

# Performance and Possible Deployment of HTTP Session Mobility Service using SIP

Michael Adeyeye, Neco Ventura  
Communication Research Group  
Department of Electrical Engineering  
University of Cape Town  
{micadeyeye, neco}@crg.ee.uct.ac.za

**Abstract**— This paper presents results of the implementation of HTTP Session Mobility Service using SIP. The results are based on HTTP session mobility test on some notable websites. More precisely, web session hand-off between two web browsers was carried out. A loosely-coupled approach whereby an extension was developed and integrated into a web browser was used. This implementation leverages Session Initiation Protocol (SIP) Transportation and Mobility to transfer web sessions between two web browsers. Results showed that the service could not work on all websites, most notably websites based on FRAME/IFRAME HTML Tags, AJAX and other Web 2.0 technologies. The service, however, can work in a Peer-to-Peer environment. It can also be commercialized if the privacy and security of session data can be assured by the implementers. Some recommendations are made for the implementers of this service. Most important of all, it offers more Personal Mobility to web surfers, and SIP functionalities such as voice call can be achieved via a web browser.

**Index terms:** HTTP Session Mobility, Web browser

## I. INTRODUCTION

A number of architectural schemes have been proposed and implemented in a bid to transfer session state between web browsers. They include Server-based, Client-based and Proxy-based Architectural Schemes [1, 2, 3]. The transfer of session state between User Agent Clients (UACs) is referred to as HTTP Session Mobility. In this paper a web browser and a User Agent Client (UAC) are interchangeably used, also a web server and a User Agent Server (UAS). Here a Hybrid-based Architectural Scheme is implemented. That is, a web browser is extended and a Proxy Server is improvised. It leverages the merits of the Client-based and the Proxy-based Architectural Schemes. This is a novel approach that uses SIP as the carrier of the session state information.

Session Initiation Protocol (SIP) is an application-layer control protocol that can establish, modify and terminate multimedia sessions or calls [4]. Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative and hypermedia information systems [5]. Both protocols are application layer protocols. However, HTTP is a stateless protocol while SIP can be a stateful or stateless protocol. HTTP was not developed with the intention of migrating a web session between a web server and a web browser to another web browser. In this research work, HTTP

Session Mobility between two or more UACs using SIP has been achieved. This paper presents the results of the implementation during a web session hand-off, discusses the HTTP session mobility test, and makes some recommendations should this service be commercially deployed. The paper is arranged thus: Section Two gives the motivation for this research work, Section Three describes the service, Section Four discusses the immediate results, and Section Five provides the conclusion of this paper.

## II. MOTIVATION OF RESEARCH WORK

At first, web browsers were single-principal platform software for viewing a website at a time [6]. Today, they are multi-principal platform software that can be used to view more than one website at a time. With the widely available Rapid Application Development (RAD) tools, software, most especially web browsers are becoming easily extensible. The emergence of adaptive UACs or web browsers that have the capabilities of making voice calls, video conferencing and text chat alongside their browsing purpose is anticipated. This innovation is obtainable by simply installing an extension that integrates a new Protocol such as SIP into a web browser.

Mobility is a prevalent feature desired by end users and it is provided in most of the Service Delivery Platforms (SDPs). IP Multimedia Subsystem (IMS), one of the available SDPs, has been adopted by the Third Generation Partnership Project 2 (3GPP2) as the platform for service delivery to the end users in the Next Generation Networks (NGN). It is meant to merge the internet and the cellular worlds. It will also make existing and new services on the internet accessible from most of the network access technologies. The different types of mobility are Service Mobility, Personal Mobility, Terminal Mobility and Session Mobility. Although the differences are subtle, most especially among Service Mobility, Session Mobility and Personal Mobility, there is a need to move web sessions between Personal Computers (PCs). This will offer more Personal Mobility to end users. There are already client-based, proxy-based and server-based architectural schemes for moving existing web sessions [1, 2, 3].

