

# Mapping Third Party Call Control and Session Handoff in SIP Mobility to Content Sharing and Session Handoff in the Web Browsing Context

Michael Adeyeye, Neco Ventura and \*David Humphrey

Department of Electrical Engineering

University of Cape Town

Private Bag X3, Rondebosch 7701, South Africa

{micadeyeye; neco}@crg.ee.uct.ac.za

\*School of Computer Studies

Seneca College, Toronto, Canada

david.humphrey@senecac.on.ca

**Abstract-** This paper presents a new service to web browsing. The service, referred to as Session Handoff and Content Sharing between two web browsers, requires extending the capabilities of existing web browsers by integrating a Session Initiation Protocol (SIP) stack into them. An optional SIP Application Server (SIP AS) that co-ordinates HTTP session mobility between web browsers is introduced. In addition, Third-party Call Control and Session Handoff in SIP Session Mobility are successfully mapped to Content Sharing and Session Handoff between two web browsers, respectively. While content sharing refers to the ability to view same web resource on two web browsers, session handoff refers to the ability to migrate a web session between two web browsers. Until now, SIP had only been integrated into application servers. Integrating a SIP stack into a web browser has helped improve collaboration and mobility among the web users. In addition, it encourages developing adaptive User Agent Clients (UACs) that can serve two or more purposes. In this case, a web browser can also be used as a SIP client to make voice calls and be extended to perform other SIP functionalities.

## I. INTRODUCTION

As the web browsing experience is improving and web browsers' capabilities are increasing, networks and services in the telecommunications and Internet world are converging. The popular online service providers rendering free email service such as *Yahoo*, *Hotmail* and *Gmail* have proprietary programs that must be installed in order to establish multimedia sessions. Recently, these companies have opened up their services via Application Program Interfaces (API) to encourage third parties to use their services and develop new ones.

Today, websites such as *www.meebo.com* and *www.iloveim.com*, owned by third parties, have integrated these APIs to offer visitors access to their accounts and establish multimedia sessions on their websites rather than visiting each proprietary website. Although the ability to access various user's accounts via a single website has reduced

running multiple proprietary programs, new software is still required on the user's Personal Computers (PCs) to enjoy these services. For example, *www.meebo.com* requires that users must have the latest versions of Adobe Flash Player and Java Virtual Machine before they can make multimedia calls. A visitor, however, browsing *www.meebo.com* can make multimedia call or start a chat session with any of his friends on *www.gmail.com*, *www.yahoo.com* and *www.hotmail.com* rather than running each Instant Messaging (IM) engine on his PC. Standalone applications such as Pidgin on Linux OS also provide this service.

The built-in features added to web browsers to improve the web browsing experience include tabbed browsing and spell-check. On the features added by third parties to web browsers, there are File Transfer Protocol (FTP) and Internet Relay Chat (IRC) extensions in the Mozilla Firefox web browser, to mention a few. The ability to access multiple social networking services or multiple IM accounts via a single website is an example of converging services. Unceasingly, the web is rapid evolving with new set of applications and services being pushed to the market day in day out. Visitors or web users have enormous user accounts with different credentials, most notably passwords, for these services. The convergence of services has introduced Single Sign-on, which is one way of alleviating the burden of having multiple user accounts. Another way that has alleviated this problem is the use of Open ID [1]. Open ID is a shared identity service that eliminates the need for multiple usernames and passwords across different websites, simplifying the online experience. These solutions have not completely resolved the multiple accounts problem, especially when a web application does not support these technologies. Today, another way of improving the web browsing experience, known as web session mobility between PCs, has arisen.

Numerous works have explored web session mobility. Session Mobility would enable seamless transfer of a web session between different devices based on user preferences.

The reasons for session transfer include cheapest access cost, better user experience and physical user mobility [2]. As convergence is now moving into the mobile space, there is a strong push from Triple play of voice, video and data to Quad play of voice, video, data and mobility services [2]. The advancement from Triple play to Quad Play shows that mobility is a key feature required in the convergence of networks and services.

## II. RELATED WORK

HTTP Session Mobility has been carried out using different schemes namely Client, Server and Proxy Architectural Schemes [3, 4, 5]. Each scheme requires modifying the User Agent Client (web browser) or User Agent Server (web server) or both. In some cases, a Proxy could be placed along the communication path of the client and server, as found in the Proxy Architectural Scheme. In previous works, the identification or authentication mechanism for users was not clearly stated, the session mobility paradigms were not based on standards but personal initiatives and the HTTP/1.1 security as stated in RFC 2616 [6] was broken in two of the implementations [3, 5].

