

**A comparative study of the network traffic generated from
Traditional Internet Applications versus Rich Internet
Applications**

Submitted in partial fulfilment of the requirements of the Bachelor of
Science Honours degree in Computer Science



RHODES UNIVERSITY

Stuart Thackray

Project Supervisor: Prof. Greg Foster

Project Co-supervisor: Mr. Barry Irwin

Computer Science Department, November 2007

Acknowledgements

This project would not have been possible if it were not for the following people:

- Prof. Greg Foster, my supervisor. For his many hours of help with direction and changes throughout the year.
- Mr. Barry Irwin for his network expertise and the constructive criticism with regard to how tests were run.
- My parents for my funding and supporting me throughout my Honours year.
- The numerous proof readers, special thanks to Thomas and Matthew.
- Additionally I acknowledge the financial and technical support of this project of Telkom SA, Business Connexion, Comverse SA, Verso Technologies, Stortech, Tellabs, Amatole, Mars Technologies, Bright Ideas Projects 39 and THRIP through the Telkom Centre of Excellence at Rhodes University.

Abstract

The Internet was originally designed to enable sharing of documents as static pages of link text, as it has grow need for greater functionality has developed. This need has outpaced traditional Internet applications ability to deliver the functionality required; this has resulted in what are known as Rich Internet Applications. Rich internet applications are able to offer comparable features to those of traditional desktop applications.

This project does a study comparing the network utilization of rich Internet applications and that of tradition Internet applications. To accomplish this two applications each with two versions were developed. The applications are identical in terms of functionality, but differ in their implementation platform. . The RIA has been developed using Flex, whilst the traditional Internet application was developed using a combination of HTML with JavaScript. The reason for this choice is down to the fact that in a continuum of web technologies they are polar opposites.

This work compares the applications base on their TTFB and TTLB with varying number of concurrent users. Additional the quantity of network traffic generated is compared. The results are then summarized, and relevant conclusion are drawn.

Contents

<u>CHAPTER 1 – INTRODUCTION.....</u>	<u>1</u>
1.1.PROBLEM STATEMENT.....	1
1.2.BACKGROUND.....	1
1.3.METHODOLOGY.....	2
1.4.DOCUMENT STRUCTURE.....	2
<u>CHAPTER 2 – RELATED WORK.....</u>	<u>3</u>
2.1 ARCHITECTURE OF RIA AND TRADITIONAL WEB APPLIATIONS.....	4
2.2 TRADITIONAL WEB APPLICATIONS.....	5
2.3 RICH INTERNET APPLICATIONS.....	6
2.3.1 JAVA APPLETS.....	7
2.3.2 FLASH AND FLEX.....	8
2.3.3 AJAX (ASYNCHRONOUS JAVASCRIPT AND XML).....	9
2.4 NETWORK METRICS FOR EVALUATING AND REPRESENTING PERFORMANCE.....	11
2.4.1 LATENCY.....	13
2.4.2 TTFB AND TTLB.....	13
2.4.3 AVERAGE QUANTITY OF DATA TRANSFERRED.....	14
2.5 RELATED WORK.....	14
2.6 SUMMARY.....	15
<u>CHAPTER 3 – WEB APPLICATION IMPLEMENTATION.....</u>	<u>16</u>
3.1 INTRODUCTION.....	16
3.2 THE APPLICATIONS.....	16
3.2.1 TRADITIONAL INTERNET APPLICATION.....	20
3.2.2 RICH INTERNET APPLICAION.....	21
3.3 SUMMARY.....	24

<u>CHAPTER 4.....</u>	<u>25</u>
4.1 INTRODUCTION.....	25
4.2 METHOD.....	25
4.3 ORIGINAL VERSION.....	26
4.3.1 TIME TO FIRST BYTE AND TIME TO LAST BYTE.....	26
4.3.2 QUANTITY OF NETWORK TRAFFIC GENERATED.....	29
4.4 VERSION WITH SEARCH FUNCTIONALITY.....	30
4.4.1 TIME TO FIRST BYTE AND TIME TO LAST BYTE.....	30
4.4.2 QUANTITY OF NETWORK TRAFFIC GENERATED.....	34
4.5 SUMMARY.....	36
<u>CHAPTER 5 – CONCLUSIONS AND FUTURE WORK</u>	<u>37</u>
5.15 CONCLUSIONS.....	37
5.2 POSSIBLE EXTENSIONS.....	38
<u>REFERENCES.....</u>	<u>39</u>
<u>APPENDIX I – GLOSSARY.....</u>	<u>43</u>
<u>APPENDIX II – SCREENSHOTS OF THE TEST PROGRESSION</u>	<u>44</u>
<u>APPENDIX III – PROJECT POSTER.....</u>	<u>47</u>

Chapter 1 – Introduction

1.1 Problem Statement

The primary objective of this project is to compare network traffic generated from traditional web applications versus Rich Internet Applications (RIAs). This has been done through monitoring of TTFB (Time to first byte), TTLB (Time to first byte), and network load with multiple simultaneous users. The methodology used is centred around the construction and testing of two versions of two web based applications. These applications are identical in terms of functionality and workflow but differ in terms of their implementation platform.

1.2 Background

The Internet was originally developed in order to enable sharing of documents as static pages of linked text [3]. Over time, however, a need for greater functionality developed, which has resulted in new platforms evolving to meet this need for content. Rich Internet Applications are just the next step in the evolution, offering similar features and all the functionality of traditional desktop applications.

Research has been done in this area such as an Adobe study that measured the CPU usage of servers between RIA and traditional Internet applications [14], as well as comparisons between RIA. This includes research that has been done by Microsoft in their comparison of an ASP.NET and AJAX [27]. There has, however, been little research into the comparison of network traffic of traditional Internet applications and RIAs; as such, this project aims to delve deeper into this area.

Research has suggested that RIAs can use less bandwidth than traditional applications [2], of additional interest is latency of the platforms. This is due to the fact that whole pages don't need to be refreshed when updating data using RIAs, this may result in smaller data transfers which are needed. Additionally, faster data transfers are possible.

1.3 Methodology

For the comparison of network traffic generated, it was necessary to build applications with the same functionality and workflow using the two web technologies chosen. The RIA has been developed using Adobe Flex Builder 2, the most recent version of Flex Builder available at the start of this project. The traditional internet application was built using a combination of HTML with JavaScript, delivered through the use of a Servlet. The reason for these choices is the fact that in a continuum of web technologies they are polar opposites.

Two version of both the traditional and RIA were developed in order to determine how processing offloading could affect the network traffic. They are based on a pet store because there are numerous blue print applications available from software developers. These versions were then tested based on quantity of network traffic, TTFB and TTLB with varied concurrent users. The results collected were then compared with each other and the research, relevant conclusions were then drawn.

1.4 Document Structure

Chapter 2 offers insight into what Internet implementation platforms are available. This chapter demonstrates the architecture of both tradition and RIA platforms; it goes on to explain the major traditional and RIA platforms and the advantages and disadvantages of these platforms. Special attention is given to the major RIA platforms, in order to give an indication of why the implementation platform was chosen.

Chapter 3 explains how the applications were developed and implemented, and the reasoning behind creating two versions to be compared. An explanation of which pages are compared and why they are compared is explained.

Chapter 4 details how the tests run, what metrics were recorded and what they are designed to measure. Results are displayed in the form of graphs, and patterns are discussed.

Finally, chapter 5 outlines the outcomes of the project and draws relevant conclusions.

Chapter 2 – Related Work

This chapter gives an overview of traditional and Rich Internet Applications, with additional attention given to the major platforms. The advantages and disadvantages of the different technologies are also explained. An explanation of what network matrices may be used to compare technologies and what they are designed to measure will also be covered.

The internet has grown up on the notion of a network of loosely coupled, unintelligent clients that communicate with increasingly intelligent servers by sending requests for pages [3]. The Internet was originally intended to help researchers share documents as static pages of linked text formatted in Hypertext Markup Language (HTML). However web pages quickly evolved to include complex structures of text and graphics, with plug-in programs to play audio and video files or to stream multimedia content. Web developers soon supplemented the basic browser function of rendering HTML by invoking code scripts on the client's computer. These scripts are able to create interface elements such as rollover effects, custom pull-down menus, and other navigation aids. They can also execute UI methods, for example, to validate a user's input in an HTML form [9].

The Web is already the platform for doing business efficiently and quickly. As the penetration of high-speed and broadband Internet access increases, web technologies continue to evolve to deliver new user experiences and increased application utility. RIAs are just the next step in that evolutionary process, this is according to Chris Loosley [9].

2.1 Architecture of RIA and traditional web applications

A typical architecture for RIAs is shown in Figure 2.1, XML is generally used as the data transfer format and is sometimes also used to describe form layouts. In many instances, the client can stay connected to the data source, so a server can update the client in real time a process known as server side pushing. Database access is achieved through the use of web service calls [2].

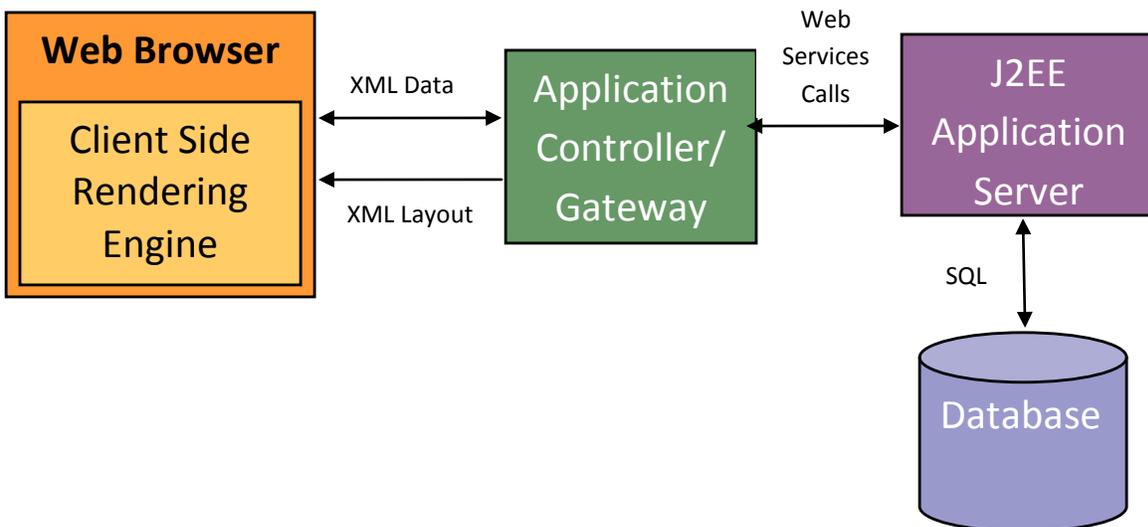


Figure 2.1 - Typical RIA Architecture. [2]

Figure 2.2 shows an example of traditional web architecture. The client would be a web browser in this case. It usually consists of HTML traffic being requested by the client and the server would then return the requested page in this case using another application in order to provide the dynamic content requested.

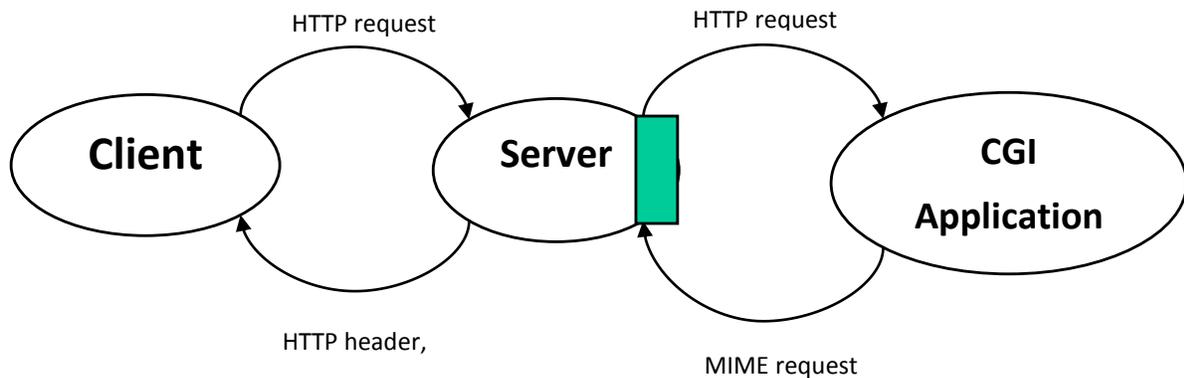


Figure 2.2 - A backend CGI program provides services to the WWW server on behalf of the client. [16]

As the diagram shows the RIA architecture needs a rendering engine on the client side, while in the traditional application this is not necessary due to the fact that traditional technologies only require a browser to display the information.