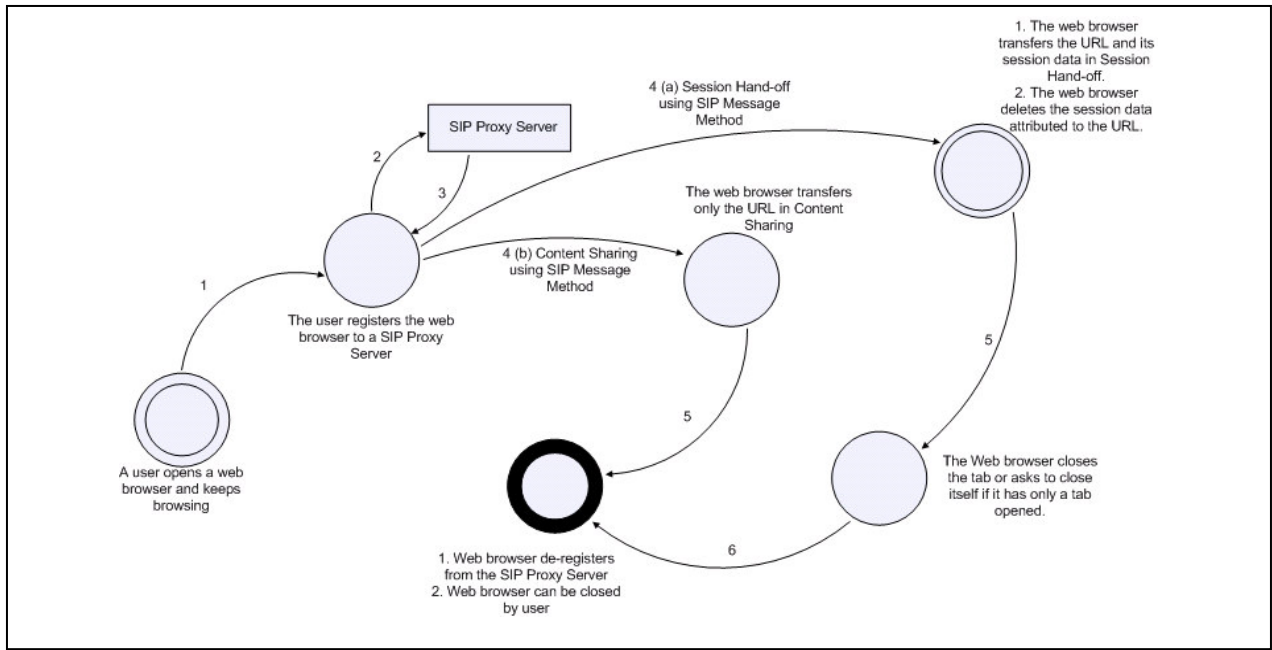


Figure 1. The Web Session Transfer Process

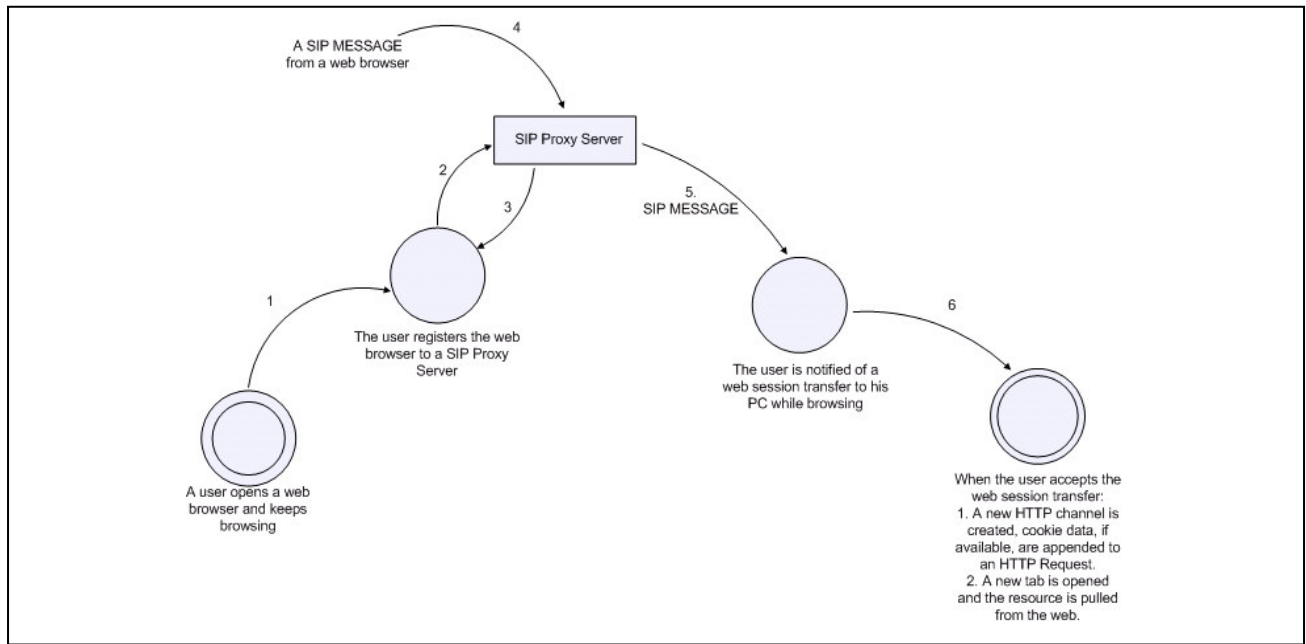


Figure 2. The Web Session Receipt Process

### III. SERVICE DESCRIPTION

This newly introduced service is session hand-off and content sharing. While session hand-off means the ability to move existing web session between two UACs, content sharing means the ability to simultaneously view the same web

page on two UACs. Content sharing requires transferring only the Universal Resource Locator (URL) of a web page. In session hand-off, the URL of a web page and its session data, usually cookies and hidden input elements, are transferred. The compositions of the session data can be a URL, cookies and hidden input elements. These data are sent in an XML format using SIP MESSAGE method.

Figure 1 shows the process flow of how a web session is transferred. In this paper, the term “web session transfer” generally refers to session hand-off and content sharing. Processes 1, 2 and 3 in Figure 1 are the same for session hand-off and content sharing. These involve launching a web browser and registering it to a SIP proxy server. Processes 4(a), 5 and 6, as indicated in the flow, are peculiar to web session hand-off. Here the web browser moves the URL and the session data to another UAC. The session data are automatically deleted from the source UAC after composing and sending the data in an XML format. Also the current tab is closed, and where it is the only one, the user is asked if he wants the web browser closed. The session data are sent to the destination UAC via the SIP proxy server. Should the user want to continue browsing the website, he will be required to log in again because his session data, usually cookies, have been deleted. Processes 4(b) and 5, as also indicated in the flow, are peculiar to content sharing. In content sharing, only the URL of the source UAC is copied and transferred. In addition, the current tab of the source UAC does not close. Hence, both the source UAC and the destination UAC can have access to the same web resource.