Projects that took advantage of SIP extensibility include the Akogrimo Project [7] that involves embedding web service data in a Session Description Protocol (SDP). In addition, in an expired IETF internet draft [8], two approaches were identified for transferring Universal Resource Locator (URL) between two web browsers. The first approach was by sending the URL via a SIP MESSAGE method. This first approach is what this research work is based on. The second approach was by using SIP NOTIFY method. This second approach was described as a way of achieving conference model of web browsing whereby a browser could be notified when a web page, on another browser, has changed. Although the approaches were not standardized, they showed different ways of achieving web share using SIP MESSAGE and NOTIFY methods.

In this work, SIP is used to achieve HTTP Session Mobility. It also provides standard identification mechanism where each web browser has a unique SIP address. Interaction between web browsers is based on their unique SIP addresses. This is synonymous to a Local Area Network (LAN) in which PCs interact with one another based on their unique Internet Protocol (IP) addresses. It is an application-layer approach to achieve session mobility unlike session continuity in Mobile IP which is a lower layer approach.

To implement this project, an extension that integrates a SIP Stack into the present-day web browsers was developed. The extension runs on the Mozilla Firefox web browser as a proof of concept. This work is different from the existing crash recovery or session restoration in Firefox. It focuses on transferring existing web session between PCs but not restoring web session on the same PC.

## III. THE HTTP SESSION MOBILITY SERVICE

In this research, HTTP session mobility using SIP is achieved by extending the capabilities of a web browser, and

using an optional SIP AS. This scheme is referred to as a Hybrid-based Architectural Scheme. No changes are made to the present-day web servers architectures. In this project, excerpts of HTTP Responses are encapsulated and sent in a SIP MESSAGE.

## IV. DISCUSSIONS

### A. Use Case Scenarios

The new service is called content sharing and session handoff. The service is introduced via the web browser interface to address common problems faced by users. The problems are regularly encountered, most notably among peers, and in some cases an individual. Two scenarios are described below to illustrate the problems.

1. Bob is in a laboratory at school browsing a newspaper website when he comes across an interesting article that he wants to share with Alice. Alice, his colleague, is in a coffee shop also browsing, though a different website, via a wireless access point provided in the coffee shop. Assuming that both of them are online on Facebook and Yahoo Messenger, Bob quickly copies the URL of the article and sends to Alice via one of the Instant Messaging services. At the same time, he invites Alice to a voice chat to discuss the article. In a situation where Alice is not online, Bob sends the URL in an email to Alice and asks if they could discuss the article later.

2. Alice, researching on her project, is asked to fill in a form before she can download software she needs for the project. She has only logged in minutes ago and now asked to fill a form that requires a number that she cannot recall off-hand. From all indications, she realizes that she would have to restart the whole processes when she reaches home. Finally, she quits the website and tries to do something else until she gets home when she will be able to continue her project.

Scenario one depicts content sharing, and scenario two depicts session handoff. These problems are very similar to what individuals face today when surfing the Internet. Scenario one shows slow and in-efficient ways to referring someone else to view the same web page being viewed at the same time. This problem is identified as one of the ways the web browsing experience could be improved if an efficient solution exists. The solution should cater for sending the URL to the intended recipient by a click or not more than two processes rather than copying the URL and using another software or service to accomplish the task. Better still, the solution should also cater for the voice interaction.

In scenario two, referred to as session handoff, the user, Alice, would like to continue filling the form at home without having to log-in again and navigate the website to the form page. Most times, individuals want to continue viewing the same web page later and at a different place. A large amount of HTTP signalling is involved moving from one link to the other, and a cost is incurred in the signalling, most notably where Internet access is expensive, though the cost could be small.

### B. The Implementation Framework

This framework is based on a Hybrid-based Architectural Scheme. It requires modifying web browsers' architecture in order to support SIP and the use of optional SIP proxy for client authentication and data encryption. No changes are made to web servers' architecture, and each web browser can interact with its respective web server via an HTTP proxy server. Session handoff between devices owned by the same person and content sharing, which requires interaction between devices owned by different people, are also indicated here.

