International Journal of Principles and Applications of Information Science and Technology
July 2008, Vol.2, No.1

Measuring AJAX Performance on a GPRS Mobile Platform

Feng Xie[1] and David Parsons[2]

Institute of Information and Mathematical Sciences, Massey University
Auckland, New Zealand
[1]xiefeng401@gmail.com [2]d.p.parsons@massey.ac.nz

Abstract. Today, mobile technology is rapidly affecting our society, with increasing numbers of services supported by mobile phones, including mobile Internet access. However, mobile Internet access performance 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 a new desktop approach to web application development that uses client-side scripting to provide a seamless user application experience and reduce traffic between client and server. In this paper we address the question of whether mobile Ajax provides measurable performance advantages over non-Ajax mobile applications. A study of web application performance over a GPRS network was undertaken, based on comparing an Ajax application and an Active Server Pages (ASP) application with identical functionality. Our results suggest that mobile Ajax over GPRS can reduce the bandwidth requirement by nearly 70% and cut the server's response time in half. Usability testing also suggests that task performance time can be reduced.

Keywords : Ajax performance, Mobile Phone, GPRS network, Mobile Internet

1 Introduction

Over the past few years, an increasing number of people have been using mobile Internet access, but due to the limitations of mobile phone web applications and hardware, it is still very problematical. There are a number of reasons, for example; small screens, high network latency, low bandwidth, interface complexity and so on. To address these issues, we may look to recent developments in desktop browser technologies that might also be implemented in the mobile environment to improve the mobile Internet access experience.

In 2005, Garrett [1] introduced a new web application approach, Ajax (Asynchronous JavaScript and XML), on the desktop. Compared with the traditional web application model, Ajax can significantly increase a web page's download speed in the desktop environment. It has been shown by White [2] and Smullen and Smullen [3] to reduce the data transmission volumes between the server and client device, and also improves the user experience on the desktop. However, 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 to address the question; does mobile Ajax provide measurable advantages over a non-Ajax mobile application? The importance of this study is based on the two major proposed benefits of Ajax, which are seamless updates to the interface and a reduction in data transfer volumes. These are issues that are particularly important in the context of mobile Internet access, where a reduction in transmission volumes when using expensive and low speed connections can be very beneficial, and seamless updates to the interface may provide a better user experience in browsers with limited screen real estate and navigation tools.

The goal of this research is to evaluate the benefits of Ajax over non-Ajax applications, when users access websites via mobile devices through a General Packet Radio Service (GPRS) network. First, we developed two websites with identical functionality, using both an Ajax approach and a non-Ajax approach based on Microsoft Active Server Pages (ASP). These two websites have the same user interface and can be accessed both by desktop and mobile browsers. Second, we measured these websites' performance over a GPRS network based on the data collected from the web server's log files. Finally, we administered some small group usability tests to measure task performance times and user responses.

2 Related Work

Previous related work falls into three main areas of research; the development of the Ajax web interaction model, empirical research into the performance of GPRS networks, and empirical research into the performance of Ajax based web applications when compared to more traditional web application architectures. So far, there has been no previous work on the measurement of Ajax applications using a mobile GRPS platform.

2.1 The Ajax Approach

Garrett [1] defined Ajax as a set of powerful, widely-used, well-known and mature technologies, combining them together to create a new interaction approach:

- Standards-based presentation using XHTML and CSS;
- Dynamic display and interaction using the Document Object Model;
- Data interchange and manipulation using XML and XSLT;

- Asynchronous data retrieval using the XMLHttpRequest object;
- JavaScript binding everything together.

The Ajax web application approach alters two features of the traditional web model to produce high performance and more interactive web applications. First, partial screen updates takes the place of the 'Click, wait, and refresh' user interaction model. Second, the synchronous request/response model is replaced by an asynchronous communication model.

Traditional web pages refresh the whole page, every time some new information arrives from the server. During this time users must wait for the page to refresh. Ajax only updates the part of the user interface on the screen that contains the new information, retaining the current web page in the browser [4].