2.2 Traditional web applications

The major traditional web application platform is HTML and it can be used in conjunction with Cascading Style Sheets (CSS) and JavaScript [12]. CSS allows styling of HTML pages, which helps to ensure a constant look and feel of a website, as they are all able to use the same style sheet for every page. HTML can also be delivered with JavaScript [12]. For example, the model view controller architecture uses JavaScript to generate the presentation layer as well as perform process-intensive tasks such as validations, handling posting, and report execution [21].

HTML-based web applications have become so popular because of the low cost of deployment, the simple architecture, and HTML was trivially easy to learn and use [2]. This is because “elements that can be specified in HTML are familiar to the majority of web users” [12]. HTML

is also well integrated with the reload, back, history, and book marking as it complies with the one URL one page rule. HTML also allows search engines to index content easily [12].

HTML has the biggest reach, due to the fact that it may be rendered by any browser; in addition it can be effectively used by users with slow connections. This is primarily due to it having been standardized and therefore available to users running all types of software and operating systems. HTML can delivered through the use of Servlets, ASP, ASP.NET, JavaServer Pages (JSP), Coldfusion, PHP, and more. Moreover, it is also relatively easy to find people with HTML, and XHTML, programming skills so development costs are low [12].

HTML however exhibits various limitations as the demand to build applications of increasing complexity has continued to outpace the ability of traditional Web applications to represent this complexity [3]. This is due to HTML only being sufficient for relatively static content and trivial tasks, such as form entry, data display and link navigation. Integration with local resources (hardware and software) is nearly impossible [12]. Client requests require full reloads of Web pages in order to update displays this increases download times and server loads. Unfortunately Hypertext Transfer Protocol (HTTP) requests do not support guaranteed message delivery, or guaranteed order of message delivery, nor server-initiated communications (push). HTML components are also limited to those available through HTML markup which is very limited in comparison with desktop applications. Web page code is exposed and this makes it very easy to copy [12], this results in their websites user interface being used in illegal activities such as phishing [22].

2.3 Rich internet applications

Rich internet applications have been implemented in many websites. Some of the more notable of these are Google AdSense, Flickr, Napster, Wikipedia, blogging, Facebook and upcoming.org [4]. There are many different RIA platforms, the major ones being AJAX, Flash with Flex [11], Applets, Windows Presentation Foundation (WPF), OpenLaszlo, and NexaWeb [13]. The main

contenders are Adobe’s Flash and Flex suite, Java applets, and the collection of Web technologies known as AJAX, a term coined in 2005 by Adaptive Path’s Jesse James Garrett [9]. These will be further discussed below.

When evaluating RIA platforms, Ramirez Design [12] suggest that RIAs should be evaluated on user experience, deployment and reach, processing, interface components and customization, back-end integrations, unique features, future proofing and staffing and costs [12]. This criteria is used in the following explanation of the advantages, disadvantages and the problem areas each technology best deals with.

2.3.1 Java Applets

Java Applets have the advantage of offering a rich set of features for engaging interactions, including drag and drop, animation, and other User Interface (UI) elements found in traditional desktop applications. They enable real-time updates and validations as the user completes forms. Java applets run on a virtual machine; either the Java plug-in or Java Virtual Machine (JVM). This allows applets to run unchanged on any device that is able to run the JVM. Applets have the ability to go beyond HTTP for remote data requests and responses. A plug-in only requires one installation as they can be cached. Applets allow very good, processor-intensive, visualization rendering which allows for good interactive and dynamically generated graphs, charts, etc. Applets enable the server to offload processing onto the clients machine, which can save both CPU overhead and network bandwidth. Staffing is not a problem due to there being many developers who are familiar with Java [12].

Security is a concern however as applets are able to write to the local disk if they are digitally signed. However, as applets run in a “sandbox” few other security concerns exist; as there exist strict rules on how applets are able to interact with your computer and the network. Due to the size of applets, latency is high due to relevantly large initial download and plug-in launch.

Additionally linking to, saving, and book-marking content can be tricky, and search engines do not index applet content [12].

2.3.2 Flash and Flex

Flash and Flex offer developers a rich set of features for engaging interactions including drag and drop, animation, transparency, layering, audio and video streaming. As well as allowing for real-time updates and validations as user completes form fields [12]. Like applets, the Flash player is also able to run identically on all the major operating systems. According to Adobe, Flash Player 8 has broad reach and is installed on over 90% of Internet enabled desktops as of February 2007 [19]. It also offers near seamless upgrade process for the plug-in, whilst still supporting previous versions. Through the use of built-in ActionScript dynamic processing can be achieved without the need for page reloads. It also natively supports vector graphics, streaming audio and video which makes it popular choice for many websites, and games. There are also many additional Flash UI toolkits and components available from 3rd parties, [12] these UI components are often better than the UI applets offer [14]. Reduction of server loads is also achievable due to processing on the client side and the fact that it only requires pure data requests after the initial load [12]. Socket connections allow server initiated communications to client application. Flash can be used as the presentation layer with common server technologies such as Java Servlets, JSP, PHP, ASP, and many more. Decreased costs of developing and testing are able to be achieved due to the fact that they are able to be done once for all platforms. Developers familiar with JavaScript will find learning ActionScript easy [12].

Flex differs in the way applications are developed, as it uses both ActionScript and MXML. MXML is used mainly to declaratively lay-out the interface of the application, as well as implement complex business logic and behavior of the RIA [29]. Flex Builder 2 IDE offers developers an incredibly productive environment for building RIAs [7]. Additionally it allows for the programming to be improved through the use of “state-of-the-art coding and debugging environment; intuitive layout and styling” [7]. Through effective use of this development environment you are able to develop web applications that have the features and functionality of

traditional desktop applications [6]. Flex application are compiled down into byte code (Flash file) with a HTML wrapper.

Through the use of Adobe Integrated Runtime (AIR) both Flash and Flex programs can be deployed as desktop applications [33]. This has the added benefit of allowing them to store information on the client's machine and reducing latency as there is no need for the Flash file to be downloaded.

As with applets there is the potential for high initial latency due to relevantly large initial download and plug-in launch. Unfortunately Flash is not fully integrated with the browser environment, for example reload, back buttons and bookmarking, although workarounds do exist. As with applets, Flash content is not indexed by search engines, but workarounds exist to make this possible. Development time can take longer because everything must be created from scratch: concept of pages, links, browse history, scale, etc [12].

2.3.3 AJAX (Asynchronous JavaScript and XML)

AJAX is a popular technology used on many websites. The most notable are Google Maps, Gmail or Microsoft's Outlook Web Access. [13] Compared with classic page-based web applications, AJAX introduces a new web presentation tier model that is different in three important ways, namely: use of a client-side engine as an intermediate between the UI and the server; user activity leads to JavaScript calls to the client-side engine instead of a page request to the server; XML data transfer between server and engine [10].

AJAX incorporates standards-based presentation using XHTML and CSS which improves the user interface; as well as providing dynamic display and interaction using the Document Object Model (DOM); data interchange and manipulation using XML and XSLT. AJAX allows

asynchronous data retrieval through the use of XMLHttpRequest; using JavaScript to bind everything together [9].

Due to large interest in AJAX there is a lot of community development, therefore are numerous frameworks to choice from [12]. These range from Grade A to Grade E, Grade A frameworks implying they have the broadest support of browser compatibility [18]. According to Bict [18] the following fall into Grade A AJAX frameworks: AJAX Dojo Toolkit, Echo 2, Google Web Toolkit, JavaScript/Ajax Toolbox [23], jQuery, Moo.fx, Prototype, Rico, Sardalya, Script.aculo.us, Tacos, TurboWidgets, TwinHelix, Wicket, Yahoo User Interface Library [18]. This is important due to the fact that incompatibilities of the JavaScript object models supported by various browsers can make coding quite a chore, [13] which leads to higher development costs due to sophisticated, branching code required to support multiple browsers [12].

A major advantage of AJAX is that it enables backchannel communication in Web applications so that only small portions of web pages need to be updated in response to user activity [13]. Web developers however ultimately need a complete, robust AJAX focused development stack that will marry client side and server side programming in a neat package [13].

AJAX applications often break the "one URL equals one resource" model the web has long employed. This condition can break the semantics of the browser "back" button from the user's point of view, make book marking problematic, can make caching difficult or impossible, render log files less useful for user tracking, search bots less effective than with HTML and makes disabling access harder to code for [13]. This means that content may be incorrect when users click reload and back buttons [12].

AJAX Web applications are as insecure as traditional Web applications, as they are far too trusting of user inputs. SQL injection or other data manipulation attacks are common in poorly

coded AJAX applications, and maybe more so because there is greater reliance on client activity [13]. A weak security posture is most evident, in that many AJAX applications are delivered with unobfuscated and unoptimized JavaScript. This condition allows easy access to intellectual property by unscrupulous developers and sets no barriers for code inspection by potential intruders and consequently, can use more bandwidth [13]. Also due to different parts of AJAX code having to be interpreted, it can be many thousands of times slower at data access [14].

AJAX however enhances user experience with better, faster forms, than are available with traditional web applications. AJAX enables real-time updates and validations as user moves from field to field or even after each character entry [12]. AJAX also reduces server loads due to processing on the client side and pure data requests after the initial load [12].

Another disadvantage of AJAX is that content requested through XMLHttpRequest objects cannot be indexed by search engines or read by some screen readers. In addition to this it does not have a socket connection which means the server can not initiate communications to client application [12].

2.4 Network metrics for evaluating and representing performance

Measuring the performance of a distributed application is necessary to determine how quickly users can achieve their goals, and to discover how a system behaves under increasing load. The first focus is directly on the users' experience, the second investigates underlying server behaviors that, in turn will determine what users experience. Network performance of websites can be determined when you measure server and user side latency [5], TTFB, TTLB, concurrent users and network utilization, CPU utilization of different web based applications.

The diagram below illustrates the communication models of traditional and rich web application [9].

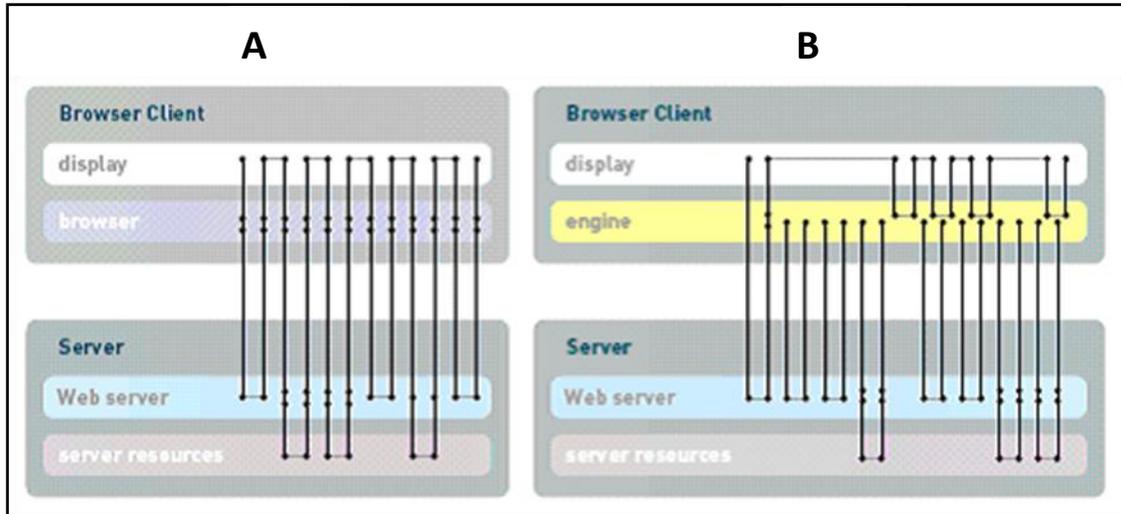


Figure 2.3 - The communication model of traditional web application (A) and RIA (B) [9]

Reviewing Figure 2.3 (A), we can characterize the response time of a traditional Web application as the time to complete the synchronous round trip of:

Click(C)=>Browser(B)=>Request(Q)=>Server(S)=>Response(R)=>Display(D)

According to Loosley [9], network measurements are surprisingly more accurate when it is done near or on the server for traditional web applications.