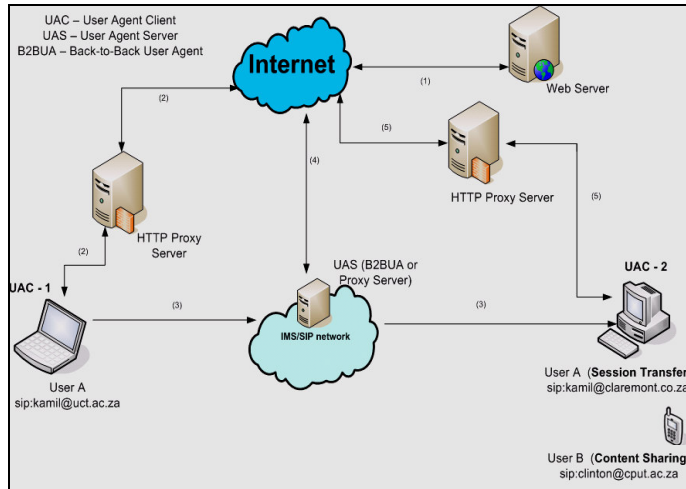


Figure 1. The Implementation Framework

### C. Mapping Third Party Call Control in SIP Mobility to Content Sharing in Web Browsing

Third Party Call Control (3pcc) in SIP [9] involves a controller transferring a session between it and its correspondent UAC to another UAC. The signalling, however, traverses the controller, and the controller is responsible for terminating the new call session between the two UACs. To describe content sharing between UAC 1 and UAC 2 using Figure 2, UAC 1 and the HTTP web server represent two UAs that are already interacting. UAC 1 now wants UAC 2 and the same HTTP web server to interact. To do so, UAC 1 first registers to the SIP proxy server and the user continues browsing.

When UAC 2 has successfully registered with the SIP proxy server, a SIP MESSAGE containing only the content of the web browser's address bar is sent to the destination SIP Universal Resource Identifier (URI) via the SIP proxy server. Should necessary application function policies be met, the SIP proxy server will forward the SIP MESSAGE to UAC 2, which is the destination SIP URI. Application function policies, in this case, include checking the source UAC address if it is allowed to transfer web session to the destination UAC address.

Where UAC 1 and the web server in Figure 2 represent the controller and UAC 2 in 3pcc, respectively, successful mapping of 3pcc in SIP [9] to content sharing in web browsing is based on the following assumptions:

- Initial call session in 3pcc refers to existing web session between UAC 1 and the HTTP web server.

- In 3pcc in SIP, a controller wants its call session with UAC 2 transferred to UAC 1 yet has access to the signalling. In Figure 2, UAC 1 wants its web session with the HTTP web server transferred to UAC 2 while still maintaining access to the signalling, that is a copy of the HTTP Request.

The key point in these assumptions is that the source UAC, like the controller in 3pcc, also has access to the signalling (in this case, the same HTTP Request) after a transfer. Having access to the signalling here refers to the ability to continue making HTTP Request to the web server after the transfer. This content-sharing service refers a user to a page that requires no identification or authentication, and no session data is intended to be transferred save the content of the web browser's address bar. Hence, only the URL of the web resource is transferred. The user at a destination UAC might be required to sign in before accessing the web resource. This will ensure that no two UACs have the same session data such as cookies. There are some web applications, however, that append or encode session data, such as a session ID, into a URL. This is a process referred to as URL Re-writing or Encoding. When session data are encoded, it is impossible to segregate the session data from a URL because the web browser or implementer of this service does not know the encoding scheme used by the application. Besides, the encoding scheme varies for different websites. In this case, the URL is transferred irrespective of the session data in it. Similarly, there are one-page dynamic websites that change their contents based on HTTP GET Requests. For this reason, the entire URL is sent without any alteration in order to refer to the same web resource.

