response time and the browser's display time. The problem is that users cannot do anything when they are waiting, and this waiting time can be up to minutes depending on the network status. The 'click, wait and refresh' approach makes a lot of technical sense, because it makes it easy for Web developers to build applications, but it is not efficient or good for the user's experience. It forces the user to act like a machine; input, wait for the process to finish, and finally get the output.

The traditional Web application communication model is based on synchronous requests and responses. The client initiates a request and the server responds to it. The communication is always initiated one way, from the client to the server. During communication, users cannot do anything; they have to wait until the server responds to the request. This approach is like a car running on a freeway connected between the client and the server. The freeway allows two lines of traffic to travel at the same time, but due to the approach we are using, at any point in time only one car is actually running on the freeway (Figure 3). The traditional Web application cannot therefore provide a seamless user experience.

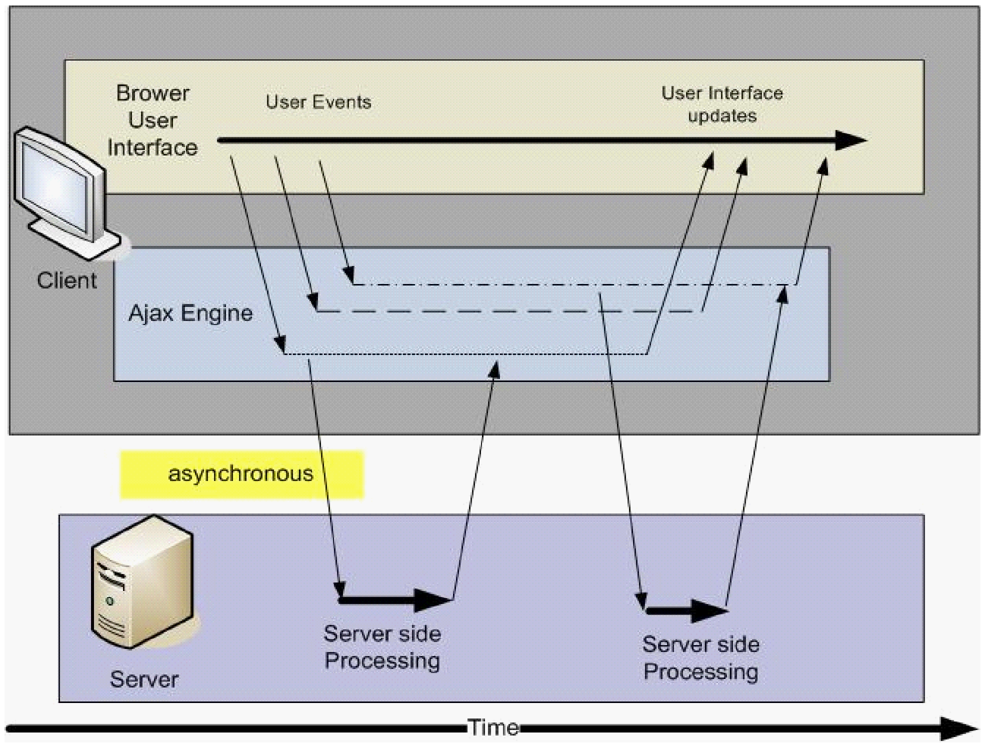
According to Watson et al. (1991), most users will run out of patience after six seconds, so we need some way of avoiding such delays. Under the limitations of network bandwidth and latency, the only way to improve Web application performance is to break down the synchronous model, and jump off the 'click, wait and refresh' cycle. One approach to this problem is Ajax (asynchronous JavaScript and XML).

Asynchronous JavaScript and XML (Ajax)

Ajax is a Web development technique for creating interactive Web applications. It is not a single new technology but combines a set of powerful, widely-used, well-known and mature technologies, mostly hosted by the client browser and largely independent of the server. The intent of Ajax is to make Web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire Web page does not have to be reloaded each time the user makes small changes. This increases the Web page's interactivity, speed, and usability.

Garrett (2005) first defined Ajax as a combination of:

Figure 4. Asynchronous communication and partial user interface updates (adapted from Wei, 2005)



- Standards-based presentation using extensible hypertext markup language (XHTML) and cascading style sheets (CSS);
- Dynamic display and interaction using the document object model (DOM);
- Data interchange and manipulation using the XML and extensible stylesheet language transformations (XSLT);
- Asynchronous data retrieval using the XMLHttpRequest object;
- JavaScript binding everything together.

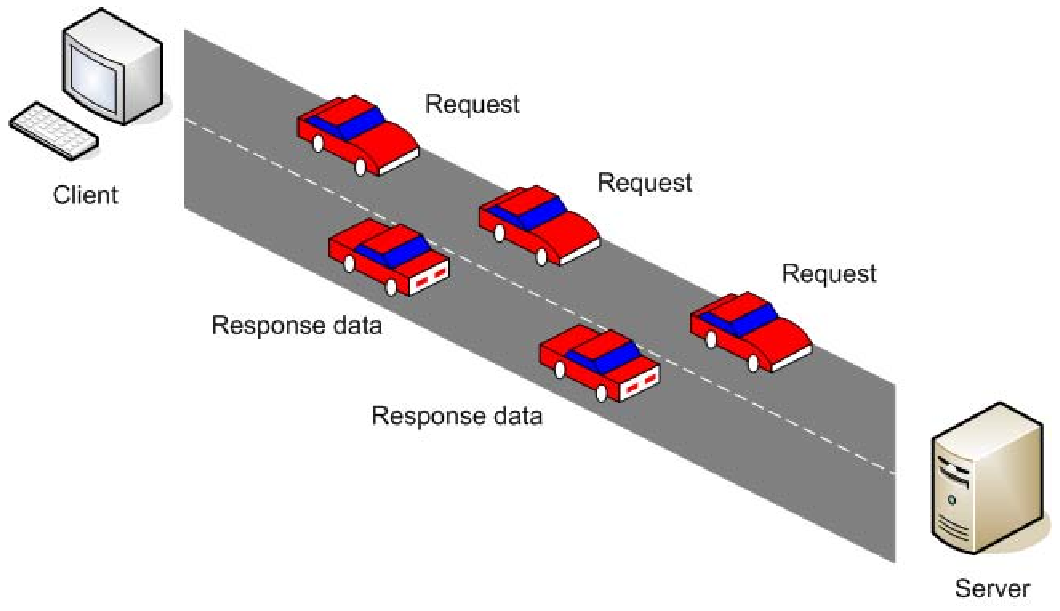
By controlling the submission and properties of XMLHttpRequest objects using JavaScript, partial screen updates replace the click, wait, and refresh user interaction model, and the synchronous request/response model is replaced by asynchronous communication (Wei, 2005). Asynchronous communication allows users to continue to interact with Web applications without waiting for server responses. The Ajax engine will automatically request information from the server in the background. Meanwhile, users can keep using other functions of the Web application. When the new information arrives, the Ajax engine will partially update the user interface (Figure 4).

With this interaction model, the user can enjoy a seamless browsing experience. Compared to the analogy of freeway with a single vehicle, it is like a busy and efficient freeway with multiple vehicles travelling in both directions (Figure 5).

The weaknesses of the traditional Web interaction paradigm and communication model are the strengths of the Ajax model. The Ajax approach for the first time introduces some common features of desktop applications to Web-based applications. The partial update and asynchronous features provide a number of advantages.

- Instead of submitting form data a page at a time when explicitly requested by the user, Ajax does submissions automatically, when the user triggers some event.
- Requests can be sent asynchronously, with the browser receiving results continually.

Figure 5. The asynchronous request and response model



- Smaller, incremental in-place partial Web page updates can be made, instead of full page rebuilds.
- The size of data transmission can be reduced as the Ajax engine only downloads new content, not new pages.

In summary, the key characteristics of Ajax are asynchronous requests to the server, retrieving data continually, and partial page updates. Theoretically, these features can significantly improve the performance and user experience of Web applications.

The Ajax Engine

Not only is the Ajax communication model different from the traditional Web application, but the structural layers of Ajax are different. A traditional Web application architecture has two layers; user interface and server (Figure 6) while Ajax has three layers; user interface, Ajax engine, and server (Figure 7).

