A SIP Integrated Web Browser for HTTP Session Mobility and Multimedia Services

Michael Oluwasegun Adeyeye

This thesis is submitted in fulfillment of the academic requirements

for the degree of

Master of Science in Electrical Engineering

in the Faculty of Engineering and The Built Environment

University of Cape Town

December 2008

As the candidate's supervisor, I have approved this dissertation for submission.

Name: Full name and title of thesis advisor

Signed: Signed by candidate

Date: _____

Declaration

I declare that this thesis is my own work. Where collaboration with other people has taken place, or material generated by other researchers is included, the parties and/or materials are indicated in the acknowledgements or are explicitly stated with references as appropriate.

This work is being submitted for the Master of Science in Electrical Engineering at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

Michael Oluwasegun Adeyeye

10th June 2008

To God, who is making a way for me.

Abstract

Web session migration is one way of improving web-browsing experience. Other ways include bookmarking and web history synchronization. This project introduces a new service to web browsing namely, Session Handoff and Content Sharing. The service requires extending the capabilities of existing web browsers by integrating a Session Initiation Protocol (SIP) stack into them. 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 the same web resource between two web browsers, session handoff refers to the migration of a web session to another web browser.

The implementation is a loosely-coupled approach in which a SIP stack is not integrated into the core of a web browser, rather an abstraction is provided for a web browser and a SIP stack to interact. This implementation leverages SIP Transportation and Mobility mechanism to transfer session data between two web browsers. Session data could compose of URL, cookies and hidden input elements. A graphical tool, Data Flow Diagram, is used to explain how a web session is transferred and received, and the technologies used in implementation are mentioned.

On the implementation, a small footprint SIP stack and an Open Source web browser are used. The SIP stack, which is compiled into a shared library, has a file size of 2.2MB, while a typical web browser's footprint could be 8MB. A number of tests, namely Upload, Download and Memory Consumption Tests, are carried out. Results show that the memory consumption of the web browser does not increase significantly to make the web browser freeze or crash. In addition, the speed of the web browser is not impeded when the web browser is used to upload

and download files. In terms of HTTP session mobility, results show 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 implementation, based on a Hybrid-based Architectural Scheme, is compared with other existing web session migration schemes.

Regarding commercialization, it is found that if the privacy and security of session data could be guaranteed by the implementers, a flat rate could be periodically charged, regardless of the varying session data sizes. On the other hand, it could be rendered as a Value Added Service (VAS) to customers.

HTTP session mobility offers Personal Mobility to end users. That is, as end users move from one place to another, they can now move their web sessions from one Personal Computer to another without re-authenticating. In addition, a web browser can now act as an adaptive User Agent Client to surf the Internet and make voice calls as a SIP client. The thesis introduces a novel mechanism, which uses SIP to transfer session data and borrows SIP Mobility types, for content sharing and session handoff between web browsers.

Acknowledgements

My sincere gratitude goes to my parents for their unceasing prayer, support and encouragement. In return, I think I have lessons to pass on to my younger ones.

I appreciate the steadfast support I got from my supervisor, Neco Ventura, during my period of study in the Communication Research Group (CRG) Laboratory.

David Humphrey, a Professor in the School of Computer Studies, Seneca College, Canada, has been a resourceful person to date. He pointed me at the resources needed for my research work and introduced me to the Mozilla Community. I feel I have gained so much afterwards.

Some researchers in the same research areas were of tremendous help during my research. They included Henning Schulzrinne, Paolo Bellavista and Chien-Chao Tseng. They, out of their busy moments, replied my mails and made contributions.