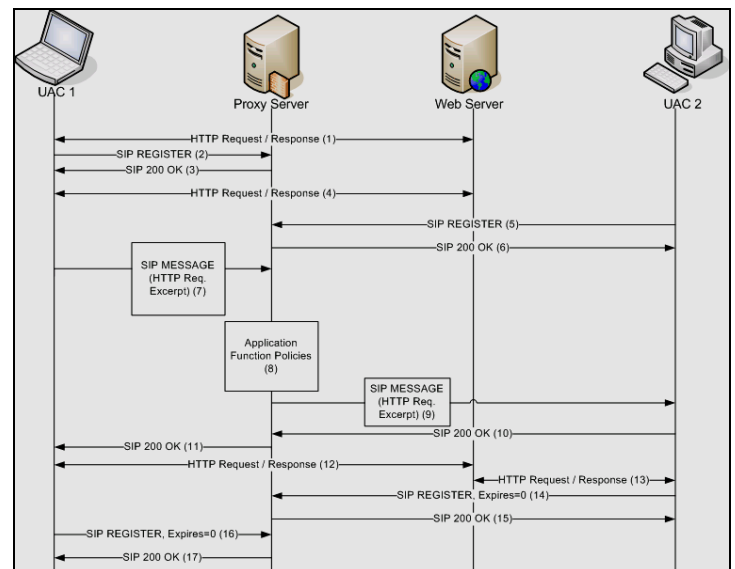


Figure 2. The Content Sharing Process Flow

### D. Mapping Session Handoff in SIP Mobility to Session Handoff in Web Browsing

Although session handoff described in [10] is classified into session handoff mode flow for transfer to a single device and session handoff to a multi-device system, only the former

mode flow is considered in this research work. This is because session handoff in this project is transferred to only one device. Session handoff in SIP [10] also involves a controller transferring a session between it and its correspondent UAC to another UAC. Unlike 3pcc, the signalling, however, does not traverse the controller and the controller is not responsible for terminating the new call session between the two UACs. In Figure 3, UAC 1 and HTTP web server represent two UACs in interaction. UAC 1 wants UAC 2 and the HTTP Web Server to interact. First, UAC 1 registers with the SIP proxy server and the user continues with his web browsing. An excerpt of each HTTP Request Header from UAC 1 can be forwarded to the SIP proxy server using the SIP MESSAGE method as the user continues to browse the web. With this, all HTTP Requests can be monitored by the SIP proxy server.

When UAC 2 has successfully registered with the SIP proxy server, a SIP MESSAGE containing the content of the web browser's address bar, and session data is sent to the destination SIP Universal Resource Identifier (URI) via the SIP proxy server. Unlike the content sharing earlier described, session data is sent alongside to the destination SIP URI. Should necessary application function policies, like filter criteria [11], be met, the SIP MESSAGE will be forwarded by the SIP proxy server to the UAC 2.

Similarly, where UAC 1 and the web server in Figure 3 represent the controller and the UAC 2 in session handoff in SIP, respectively, successful mapping of session handoff in SIP [10] to session handoff in web browsing (Figure 3) is based on the following assumptions:

- Initial call session in session handoff in SIP refers to existing web session between UAC 1 and the HTTP web server.
- In session handoff in SIP, a controller wants its call session with the UAC 2 transferred to UAC 1, but thereafter, does not have access to the signalling. In Figure 3, UAC 1 wants its web session with the HTTP web server transferred to UAC 2, but thereafter, does not have access to the signalling, that is no copy of the HTTP Request.

Here, the key point in these assumptions is that the source UAC, like the controller in session handoff in SIP, also does not have access to the signalling (in this case, the same HTTP Request) after a transfer. This is called the session handoff service. The implementation could break HTTP Security if two UACs, that is web browsers, have the same session data at the same time. Hence, session data are deleted from the source UAC when transferred to another UAC. At this point, the source UAC loses a stateful HTTP connection with the HTTP web server. As earlier stated, session data can be in the form of cookies, hidden HTML elements and rewritten or encoded URL. These session data are transferred between UACs using the SIP MESSAGE method.

### E. The 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 successful integration, the web browser was subjected to a performance test to determine effects when the SIP stack was running as a background service. The browser's performance was not hindered and the results are later discussed. This implementation takes advantage of a web browser called Mozilla Firefox. Mozilla Firefox is a FOSS that has a large community of developers writing array of extensions for it. 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 [12]. This approach provided a new layer of abstraction to the web browser in order to integrate the new protocol, SIP.