Figure 2 shows the web session receipt process. When a SIP MESSAGE is sent from the SIP proxy server to the destination UAC, a notification appears at the status bar of the web browser. In the case of a session hand-off request made to the destination UAC, cookies are generated from the XML data in the SIP MESSAGE body. These cookies, transferred from the source UAC, are stored and appended to a new HTTP Request using the URL in the XML data at the destination UAC. Thereafter, a new tab is automatically opened, a new HTTP Channel is created and the resource is pulled from the web. In the case of a content sharing request made to the destination UAC, only the URL is sent from the source UAC. That is, no cookie is transferred from the source UAC and generated at the destination UAC. A new tab, however, is automatically opened, a new HTTP Channel is created and the resource is pulled from the web.

#### IV. RESULTS AND DISCUSSION

Figure 3 shows the installed extension in a web browser. A new menu, “HTTP Mobility,” is introduced at the menu bar. A “Preferences” sub-menu can be found under the “HTTP Mobility” menu. In the Preferences, a user sets among other things, his SIP address, SIP proxy address, username and password. Figure 3 shows a typical web session transfer notification at the status bar. Next to this notification message is a text field which accepts the destination UAC’s SIP Address. A drop-down menu, with options Register Client, Make a Call, Content Sharing, Session Hand-off and De-register Client, exists next to the text field. This implementation also provides a voice communication between two UACs when an option “Make a Call” is chosen after the

UAC must have registered to a SIP proxy server. The following sub-sections discuss the implementation, HTTP session mobility test and requirements for the deployment of this new service.

#### A. IMPLEMENTATION

A small footprint SIP stack was used in this implementation. Its size was 2.2MB while a typical web browser installer could be 9MB in size. After a successful integration, the web browser was subjected to a performance test to determine its effects when the SIP stack was running as a background service. Results showed that the web browser performance was not hindered. This implementation takes advantage of the Free and Open Source Software (FOSS), in this case Mozilla Firefox, and the large community of developers writing array of extensions for the web browser. The SIP stack was wrapped as a shared library and a new Cross Platform Component Object Model (XPCOM) was written to interact with it. XPCOM makes it possible to write language-agnostic components thereby separating an implementation from its interface [6]. This approach provided a new layer of abstraction to the web browser in order to integrate the new protocol, SIP.