However, as we see in Figure2.3 (B), the client-side engine breaks apart this cycle into two separate cycles operating asynchronously—a user/client-engine cycle, and a client-engine/server cycle [9]:

Click(C)=>Engine(E)=>Display(D)

Request(Q)=>Server(S)=>Response(R)

There being two cycles might be as foreground (C-E-D) and background (Q-S-R), this complicates how to measure responsiveness [9]. To measure responsiveness therefore it requires that tools must drive the C-E-D cycle, not the Q-S-R cycle, because RIAs can generate back-end traffic in response to any user action, and not only when the user clicks [9]. This technique can also incorporate user think time if required.

2.4.1 Latency

User perceived latency is the time spent by the user waiting for the web page that they have requested [5]. Impatience with poor performance is the most common reason that visitors terminate their visit to web sites, [5] this can be attributed to the 8 second rule [6]. The 8 second rule states the average web customer will wait about eight seconds for a page to download [6].

2.4.2 TTFB and TTLB

The TTFB measurement is the amount of time from when the client sends the request (GET command) until it sees the first byte back from the server. This is a single round trip across the Internet [7]. TTFB is important due to the fact it can be used to give an indication of user wait time, and to discover how a system behaves under increasing load [9]. Time to last byte is the measurement of time from when the first byte arrives and ends when the last byte of the file arrives at the client [7].

In [7] it was concluded that, HTTP/1.0 interacts badly with TCP. It incurs frequent round-trip delays due to connection establishment, performs slow start in both directions for short duration connections, and may incur heavy latency penalties due to the mismatch of the typical access profiles with the single request per transaction model [7]. However this has been solved with HTTP/1.1 [20]. AJAX provides similar TTFB and TTLB when compared to HTTP, [13] however platforms such as applets and flash require a larger initial downloads which drastically reduces on subsequent communication [12].

2.4.3 Average quantity of data transferred

Average quantity of data transferred is important consideration for users that are limited either by speed and/or quantity of internet traffic they can transfer. RIAs promise that smaller amounts of data needs to be transferred per session [7]. They do this by only requiring a part of a web page to be reloaded, for example with AJAX, and after the initial load of Java Applets and Flash applications [12].

2.5 Related Work

Adobe conducted a study comparing a traditional web application built with JavaServer pages (JSP) with an identical application built with Adobe Flex. Server CPU utilization was measured as it is an important indication of the scalability, and number of concurrent users a server can handle. The study revealed that both server requests and CPU usage were dramatically decreased in a Flex Application, as shown in figure 2.4. The Flex application used .8% average server CPU utilization when under heavy load with minimal peaks of 50% while the JSP interface under the same load used 21.2% server CPU utilization with peaks up to 100% [14].

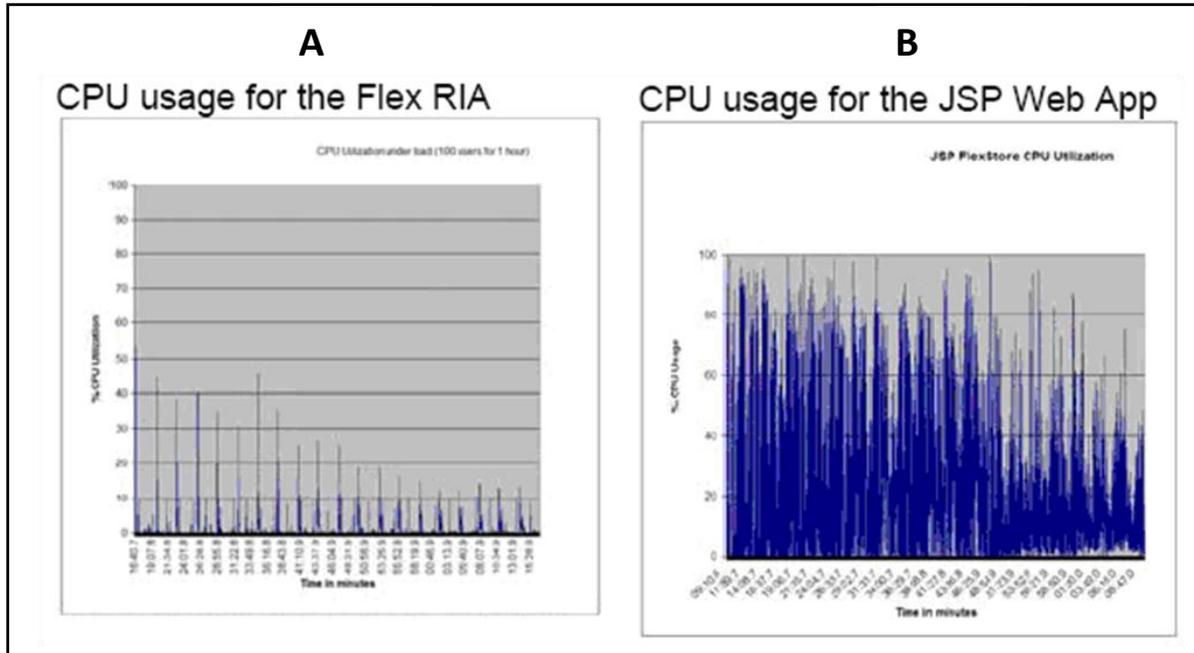


Figure 2.4 - CPU usage of the Flex RIA (A) with 100 users over one hour and CPU of a JSP Web Application (B) under the same load [14].

2.6 Summary

Rich Internet Applications are the next step in a logical progression towards better web application interfaces. However RIAs are not meant to replace HTML, as it is still the preferred model for storing and organizing content for the web. However, HTML based web application interfaces do have many limitations. RIA tools are capable of offering an entirely new paradigm for modern web application interfaces, and translate into more engaging and effective user experiences and better customer retention and service [14]. RIAs also allow reduction of network traffic and the ability to offload processing power to the client which significantly increases the scalability of the web application. Security can also be improved for example applets and flash don't allow copy and paste of code, this reduces the likeliness of SQL injection and other malicious attacks.

Chapter 3 – Web Application Implementation

3.1 Introduction

In this chapter will explore the applications that have been developed as a test-bed for evaluation of the platforms. We will explore the differences in the applications between the two versions, which have been developed. Advantages and disadvantages of developing a RIA and the traditional Internet application will also be discussed.

The applications were developed from the start based on a pet store. The pet store was developed due to blueprint applications being available in many of the rich Internet platforms namely .NET [26], Flash MX [25] and Java [24]. Additionally there have been tests in order to compare these applications [26]. It was indicated however that the results were biased based on the use of a test bed that suited to a specific platform, which affected the results [27].

3.2 The Applications

For the purpose of this project two applications have been built in the traditional and rich Internet platform, each application has two versions in order to test what effect offloading processing to the client would have on the network traffic. As mentioned the applications are modeled on a pet store and have been built from the ground up, using a similar UI to that of the Java pet store [24] as well as a modified version of the pet store's database. Some functionality has been omitted, for example the shopping cart, this decision was taken to ensure both versions were optimized and the shopping cart is seen as unnecessary for this project.

The traditional page was built using a combination of HTML and JavaScript using the Netbeans 5.5 IDE. The RIA was built using a combination of MXML [29] and ActionScript using Adobe Flex Builder 2 IDE with database access achieved using XML outputted via a Servlet. HTML

was predominantly used for the traditional site due to it being suited best for thin clients, with Flex is being used as it is on the opposite side of the web continuum as it is best suited for powerful clients.

Two versions were developed and their differences are displayed in Table 3.1. The first version of the pet store was developed in order to access and display information from a database containing pet information. The traditional and RIA have been required to include the same information, and access the same database to ensure consistency of the comparison. Both sites use the Java database connectivity (JDBC) class, in order to facilitate database access.

A second version of the pet store was developed not only to access and display information but also to include search capability this added facility was order to compare the ability of each platform. The same database is used; images are increased between the platforms however.

The architecture of the application is shown in Figure 3.1, bold is used to display the progression the test will follow, and the black box is a page that is only included in the version with search functionality. The differences in the versions and platforms for the test progress are show in table 3.1 and screenshot are shown in appendix B.

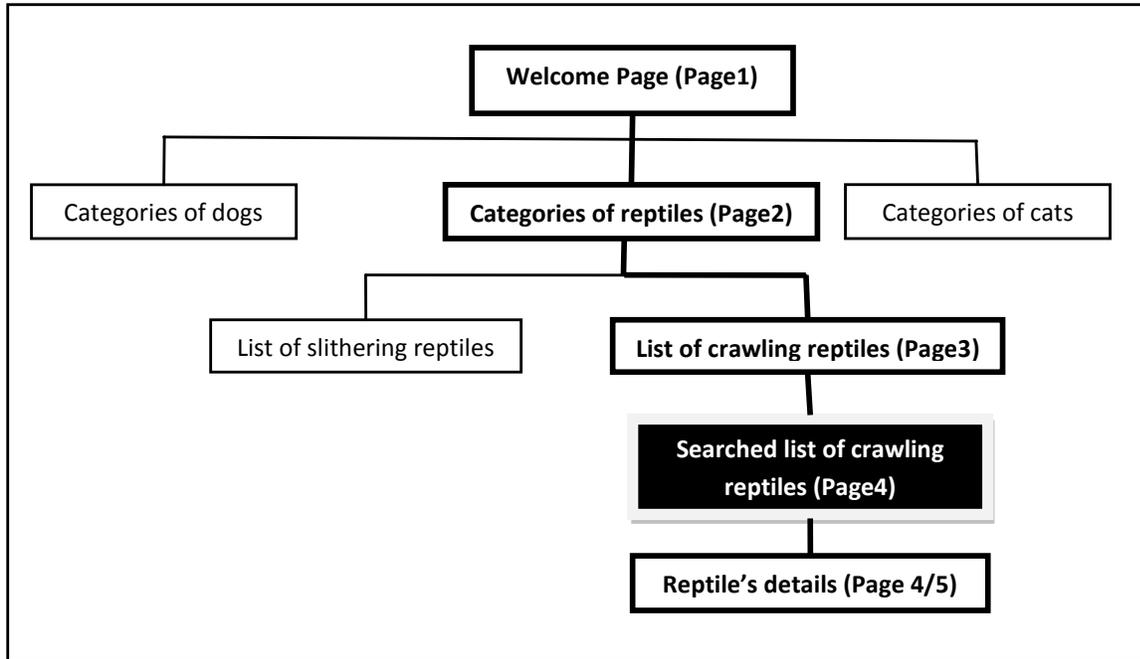


Figure 3.1 – Architecture of the two web applications

	Original Version		Version with Search functionality	
	Traditional	RIA	Traditional	RIA
Page 1	Welcome page (6.02KB)	Flash file (232KB) with HTML wrapper	Welcome page (same as original)	Flash file (240KB) with HTML wrapper
Page 2	HTML Page	XML document	same as original	same as original
Page 3	HTML Page	XML Document	HTML page with an added image	same as original with an added image
Page 4	HTML page	XML document	HTML Page	No data transferred
Page 5	N/A	N/A	same as original's page 4	same as original's page 4
Database	Identical in all versions			

Table 3.1 – Differences between versions and platforms

The architecture of the two web applications do not differ considerably, as can be seen in Figure 3.2. This demonstrates the architecture used to construct the two platforms; as can be seen the traditional site requires one step less to accomplish its task. The only difference is the RIA requires that a Flash Player be installed on the client’s machine.