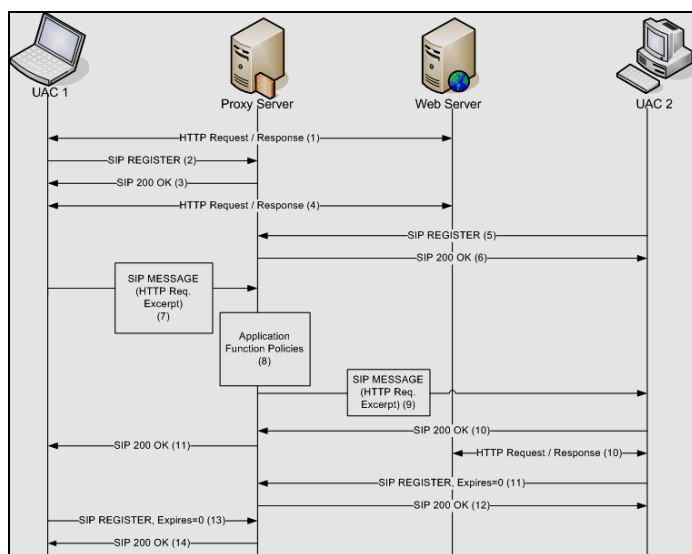
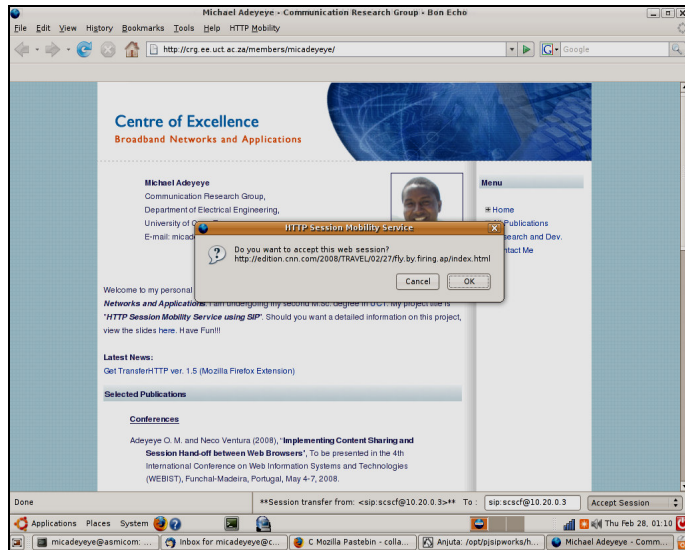


Figure 3. The Session Handoff Process Flow

XML Path Language (XPath) is a non-XML language for selecting nodes in an XML or an HTML document, and Dynamic Hypertext Markup Language (DHTML) is used to add behaviour to web experience that HTML 4.0 could not offer. Although DHTML 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 form data are sent alongside the session data transferred between two UACs. On receipt of the data in an XML format, XPath was used to extract the necessary data.

Figure 4 shows the installed extension in a web browser. A new menu, "HTTP Mobility," is added to the menu bar. A sub-menu, "Preferences," could be found under the "HTTP Mobility" menu. In the Preferences, a user sets among other things, his UAC's SIP address, SIP proxy address, username and password. Also, Figure 4 shows a typical notification in the status bar. Next to this notification message is a text field that accepts the destination UAC's SIP address. A drop-down menu, with options Register Client, Make a Call, Content Sharing, Transfer Session, Accept Session and De-register

Client, exists next to the text field. This implementation also provides a voice interaction between two UACs when the option “Make a Call” is chosen after the UAC must have registered to a SIP proxy server.



**Figure 3. A Content Sharing Instance in Mozilla Firefox**

## V. PERFORMANCE EVALUATION

The integration of a SIP stack into the Mozilla Firefox web browser was achieved in a loosely-coupled approach. The SIP stack was compiled as a shared library and an XPCOM, written in C++ programming language, was also compiled to interact with it. The XPCOM formed an abstraction over the SIP stack with which the web browser could interact with the SIP stack.

Two performance tests were conducted. First, there was a need to know if the browser required more memory to run the SIP stack as a background service. Table 1 shows the web browser’s memory consumption test. A package called “System Monitor” on Ubuntu Operating System was used to gather the memory consumption data. It is used to view current processes and monitor system state in the Operating System.