Until now, SIP was only integrated into application servers. For example, IBM Websphere Application Server Version 6.1 was built on SIP Servlet 1.0 specification. The SIP Servlet 1.0 specification was standardized through Java Specification

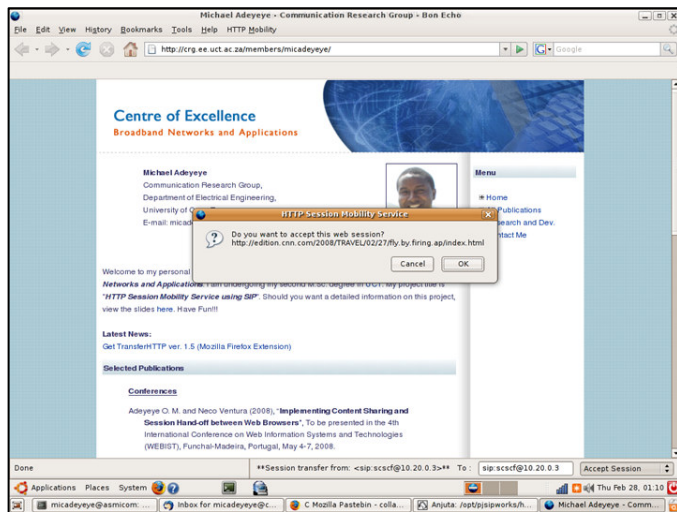


Figure 3. The TransferHTTP Web Browser Extension

Request (JSR) 116 [7]. XML Path Language (XPath) is a non-XML language for selecting nodes in an XML or HTML document. Dynamic Hypertext Markup Language (DHTML) is used to add behaviour to web experience that HTML 4.0 could not offer. It involves JavaScript and Cascading Stylesheet. Asynchronous JavaScript and XML (AJAX) and Web 2.0 are technologies similar to DHTML in purpose. In this implementation, a JavaScript was written to extract all

form data in a web page. These data are sent alongside the data transferred between two UACs. On receipt of the data in an XML format [8], though slightly modified during the implementation as shown in Figure 4, XPATH was used to extract the necessary data.

```
<xml version="1.0">
<httpsession>
<url>http://mail.live.com/TodayLight.aspx?FwaD15723</url>
<inputs>
<input name="en" type="text">english</input>
<input name="mOptionsList" type="select-one">3</input>
<input name="VIEWSTATE" type="hidden">79gh8</input>
</inputs>
<cookies>
<cookie name="MSPRequ" domain="true" host=".live.com"
path="/" isSecure="false" expires="12831">3gJeDI</cookie>
<cookie name="PHPSESSIONID">HDYK7DJ6K3</cookie>
</cookies>
</httpsession>
```

Figure 4. The SIP Message Body in an XML Format

**B. HTTP SESSION MOBILITY TEST**

Figure 5 shows that the web session data vary among websites. This web browser extension was tested on some notable websites. The intentions were to gather information on the size of data transferred during a session hand-off and find out if session hand-off could fail on some websites.

The packet size of the SIP MESSAGE body was measured when transferring each website’s homepage before and after signing in. Although the websites already set some cookies before signing in, analyzing the XML data showed that Gmail and Hotmail had more hidden input elements than cookies. In Yahoo there were less hidden input elements than cookies. After signing in, more cookies were generated in Yahoo than in Gmail and Hotmail. Also, Yahoo had more cookies than its hidden input elements. Gmail and Hotmail, after signing in, still had more hidden input elements generated than cookies.

Facebook, before and after signing in, had more cookies than its hidden input elements. Session hand-off was also successfully tested on some websites where each packet size was less than 500B. Examples were the department’s private web mail service and some eCommerce websites.