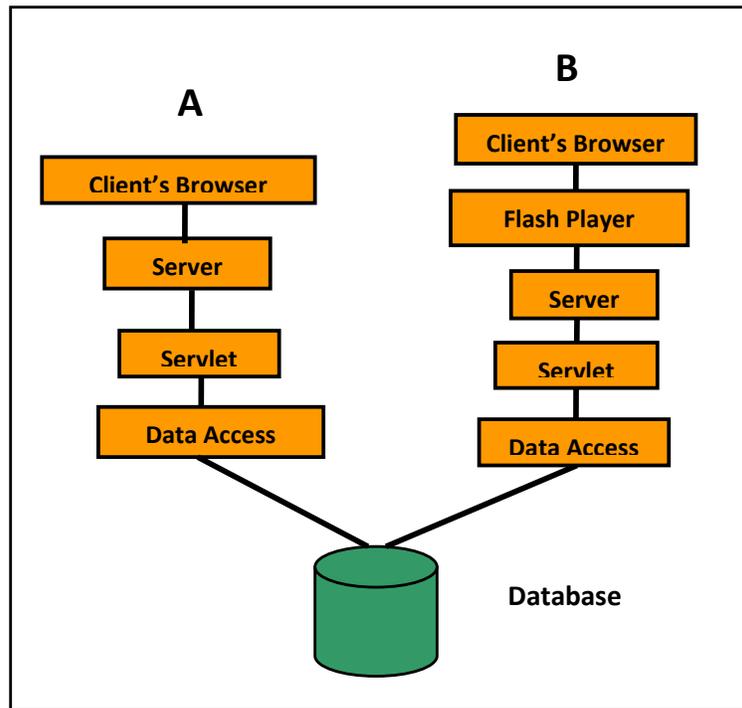


Figure 3.2 – Architecture of the traditional (A) and rich (B) internet applications

Servlets have been used to answer client requests; a Servlet is a module of Java code that runs in a server application [30]. They are typically used to include processing and/or storing of submitted data, providing dynamic content (returning results of a database query) and providing state information [30]. Servlets have been chosen over CGI as they offer numerous advantages namely, Servlets run in a single process which stays in memory between requests, there is always a single instance to answer all requests, this instance is threaded, and lastly they can be run in a sandbox [30]. For the traditional Internet application the servlet was used in order to process requests, generate the page’s source code as well as constructing the Structured Query Language (SQL) statements. For the RIA the servlet is just required in order to process the requests and create SQL statements.

The data access layer is used to control the access to the database; it does this by accepting SQL statements, controlling the opening and closing of the database (which in this case is a Microsoft Access database) and queries the database. The data access class varies for the traditional and rich Internet applications, but performs the same functions, that is formatting and outputting the results of the SQL query. For the traditional Internet application this is in the form of a HTML table, whilst the RIA this is in the form of an XML document.

3.2.1 Traditional Internet Application

The Traditional Site was built using HTML and JavaScript, with requests and responses being controlled by a Servlet and was developed using the NetBeans IDE. JavaScript is used to facilitate the roll over buttons for the different categories of animals; additionally in the second version it is used to facilitate the searching capability (Figure 3.3). This is accomplished by the user filling in the text boxes corresponding to the price range they are interested in viewing, these values are then passed as parameters and a new page displaying the animals that fall into this category are displayed.

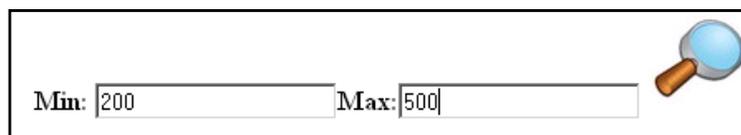


Figure 3.3 - Traditional Site's filter functionality

During development of the sites it was found coding HTML was nominally easy to code, however adding dynamic content does become more complex. Ensuring cross compatibility of HTML code can be without difficulty through the use of any of the numerous tools and IDEs that provide this functionality. Due to both HTML and JavaScript being mature technologies there is a lot of code examples, explanations and demonstrations freely available. There are few tags

available which makes code easy to write as well as reducing the complexity. However the powers of the tags are limited therefore workarounds are often required.

JavaScript allowing dynamic, visual features to be added but comes at the cost of having to debug the code which can only be done at runtime. This is a consequence of JavaScript being an interpreted, loosely typed, object-oriented programming language that is mainly used on web pages and runs in the browser [18]. However JavaScript has a number of disadvantages namely it has been implemented in browsers differently and can be disabled in the browser which results in pages not being displayed as expected. Graphical users interface although of limited importance in this project do not have the broad range of features that are offered by RIAs platforms.

3.2.2 Rich Internet Application

The Rich Internet Application was built using ActionScript and MXML in Adobe Flex Builder 2 IDE, database access is achieved using a Servlet to output XML. MXML was used mainly to declaratively lay-out the interface of the application, as well as implement some business logic and behavior. The RIA works differently from the traditional site due to the fact that the majority of the site's components are delivered in the Flash file. After the Flash file is transferred the only interactions required with the server are requests for images and XML.

For all but the first and last page DataGrids (Figure 3.4) are used to display the data received from the server, which is in the form of XML (Listing 3.1). XML is a widely used method of data transfer by RIAs, Flex is no different and offers powerful functions that allow XML manipulation. A power default feature that Flex's DataGrid is that sorting of rows can be done by simply clicking either title bar. The power of DataGrids has further been improved in Adobe Flex Builder 3 Beta [28], which supports hierarchical data, and basic pivot table functionality.

```
...  
<animal>  
  <ProductID>reptile01</ProductID>  
  <ProductName>Slithering Reptile</ProductName>  
</animal>  
...
```

Listing 3.1 – extract of XML produced to fill DataGrid

Product ID	Product Name
reptile01	Crawling Reptile
reptile01	Groomed Cat
reptile01	Hairy Cat
reptile01	Large Fish
reptile01	Slithering Reptile
reptile01	Small Fish
reptile02	Crawling Reptile
reptile03	Slimy Reptile

Figure 3.4 - DataGrid

The RIA’s last page works differently from the traditional site, as the full address of the image is not delivered to the client. It however computes the address, for example if the name is “foo.jpg” then the image URL requested would be “images/foo.jpg” in this case seven bytes can be saved but with considerably longer URLs and more images greater bandwidth would potentially be saved.

The version of the RIA with search capabilities (Figure 3. 5) has been achieved using a slider bar which filters the XML. This is done through the slider bar calling an ActionScript method when either of them are moved. The ActionScript method filters the XML, thus removing the need for a page reload and updates in real time.



Figure 3.5 – RIA’s filter functionality

There are numerous advantages of developing RIAs using Adobe Flex Builder 2 IDE; one being the graphical manipulation of objects; accomplished through the preview capability. The preview capability allows for the positioning of components through the use of graphical manipulation. ActionScript libraries are another advantage as they are powerful and extensible and may be used to improve both presentation and performance of a web site. States and their transitions are also important features which allow separation of concern to be addressed; allowing each page to be changed with little to no effect on other pages. Numerous layout, chart, data and control objects are available and are easily changeable and extensible, these components are comparable with those of desktop application. The client processing power for tasks such as filtering, and validation of inputs both reduces the load on the server and network traffic. This results in the ability to increase the number of concurrent users.

Flash does not handle any database access which is positive in terms of division of responsibility, although developing the complete application this has been a drawback. This is due to the lack of help and examples in order to retrieval and display of dynamic information for an intermediate user, for what is a common task. However this problem easily solved after consultation with other reference material [31] and has been addressed in Adobe Flex Builder 3 Beta [28].

3.3 Summary

Two identical applications in terms of functionality have been built. Neither the traditional technologies nor Flex are perfect and are not able to offer what traditional desktop applications can. Flex does however offer significant improvements and easy of design especially with the user interface, changes to the site are also a lot easier to accomplish. There are also libraries of common task such as validation of fields, for example email addresses validation which simplifies the development of sites.

Traditional sites do however have the advantage of being simple therefore easy to program and are familiar to the greater number of web users. Addition or removal of code has a larger effect on the traditional site than on the Flex IDE. The testing of JavaScript and HTML browser compatibility testing is another drawback, which is not a problem for Flash it is browser and operating system independent as the Flash player can be run on all operating systems.

Chapter 4 - Network Traffic Comparison

4.1 Introduction

This Chapter covers the results of measuring the TTFB, TTLB and the quantity of network traffic generated of the sites built. The TTFB and TTLB are measured in order to give an impression of a user's wait time; this is non-trivial as user experience improves as these figures drop [5]. Quantity of network traffic is measured as it is important to many users that have limits on bandwidth and/or speed of traffic. TTFB and TTLB have been measured using Microsoft web application stress testing tool [34], it is able to calculate the average TTFB and TTLB as well as simulate concurrent users. Quantity of network traffic was measured using Wget [32] in conjunction with Wireshark [35]. These tests were run using a client and server both with Core 2 duo processors on a 100Mbs network; neither of these components was taken near their limits as this could produce inaccurate results, due to additional processor usage and other network traffic.

4.2 Method

The Microsoft web application stress testing tool was used for the measurement of the TTFB and TTLB with varying numbers of concurrent users. TTFB and TTLB are important metrics as they are used to give an indication of user wait time, and to discover how a system behaves under increasing load [9]. A traversal of the websites (for both versions and both platforms) was recorded, without capturing any cookies or headers, as the testing did not require authentication, tracking, maintenance of specific information about users or simulation of any particular browser. For the measurement, each of the scripts generated was given a 200ms delay between request for a pages resources and the following page. This was found to give enough time for the page to be delivered to the client, and therefore the server was only working to generate and deliver the resources for one page at a time. The tests were run for a duration of five minutes resulting in an adequate number of requests for each resource. These tests were run for 1, 2, 3, 5, 10, 15, 20 concurrent users, however it was discovered that the server was unable to handle up to 20 concurrent users without crashing or producing an error.

Quantity of network traffic tests were conducted in order to prove or disprove the research that suggests RIAs requires reduced amounts of network traffic [7]. This was tested using Wget to retrieve a page's resources which was accomplished page by page, based on the script recorded by the Microsoft web application stress testing tool. Page by page basis was used in order for the transmission control protocol (TCP) as well as other protocol's overhead to be taken into account as this ensures realistic results. Wireshark was used in order to capture the network traffic and filter it to give the number of bytes the page required to load. This repeated numerous times in order to ensure that the results are accurate.

4.3 Original version

4.3.1 Time to first byte and time to last byte

The TTFB and TTLB for the original versions of both HTML and Flex, which does not include search functionality, this is displayed in figures 4.1 through 4.4 for the four pages tested using a maximum of 15 simulated users. This is due to the fact that the server is only capable to handle a maximum of 15 simultaneous users for the traditional internet application without errors or the server crashing. The reason for this is that the traditional internet application requires greater server processing.

As shown for the first page (figure 4.1), the RIA and the TTFB is comparable with the traditional site, but the TTLB is far larger in this version. This is a result of the Flash file size which is considerably larger (226 KB larger excluding headers) than the traditional Internet application requires.

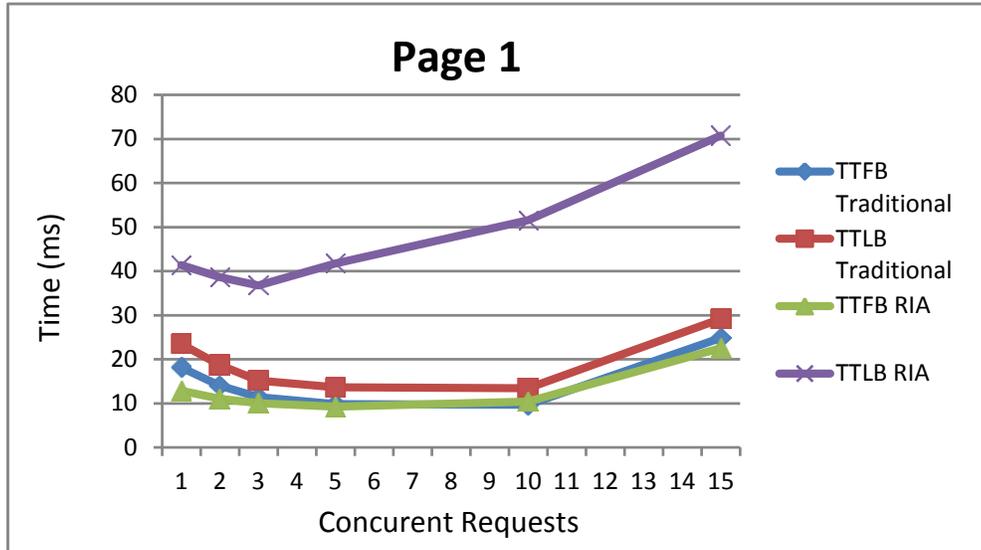


Figure 4.1 – TTFB and TTLB Comparison of the original versions of the sites for page 1

Page 2 (figure 4.2) of the original site displays the different sub categories that pets can fall into. As shown the page’s weight is not large enough to show any discernable difference between TTFB and TTLB, and is just shown for completeness. Due to the lack of page weight a large variance in the traditional application measurements can be seen.