The Ajax engine plays the role of middleware sitting between the user interface and the server. It controls the user interface and the data trans-

Figure 6. Classic Web application architecture (adapted from Wei, 2005)

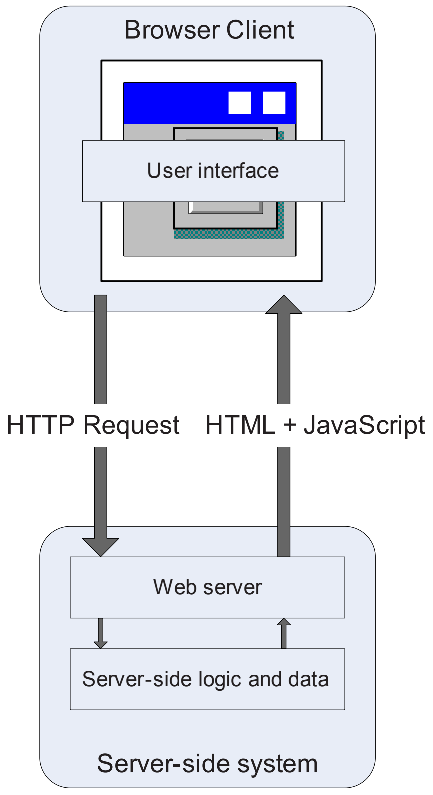


Figure 7. Ajax architecture (adapted from Wei, 2005)

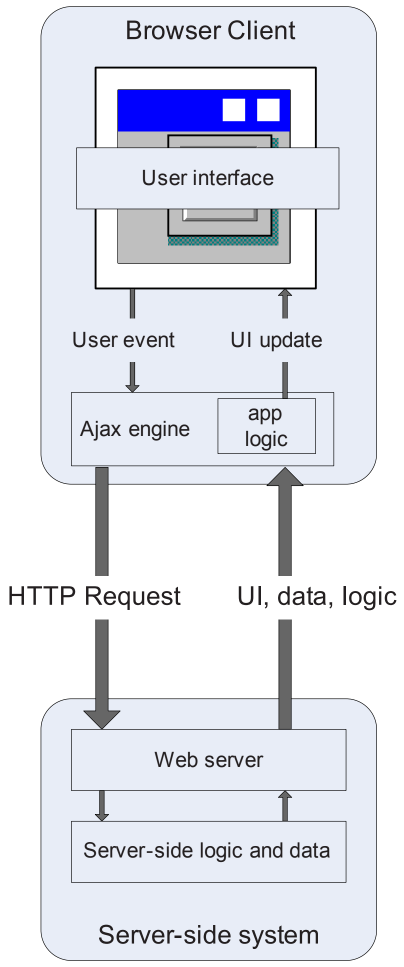
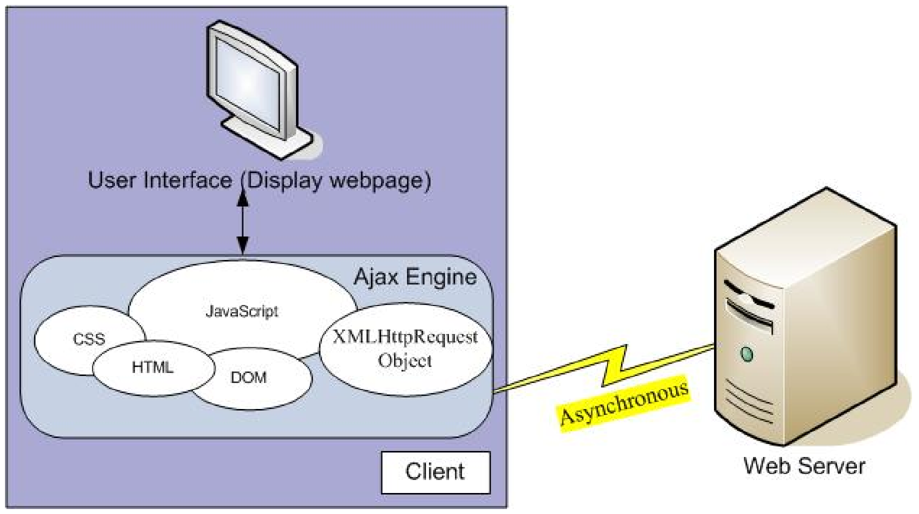


Figure 8. Ajax engine



mission between client and server, and also some logical functions. When the user initiates his or her request, this request will pass to the Ajax engine first, instead of the server directly as it does in a traditional client-server model. Then the Ajax engine will combine the data from the Web browser cache and data requested from the server to update the page. The main advantage of this combining data process is that it can reduce the size of data downloaded from the server, and increase the response speed.

Data transfer between the server and the Ajax engine is via XML documents (or character streams), not entire Web pages. These XML documents only contain the new content required by the client; they do not include any Web page structure or presentation information. This can also reduce the client-server data transfer volume and time.

The Ajax engine is written in JavaScript and uses the document object model (DOM), cascading style sheets (CSS), HTML, and the XMLHttpRequest object to control the Web application (Figure 8). The Ajax engine can be loaded into the client browser at any time, depending on the application design.

The XMLHttpRequest object is a JavaScript object. It supports a set of operations that that allows JavaScript to perform HTTP client functionality for transferring data between a client and a server, such as submitting form data or loading data from a server (W3C, 2007). By using the XMLHttpRequest object, a Web developer can partially update the current page with data from the server after the page has loaded using asynchronous client-server communication.

XMLHttpRequest objects can be used by browser scripting languages to transfer XML and other text data to and from a Web server using HTTP, by establishing an independent communication channel between client and server. The user's requests are received by the Ajax engine, the engine analyses the request and sends out a new request to the server; when the client sends the request to the server via an XMLHttpRequest object, the server will respond to the client using an XML document or character stream rather than a new page of markup. The Ajax engine will use this content to partially update the user interface using the DOM.

In traditional HTML Web applications the 'submit' action is manually controlled by the user's clicking a button following the 'click, wait, and refresh' interaction model, but with Ajax, the JavaScript can directly communicate with the server through the XMLHttpRequest object. It means that not only can 'click events' activate submit actions, but also other user activities, such as mouse or keyboard events. XMLHttpRequest objects allow asynchronous requests to be sent to the server, so requests can be sent continuously every time a user triggers an event, and one does not need to wait until the previous request returns before sending out another request. When the new content arrives, JavaScript uses the received XML document to partially rebuild the Web page using the DOM and CSS without disrupting the user's activities.

Ajax Performance

Previous research suggests that Ajax can significantly improve Web application performance on the desktop. One commonly cited Ajax performance evaluation is the one reported by White (2005). In this study, the Ajax application transferred on average just 27 percent of the bytes that were transferred by a traditional HTML application. Not only was there an improvement in the transferred byte volume, but there was also an improvement in performance, a 68 percent overall improvement in data transfer time. The two applications used in this study, however, did not have the same user interface. The users' skill levels and training were also not assessed in the report, and these factors may have affected the experimental outcomes.

A more controlled Ajax performance measurement comes from Smullen and Smullen (2006). They compared the client-side performance of a real-life HTML application and an Ajax application that implemented the same user interface. In this case, the variable of training and users' skill level can be ignored. Experimental data was collected on the performance of each when presented with the same set of tasks. Performance measures were computed for the HTML application and for the Ajax application.

Later they extended their study by collecting data on a statistically significant sample size and included server performance results. Response size and service time performance measures computed for the applications provided significant performance improvements in response size for the Ajax application (56%), thereby reducing bandwidth requirements. Ajax provided a mean service time improvement of approximately 16 percent (Smullen & Smullen, 2007).

Another Ajax performance study used a Web-chat application, and compared a traditional client-server Web chat model with an asynchronous Ajax model (Angelaccio & Buttarazzi, 2006). Their results showed better performance with the Ajax system, measured by the ratio of chat messages to the second.

Although Ajax performance has been measured on the desktop, there is no currently available performance measurement research about Ajax on mobile devices. Therefore we do not know if it can improve mobile Web-based application performance, or indeed if Ajax is effective in the mobile environment. Ajax is a client-side technology, which means when the client platform moves from the desktop to a mobile device the results may be different. Therefore this study focuses on measuring Ajax performance in the context of mobile learning.

Mobile Browser Support for Ajax

Nokia studies of S60 Smartphone users show browsing is generating over 60 percent of data traffic (Nokia, 2005). Today, hundreds of mobile phone models are available on the market, and more and more mobile phones are offering mobile browsers. A mobile Web browser is designed for the constrained environment of mobile devices, such as mobile phones, PDAs, or PPC (pocket PCs). These mobile browsers are device-oriented. They need to be optimized or modified for the devices they are going to run on, because different vendors' mobile phones have different kinds of hardware and embedded operating systems and applications.