**Table 1. Web browser memory consumption test**

	MEMORY CONSUMPTION (MB)		
	IMMEDIATELY	2HRS	4HRS
Without SIP stack integration	11.3	11.6	11.9
With SIP stack integration	16.7	17	18.6

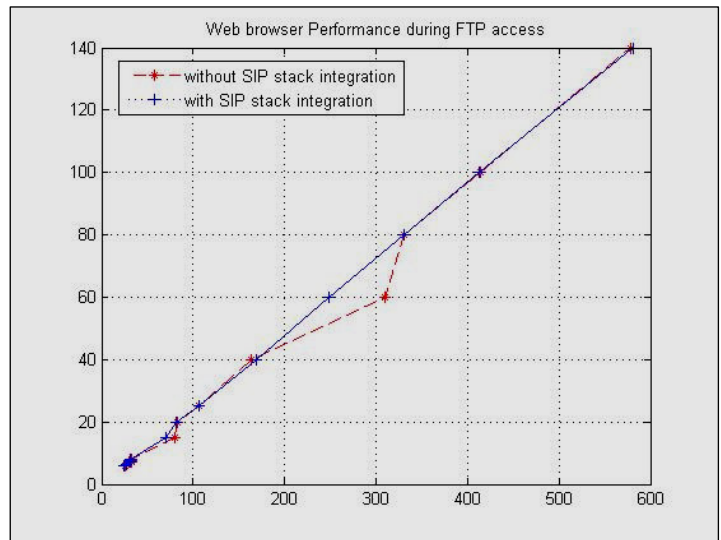
In this first performance test, the browser’s memory consumption when the SIP stack was not integrated was

compared with when the SIP stack was running as a background service. The first column of results in Table 1 indicates the results gathered when the browser was immediately launched and closed. The second and third columns of the results show the results gathered when the browser was allowed to run for two and four hours, respectively.

Results gathered without the SIP stack running as a background service indicated the normal memory consumption of the browser. The memory consumption at start was 11.3MB and progressively increased overtime. When the SIP stack was allowed to run as a background service, the memory consumption at start was 16.7MB. In addition, the memory consumption progressively increased. Analytically, running the SIP stack as a background service caused the browser’s memory consumption to increase averagely by 5.83MB.

The browser was used to browse the Internet during and after the memory consumption tests were carried out. Findings showed that the browser did not crash either freeze when running the SIP stack as a background service over a long period.

The second test compares the web browser’s performances. Figure 5 shows a comparison made between the web browser’s performances when the SIP stack was running as a background service and when the SIP stack was not running. At both periods, the browser was used as an FTP Client to transfer picture files of varying sizes to a web server. The average download and upload speeds of the Internet connection were 56kbps and 241kbps respectively. The graph shows no changes in the browser’s performances when the SIP stack was running as a background service and when it was not running as a background service. In summary, the test was only conducted to check if the performance of the web browser was impeded by integrating a SIP stack.



**Figure 4. Comparison of the web browser performances during an FTP access**

While this implementation successfully runs in a P2P environment, it is strongly recommended that it is used in a

client-server environment. To provide data integrity and confidentiality, it is advisable that the SIP server should implement TLS in order to provide HTTP Digest Authentication and encryption of data during UACs interaction. An alternative is the use of UACs that support Secure Multi-purpose Internet Mail Extension (S/MIME).

S/MIME is a feature required in a SIP stack that might be used in the implementation, though it is not currently found in most of the available SIP stacks. It offers end-to-end data encryption while TLS-supported proxy servers offer a hop-by-hop data encryption. When deploying this service over a large network, S/MIME is preferable to TLS. The reason is that on the use of TLS for data security, the last SIP proxy server that sends the session data to the destination UAC might not implement TLS or offer any encryption [13]. Hence, session data in transit can be hijacked.

## VI. CONCLUSIONS

This implementation [14] has provided a fast and efficient way of referring someone else to the same web page rather than the slow way of pasting and sending the URL in chat session or an email. With this implementation, web browsers can now have unique SIP addresses to interact with one another like PCs with unique IP addresses. Modifications made to the web browser architecture were discussed in previous papers [15, 16]. This implementation helps improve collaboration and mobility among the web users and encourages adaptive UACs. In this case, the web browser can also be used as a SIP client to make voice calls and be extended to perform other SIP functionalities.