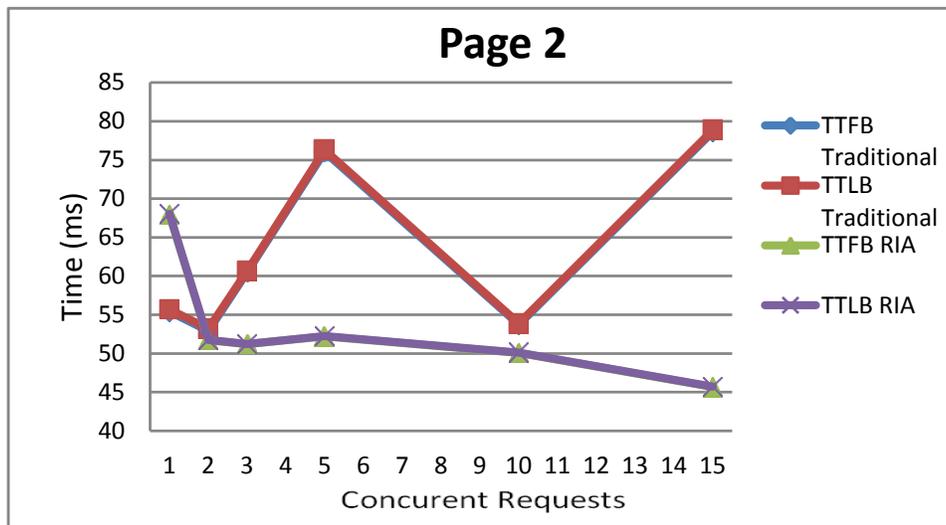


Figure 4.2 – TTFB and TTLB Comparison of the original version of the sites for page 2

Page 3 (Figure 4.3) is a navigation page and displays information regarding a specific sub category of pets available. As shown, the metrics stay relatively flat for the RIA but increase for the traditional site as concurrent users increase; this is largely due processing required by the server. The RIA is transferred faster due to it being computationally simpler and less bandwidth intensive. It is computationally simpler as not a whole page needs to be deliver instead an XML document is all that is required.

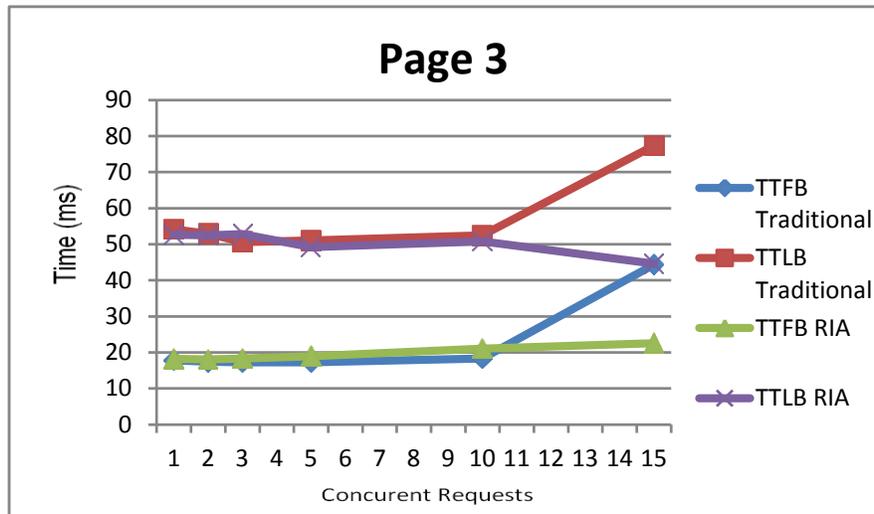


Figure 4.3 – TTFB and TTLB Comparison of the original version of the sites for page 3

Page 4 (Figure 4.4) is a details page which gives the details and explanations of a specific pet. Although there is a large difference between the TTFB and TTLB of the RIA still gets delivered faster than the traditional internet application. Additionally the traditional internet application exhibits a faster escalation in these times suggesting that with more users the response time would deteriorate.

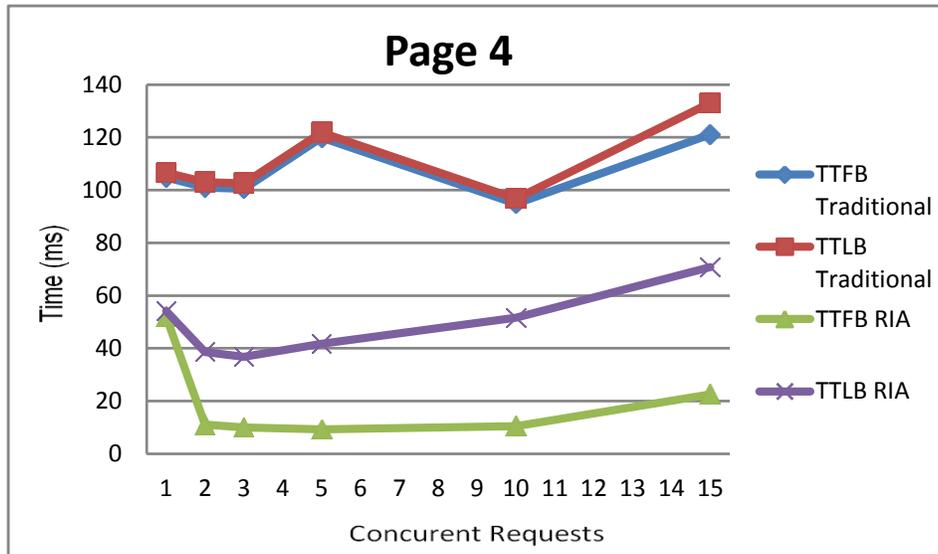


Figure 4.4 – TTFB and TTLB Comparison of the original versions of the sites for page 4

4.3.2 Quantity of network traffic generated

The quantity of network traffic can be seen in figure 4.5, this displays the difference in network utilization of the traditional sites in comparison to the RIA for the original version. As can be seen in figure 4.5, the RIA requires that considerably greater traffic is transferred for page 1 and this is due to the fact that the Flash file is noticeably larger than the traditional site. However as shown, the subsequent page request's traffic is minimized. This results in the RIA requiring that 327.8% more traffic be transferred on the first page, however subsequent pages average traffic quantity is reduced by 25.8%. Overall after the loading of four pages, total network traffic for the traditional internet application is 132.6 KB whilst the RIA required 361.6 KB. This results in approximately 12 pages having to be viewed in order for traffic to be equal after the load of the Flash file.

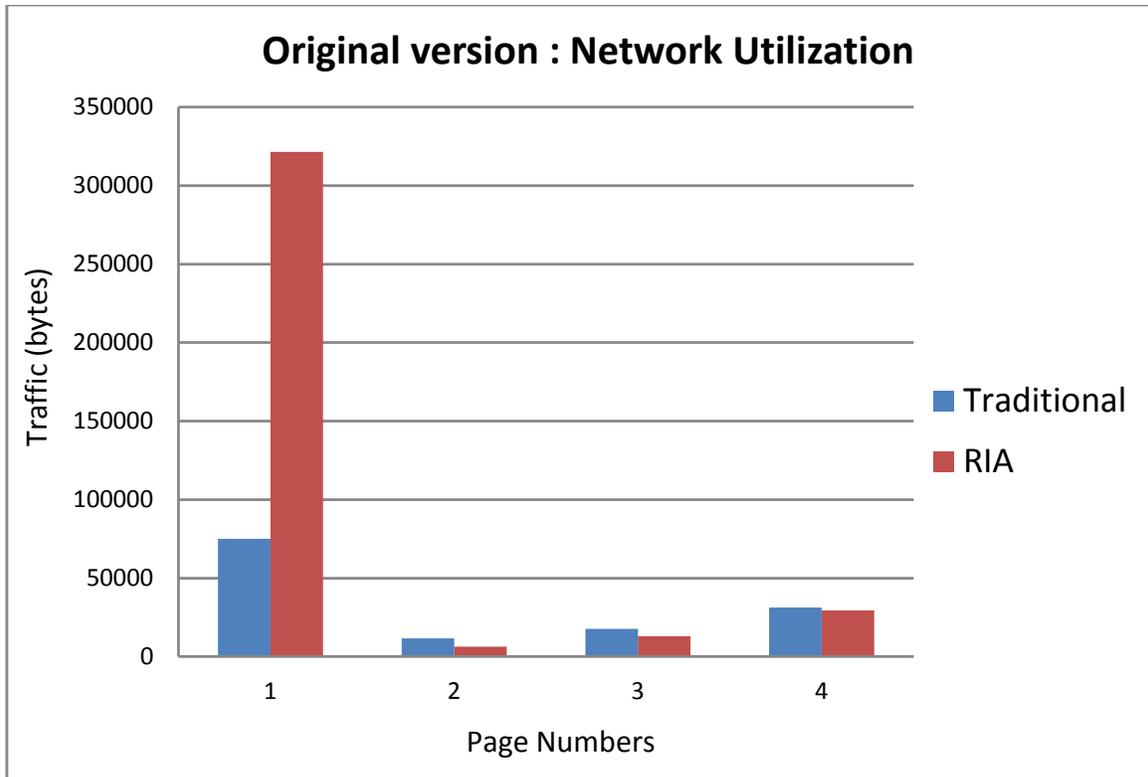


Figure 4.5 - Network Utilization of the original version traffic comparison.

4.4 Version with search functionality

4.4.1 Time to first byte and time to last byte

This version contains search functionality, and the TTFB and TTLB are shown in figures 4.6 through to figure 4.10, and the five pages were tested due to the search capability having been added.

This version's page 1 (figure 4.6) corresponds to the graph in the original version; however the RIA's TTLB escalates faster due to the extra load put on the server as a result of the larger Flash file size. Again, the Flash file's size results in the difference between the TTFB and TTLB being considerable.

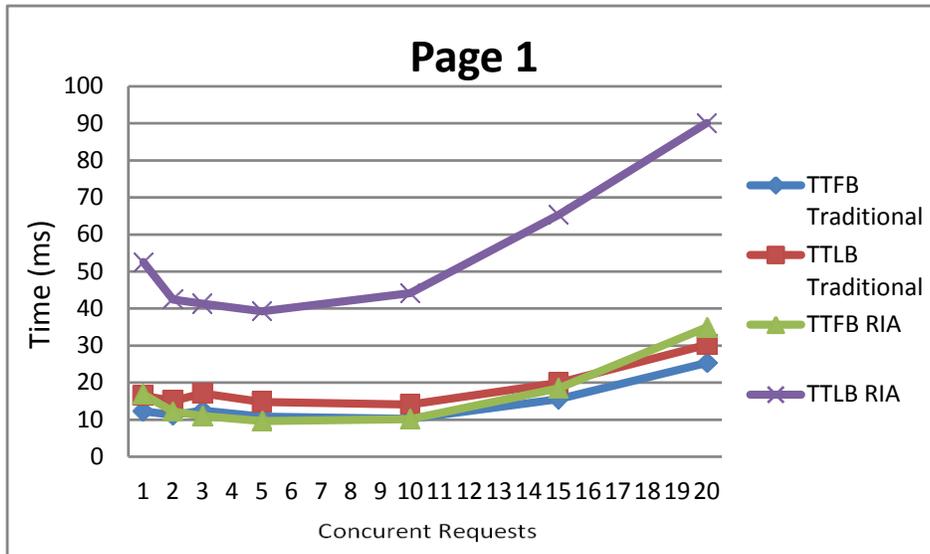


Figure 4.6 – TTFB and TTLB comparison for the versions with search functionality for page 1

Figure 4.7 show this version’s Page 2 , as with the same graph for the original version the weight on this page is too small to show any discernable difference between the TTFB and TTLB. This again results in the graph showing no disenable pattern, and is shown for completeness.