There is no common standard for mobile client development; each mobile phone or mobile device is unique. To illustrate, the size of the screen can vary greatly. The Nokia N70 uses 176 x 208 pixels, 35 x 41 mm TFT and a 256K color screen, but the Motorola RAZR2 V8 uses a 2.2" QVGA (320 x 240 pixels) internal display with a 262K color screen. Their display sizes and resolutions are totally different. Not only is the hardware different for each mobile phone, but operating systems also vary. For example the Nokia N70 uses the Symbian operating system while the Motorola RAZR2 V8 uses Linux. Because of these variations in both hardware and software, a mobile application cannot run on both mobile phones without some customization. However, an increasing number of mobile Web browsers are being developed that can overcome these hardware and operating system issues and provide a common platform for mobile Ajax. To support Ajax, the mobile Web browser must be JavaScript enabled and support the XMLHttpRequest object. It also needs to support DOM, CSS and XML. Opera Mobile, Nokia's S60 Browser and Microsoft Internet Explorer Mobile all claim to support JavaScript and Ajax, and more mobile browsers will support Ajax in the future, which should help the acceptability of Ajax as a pervasive mobile learning platform. For our experiments we used Opera Mobile, a stand-alone Web browser that can run on a number of different mobile devices. Opera Mobile supports all of the necessary desktop browser technologies that enable Ajax.

METHODOLOGY

The methodology used in this study was design science, which has five basic steps: awareness of the problem, suggestion, development, evaluation, and conclusion. Design science research is sometimes called "improvement research," emphasizing the problem-solving/performance-improving nature of the activity. Suggestions for a problem solution are abductively drawn from the existing knowledge/theory base for the problem area. An attempt to build an artefact that implements the solution follows (development). Partially or fully successful implementations are then evaluated. This cycle of development, evaluation and further suggestions is frequently performed iteratively. The basis of the iteration, from partial completion of the cycle back to awareness of the problem, is *circumscription*. Conclusion indicates the termination of a specific design project (Vaishnavi & Kuechler, 2007).

Our study consisted of a single iteration with two evaluations (technical performance measures and usability testing). The results of these measures can be used in further research. In terms of awareness of the problem, we know that mobile Web access is hampered by issues like small screens, high network latency, small bandwidth and interface complexity. One suggestion for addressing this problem is to develop a mobile Ajax system, but so far Ajax application performance over wireless networks has remained untested. To address this issue, two mobile quiz Web applications were developed. The first mobile quiz system is an Ajax enabled Web application, while the second uses a more traditional ASP (active server pages) Web technology. Although these two Web applications use different technologies, they have the same user interface. Because we were interested in comparing the performance of these two applications we needed to exclude other aspects that might affect system performance, such as the user interface. Also, the learning process had to be the same, including very similar questions and answers. Finally, both systems had to be accessed from the same mobile device over the same GPRS network. We evaluated the performance of the two Web-based mobile learning applications from three aspects: theoretical evaluation, real-life performance evaluation, and usability testing.

We began with a theoretical evaluation (quantitative study) where we modeled the mobile learning process and calculated all the data on paper to find out how different system implementations mapped to our design. First, we assume there is a user visiting the Web application using a Web browser and predict the data transfer usage between client and server. Then we compare the ASP Web application's predicted data transfer volumes with the Ajax enabled Web application's data. According to this data, we might consider modifying the Web application's design. This can be considered a tentative evaluation.

The second part of the study was a real-life computer logging evaluation (quantitative study). Here, we set up an experimental test system (using a mobile device, a GPRS network and a Web server) for physical interaction, and deployed the Web applications onto a server. We tested the Web applications over a GPRS network and collected all the data transfer information between the Web server and the client using the Web server's log. Finally, we analyzed the log data to evaluate the performance of the Ajax and non-Ajax Web applications over the GPRS network.

The final part of the study was user evaluation (qualitative study). A small group of users was invited to participate in an observational usability test. Cooperative evaluation, a variation of the think-aloud observational technique, was used in this test. Participants were asked to use both the Ajax and non-Ajax mobile quiz sites using a single mobile device (an i-mate™ SP5 mobile phone). Their comments were recorded and analyzed, with particular emphasis on comments relating to Web application performance. This user evaluation attempted to find out whether the users thought that the Ajax application's performance was noticeably better than the non-Ajax version.

According to March and Smith (1995), there are four outputs from the design research method: constructs, models, methods, and instantiations. Later, Rossi and Sein (2003) and Purao (2002) have set forth their own list of design research outputs. However, they can be mapped directly to March and Smith's list as a fifth output, better theories (Table 1).

In this research, the main constructs are Ajax and non-Ajax Web application technologies using mobile devices over the TCP/IP GPRS network. Our models describe the traditional Web application interaction model and the Ajax interaction model. Our methods are based on software engineering approaches to developing Web-based applications while our instantiations are an Ajax enabled Web application and a non-Ajax application. Measuring the comparative performance of these applications gives us better theories about how to optimize the performance of mobile learning applications.

User Evaluation Design

To evaluate the two systems from the user perspective we used within-subject design, using coopera-

Table 1. The outputs of design research (March & Smith, 1995; Purao, 2002; Rossi & Sein, 2003)

	Output	Description
1	Constructs	The conceptual vocabulary of a domain
2	Models	A set of propositions or statements expressing relationships between constructs
3	Methods	A set of steps used to perform a task – how-to knowledge
4	Instantiations	The operationalization of constructs, models and methods.
5	Better theories	Artefact construction as analogous to experimental natural science

tive evaluation and the post-task walkthrough as our observational techniques.

Because the goal of this usability test was to find out if end users gain any benefits from mobile Ajax, we first measured task completion performance. We also gathered preference data to see if users preferred using Ajax over the ASP application. In a within-subject design, the values of the dependent variable for a task or a set of tasks are compared with the values for another task or another set of tasks within one participant's data. In other words, the same variable is measured repeatedly on the same participant under different task conditions (Ergosoft Laboratories, 2003). Applying within-subject design in this usability test, participants needed to complete the same task using both Web applications. The task was to finish all five multi-choice questions in the quiz (all questions are of equal difficulty). Since the users will have had experience of both versions of the system, they could compare them and make some statements about them. At the same time we could collect some performance data (task completion time). To avoid users transferring learning from the first application to the second, we provided the users with a demonstration Web site with the same interface but different questions, to help users get familiar with the test device and the task process.

Observational Techniques

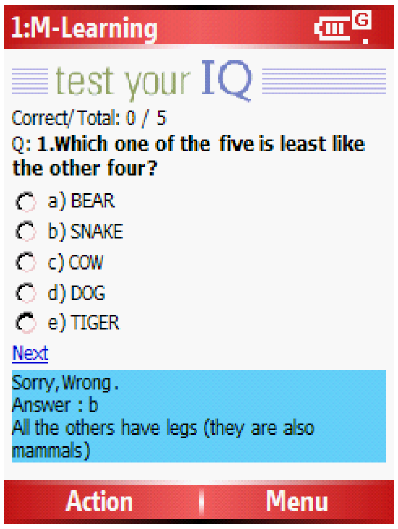
A popular way to gather information about actual use of a system is to observe users interacting with it. The think-aloud method (Lewis & Reiman, 1993) is used to gather data in usability testing in product design and development, in psychology and a range of social sciences. Think-aloud protocols involve participants thinking aloud as they are performing a set of specified tasks. Cooperative evaluation is a variation of think-aloud, encouraging the user to be a collaborator in the evaluation. It allows the user to ask the evaluator any questions if a problem arises and the evaluator can encourage them to express themselves. The protocol for think-aloud sessions can include paper and pencil, audio recording, video recording, computer logging, and user notebooks (Dix, Finlay, Abowd, & Beale, 1998). Paper and pencil plus video recording were applied in this usability test. A post-task walkthrough was also used to reflect their actions back to the subjects after the event. The advantage of a delayed walkthrough is that the analyst has time to frame suitable questions and focus on specific incidents (Dix et al., 1998). In our case, the post-task walkthrough helped users become more aware of some performance problems.

IMPLEMENTATION

This study is based on a controlled measurement of data transfer volumes and times, comparing an Ajax Web application with a similar non-Ajax implementation. First, we predicted system performance based on paper calculations, theoretically analyzing the data volumes and response times. Second, we analyzed real-life data collected from the system log of the server including bandwidth usage, response times, and mobile unit storage requirements. Finally, small group usability tests were conducted to find out if the users noticed any difference between the Ajax and non-Ajax Web applications. In order to compare the actual performance of a traditional Web application and Ajax, two mobile learning quiz systems were created: an Ajax Web application and a similar non-Ajax implementation (ASP). They have the same user interface and functionality (Figure 9).