The other key feature for Ajax technology is an asynchronous communication model, using JavaScript to manage all requests to the server. As a result, requests will continue to be sent out if necessary, but users do not need to wait for the responses. All communication will be done in the background; meanwhile the user can continue to use the Web application online. When the new information arrives, JavaScript will partially update the user interface to provide the user with the latest information so the user can enjoy a seamless browsing experience [4]. In summary, Ajax can theoretically increase a web page's interactivity, speed and usability.

2.2 GPRS network (Bandwidth and Latency)

GPRS (General Packet Radio Service) is the world's most ubiquitous wireless data service, available now with almost every GSM network. GPRS is a connectivity solution based on Internet protocols that support a wide range of enterprise and consumer applications [5]. GPRS can be utilized for services such as WAP (Wireless Access Protocol) access, SMS (Short Message Service) and MMS (Multimedia Message Service), but also for Internet communication services, such as email and web access. Unfortunately, web access over GSM (GPRS) using the TCP/IP protocol is problematical, with users experiencing very poor performance [6]. There are various reasons for this, for example high and variable latency, fluctuating bandwidth, occasional link 'blackouts' [7], packet loss, and link outages. Sometimes, a simple request can take several seconds [8].

It is recognized that the performance issues of GPRS networks are to some extent resolved by the continuing rollout of 3G (third generation) wireless networks. However in this study we have chosen to focus on GPRS networks for a number of reasons. GPRS is the most commonly used network in the world, with the widest coverage, considerably larger globally than 3G. Using this network 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. It should also be noted that network latency does not improve with increased bandwidth, and deploying a 3G mobile phone network is not necessarily the solution to all performance issues [9].

2.3 Ajax Performance on the Desktop

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 [2]. In his study, the Ajax application transferred on average just 27% 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% 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[3]. They compared the client-side performance of a real-life HTML application and an Ajax application that implemented the same user interface. 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%.

In this study we build on this previous work by measuring Ajax performance in the context of the mobile Internet using a GPRS connection.

3 Methodology

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. We developed two IQ test (multi-choice) web applications with the same interface and functionality (Fig. 1), to compare the performance of an ASP based website and an AJAX based website in a GPRS mobile network environment. Although the two applications were very similar in appearance, their architectures were very different, with the Ajax application receiving asynchronous updates to parts of the screen, while the ASP application followed a more traditional 'click, wait and refresh' approach. This meant that, for example, the display of the answers

on the screen (Fig 1) did not require a complete screen refresh for the Ajax version.

To measure the performance of the two applications, we collected data from the server and mobile phone, including bandwidth usage, response times, and mobile unit storage requirements. We also gathered some usability data from the participants.

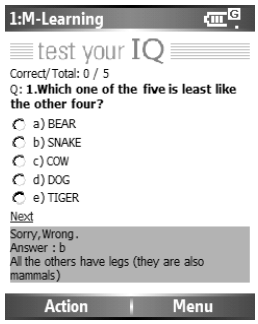

Fig. 1. Web application interface

The experimental system setup used to measure system performance is shown in Fig. 2. We used a laptop connected to an 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 same base station approximately 500 meters from the test site.

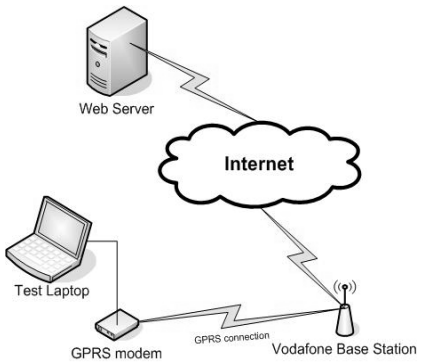

Fig. 2. System configuration for performance testing

4 Data Collection and Analysis

The Web server used for the tests 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 the mobile terminal. These logs, comprising data from over 200 requests, were analyzed using Microsoft Log Parser 2.2.

4.1 The Web Applications

The ASP Web application only contains three ASP pages; a start web page, a question display webpage, and a finish web page. These pages interact with the database to provide dynamic 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, it activates a JavaScript function that sends a request to the server to fetch a new question or answer.