My colleagues, Uche Onumajuru and Gabriel Andrews, helped proofread the thesis after my first draft. David Waiting and Richard Good gave constructive feedback on the first draft, and it has resulted in a better thesis. Lastly, I also appreciate other members of the CRG whom I met during my study and those that made contributions to my research work.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AJAX	**-**	**Asynchronous JavaScript and XML**
AS	**-**	**Application Server**
API	**-**	**Application Program Interface**
BSMP	**-**	**Browser Session Migration and Preservation**
CERN	**-**	**European Nuclear Research Center**
CSS	**-**	**Cascading Style Sheet**
DFD	**-**	**Data Flow Diagram**
DHTML	**-**	**Dynamic Hypertext Markup Language**
FOSS	**-**	**Free Open Source Software**
FTP	**-**	**File Transfer Protocol**
GNU	**-**	**GNU is not UNIX**
HTML	**-**	**Hypertext Markup Language**
HTTP	**-**	**Hypertext Transfer Protocol**
IBM	**-**	**International Business Machine**
IDL	**-**	**Interface Definition Language**
IE	**-**	**Internet Explorer**
IETF	**-**	**Internet Engineering Task Force**
iMASH	**-**	**Interactive Mobile Application Session Handoff**

IMS	**-**	**IP Multimedia Subsystem**
IP	**-**	**Internet Protocol**
IPTV	**-**	**Internet Protocol Television**
IRC	**-**	**Internet Relay Chat**
JSR	**-**	**Java Specification Request**
MAC	**-**	**Media Access Control**
MPA	**-**	**Mobile People Architecture**
MPL	**-**	**Mozilla Public License**
MSRP	**-**	**Message Session Relay Protocol**
NAT	**-**	**Network Address Translation**
NCSA	**-**	**National Center for Supercomputing Applications**
NGN	**-**	**Next Generation Networks**
OS	**-**	**Operating System**
OSI	**-**	**Open Systems Interconnection**
PC	**-**	**Personal Computer**
PDA	**-**	**Personal Digital Assistant**
P2P	**-**	**Peer-to-Peer**
RAD	**-**	**Rapid Application Development**
RFC	**-**	**Request for Comments**
RTSP	**-**	**Real Time Streaming Protocol**

SCTP	**-**	**Stream Control Transmission Protocol**
SDP	**-**	**Session Description Protocol**
SHOC	**-**	**Session Handoff Component**
SIP	**-**	**Session Initiation Protocol**
SOP	**-**	**Same Origin Policy**
SRC	**-**	**Source**
SSL	**-**	**Secure Sockets Layer**
S/MIME	**-**	**Secure Multipurpose Internet Mail Extension**
TCP	**-**	**Transport Control Protocol**
TLS	**-**	**Transport Layer Security**
TraSH-SN	**-**	**Transport Layer Seamless Handoff Scheme for Space Networks**
UA	**-**	**User Agent**
UAC	**-**	**User Agent Client**
UAS	**-**	**User Agent Server**
UI	**-**	**User Interface**
URI	**-**	**Universal Resource Indicator**
URL	**-**	**Universal Resource Locator**
VAS	**-**	**Value Added Service**
VOD	**-**	**Video on-Demand**
WWW	**-**	**World Wide Web**

W3C	**-**	**World Wide Web Consortium**
XML	**-**	**Extensible Markup Language**
XPATH	**-**	**XML Path Language**
XPCOM	**-**	**Cross Platform Component Object Model**
XPIDL	**-**	**Cross Platform Interface Definition Language**
XUL	**-**	**XML User Interface Language**
3GPP	**-**	**Third Generation Partnership Project**
3GPP2	**-**	**Third Generation Partnership Project 2**
3pcc	**-**	**Third Party Call Control**

Chapter 1

1 Introduction

The World Wide Web (WWW) was first described in a proposal written by Tim Berners-Lee in 1990 at the European Nuclear Research Center (CERN) [1]. The following year, he wrote the first web browser, which was graphical and also served as a HyperText Markup Language (HTML) editor. Lynx, a well supported text-only browser, was adapted to support the web in 1993. At the same time, a graphical browser called Mosaic was released by the National Center for Supercomputing Applications (NCSA). Berners-Lee founded the World Wide Web Consortium (W3C) in 1994 to guide the evolution of the web and promote interoperability among web technologies. Microsoft released Internet Explorer in 1995, based on Mosaic, and Netscape released its browser in 1998 as open source under the name Mozilla.