The mobile learning Web application was based on a three-layer architecture: the mobile phone user interface (Web browser), server pages and a database server. The Web server was Microsoft Internet Information Services (IIS), the server pages were written using ASP 2.0 and the test database was stored using Microsoft Access. All dynamically generated Web pages were valid

Figure 9. Mobile learning Web application interface (in the Opera Mobile browser)



Correct/ Total: 0 / 5
Q: 1.Which one of the five is least like the other four?
a) BEAR
b) SNAKE
c) COW
d) DOG
e) TIGER
Next
Sorry,Wrong.
Answer : b
All the others have legs (they are also mammals)

XHTML 1.1, compatible with the mobile browser (Opera Mobile) used in the test. All data transfer was through the Vodafone New Zealand GPRS network by Class 2 '2+1' multi-slots (2 downlink, 1 uplink channels). The system logs were activated in the IIS server, so all traffic going to and from the server was recorded. The collected data was used to analyze and measure the Web applications' performance over the GPRS network.

The ASP Mobile Learning System

The ASP Web application only contains three server pages: a start page, a question display page, and a finish page. These pages interact with the database to provide page content. Other supporting files are an image file (in GIF format) and a JavaScript file to manage the question fetching process. When the user clicks on the submit button on the screen, it activates a JavaScript function that sends a request to the server to fetch a new question or answer. The storage sizes of all the files on the Web server are shown in Table 2.

The ASP Web application is developed in the traditional way. An ASP file includes CSS and JavaScript file paths in its header (the CSS is used to manage the page's presentation). It also contains the Web page layout information, Web content, and ASP code to dynamically generate the HTML pages. The ASP code also includes a function to verify the user's answers as they are submitted.

The JavaScript file ('asp_m.js') only has two main functions. The first fetches the next question for the client, using a question number attached to the URL. The second validates the user's submission to avoid empty or error submissions being sent to the server.

The database file is used to store all the mobile learning multi-choice questions. In the question database, there is only a single table. All questions, answers and explanations are included in this table.

The Ajax Mobile Learning System

The Ajax implementation is similar to the ASP Web application. It has the same user interface, the same image file ('iq_header.gif'), the same CSS file ('hello.css') and shares the same database. However on the client side it includes three HTML pages and has a different JavaScript file. The ASP file used in the Ajax application is also different from those in the ASP application. It is used as both a database connector and XML file generator, because the communication between client and server is via XML. It is not used to generate HTML pages. The storage sizes of all the files on the Web server for the Ajax application are shown in Table 3.

The total size of the ASP Web application is about 2Kb bytes smaller than the Ajax version. The major difference between them is the size of the JavaScript file. The ASP Web application's JavaScript file is only 1.5Kb but the Ajax Web application's JavaScript file is almost 5Kb. That is because the Ajax JavaScript file not only contains the same functions as the ASP version, but also

Table 2. ASP Web application file sizes (server side)

File Name	Size(bytes)	Detail
index.asp	818	Start Web page
mlearning.asp	3,010	Display questions
iq_header.gif	1,470	Header image file
asp_m.js	1,530	JavaScript file
hello.css	45	CSS file
learning.mdb	496,000	Question database
end.asp	744	Finish Web page
Total size	504,309 bytes	

Table 3. Ajax Web application file sizes (server side)

File Name	File Size (bytes)	Detail
majax.js	4,750	JavaScript file(Ajax Engine)
Hello.css	45	CSS file
2.asp	1,040	Database connection & XML generator
end.htm	1,530	Finish Web page
index.htm	526	Start Web page
iq_header.gif	1,470	Header image file
learning.mdb	496,000	Questions database(same as asp version)
mlearning.htm	1520	Display questions
Total size	506,836 bytes	

has some logical functions, such as verifying the user's answer.

Because Ajax uses a different Web development approach, the purpose of some its files are different from those in the ASP Web application. In the Ajax application, the HTML file size is very small because it is only used to lay out the main structure for the Web page, such as where to display the questions and where to display the answers and explanation. The most important part of the HTML file is that it provides an entry page for the m-learning Web application and the JavaScript file path. The HTML can be considered as a content template.

The JavaScript file, which is acting as an Ajax engine, is very important for the Ajax Web application and provides most of its functions. There are two main aspects and five basic functions provided by this JavaScript file (Ajax engine). The first part of the Ajax engine is used to asynchronously transfer the data between the client browser and server. The second part is used to manage the client side user interface, with a logical function used to verify the user's answer. When the Ajax engine submits a request to the server, the ASP file will dynamically generate an XML document, containing the data returned from a database query. In this case, just a few characters need to be returned to the client browser. The response only includes content data; it does not include any page mark-up information. Back on the client side, when the Ajax engine receives the XML response from the server it will partially update the user interface based on the latest information.

Using the Mobile Learning System

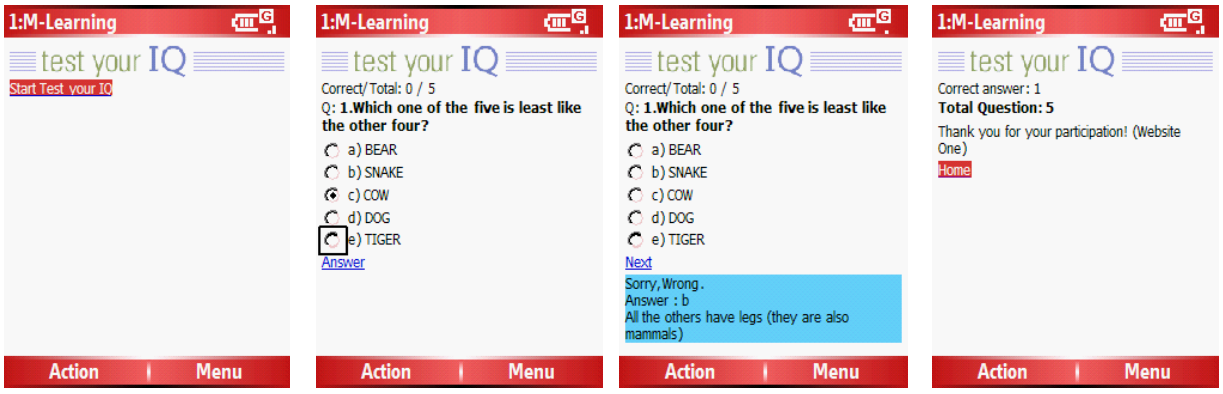
Because both mobile learning quizzes are Web-based, users do not need to install or setup anything as long as their mobile device supports an Ajax enabled browser. Because the Ajax and ASP mobile learning systems are almost identical, both systems are used in the same way. Figure 9 shows some screenshots from the mobile quiz. It begins with a welcome page (1) with a hyperlink to start the quiz. Each answer is selected using a radio button (2). The 'Answer' hyperlink will provide the result. The correct answer will appear under the questions, with the explanation for the answer. Meanwhile, the 'Answer' link will change to a 'Next' link for the next question (3). When all questions have been completed, a finish page is displayed with a link back to the start page (4).

Theoretical ASP System Performance

In this section, we predict the Web applications' performance based on application analysis and paper calculations, theoretically analyzing the data volumes and response times.

For the ASP mobile learning Web application, when the browser requests an ASP file, the server passes the request to the ASP engine. The ASP engine reads the ASP file and executes the scripts in the file to dynamically generate a client Web page. Finally, the generated page is returned to the browser as plain XHTML. These XHTML files also contain references to other required resources such as stylesheets and script files, so we can predict the total file load size based on the source code (Table 4).

Figure 10. Using the mobile learning quiz Web application (screen captures from Opera Mobile browser)



(1)
Start Test your IQ

(2)
Correct/ Total: 0 / 5
Q: **1.Which one of the five is least like the other four?**
a) BEAR
b) SNAKE
c) COW
d) DOG
e) TIGER
Answer

(3)
Correct/ Total: 0 / 5
Q: **1.Which one of the five is least like the other four?**
a) BEAR
b) SNAKE
c) COW
d) DOG
e) TIGER
Next
Sorry, Wrong .
Answer : b
All the others have legs (they are also mammals)

(4)
Correct answer: 1
Total Question: 5
Thank you for your participation! (Website One)
Home

Table 4. ASP application initial load file sizes

File Name	Load Size (bytes)	Detail
mlearning.asp	2K × 2	Loads questions, and refreshes the whole page after submitting answers.
iq_header.gif	2K	Header image file
asp_m.js	2K	JavaScript file
hello.css	45	CSS file
Initial Load Size	8K	

When the ASP application is running, a welcome page and a finish page are used to start and end the application; 'index.asp' (load size: 1K) and 'end.asp' (load size: 1K). These two files are not included in the calculated load size of the application.

The initial load size of the ASP application is about 8K, including the main ASP application file ('mlearning.asp') and a header image file. The JavaScript file controls the downloading of questions, verifies the submitted form and submits the answer back to the server. The initial file size includes the load size for the first question.

The header image file, JavaScript file and the CSS file need to be loaded only once if the browser supports caching. In our measurements, all browsers support caching.