Until now, SIP had only been integrated into application servers. For example, IBM Websphere Application Server Version 6.1 was built on SIP Servlet 1.0 specification [17]. The disparate clients for web access and voice or video communication have led to enormous UACs on users' PCs. As services and networks converge in the Next Generation Networks (NGN), it is desired that a UAC should be easily extensible to meet other needs. This is a research work that entails modifying present-day web browsers' architectures to integrate a SIP Stack and using an optional SIP Application Server in order to achieve content sharing and session handoff.

Only the client-side requirements of the service have been exhaustively discussed in this paper. This is a custom implementation and can be deployed in a P2P environment. While this implementation successfully runs in a P2P environment, it is strongly recommended that it is used in a client-server environment. To provide data integrity and confidentiality, it is advisable that the SIP server should implement TLS in order to provide HTTP Digest Authentication and encryption of data during UACs interaction. An alternative is the use of UACs that support Secure Multi-purpose Internet Mail Extension (S/MIME).

In a situation where media negotiation is already done, such as a voice call or video conferencing, HTTP Request excerpts can be sent using SIP INFO Method or Inside-dialog Instant Messaging. SIP has been chosen because it is widely used in

the industry and extensible to support additional features. This is not a killer application but it will encourage unified or adaptive User Agents.

Integrating a SIP stack into a web browser could modify or extend how click-to-dial works [18]. With a SIP stack integrated into a web browser, a call session could be set-up between a telephone or a mobile phone and a web browser, which acts as a SIP client. Having standardized the command "tel:" in a hyperlink [19], when such hyperlink is clicked, it could initialize the built-in SIP stack in a web browser and set up a call between the SIP address and the web browser. In click-to-dial, an application server is responsible for establishing a call session between two phones.

## REFERENCES

- [1] The Open ID Project, <http://openid.net/what/>, July 14, 2008.
- [2] Sujet Mate, Umesh Chandra, Igor D. D. Curcio, "Moveble-Multimedia: Session Mobility in Ubiquitous Computing Ecosystem," Proceedings of the Mobile and Ubiquitous Multimedia (MUM '06), Stanford, California, 2006.
- [3] 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.
- [4] H. Song "Browser Session Preservation and Migration," In Poster Session of WWW 2002, Hawaii, USA, 7-11 May, 2002, pp. 2.
- [5] 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.
- [6] 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.
- [7] The Akogrimo Project, <http://www.mobilegrids.org>, June 19, 2008.
- [8] Xiaotao Wu and H. Schulzrinne, "Use SIP MESSAGE method for shared web browsing," <http://www3.tools.ietf.org/id/draft-wu-sipping-webshare-00.txt>, November 14, 2001.
- [9] J. Rosenberg, J. Peterson, H. Schulzrinne and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)," IETF RFC 3725, April 2004.
- [10] R. Shacham, H. Schulzrinne, S. Thakolsri and W. Kellerer, "Session Initiation Protocol (SIP) Session Mobility," Internet-Draft: draft-shacham-sipping-session-mobility-05, <ftp://ftp.rfc-editor.org/in-notes/Internet-drafts/draft-shacham-sipping-session-mobility-05.txt>, November 18, 2007.
- [11] Gonzalo Camarillo and Miguel Garcia-Martin, The 3G IP Multimedia Subsystem, Wiley Press, England, Second Edition, 2006, pp. 97, 155-162.
- [12] 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.
- [13] Gonzalo Camarillo and Miguel Garcia-Martin, The 3G IP Multimedia Subsystem, Wiley Press, England, Second Edition, 2006, pp. 155-162, 345-349.
- [14] The TransferHTTP Extension, <http://transferhttp.mozdev.org>, September 5, 2008.
- [15] M. Adeyeye and N. Ventura, "Extending Web Browsers Architectures to Support HTTP Session Mobility," In *Proceedings of the 3<sup>rd</sup> Annual CoNEXT Conference*, New York, U.S.A., Dec. 2007.
- [16] M. Adeyeye and N. Ventura, "Implementing Content Sharing and Session Hand-off between Web Browsers," Proceedings of the 4th International Conference on Web Information Systems and Technologies, Funchal, Madeira, Portugal, May. 4-7, 2008.
- [17] 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].
- [18] R. Jain, J. Bakker and F. Anjum, Programming Converged Networks: Call Control in Java, XML, and Parlay/OSA, Wiley Interscience, Canada, First Edition, 2005, pp 27-34.
- [19] H. Schulzrinne, "The tel URI for Telephone Numbers," IETF RFC 3966, December 2004.