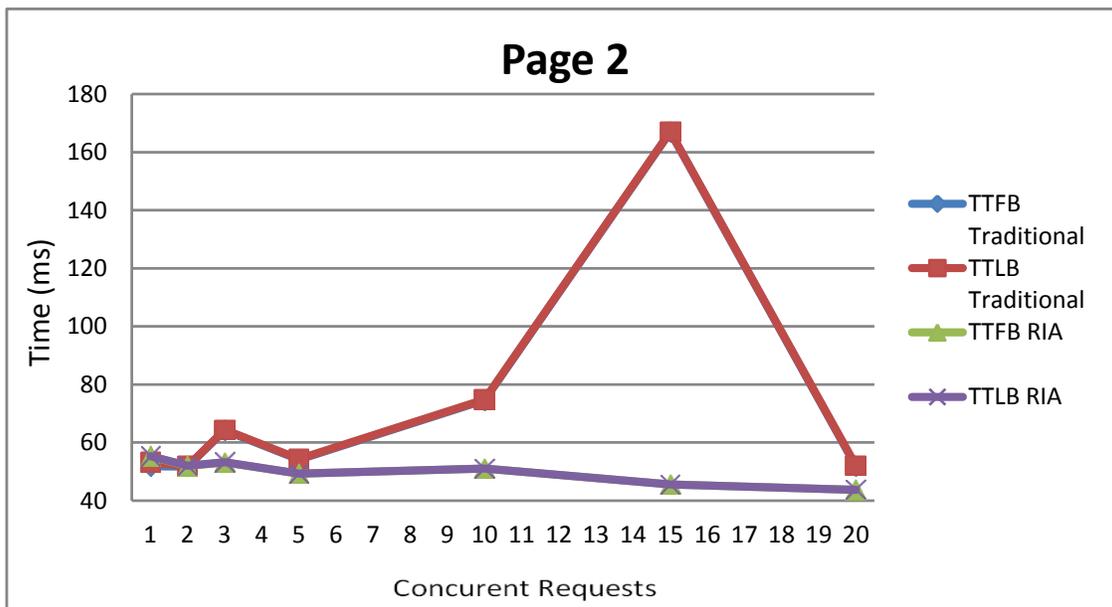


Figure 4.7 – TTFB and TTLB Comparison of the versions with search functionality for page 2

Page 3 (figure 4.8) demonstrates a relatively linear line for both platforms. However the traditional internet application shows an inconsistency at 10 concurrent users for the TTLB and TTFB. This inconsistency was a result of one of the images taking considerable longer to be transferred than it did on other pages, as only average time was available it is unclear if every request it took as long. As is also shown the TTFB of the RIA is considerably faster than the traditional internet application however the TTLB is still comparable.

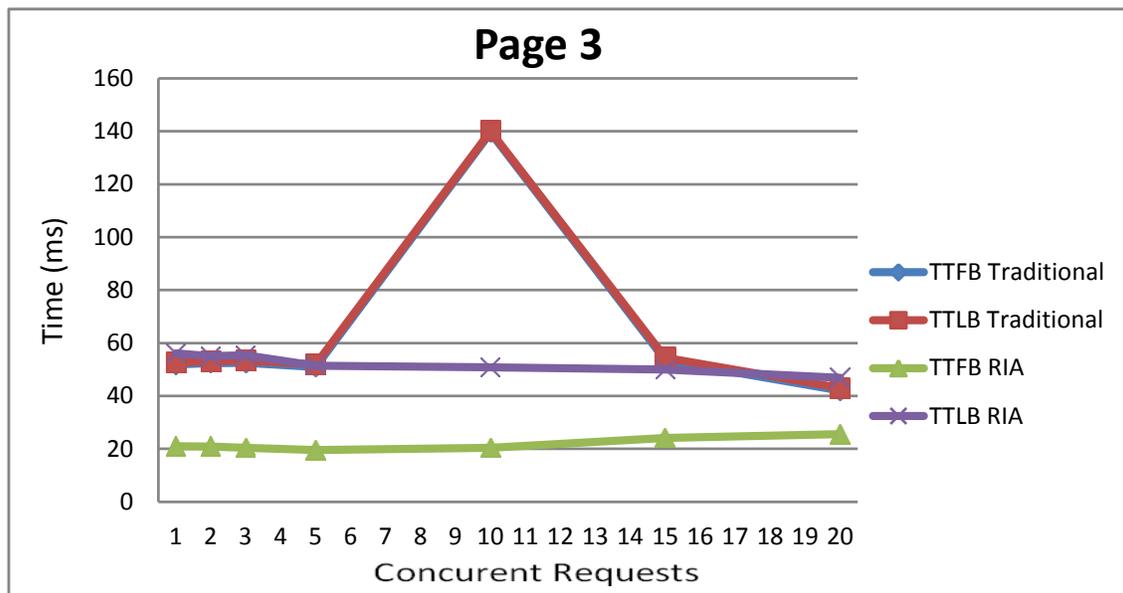


Figure 4.8 – TTFB and TTLB comparison of the versions with search functionality for page 3

Page 4 (figure 4.9) takes advantage of the search capability, it shows similar results for the traditional site compared to that of page 3; though through the use of ActionScript the RIA is able to bypasses any data transfer. This addition results in processing being passed to the client machine and data may be filtered repeatedly without generating any data requests.

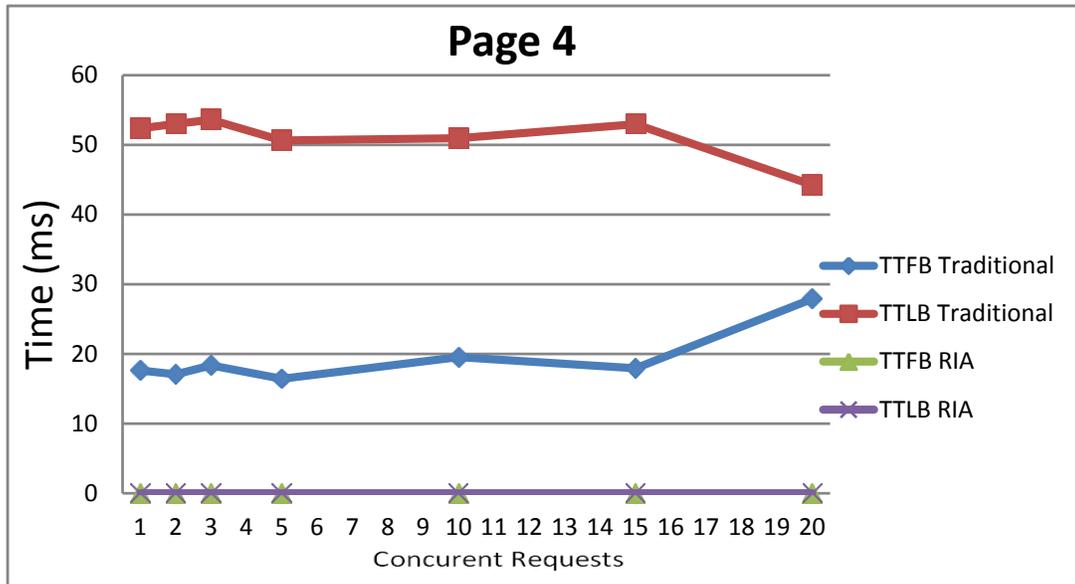


Figure 4.9 – TTFB and TTLB comparison of the versions with search functionality for page 4

Page 5 (figure 4.10) is the same page as is seen in the original version’s page 4 and therefore exhibits the same trend; that is, the traditional site’s TTLB and TTFB escalates rapidly whilst the RIA shows little variation with the increasing number of concurrent users.

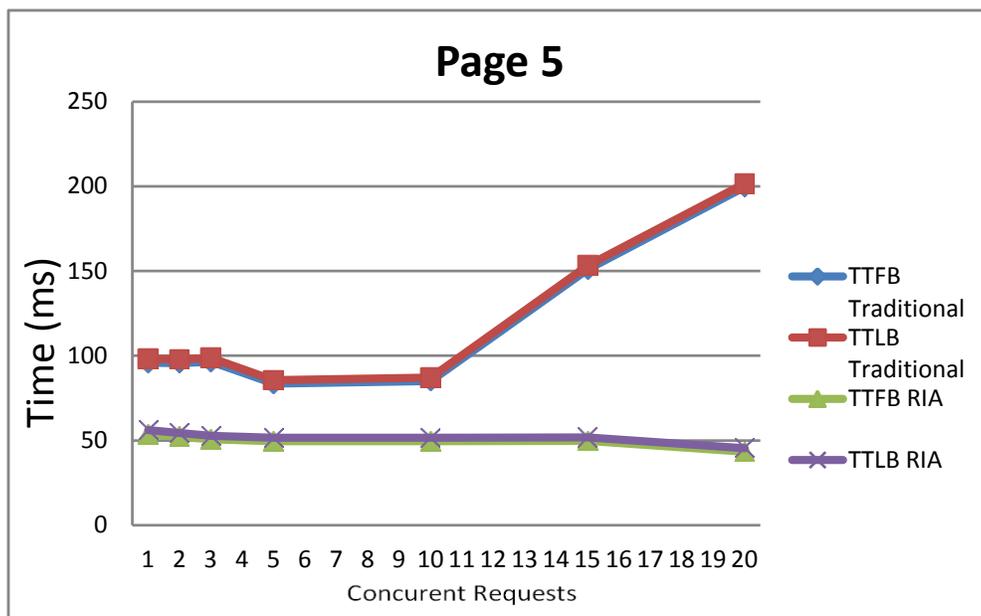


Figure 4.10 – TTFB and TTLB Comparison of the versions with search functionality for page 5

Overall the TTFB of both platforms shows that the RIA's requests for data are able to be delivered faster due to the reduced processing the server requires. Additionally the RIA is faster in terms of TTLB, except for the first page, when compared to the traditional internet application. This can be attributed Flash file size in the first page being significantly larger. When playing Flash files the user is in fact informed of the loading of the file and its progression, which reduces the risk of them abandoning their session due to the extra load time required. Through the running of these tests it was discovered the server could handle at least five more concurrent users for the RIA in both of the versions. This is the result of the traditional site processing having to be done solely by the server, whilst with the RIA this processing is offloaded to the client. This suggests that RIAs are significantly more scalable for the hosting of dynamic sites than otherwise maybe be possible with traditional technology.

4.4.2 Quantity of network traffic generated

For page one (figure 4.11) of this version, the RIA is 3.6% (11682 bytes) larger than the original version this is a result of the Flash file being delivered with search functionality. The search (page 4) using Flex results in no requests for data being necessary which saves considerable bandwidth. In this case on the first page the RIA requires 342.7% more data due to the Flash file being larger; however the saving is increased significantly with the subsequent pages requiring 42.9% less traffic. Overall for the five pages loaded the total traffic transferred for the traditional internet application is 149.2 KB, whilst the RIA required 384.4 KB. This results in approximately 8 pages needing to be loaded for the network traffic quantity to be equal.

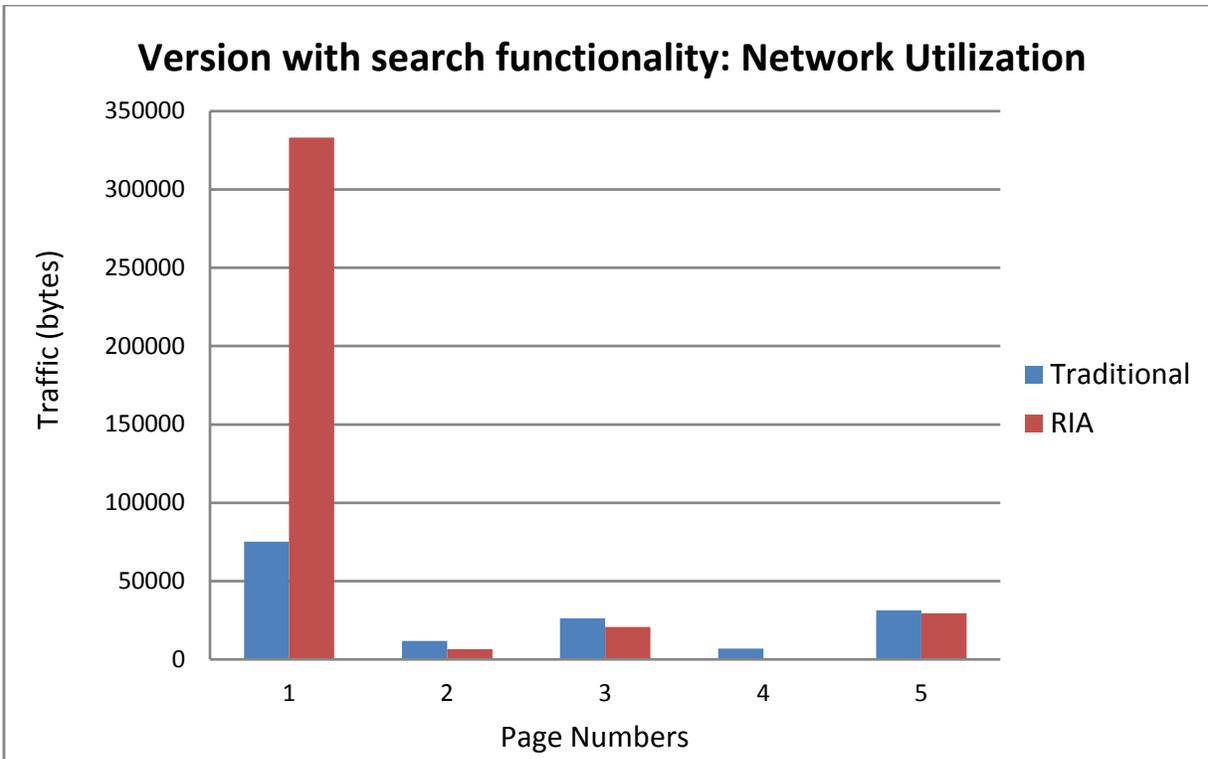


Figure 4.11 Network Utilization of the two version 2 sites

Comparing the two versions suggests the more processing of data on the client side offered by Flash would allow considerable greater saving of network traffic than shown here. Another aspect that would reduce bandwidth is caching; dynamic requests can not be cached, but the Flash file, which is the biggest contributor to the RIA traffic, can be stored in cache. Dynamic requests for the RIA include all data requests for the information regarding the pets, whilst the traditional internet application all requests for pages excluding the welcome (first) page are dynamic requests.