The size of the XHTML file generated by 'mlearning.asp' is 2K, smaller than the ASP file size of 3K (Table 1). This is because the code generated by the scripts in the page is smaller than the source code. Because of the nature of the ASP file, the ASP scripts need to be executed on the server and rebuild the whole page each time. Since the question and answer pages are generated separately, the client browser will download 2K × 2 = 4K bytes for each question.

The mobile learning application contains 10 multi-choice questions in total, but when the application initially loads, only one question has been loaded. Subsequently the 'mlearning.asp' file needs to generate the other nine questions for the user. Additionally, it sends back nine answer pages, so the ASP file will generate 18 pages in total comprising about 2K × 18 = 36K bytes of XHTML mark-up. The formula to calculate the total download size of a Web application is:

Initial Load Size + Process Load Size = Total Size

Therefore the total download size for the ASP mobile learning Web application is:

8K + 36K= 44K

As a result, we expect that about 44K of data needs to be sent back to the browser from the m-learning Web server.

Theoretical Ajax System Performance

When an Ajax application is requested by a browser, an XHTML page is loaded into the browser. This XHTML page contains the JavaScript needed to run the user interface and to issue XMLHttpRequest calls, the XML handler to format and present the results, and any other elements needed for the user interface, such as XHTML mark-up and CSS. The client interacts with the user interface presented on the browser

Table 5. Ajax application initial load file sizes

File Name	Load Size (bytes)	Detail
2.asp	500	Send XML back to browser (XML file size)
majax.js	5K	JavaScript file (Ajax Engine)
hello.css	45	CSS file
iq_header.gif	2K	Header image file
mlearning.htm	2K	Load first questions
Initial Load Size	10K	

page (e.g., clicking hyperlinks, submitting forms, or triggering an event). The Ajax engine handles all interaction events between user and server by generating an XMLHttpRequest message to the server. The server returns new content in XML file format. This XML data is handled by the Ajax engine and presented to the user by partially updating the current page in the browser.

As with the ASP application, the Ajax system displays a welcome page at the beginning (index.htm) and a finish page at the end (end.htm) which are both about 1K. These two files are again excluded from the download calculations.

The main application file (mlearning.htm) and associated resources (e.g., JavaScript file, CSS file, and header image file) will be loaded into the Web browser for the first question. This initial load of the Ajax application requires 10K of files, but unlike the ASP Web application, only a small XML file needs to be downloaded for subsequent questions. The file sizes for the Ajax application are shown in Table 5.

As with the ASP version, the header image file, JavaScript file and CSS file only need to be loaded once if the browser supports caching.

In the Ajax version of the m-learning application, the initial load size is more than the ASP version, but the advantages of the Ajax application show up when the user is using the application. No further pages need to be downloaded from the Web server, only the question's content and answers in XML file format. All the answers from the user can be verified locally by the Ajax engine; the browser does not need to submit any data back to the server. The size of the XML file sent back from the Web server is only about 500 bytes. With this data the Ajax engine can partially update the Web page to tell the user the correct answer. Therefore the main part of the Ajax application only needs to transfer about 9 × 500 = 4.5K of data during the quiz process. So, according to our formula the total download size for the Ajax application is:

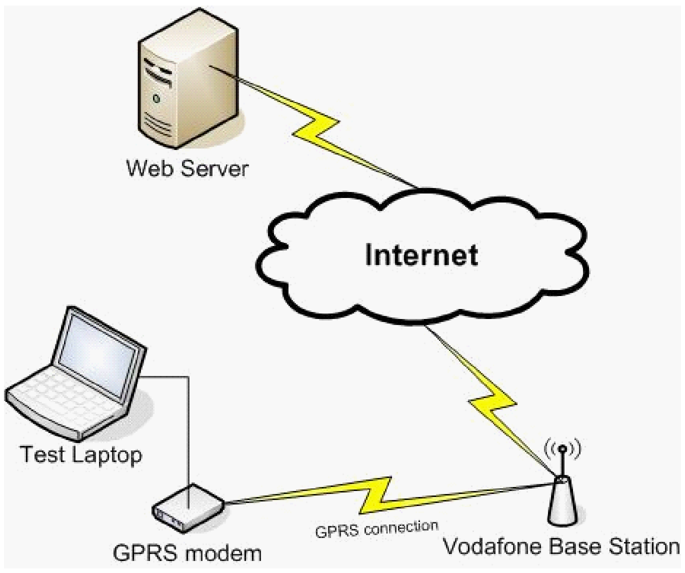
10K + 4.5K= 14.5K

In summary, according to our calculations, the ASP m-learning system needs to transfer 44K of data; Ajax can provide the same interface by only transferring 14.5K of data. That is a big theoretical improvement, but can it be replicated using empirical testing? To find out, a practical performance measurement was carried out on an experimental test system.

Experimental Performance Measures

The experimental system setup used to measure the m-learning system performance is shown in Figure 11. We used a laptop connected to an

Figure 11. Experimental setup for performance testing



iTegno 3000 GPRS modem by USB 1.1 through a serial PPP (point-to-point) link to act as a GPRS mobile terminal to emulate the GPRS connection from mobile devices to the Web server in the Vodafone New Zealand (GPRS/GSM) network. The laptop used the Windows XP operating system and the Web browser was Opera Mobile 8.65 for Windows mobile 5/6 PPC running in a Windows Mobile 6 emulator. The Vodafone New Zealand GPRS network (two download channels and one upload channel, Coding Scheme 4) was used. Since wireless network signal levels may vary across different locations, all access to the Web server from the test laptop was via the Albany Stadium base station adjacent to Massey University's Auckland campus.

The Web server was running Internet Information services 6.0 (IIS 6.0); system logs were active in IIS so all traffic going to and from the Web server was recorded in the system log, monitoring all data transfer between the Web server and mobile terminal. These logs, comprising data from over 200 requests, were analyzed using Microsoft Log Parser 2.2. (Microsoft TechNet 2007). All log data was collected from the dedicated test domain.

Table 6 shows a summary of part of the server log for the ASP Web application. 'cs-method' means the HTTP request type, 'sc-bytes' means the number of bytes that the server sent and 'cs-bytes' means the number of bytes that the server received. 'Time-taken' means the length of time that the action took in milliseconds. This table illustrates one of the most important aspects of the ASP Web application, which is that the 'mlearning.asp' file needs to be executed twice for each question; the first time is the 'GET' request for the question, and the second time is the 'POST' request for the answer, which refreshes the whole page.

Table 7 shows an example of the IIS log for the Ajax application. This uses only 'GET' requests.

ASP Application Response Size

Response size means the number of bytes sent by the server, which is one of the most useful

Table 6. ASP application log example

cs-method	cs-uri-stem	sc-bytes	cs-bytes	time-taken
GET	/asp1/index.asp	1,117	466	468
GET	/asp1/hello.css	346	454	156
GET	/asp1/iq_header.gif	1,815	459	296
GET	/asp1/asp_m.js	1,893	469	187
GET	/asp1/index.asp	1,117	407	140
GET	/asp1/iq_header.gif	1,815	352	156
GET	/asp1/mlearning.asp	2,181	493	265
POST	/asp1/mlearning.asp	2,274	686	499
. . .				
GET	/asp1/mlearning.asp	2,184	495	203
POST	/asp1/mlearning.asp	2,280	690	343
GET	/asp1/end.asp	963	460	156

Table 7. Ajax application log example

cs-method	cs-uri-stem	sc-bytes	cs-bytes	time-taken
GET	/ajax1/index.htm	851	508	171
GET	/ajax1/hello.css	346	350	140
GET	/ajax1/iq_header.gif	1,814	354	281
GET	/ajax1/mlearning.htm	2,166	515	374
GET	/ajax1/majax.js	5,190	353	296
GET	/ajax1/2.asp	518	377	296
GET	/ajax1/2.asp	514	377	203
GET	/ajax1/2.asp	500	376	187
GET	/ajax1/2.asp	621	377	187
GET	/ajax1/2.asp	659	377	187
GET	/ajax1/2.asp	528	376	234
GET	/ajax1/2.asp	533	377	203
GET	/ajax1/2.asp	571	378	249
GET	/ajax1/2.asp	583	377	203
GET	/ajax1/2.asp	524	379	187
GET	/ajax1/end.htm	1,951	465	156

Table 8. ASP application 'sc-bytes' statistics

cs-uri-stem	sc-bytes		
ASP application	AVG	MIN	MAX
/asp1/mlearning.asp	2 x 2,305	2,256	2,406
/asp1/asp_m.js	1,893	1,833	1,897
/asp1/iq_header.gif	1,815	1,800	1,880
/asp1/hello.css	346	356	396
Initial load size.	≈ 8.7Kb		

Table 9. Ajax application 'sc-bytes' statistics

cs-uri-stem	sc-bytes		
Ajax application	AVG	MIN	MAX
/ajax1/majax.js	5,190	5,100	5,280
/ajax1/mlearning.htm	2,166	2,007	2,356
/ajax1/iq_header.gif	1,814	1,754	2,014
/ajax1/hello.css	346	326	386
/ajax1/2.asp	555	500	614
Initial load size.	≈ 10kb		

measurable features of application performance. In the IIS log, its identifier is 'sc-bytes.'