However, web session hand-off on some websites was partially successful. Although session data was successfully transferred, the exact web page with current interaction such as a web chat session could not be continued on some websites. An example was Meebo which is based on AJAX. Websites like Meebo use XMLHttpRequest to asynchronously post and retrieve data via HTTP without changing a URL. Some Mash-ups, websites that present information gathered from other websites to users, also failed. This failure can also be attributed to their abilities to update web page contents without changing its URL during a HTTP POST or GET request. Other websites that achieved partially successful session hand-off were those made up of FRAME or IFRAME HTML tags. In such cases, the source (SRC) of a FRAME or IFRAME pointed to a website that required another authentication. Only cookies of the parent FRAME or IFRAME were transferred when session hand-off was tested on these websites. This resulted in a prompt or log-in request in the FRAME or IFRAME pointing to an external URL.

In summary, web browsers are based on Same Origin Policy (SOP) in which cookies are selectively sent to a web server based on its domain or sub-domain [9]. RFC 2965 [10] provides information on how cookies should be selectively sent to a web browser and what modifications to cookies are not encouraged from end users. This implementation makes no changes to cookies. It also deletes cookies from the source UAC during web session hand-off to prevent having same session data on two or more UACs. The current web browser tab is also closed. This measure clears all hidden input elements and will require logging-in again to access the newly generated cookies and hidden input elements.

**C. DEPLOYMENT AND COMMERCIALIZATION**

While this implementation successfully runs in a Peer-to-Peer (P2P) environment, it is strongly recommended that it is used in a client-server environment. To provide data integrity and confidentiality, it is recommended that the SIP proxy server should implement a Transport Layer Security (TLS) in order to provide HTTP Digest Authentication during UACs interaction and encryption of session data. A SIP proxy server, however, is required in a client-server environment when the UACs are behind Network Address Translators (NAT). In this project, a SIP proxy server was only set-up for the UACs registrations, and the UACs were not behind NATs. SIP stack that support Secure Multi-purpose Internet Mail Extension (S/MIME) can also be used to provide data integrity and confidentiality. This feature is not currently found in most of the available SIP stacks. S/MIME offers end-to-end data encryption while TLS-supported proxy servers offer a hop-by-hop data encryption. S/MIME is preferable because in TLS-

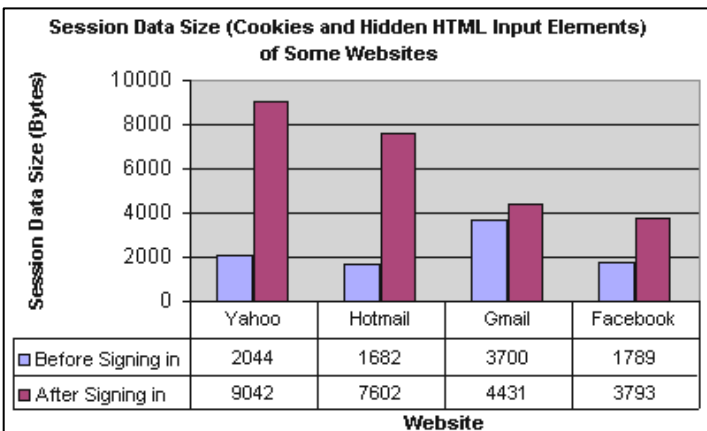


Figure 5. Session Data Size (Cookies and Hidden Input Elements) on prevalent Websites.

supported proxy servers, the last SIP proxy server that provides session data to a destination UAC might not implement TLS or offer any encryption [11].

The data showed in Figure 5 represent the number of characters transferred during web sessions hand-off. This number of characters or session data size could be the yardstick for charging users of this service. These figures account for the cookies and the hidden input elements. It was observed that the session data size varies among web pages in the same website. An example was Youtube. Although not indicated in the table, its session data size, after signing in, at the homepage was 2kB, and at a randomly selected page, it was 15kB. Additional feature available in this service is the ability to maintain the same state of a web form during content sharing or session hand-off. This feature enables a user to move a web form that he is filling between web browsers without starting afresh. A function written in JavaScript steps through the Document Object Model (DOM) of the web page, retrieves the names and values of its FORM elements, and sends alongside the session data. On getting necessary information at the destination, another function updates the DOM of the web page with what was sent to it. While it might be controversial if hidden input elements should be overwritten, they often hold posted information by users, most especially during an unsuccessful web form submission. Transferring hidden input elements' values makes this implementation maintain the same information or state between web browsers. The session data size in this case also depends on the number of HTML Form elements in the webpage. A charging function can be integrated into the SIP server and a web interface can be provided for end users to monitor their credit and usage.