Open source web browsers, such as Mozilla Firefox and Konqueror, have been reused into new ones [2]. Apple Computers has integrated Konqueror's core subsystems into its OS X web browser, Safari. Similarly, Mozilla Firefox has been reused into new ones, such as Flock web browser. Internet Explorer, a proprietary web browser, has also been reused into several new browsers like Maxthon, NetCaptor and Avant. These new web browsers are pushed to the market with numerous new features that are lacked in the old web browsers. On the other hand, the new web browsers may be refined to meet specific requirements of a group, community or set of people.

Although the 1990's was referred to as the "browser wars" period [2], the competition among web browsers has not ended. In general, today, features present in the web browsers in order to improve the web browsing experience include tabbed browsing, pop-up blocking, spell-check, page zooming and bookmarking. Although the principal protocol of the web browsers is

Hypertext Transfer Protocol (HTTP), more protocols are supported in the Mozilla Firefox web browser. Mozilla Firefox supports extension or add-on development by third parties. Integrating File Transfer Protocol (FTP) [3] and Internet Relay Chat (IRC) protocol [4] into the Mozilla Firefox web browser makes it work as a full-fledged FTP or IRC client. Although most web browsers can act as FTP clients, they, however, lack features of a standalone FTP client, such as secure FTP connection, account management and tree hierarchy of directories and subdirectories.

1.1 Converging Networks and Services

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 (APIs) to encourage third parties to use their services and develop new ones.

Today, websites, such as MEEBO and ILOVEIM, owned by third parties have integrated these APIs to offer visitors access to their accounts and chat or to 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, MEEBO requires that a user must have the latest versions of Adobe Flash Player and Java Virtual Machine before he/she can make multimedia calls. A visitor, however, browsing MEEBO can make multimedia call or start a chat session with any of his/her friends on GMAIL,

YAHOO and HOTMAIL rather than running each Instant Messaging engine on his/her PC. Standalone applications, such as Pidgin [5] on Linux OS, also provide this service.

1.2 Improving web browsing experience through session mobility

As earlier mentioned, 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 FTP and IRC extensions, to mention a few. The ability to access multiple social networking services or multiple instant messaging accounts via a single website is an example of converging services. Unceasingly, the web is rapidly evolving with new set of applications and services being pushed to the market day-in day-out. Visitors or web users have many 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 been proposed to alleviate this problem is the use of Open ID [6]. 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 the 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 enables 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.

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 [7]. Many efforts to achieve mobility between user devices are discussed in Chapter Two. The advancement from Triple play to Quad Play shows that mobility is a key feature required in the convergence of networks and services.

1.3 Content Sharing and Session Handoff in the web browsing context

Two problems are regularly encountered on the Internet, and these problems are described below using two Use Case Scenarios.

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 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 and using one of the PCs in a public library, 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. She realizes that she would have to restart the whole processes on her PC 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.

These problems are very similar to what individuals face today when surfing the Internet. Scenario one shows slow and inefficient 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.

This thesis proposes a new service called Content Sharing and Session Handoff. The service is introduced via the web browser interface to address these common problems faced by users.

1.4 Using SIP to achieve Session Handoff and Content Sharing

Session Initiation Protocol (SIP) is an application-layer control protocol that can establish, modify and terminate multimedia sessions [8]. On the other hand, Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative and hypermedia information systems [9]. Both protocols are application layer protocols; however, HTTP is a stateless protocol, while SIP can be made stateful or stateless. HTTP, identified as the principal protocol of web browsers, 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 project, HTTP session mobility between two or more web browsers has been achieved using SIP.