According to the IIS system log (Table 8), the average ASP application initial load size is about 8.7K (including the first question). As expected, the main ASP application file needs to load 10 times to display the questions and load another 10 times after users submit the answer. The average load size of the file generated by 'mlearning.asp' is 2,305K, so the total bytes needed to be sent back from the Web server are calculated as follows:

$8.7K + 9 \times 2 \times 2305 \approx 50K$

So, in real-life performance tests, the ASP M-learning application needs to load about 50K of data from the server during the whole application process, somewhat larger than our estimate of 44K.

Ajax Application Response Size

The number of bytes returned for Ajax application is different from the ASP application. First, the load size is very different from the ASP version, about 10K of initial load data. The major file is the JavaScript (Ajax engine) file which is about 5K, and the other one is an HTML layout file that is about 2K (Table 9).

However, while the Ajax application is running, just a few bytes need to be transferred between the client browser and the server. All

Table 10. ASP application 'time-taken' statistics

cs-uri-stem	Time-taken		
ASP Application	AVG(ms)	MIN(ms)	MAX(ms)
/asp1/mlearning.asp	285×2	187	499
/asp1/iq_header.gif	207	156	296
/asp1/asp_m.js	202	187	218
/asp1/hello.css	163	156	171
Initial Time taken(ms)	1,142		

Table 11. Ajax application 'time-taken' statistics

cs-uri-stem	Time-taken		
Ajax Application	AVG (ms)	MIN (ms)	MAX (ms)
/ajax1/hello.css	319	140	499
/ajax1/majax.js	296	296	296
/ajax1/iq_header.gif	281	201	381
/ajax1/mlearning.htm	272	171	374
/ajax1/2.asp	210	171	343
Initial Time taken(ms)	1,378		

this data is XML and the average size is only about 555 bytes (generated by the '2.asp' file), including the question and the answer. It allows the Ajax engine to verify the answer locally, and provides users with a very fast response time. Also, because the question and answer load at the same time, each question only needs one post back from the server. The total load size is calculated as follows:

$10K + (9 \times 555K) \approx 15K$

This is only slightly larger than our estimate of 14.5K. The Ajax m-learning application only needs to transfer about 15K of data between the server and the client, whereas the ASP Web application needs the server to transfer 50K of data. There is therefore a significant difference between the two applications in terms of data transfer from server to client.

ASP Application Response Time

As well as measuring the size of downloaded data, the server's response time is another important feature that can be used to measure a Web application's performance. In the IIS system log, there is a field called 'time-taken,' which is the length of time that the action took in milliseconds on the server, and can be considered the server's action response time.

The measurement formula is similar to measuring the bytes sent back from the server. According to the IIS system log (Table 10), the initial response time for the ASP m-learning application is about 1,142ms (initial load time, including the first question load time), and the main application file needs to load twice the number of times as there are questions in the quiz. The average response time for the 'mlearning.asp' file is 285ms; the 'mlearning.asp' page needs to process twice for each question, so for a total of 10 questions, minus the response time for first question, the process response time is

9 × 2 × 285 = 5,130ms

Similar to the total load size formula, the total response time for the whole application formula is:

Initial Response Time + Process Response Time = Total Response Time

So, the total Web server response time for the ASP mobile learning Web application is:

1,142ms + 5,130ms = 6,272ms

Ajax Application Response Time

For the Ajax m-Learning application, the IIS system log shows the initial response time is about 1,378ms (including the first question), and the average server response time for the XML generator '2.asp' file is 210ms (Table 11). However, '2.asp' only runs once for each question on the server. Therefore for a total of 10 questions, minus the response time for the first question, the process response time is

9 × 210 = 1,890ms.

According to the formula, the total server response time for the whole ASP mobile learning Web application is:

1,378ms + 1,890ms = 3,268ms

In summary, for the whole application, the ASP system takes about 6,272ms of server response time, but the Ajax application takes only 3,268ms.

ASP Application Request Size

The response size from the Web server and the server's task time show us the Ajax m-learning application transfers smaller volumes of data between the client and server and has a better response time than the ASP m-learning application. However, what about the data submitted from the client? Does it reveal a similar story?

The client's request size can affect the transfer time; if the request size is big, not only does it need more bandwidth to transfer the data, but also it takes a long time to transfer before the server can actually process the request. Then, from the user's point of view, the Web application's response time is slow. Therefore, we should also include the client's request sizes in our evaluation.

The 'cs-bytes' field in the IIS system log shows us the number of bytes that the server received, so this data can be regarded as a measure of the data sent from the client browser to request new data from server. Table 12 shows query size of submitted data from the client to the server. For the 'mlearning.asp' file there are both 'GET' and 'POST' methods, and the query size differs for these two methods.

These queries are used to request the next question for the user. The average query size for the main 'mlearning.asp' file is about 496 bytes (Table 12, 'GET' method). Of course, in the ASP version, users need to submit their answers back to the server to have them verified. The average transfer size is 686 bytes (Table 12, 'POST' method) when users submit their answers. Therefore the initial request size (including the first question) formula is:

mlearning.asp (GET) + mlearning.asp (POST) + other files = Initial Request Size

Table 12. ASP application 'cs-bytes' statistics

cs-uri-stem	cs-bytes (bytes)		
ASP Application	AVG	MIN	MAX
/asp1/mlearning.asp("GET")	496	492	566
/asp1/mlearning.asp("POST")	686	600	750
/asp1/iq_header.gif	398	352	459
/asp1/asp_m.js	423	378	469
/asp1/hello.css	416	379	454
Initial Query Size	2,428 (including first question)		

Table 13. Ajax application 'cs-bytes' statistics

cs-uri-stem	cs-bytes		
Ajax Application	AVG (bytes)	MIN (bytes)	MAX (bytes)
/ajax1/2.asp	377	373	379
/ajax1/iq_header.gif	354	350	374
/ajax1/majax.js	353	351	363
/ajax1/hello.css	350	350	370
/ajax1/mlearning.htm	515	514	519
Initial Request Size	1,949 (including first question)		

According the formula, the initial request size is therefore calculated as:

496 + 686 + 398 + 423 + 416 = 2,428 bytes

The total request size calculation formula is:

Initial Request Size + Process Request Size = Total Request Size

Which gives us the following result:

2,428 + (686 × 9 + 496 ×9) = 13,066 bytes ≈ 13K

In total, therefore, the client needs to submit about 13K of data to the Web server.

Ajax Application Request Size

Table 13 shows the size of data submitted from the Ajax client to the server.

The average request size is 377 bytes. These submissions only request the next question: one question, one submission. Therefore it is very easy to calculate the initial request size (including the first question) for the Ajax application; simply add all the request sizes together:

377 + 354 + 353 + 350 + 515 = 1,949 bytes

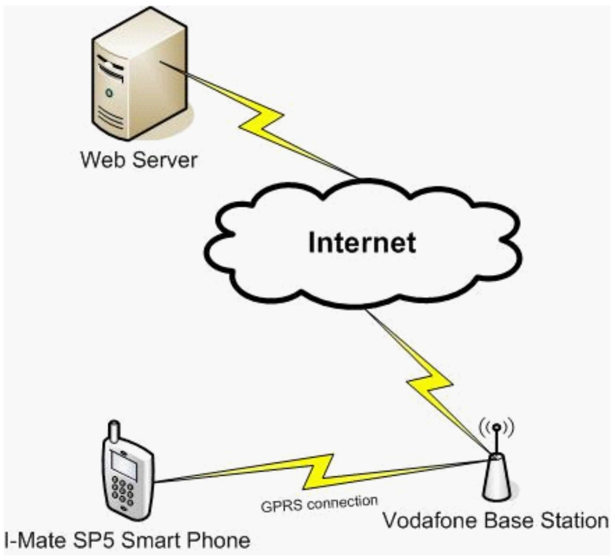
The total query size is therefore calculated as:

1,949 + 9 × 377 = 5,342 bytes ≈ 5.3K

Table 14. Performance improvement

	Response size	Response seconds	Request size
ASP	50K	6,272ms	13K
Ajax	15K	3,268ms	5.3K
Improvement	71.00%	48.00%	59.23%

Figure 12. Experimental setup for usability testing



The overall result is that the ASP Web application submitted around 13K of request data to the server; while the Ajax Web application sent only about 5.3K of data.