4.5 Summary

This chapter showed that the Flex implementation required significantly greater bandwidth to load, although subsequent requests required less bandwidth. It was shown that after the initial load, faster interaction was possible for the RIA, therefore making it ideal for any site where user interaction is prolonged. Additionally caching can improve this as the Flash file is not dynamic itself even though the data it requests may be. As the Flash file is large it takes considerable time to load especially with numerous concurrent users however the Flash player displays the progression of this loading time, which can minimize user annoyance. Once this load is completed lower TTFB and TTLB are obtained and this gives the user faster interaction. This results in greater scalability of web applications owing to them offering offloading of processing to the client.

Chapter 5 – Conclusions and Possible Extensions

5.1 Conclusions

This work compared the network traffic of a RIA pet store written using ActionScript and MXML and a traditional internet application written using HTML and JavaScript. A pet store was developed due to blueprint applications being available in .NET, Java and Flash MX. Two versions were developed in order to test how offloading processing to the client would affect the network traffic. The table (5.1) summarises the findings in terms of a comparisons between the two.

	HTML with JavaScript	FLEX
Scalability	Inferior	Superior
TTFB	Inferior	Superior
TTLB	Superior for first page Inferior for other pages	Intially Inferior due to Flash file Subsequently superior
Quantity of network traffic	Generally superior	Inferior for non dynamic data Generally superior for dynamic data
Result of caching	Superior for caching static data but not dynamic data	Superior as largest contributor is the Flash file whih is static

Table 5.1 – Summary of findings

Research [12] (chapter 2) suggested TTFB and TTLB would be significantly larger for the RIA for the intial loading of the Flash file compared to the traditional internet appliation, this was proved to be correct. Additionally it was suggested that subsequent requests this figure drops would significantly drop and the RIA was found to faster than the traditional application.

Research [7] suggest that the quantity of network traffic generated should be smaller for the RIA than the traditional internet applicataion. However in testing this was found not to be true, with the RIA requiring approximately 12 pages to be viewed for network traffic to be equal in the

original version of the applications. For the version with search functionality this dropped to 8 pages requiring to be loaded for network traffic to be equal this is due to the data manipulation that can be offloaded to the client. This suggests the more processing that can be offloaded to the client the more network traffic can be saved.

RIAs scalability is improved, this was shown by the fact that the server was able to handle at least five more concurrent users without crashing or producing errors than was possible with the traditional internet application. This is attributed to the load on the server being reduced as less processing is required from it.

5.2 Possible Extensions

Possible extensions to this project may include

- Comparing other RIA platforms such as AJAX and/or applets, based on network traffic in order to determine if any of the other platforms are able to produce less network traffic for 4 or 5 pages of the pet store.
- Addition of a shopping cart for the pet store in order to monitor the user interactions. This would allow the platforms to be compared based on their user interface and this would determine which platform would therefore be best for an online retailer.
- Comparison of a dedicated server's CPU usage between different platforms and the effect the number of concurrent users has on this. This would give an idea of how many users a server would comfortably handle without error.
- Comparisons of the platforms support for multi media content such as video and audio content, and how this support is achieved that is through plug-in or natively through built in classes.

References

References

Note: As a result of this project covered relatively new technologies there are a high proportion of web references. Additionally a broad amount of web technologies and history were covered which has resulted in a large number of references being required.

[1] Macromedia Inc. Developing Rich Internet Applications with Macromedia MX 2004. August 2003.

[2] O'Roule, Cameron. A look at Rich Internet Applications. July 2004.

[3] Duhl, Joshua. Rich Internet Applications. November 2003.

[4] O'Reilly, Tim. What is Web 2.0, 30 September-2005.

[5] Marshak, Marik and Levy, Hanoch. Evaluating web user perceived latency using service side measurements. 7 August 2002.

[6] Zone Research. Need for speed II. June 2001.

[7] Md Habib, Aham and Adams, Marc. Analysis of Sources of Latency in downloading web pages.

[8] West, Peter; Foster, Greg and Clayton, Peter et al., Content Exposure of Slide Show Presentations for selective downloads and annotations via Mobile Devices.

[9] Loosley, Chris. Rich Internet Applications: Design, Measurement, and Management Challenges. 2006. http://www.keynote.com/docs/whitepapers/RichInternet_5.pdf. [Accessed: 27 June 2007]

[10] Backbase. Rich Internet Applications "AJAX and beyond". 2006. http://www.backbase.com/#home/essays/001_ajax_and_beyond.xml. [Accessed: 20 June 2007]

[11] Webster, Steven and McLead, Allistar. Developing rich clients with macromdia Flex. April 2004.

References

- [12] Ramirez Design. Web Application Solutions: A designers guide. February 2004.
- [13] Powell, Tomas. AJAX is the future of Web Application development. 17 July 2006.
- [14] Benjamin Wigton. Rich Internet Applications for Revolutionary Interface Design. July 2004.
- [15] Webster, Stevens and McLeod, Alistar. Actionscript Design Patterns for Rich Internet Applications Development. 14 October 2003.
- [16] Thompson, Craig and Hansen, Gil. Current Web Architecture. 1996.
<http://www.objs.com/survey/WebArch.htm>. [Accessed: 5 April 2007]
- [17] Fain, Yakov. Rich Internet Applications – State of the Union. 13 February 2007.
- [18] Rood't, Bict. The effect of Ajax on performance and usability in web environments. 31 August 2006.
- [19] Patrick, Ted. Rich with Reach. 17 March 2007. <http://www.onflex.org/ted/2007/03/rich-with-reach.php>. [Accessed: 5 June 2007]
- [20] IETF Technical Report. Hypertext Transfer Protocol. <http://tools.ietf.org/html/rfc2616>. [Accessed: 2 June 2007]
- [21] Sage Software. Sage Accpac 500 ERP - Process Server.
http://www.2020software.com/products/Sage_Accpac_500_ERP_Process_Server.asp. [Accessed: 27 July 2007]
- [22] Anti-Phishing Working Group. What is Phishing and Pharming?
<http://www.antiphishing.org/>. [Accessed: 26 July 2007]
- [23] RHAPTOS: Connections' Software and Documentation Site. AJAX DHTML JavaScript Toolkits <http://rhaptos.org/devblog/bnwest/AJAX%20DHTML%20JavaScript%20Toolkits>. [Accessed 27 July 2007]

References

- [24] Sun Microsystems. Java Pet Store Blueprint. <https://blueprints.dev.java.net/petstore/>
[Accessed 11 October 2007]
- [25] Adobe. Pet Market Blueprint Application. <http://www.adobe.com/devnet/blueprint/>
[Accessed 11 October 2007]
- [26] Microsoft. Pet Store. Using .NET to Implement Sun Microsystems' Java Pet Store J2EE BluePrint Application. <http://msdn2.microsoft.com/en-us/library/ms954626.aspx>
- [27] Wampler, Dean. Cat Fight in a Pet Store: J2EE vs. .NET. 28 November 2001.
<http://www.onjava.com/pub/a/onjava/2001/11/28/catfight.html> [Accessed 11 October 2007]
- [28] Adobe Labs. Adobe Flex Builder 3 Public Beta 2.
<http://labs.adobe.com/technologies/flex/flexbuilder3/> [Accessed 12 October 2007]
- [29] Coenraets, Christophe. An overview of MXML: The Flex markup language. 29 March 2004. <http://www.adobe.com/devnet/flex/articles/paradigm.html> [Accessed 12 October 2007]
- [30] Zeiger, Stefan. Servlet Essentials. 4 November 1999.
<http://www.novocode.com/doc/servlet-essentials/> [Accessed 1 November 2007]
- [31] Lott, Joey; Schall, Darron and Peters, Keith. ActionScript 3.0 Cookbook. October 2006.
- [32] GNU. Wget. <http://www.gnu.org/software/wget/> [Accessed 4 November 2007]

References

[33] Chambers, Mike. Introducing Adobe AIR beta 2. 30 September 2007.

http://www.adobe.com/devnet/logged_in/mchambers_air_beta.html [Accessed 3 November 2007]

References

[34] Microsoft. Web Application Stress Tool. 26 December 2002.

<http://www.microsoft.com/downloads/details.aspx?familyid=e2c0585a-062a-439e-a67d-75a89aa36495&displaylang=en> [Accessed 4 November 2007]

[35] SourceForge. Wireshark. <http://www.wireshark.org/> [Accessed 4 November 2007]

Appendix A – Glossary

AIR	Adobe Integrated Runtime
AJAX	Asynchronous JavaScript and XML
CCS	Cascading Style Sheets
CGI	Common Gateway Interface
DOM	Document Object Model
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JVM	Java Virtual Machine
JDBC	Java Database Connectivity
JSP	Java Server Pages
RIA	Rich Internet Application
SQL	Structured Query Language
TCP	Transmission Control Protocol
TTFB	Time To First Byte
TTLB	Time To Last Byte
UI	User Interface
XML	Extensible Markup Language

Appendix B – Screenshots of the test progression

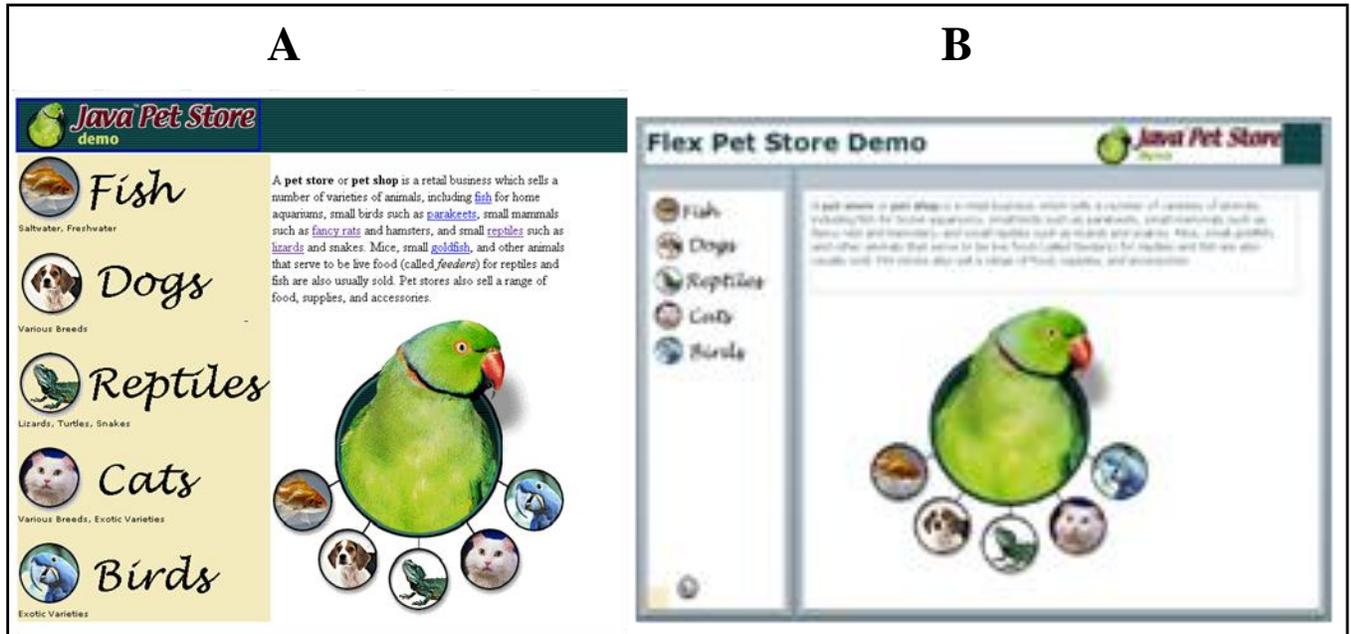


Figure A.1 – An example of page1 for both versions of the Traditional (A) and RIA (B)