The Ajax application version is similar to the ASP web application. On the client side, it includes three HTML pages, a JavaScript file, a GIF image file, and a CSS file. There are also an ASP file and an Access database running on the web server. The ASP file used in the Ajax application is 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.

These web applications have the same user interface, use the same image file and CSS file and share the same database.

4.2 Web Application Performance

Although the Ajax web application and ASP web application have much in common, they performed quite differently in our experiment, the results of which are now described. The data collected from the IIS web server system log comprised hundreds of entries, so all results stated here are averages.

The first measure was the application response size, the number of bytes sent by the server, which is one of the most useful measurable features of application performance. According to the IIS system log the average ASP application initial load size was about 8.7K. 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 generated pages was 2,305 bytes, so the total bytes that needed to be sent back from the Web server were calculated as follows:

$$8.7\text{K} + (9 \times 2 \times 2{,}305) \approx 50\text{K} \ . \qquad (1)$$

The number of bytes returned for the 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. However while the Ajax application is running, just a few bytes need to be transferred between the client browser and the server. All this data is XML and the average size is only about 555 bytes. 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 therefore calculated as follows:

$$10K + (9 \times 555) \approx 15K \ . \tag{2}$$

Our results show that the Ajax application only needs to transfer about 15K of data from the server to 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.

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. According to the IIS system log, the initial response time for the ASP application was 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 ASP file is 285ms; the server 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

$$9 \times 2 \times 285\text{ms} = 5{,}130\text{ms} \ . \tag{3}$$

The total response time for the whole application was calculated by adding the initial response time to the process response time, so the total web server response time for the ASP application was:

$$1{,}142\text{ms} + 5{,}130\text{ms} = 6{,}272\text{ms} \ . \tag{4}$$

For the Ajax application, the IIS system log shows the initial response time was about 1,378ms (including the first question), and the average server response time for the XML generator ASP file is 210ms. However, this server page only runs once for each question. Therefore for a total of 10 questions, minus the response time for the first question, the process response time is

$$9 \times 210\text{ms} = 1{,}890\text{ms} \ . \tag{5}$$

The total server response time for the whole Ajax application is therefore:

$$1{,}378\text{ms} + 1{,}890\text{ms} = 3{,}268\text{ms} \ . \tag{6}$$

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.

The response size from the web server and the server's task time show us the Ajax application transfers smaller volumes of data between the client and server and has a better response time than the ASP 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 average query size for the main ASP file is about 496 bytes. 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 when users submit their answers. In addition there are other files requested initially by the client, such as image, JavaScript and CSS files, giving an initial request size of 2,428 bytes. Adding the subsequent request sizes for each question gives the following totals

$$2{,}428 + (686 \times 9 + 496 \times 9) = 13{,}066 \text{ bytes} \approx 13\text{K} \ . \tag{7}$$

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

For the Ajax system the average request size was 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 the initial download size of 1,949 bytes to the subsequent request sizes:

$$1{,}949 + 9 \times 377 = 5{,}342 \text{ bytes} \approx 5\text{K} \ . \tag{8}$$

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 5K of data.

4.3 Summary Results

In previous studies, both White [2] and Smullen and Smullen [3] used the following algorithm to define the percentage of Ajax application performance improvement:

$$(\text{HTML}) - \text{Ajax}) \ / \ \text{HTML} \ . \tag{3}$$

According to this algorithm (substituting the ASP data for HTML in the original formula) the Ajax approach in our experiment provides a 48%

Table 1. Performance Improvement.

	Transfer bytes	From formulae	Response seconds	From formula
ASP	63Kb	1 + 7	6,272ms	4
Ajax	20Kb	2 + 8	3,268ms	6
Improvement	68%		48%	

performance improvement in overall response time when using a mobile device over GPRS, based on a reduction in the data transfer volumes (both upload and download) of 68% (Table 1).