Overall Performance Measures

When measuring a Web application's performance, the data can be separated into three parts: data transfer time (in two ways, from client to server and from server to client) and server process time. Table 6 summarizes these performance measures for the two Web applications. The average data volume sent from the Web server is 50K for the ASP version and 15K for the Ajax version during the whole in the m-Learning Web application process. The average number of bytes received by the server is about 13K for the ASP version and about 5.3K for the Ajax version. It takes about 6,272ms for the server to process all requests for the ASP version and about 3,268ms for the Ajax version. White (2005) and Smullen and Smullen (2006) use the following algorithm to define the percentage of an Ajax application's performance improvement:

(HTML-Ajax)/HTML

This formula has been used to calculate the 'improvement' data in Table 14 (in our case 'HTML' would represent the ASP application results).

The Ajax approach in our experiment provides a significant performance improvement across all three of our measures. We can therefore confidently state that our Ajax M-learning Web application has much better performance than the non-Ajax m-learning Web application when tested over a GPRS mobile phone network.

USABILITY EVALUATION

The goal of this study was to evaluate the Ajax enabled mobile learning system's performance over a GPRS network both by quantitative measurement statistics and user feedback. This was necessary because although we could demonstrate that the Ajax system performed better in quantitative terms, we also needed to find out if these differences were significant enough to be noticed by end users. The goal of the usability test was therefore to find out if users noticed any differences between the Ajax and the non-Ajax mobile learning applications, and if these differences were meaningful from an end-user perspective.

In order to simulate a real-life mobile learning environment as realistically as possible, the usability testing experimental setup (Figure 12) was somewhat different from the performance measurement setup. In this system, we used an i-mate^{TM} SP5 SmartPhone instead of the laptop. The mobile Web browser was Opera Mobile, version 8.65 for Windows Mobile 5/6 SmartPhone operating system. The same Vodafone New Zealand GPRS network (two download channels and one upload channel, Coding Scheme 4) was used. All participants used the same mobile phone, again connected via the Albany Stadium base station.

In the evaluation, users were asked to complete the quiz using the mobile device, think aloud about their feelings and answer a questionnaire. In order to reduce the interview time, we reduced the number of questions in the quiz from 10 down to five for each system. The test Web applications were again hosted at http://www.ajaxnz.com.

There were two tasks in this usability test. Task one required users to finish all the multi-choice test questions in the Ajax mobile learning system, and task two required users to finish a similar set of multi-choice test questions using the ASP mobile learning system. A training session was provided before users started their mobile learning activity. Finally, the post-walkthrough method was applied to wrap up the test. The questionnaire was completed and the Opera Mobile cache was cleared ready for the next test.

In this usability test, there were two variables we collected from the users to represent both quantitative data and qualitative data. The first quantitative variable is the task completion time, which was used to measure the two different performances (within-subject design). The second variable is a qualitative variable, the preferred data, which is used to find out if users notice any difference between the two Web applications.

The questionnaire included some user information, such as gender, the type of mobile phone they have, if they have ever used their mobile phone to access the Internet and if so, how often. Participants were also asked if they noticed any differences between the two Web applications, and if so, if they preferred to use one over the other.

Usability Test Results

Twenty people attended this small group usability test, totaling 19 valid examples; one participant did not finish the test. In all of the valid examples, 18 of the participants had mobile phones, and only one participant did not have a mobile phone. There were six female participants and 13 male participants. Approximately 40 percent of the participants had previous experience of using a mobile phone to access the Internet. Twenty percent had accessed the Internet a few times, while 20 percent have used the mobile Internet just once. Only 10 percent of the participants used the mobile Internet at least once a week.

During the usability test, all participants were comfortable about the mobile learning system itself, but had some problems with the mobile device. The first complaint was about the navigation button being hard to use and not functioning very well. Sometimes users had to keep pressing it until it worked (i.e., an action occurred on the screen). One participant aborted the test because of these problems with the navigation button. The second problem was about the small screen and font size. Observations about the device tended to overshadow observations about the application itself. However one participant observed that the loading order of questions and answer options was different between the two applications (a consequence of Ajax updating the page asynchronously).

Based on the 19 test examples, applying pairwise T-Testing to the task completion time, our participants performed significantly better with the Ajax mobile learning system.

$t\ (18) = 11.71,\ p<0.05.$

The mean value of the task completion time for the first Web application was 119.26 seconds and the task completion time for the second Web application was 64.11 seconds. This means that users finished their task on the second application much faster than the first, about 50 percent faster. The Ajax mobile learning system therefore enabled much better performance than the ASP mobile learning system when assessed by the quantitative variable.

In terms of frequency to their preference, we can use a chi-square test, but to some extent it does not seem to meet its assumption.

Chi-square (2) = 12.41, $p<0.05$

The expected sample is too small, so it would be unwise to use this statistical analysis for sure. However, analytically (not statistically) speaking, users noticed there were some differences between the two applications, even though they have the same interface and the same learning process. Twelve out of the 13 participants who noticed a difference preferred the Ajax mobile learning system. These two results from the usability test indicate that an Ajax mobile learning system can provide better performance in the mobile environment on the user experience level.

SUMMARY AND CONCLUSION

In our tests we applied the Ajax approach to a mobile learning system and compared it with a traditional ASP mobile learning system. After the evaluation of these systems in terms of their performance and usability, our results indicate that an Ajax mobile learning system can reduce data transmission volumes and the server's response time, and is preferred by users. Moreover, applying an Ajax approach to Web-based mobile applications is a much more practical way to improve system performance than updating the mobile hardware or the wireless network.

The advantages of Ajax are; first, it allows users to access to the system as long as their mobile browser supports Ajax and has connectivity to a GSM/GPRS network, meaning that it is a widely available option. Second, system performance is much faster than a traditional Web application. Third, reduced data traffic can save money for mobile learners in territories where mobile telecommunications companies charge for the amount of data sent and received. In addition, although this is not a major aspect of our research, Ajax systems also can be accessed from a desktop browser environment without modification.

In our experimental measurements, Ajax reduced network transfer traffic from the server to the client by 71 percent, saved 48 percent of the Web server's processing time and reduced submission data from the client to the server by 59 percent. Furthermore, on the user experience level, the task completion time statistics show users finished the same m-learning multi-choice questions on the Ajax system 50 percent faster than they did on the ASP system. Finally, more than 60 percent (12 out of 19) of users noticed the difference between these systems and preferred to use the Ajax system.

The results of our experiments demonstrate that the Ajax approach can significantly reduce both the data transmission size and the server's response time. Meanwhile, reducing the bandwidth required, and speeding up the user interface on the mobile device provides the user with a better mobile Internet experience. In addition, the Ajax approach has little infrastructure dependence, because it is a software technology and does not require any hardware or environment updates in order to be implemented.

FUTURE RESEARCH DIRECTIONS

There were a number of limitations to this study. Our experimental system was only tested on one GPRS network, with two download channels and one upload channel, and only one mobile device (i-mate^{TM} SP5). There are some other GPRS network configurations that could be used as an evaluation network and may give different results, as well as a number of other types of mobile data network such as 3G, WiMAX and WiFi. Other mobile devices may also perform differently from those used in our experiments. Another limitation of our study is that the Web application's functionality is very limited, only measuring the performance of a multi-choice quiz. Whilst such quizzes are an important component of mobile learning systems, other types of mobile learning interaction may well give different profiles of data transfer. In addition, the test group used in our usability experiment was too small and narrow (most of the participants were students or teachers) to enable us to draw more generalized conclusions.

There are a number of avenues for further research that could be explored. In this study, both the ASP and Ajax Web applications had the same user interface to enable us to evaluate performance statistics. However, one consequence of this is that we are not using the full potential of Ajax, simply mimicking a more traditional UI. To properly evaluate the benefits of Ajax in mobile learning systems it would be helpful to undertake both quantitative and qualitative evaluations of systems that leverage all the features of Ajax. Further study might also be undertaken using other types of wireless network, to see if the kinds of results measured here over GPRS are replicated under different network conditions. Finally it would be useful to do performance testing with a more realistic, larger scale mobile learning system than the multiple choice quizzes used in this study.

REFERENCES

Alexander, B. (2004). Going nomadic: Mobile learning in higher education. *EDUCAUSE Review, 39*(5), 28-35.

Angelaccio, M., & Buttarazzi, B. (2006, December 4-7). *A Performance Evaluation of Asynchronous Web Interfaces for Collaborative Web Services*. Paper presented at the Frontiers of High Performance Computing and Networking – ISPA 2006 Workshops (pp. 864-872). Sorrento, Italy.