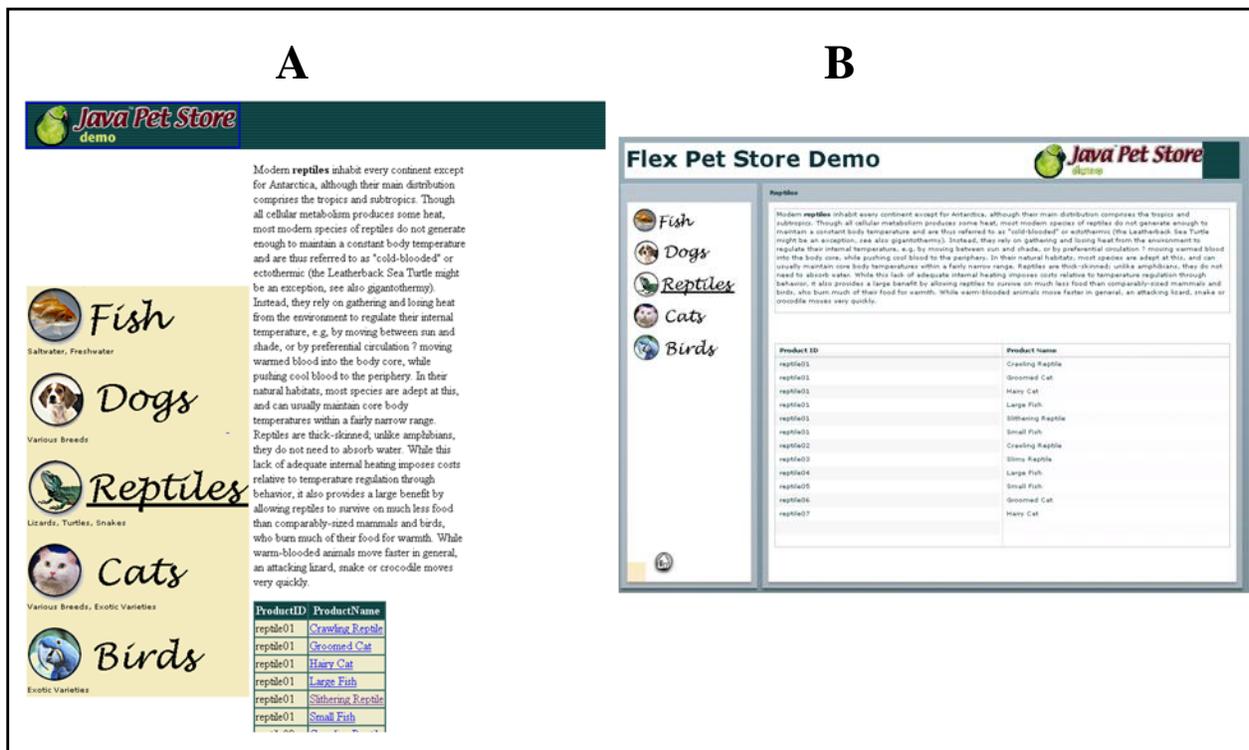


Figure A.2 – An example of page 2 for both versions of the Traditional (A) and RIA (B)

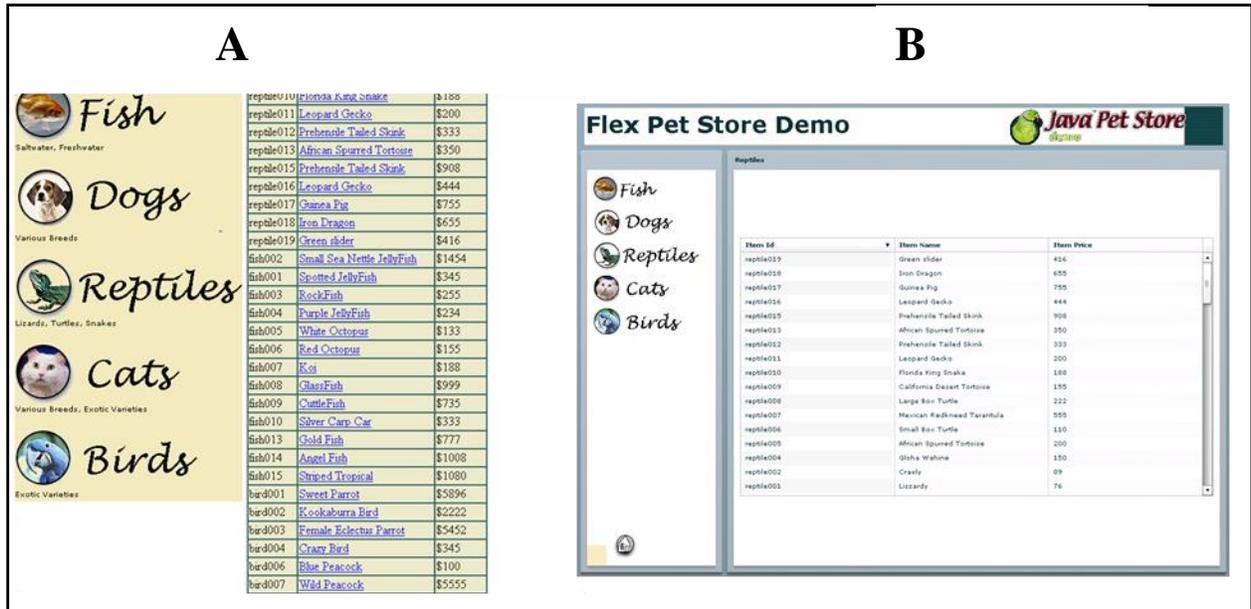


Figure A.3 – An example of original version page 3 for the Traditional (A) and RIA (B)

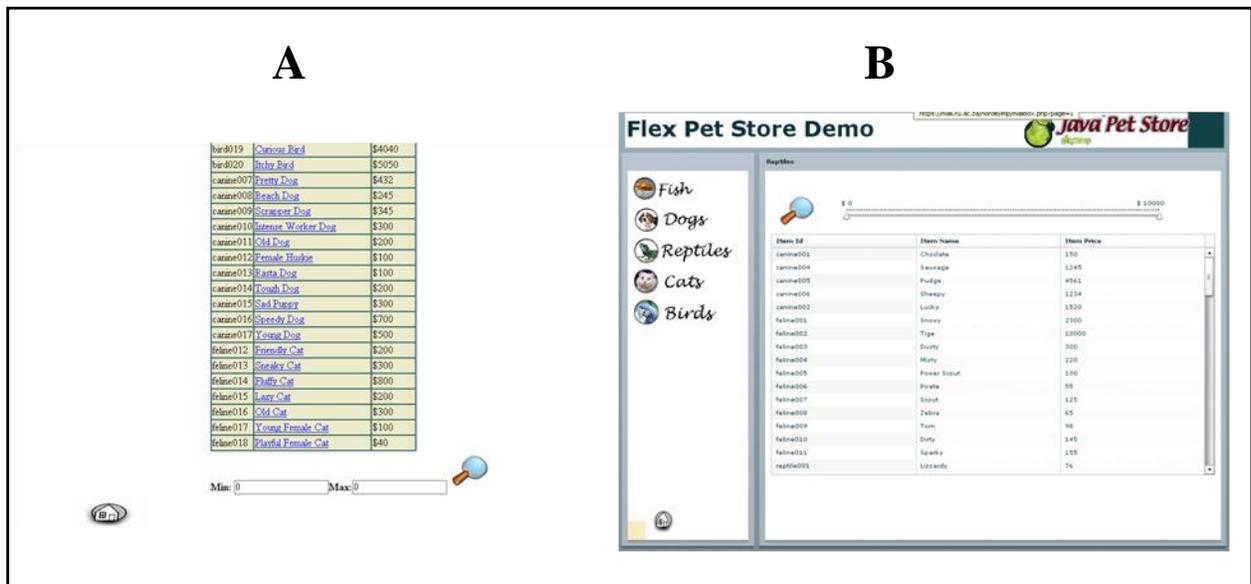


Figure A.4 – An example of the version with search functionality page 3 and page 4 for the Traditional (A) and RIA (B)

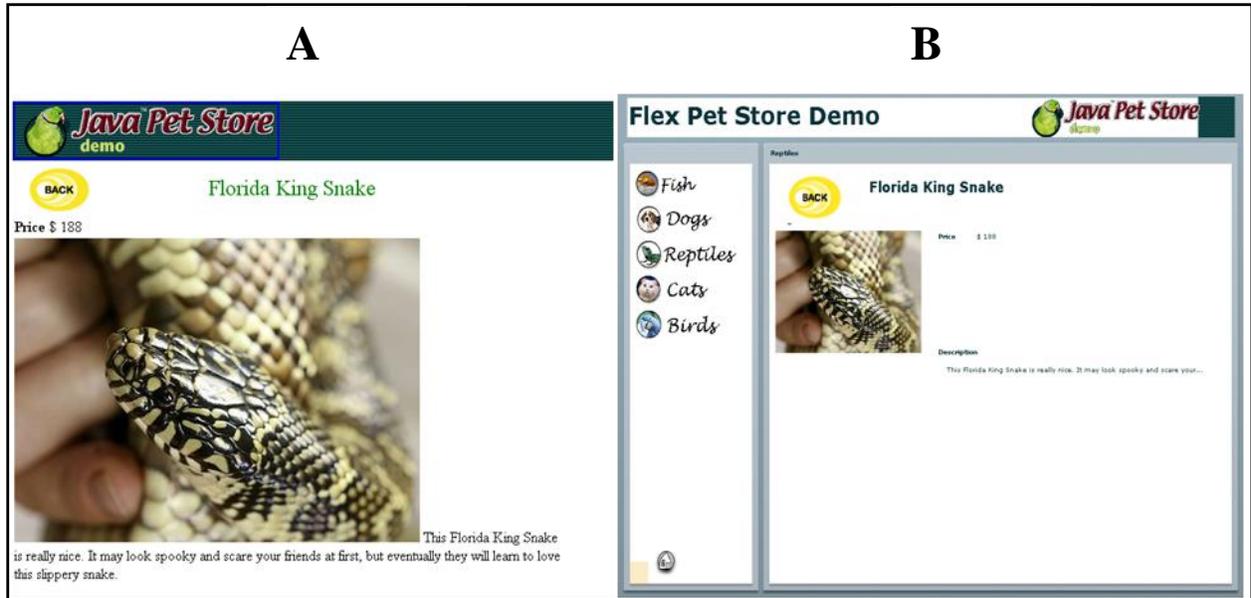
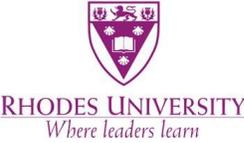


Figure A5 – An example of original version’s page 4, and the second version’s page 5 for the Traditional (A) and RIA (B)

Appendix C – Project Poster



Comparative study of the network utilization of traditional and rich internet applications

 **STUART THACKRAY**  **G03T2382@CAMPUS.RU.AC.ZA**
 **HTTP://WWW.CS.RU.AC.ZA/RESEARCH/G03T2382/**

INTRODUCTION

This project aims to compare the network traffic generated from a traditional web application and Rich Internet Application (RIA)

The metrics being used are server and user side latency, Time to First Byte, Time to Last Byte, concurrent users and the total quantity of data transferred in the differing platforms.

METHODOLOGY

Two applications identical in terms of functionality and workflow will be built.

The traditional web application using HTML and JSP delivered via a servlet has been built.

The RIA is being built using MXML and ActionScript and will be deployed as a flash file.

Stress as well as other testing will then be simulated in order to get comparative data.

Rich Internet Application (FLEX)



Flex does not access data directly therefore CFCs are needed in order to make data access possible. The flash file then has to parse and display the data.

Traditional Web Application



With the traditional web application requests are made to the server which passes the parameters to the Servlet. The Servlet then calls another class which makes calls to the database based on the user request. The web pages are then built and delivered to the client.

Flash Player

Flash Remoting

Pet Store CFCs

Cold Fusion MX 7

Cold Fusion MX 7

Servlet

Data Access

Database
MS Access Database

EXPECTED RESULTS

Research into this area has suggests that RIA applications require a larger initial download, but after subsequent requests will require considerably less bandwidth utilization.

Scalability is also dramatically increased due to server load being reduced due to processing power being ofloaded to the clients computer.

Extra functionality and more responsive interfaces due to less data transfers can be achieved. Security can also be improved for example applets and flash don't allow copy and paste of code.

SPONSORED BY



Supervisor: **Greg Foster**

Co-Supervisor: **Barry Irwin**