5. Usability Evaluation

The goal of this part of the study was to evaluate the Ajax system's performance over a GPRS network by user performance measurement. This was necessary because although we could demonstrate that the Ajax system performed better in terms of data transfer times, we also needed to find out if these differences were significant enough to have an effect on end users. The goal of the usability test was therefore to find out if users showed any differences in task performance between the Ajax and the non-Ajax mobile applications.

In order to simulate a real-life mobile environment as realistically as possible, the usability testing experimental setup was somewhat different from the performance measurement setup. In this system, we used an i-mateTM 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 (2 download channels and 1 upload channel, Coding Scheme 4) was used. All participants used the same mobile phone, connected via the same base station.

In the evaluation, users were asked to complete the quiz using the mobile device. In order to reduce the experimental time, we reduced the number of questions in the quiz from 10 down to 5 for each system.

There were two tasks in this usability test. Task one required users to finish all the multi-choice test questions in the Ajax system, and task two required users to finish a similar set of questions using the ASP system. A training session was provided before users started their activity. The task completion time was used to measure the two different performances (within-subject design).

5.1 Usability Test Results

Twenty people attended this small group usability test, totaling nineteen valid

examples; one participant did not finish the test. Based on the nineteen test examples, applying pair-wise T-Testing to the task completion time, our participants performed significantly better with the Ajax system. 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% faster. The Ajax system therefore enabled much better performance than the ASP system when assessed by the task performance variable.

Although our test sample was too small for reliable qualitative analysis in terms of user preference, responses to the test systems based on a short questionnaire indicated that users noticed there were some differences between the two applications, even though they have the same interface and the same process flow. In our tests, 12 out of the 13 participants who noticed a difference preferred the Ajax mobile learning system.

The results of the usability tests indicate that Ajax can provide better performance in the mobile environment on the user experience level.

6 Conclusion

In our tests we applied the Ajax approach to a mobile web application and compared it with a traditional ASP system. Our results indicate that Ajax 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%, saved 48% of the web server's processing time and reduced submission data from the client to the server by 59%. 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% faster than they did on the ASP 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.

6.1 Limitations and Further Work

There were a number of limitations to this study. Our experimental system was only tested on one GPRS network, 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. Other types of mobile web application may well give different profiles of data transfer. In addition, the test group used in our usability experiment was too small to enable us to draw more generalised 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 it would be helpful to undertake 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 system than the multiple choice quizzes used in this study.

Acknowledgments

Sincere thanks to the Institute of Information and Mathematical Sciences, Massey University, for their generosity in letting us use their computing resources to conduct our experiments.

References

1. Garrett, J.J.: Ajax: A New Approach to Web Applications. Retrieved December 9th 2007 from: http://adaptivepath.com/publications/essays/archives/000385.php (2005).
2. White, A.: Measuring the Benefits of Ajax. Retrieved December 9th 2007 from: http: //www.developer.com/java/other/article.php/3554271 (2005).

3. Smullen, C.W., Smullen, S.A: AJAX Application Server Performance. In: Proceedings of the IEEE SoutheastCon, Richmond, VA, USA. March 22-25 (2007).
4. Wei, C.K: AJAX: Asynchronous Java + XML? Retrieved December 9th 2007 from: http: //www.developer.com/design/article.php/3526681 (2005).
5. GSM Association: GPRS Roaming Guidelines. Retrieved December 9th 2007 from: http: //www.gsmworld.com/technology/gprs/index.shtml (2003).
6. Chakravorty, R., Pratt, I.: WWW Performance over GPRS. In: Proceedings of the 4th International Workshop on Mobile and Wireless Communications Network, Stockholm, Sweden, September 9-11 (2002).
7. Chakravorty, R., Cartwright, J., Pratt, I.: Practical experience with TCP over GPRS. In: Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '02), Taipei, Taiwan, November (2002).
8. Stuckmann, P., Ehlers, N., Wouters, B.: GPRS traffic performance measurements. Proceedings of the 56th Vehicular Technology Conference, Vancouver, Canada, September (2002).
9. Nortel. (2007). Long-Term Evolution (LTE): The vision beyond 3G. Retrieved January 3rd, 2008, from http://www.nortel.com/solutions/wireless/collateral/nn114882.pdf.