Bar, H., Häussge, G., & Rößling, G. (2007, June 23–27). *An integrated system for interaction support in lectures*. Paper presented at the 12th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE'07) (pp. 281-285). Dundee, Scotland.

Berri, J., Benlamri, R., & Atif, Y. (2006, July 3-6). *Ontology-based framework for context-aware mobile learning.* Paper presented at the International Conference on Wireless Communications and Mobile Computing (pp. 1307-1310). Vancouver, Canada.

Black, J. T., & Hawkes, L. W. (2006, July 3-6). *A prototype interface for collaborative mobile learning.* Paper presented at the International Conference on Wireless Communications and

Mobile Computing (pp. 1277-1282). Vancouver, Canada.

Cao, Y., Tin, T., McGreal, R., Ally, M., & Coffey, S. (2006, July 3-6). *The Athabasca University mobile library project: increasing the boundaries of anytime and anywhere learning for students.* Paper presented at the International Conference on Wireless Communications and Mobile Computing (pp. 1289-1294). Vancouver, Canada.

Chakravorty, R., Cartwright, J., & Pratt, I. (2002, November 17-21). *Practical experience with TCP over GPRS.* Paper presented at the IEEE Global Telecommunications Conference (GLOBECOM '02) (pp. 1678-1682). Taipei, Taiwan.

Chakravorty, R., Clark, A., & Pratt, I. (2003, May 5-8). *GPRSWeb: Optimizing the Web for GPRS Links.* Paper presented at the MobiSys 2003: The 1st International Conference on Mobile Systems, Applications, and Services, San Francisco.

Chakravorty, R., & Pratt, I. (2002, September 9-11). *WWW performance over GPRS.* Paper presented at the 4th International Workshop on Mobile and Wireless Communications Network, Stockholm, Sweden.

Church, K., Smyth, B., Cotter, P., & Bradley, K. (2007). Mobile information access: A study of emerging search behavior on the mobile internet. *ACM Transactions on the Web, 1*(1), Article 4.

Crane, D., Pascarello, E., & James, D. (2005). *Ajax in action*. Manning Publications Co.

Dix, A. J., Finlay, J. E., Abowd, G. D., & Beale, R. (1998). *Human-computer interaction* (2nd ed.). Hertfordshire: Prentice Hall Europe.

Ergosoft Laboratories. (2003). What is a within-subjects design? Retrieved January 3, 2008, from http://www.ergolabs.com/within_subjects.htm

Garrett, J. J. (2005). Ajax: A New Approach to Web Applications. Retrieved January 3, 2008, from http://adaptivepath.com/publications/essays/archives/000385.php

Kadirire, J. (2007). Instant messaging for creating interactive and ollaborative m-learning environments. *The International Review of Research in Open and Distance Learning, 8*(2) 1-14.

Kukka, H., & Ojala, T. (2006, October 25–27). *mobileDOK: Culture in your pocket*. Paper presented at the 3rd international conference on mobile technology, applications & systems (Mobility 06), Bangkok, Thailand .

Lee, T. B. (1992). What's new in '92. Retrieved January 3, 2008, from http://www.w3.org/History/19921103-hypertext/hypertext/WWW/News/9201.html

Lee, W., & Lu, C. (2003). Customising WAP-based information services on mobile networks. *Personal and Ubiquitous Computing, 7*(6), 321-330.

Leung, C.-H., & Chan, Y.-Y. (2003, July 9-11). *Mobile learning: a new paradigm in electronic learning.* Paper presented at the 3rd IEEE International Conference on Advanced Learning Technologies (pp. 76-80). Athens, Greece.

Lewis, C., & Rieman, J. (1993). Task-centered user interface design: A practical introduction. *Shareware.* Retrieved January 3rd, 2008, from http://www.hcibib.org/tcuid/

March, S., & Smith, G. (1995). Design and natural science research on information technology. *Decision Support Systems 15*(4), 251-266.

Microsoft TechNet. (2007). W3C Extended Log File Format (IIS 6.0). Retrieved January 3, 2008, from http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/bea506fd-38bc-4850-a4fb-e3a0379d321f.mspx?mfr=true

Nakahara, J., Hisamatsu, S., Yaegashi, K., & Yamauchi, Y. (2005, May 30 - June 4). *iTree:*

Does the mobile phone encourage learners to be more involved in collaborative learning? Paper presented at the 2005 conference on Computer support for collaborative learning (pp. 470–478). Taipei, Taiwan.

Nokia. (2005). Nokia introduces a new Web browser for S60 3rd Edition. Retrieved January 3, 2008, from http://press.nokia.com/PR/200511/1019239_5.html

Nortel. (2007). Long-Term Evolution (LTE): The vision beyond 3G. Retrieved January 3, 2008, from http://www.nortel.com/solutions/wireless/collateral/nn114882.pdf

Pinkwart, N., Hoppe, H. U., Milrad, M., & Perez, J. (2003). Educational scenarios for the cooperative use of personal digital assistants. *Journal of Computer Assisted Learning, 19*(3), 383-391.

Polsani P (2003). Network learning. In K. Nyiri (Ed.), *Mobile learning. Essays on philosophy, psychology and education* (pp. 139–150). Vienna: Passagen Verlag.

Purao, S. (2002). Design Research in the Technology of Information Systems: Truth or Dare. *GSU Department of CIS* Working Paper, Atlanta, Retrieved January 3, 2008, from http://purao.ist.psu.edu/working-papers/dare-purao.pdf

Rossi, M., & Sein, M. (2003, August 9-12). *Design Research Workshop: A Proactive Research Approach*. Presentation delivered at the 26th Information Systems Research Seminar in Scandinavia (IRIS 26), Porvoo, Finland.

Sayar, A., Pierce, M., & Fox, G. (2006, February 19-25). *Integrating AJAX Approach into GIS Visualization Web Services*. Paper presented at the International Conference on Internet and Web Applications and Services (AICT-ICIW '06), Guadeloupe, French Caribbean.

Seong, D. S. K. (2006, October 25–27). *Usability guidelines for designing mobile learning portals.* Paper presented at the 3rd international conference on mobile technology, applications & systems (Mobility 06), Bangkok, Thailand.

Sharples, M., Corlett, D., & Westmancott, O. (2002). The design and implementation of a mobile learning resource. *Personal and Ubiquitous Computing, 6*(3), 220-234.

Smullen, C., & Smullen, S. (2006, July 17-19). *Modelling AJAX Application performance.* Paper presented at the Conference on Web Technologies, Applications, and Services (WTAS 2006), Calgary, Canada.

Smullen, C., & Smullen, S. (2007, March 22-25). *AJAX application server performance*. Paper presented at the IEEE SoutheastCon (pp. 154-158). Richmond, USA.

Stodel, E. J., Thompson, T. L., & MacDonald, C. J. (2006). Learners' perspectives on what is missing from online learning: Interpretations through the community of inquiry framework. *The International Review of Research in Open and Distance Learning, 7*(3).

Stuckmann, P., Ehlers, N., & Wouters, B. (2002, September 24-28). *GPRS traffic performance measurements*. Paper presented at the 56th Vehicular Technology Conference (VTC 2002), Vancouver, Canada.

Theng, Y, Tan, K, Lim, E, Zhang, J., Goh, D., Chatterjea, K. et al. (2007, June 18-23). *Mobile G-Portal supporting collaborative sharing and learning in geography fieldwork: an empirical study.* Paper presented at the International Conference on Digital libraries (pp. 462–471). Vancouver, Canada.

Traxler, J. (2007). Defining discussing and evaluating mobile learning: The moving finger writes and having writ... *International Review of Research in Open and Distance Learning. 8*(2) unpaged.

Vaishnavi, V., & Kuechler, B. (2007). Design Research in Information Systems. Retrieved

January 3, 2008 from http://www.isworld.org/Researchdesign/drisISworld.htm

W3C.org. (2007). The XMLHttpRequest Object. Retrieved January 3, 2008 from http://www.w3.org/TR/XMLHttpRequest/

Watson, H., Rainer, R., & Koh, C. (1991). Executive information systems: A framework for development and a survey of current practices. *MIS Quarterly, 15*(1), 13-30.

Wei, C. K. (2005). AJAX: Asynchronous Java + XML? Retrieved March 4, 2007, from http://www.developer.com/design/article.php/3526681

White, A. (2005). Measuring the Benefits of Ajax. Retrieved August 15, 2006, from www.developer.com/java/other/article.php/3554271

Yokomoto, C. F. (2000, October 18-21). *Promoting depth of knowledge through a new quiz format to improve problem solving abilities.* Paper presented at the 30th ASEE/IEEE Annual Frontiers in Education Conference (FIE 2000), Kansas City, USA.