## V. CONCLUSIONS

This implementation is referred to as a loosely-coupled approach because the SIP stack was not integrated into the core of the web browser, rather an abstraction was provided for the web browser and SIP stack to interact. Modifications made to the browser architecture were discussed in [8, 12]. In summary, this research work has provided new service namely, content sharing and session hand-off in the web-browsing context. This implementation helps improve collaboration and mobility among the web surfers, and encourages adaptive UACs. In this case, a web browser can be used as a SIP Client to make voice call and extended to perform other SIP functionalities.

The SIP stack packet size was extended to 16kB in order to send session data at once. The data can be sent in chunks and merged at the destination UAC in order to propagate most SIP servers should it be used over a large network. This is a custom implementation and can be deployed in a P2P environment. It can also be extended to the IP Multimedia Subsystem as a service.

Further research work includes implementing a policy control to block unwanted web session transfer request. Such restriction can be based on a domain name or a SIP address. While a session-based cookie expires in a short time of inactivity, a persistent cookie can provide access to a website over a long period of time. A session management mechanism can also be integrated so that a web session transfer request can be held for a long time without expiring when the destination URL can not be reached or a session-based cookie that expires in a short time is used by a web server.

## REFERENCES

- [1] G. Canfora, G. Di Santo, G. Venturi, E. Zimeo and M.V. Zito, "Proxy-based Hand-off of Web Sessions for User Mobility," Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous '05), 2005.
- [2] Ming-Deng Hsieh, Tsan-Pin Wang, Ching-Sung Tsai and Chien-Chao Tseng, "Stateful session handoff for mobile WWW," Information Sciences, Elsevier Science Press, volume 176, 2006, pp. 1241-1265.
- [3] H. Song "Browser Session Preservation and Migration," In Poster Session of WWW 2002, Hawaii, USA, May 7-11, 2002, pp. 2.
- [4] M. Handley, H. Schulzrinne, E. Schooler and J. Rosenberg, "SIP: Session Initiation Protocol," IETF RFC 2543, March 1999.
- [5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," IETF RFC 2616, June 1999.
- [6] David Boswell, Brian King, Ian Oeschger, Pete Collins and Eric Murphy, Creating Applications with Mozilla, O'Reilly Press, USA, First Edition, 2002, pp 1-8.
- [7] Erik Burckart, "Session Initiation Protocol in WebSphere Application Server V6.1," [Online], Available: [http://www.ibm.com/developerworks/websphere/techjournal/0606\\_burckart/0606\\_burckart.html](http://www.ibm.com/developerworks/websphere/techjournal/0606_burckart/0606_burckart.html) [May 16, 2008].
- [8] M. Adeyeye and N. Ventura, "Implementing Content Sharing and Session Hand-off between Web Browsers," Proceedings of the 4<sup>th</sup> International Conference on Web Information Systems and Technologies, Funchal, Madeira, Portugal, May. 4-7, 2008.
- [9] Helen J. Wang, Xiaofeng Fan, Jon Howell and Collin Jackson, "Protection and Communication Abstractions for Web Browsers in MashupOS," Proceedings of the SOSp' 07, Stevenson, Washington, USA, October 14-17, 2007.
- [10] D. Kristol and L. Montulli, "HTTP State Management Mechanism," IETF RFC 2109, October 2000.
- [11] Gonzalo Camarillo and Miguel Garcia-Martin, The 3G IP Multimedia Subsystem, Wiley Press, England, Second Edition, 2006, pp. 97, 213-229.
- [12] M. Adeyeye and N. Ventura, "Extending Web Browsers Architectures to support HTTP Session Mobility," Proceedings of CoNEXT '07, New York, U.S.A, Dec. 10-13, 2007.

**Michael Adeyeye** is a member of the IEEE. He earned B. Tech. in Electronic and Electrical Engineering from Ladoke Akintola University of Technology, Ogbomoso, Nigeria. Also, in 2007, he earned M. Eng. in Information and Communication Technology from Covenant University, Ota, Nigeria. He is currently undergoing M.Sc. (Communications) in the University of Cape Town. He is a member of the Communication Research Group in the Department of Electrical Engineering, a Certified Internet Webmaster (CIW) Professional and the Mozilla Firefox Campus Representative for the University of Cape Town.