This project implementation leverages two Free Open Source Software (FOSS) packages, namely Mozilla Firefox [10] and PJSIP [11]. While Mozilla Firefox is a web browser developed by the Mozilla Corporation, PJSIP is a small footprint, high performance, flexible and Open Source protocol stack. The Mozilla Framework is widely used in the academic environment to develop web and desktop applications [12, 64]. An example is Zamudio et al. [12] in which a Mozilla Firefox extension called “Topaz” was developed to access Grid servers via the web browser.

1.5 Statement of Problem

Efforts from developers of web browsers are geared towards increasing the web browsers capabilities. Although synchronization of activities between two or more web browsers has now been achieved [13, 14], the two scenarios described above (in subsection 1.3) showed common problems encountered by the Internet Users when browsing the web. The two ways of further improving web browsing experience are the introduction of content sharing between users and web session handoff for users. At present, there is no smart way of referring someone else to view the same web page that a person is viewing at the same time.

Existing ways used today include sending the web page's URL via an Instant Messaging or email service, which are slow and require many processes. Content sharing entails referring someone else to view the same web page, and session handoff entails moving a stateful HTTP connection between PCs used by the same person. Although session handoff has been extensively explored by researchers, there is no standardized approach to it, and, in some cases, HTTP security rules are broken [15, 16].

Second, the convergence in networks and services has encouraged network operators and online service providers to open their services via APIs. Many online service providers rendering social networking services, however, still force users to install their proprietary software in order to use their services. Owing to this, users' PCs today have enormous software that has similar functionalities installed on them. Ways to discourage this enormous software installation are by the use of an adaptive User Agent Clients (UACs) and open standard technologies. The integration of protocols such as FTP [17] and IRC [18] into a web browser has proved that adaptive UACs are achievable. An integration of SIP into a web browser could further reduce the

multiple software installation problems. In this case, one software element, such as a web browser, could be used as a SIP client by installing the necessary extension on it rather than separately installing a new SIP client.

1.6 Motivation

Initially, web browsers were single-principal platform software for viewing a single website at a time [19]. Today, they are capable of viewing more than one website at a time. The major protocols in converged applications are SIP and HTTP, and most PCs run different UACs for different services. Web browsers, in the 1990s, only supported HTTP. But Today, Rapid Application Development (RAD) tools are making web browsers easily extensible. The emergence of adaptive UACs, in this case web browsers, which 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 feature desired by end users and offered by the IP Multimedia Subsystem (IMS) [20]. IMS, one of the available SDPs [21], has been adopted by the Third Generation Partnership Project (3GPP) and the Third Generation Partnership Project 2 (3GPP2) as the platform for delivering multimedia service to end users. It is meant to merge the Internet and the cellular worlds. It is also meant to make already existing services on the Internet and newer services accessible from most network access technologies. The different types of mobility are Terminal Mobility, Service Mobility, Session Mobility and Personal Mobility [22]. In Terminal mobility, a device can move between IP subnets and continue to be reached. In Service Mobility,

a user can maintain access to his service as he moves or changes devices and network service providers. In Session Mobility, while changing terminals, a user can still maintain a media session [22]. Personal Mobility, however, allows a single user located at different terminals to be addressed by the same logical address [23]. Although the differences are subtle, most especially among Service Mobility, Session Mobility and Personal Mobility, there is a need to be able to move web sessions between Personal Computers (PCs) or mobile devices. This will offer Personal Mobility to end users and facilitate Session Mobility in web browsing.

Khan et al. [24] proposes that the IMS should integrate HTTP Proxy, Real-time Streaming Protocol (RTSP) Proxy and Application Policy Function in order to harmonize SIP, HTTP and RTSP service delivery. While current IMS applications are mainly voice and video telephony centric, application policy functions that enable providers and subscribers to create, manage and implement policy rules, such as parental controls, privacy controls and other application sharing policies, are lacking. Such application policy functions should be able to filter application requests and deny access to applications that are not subscribed to. Khan et al. also proposes that HTTP traffic should pass the IMS media plane. The support for HTTP traffic in the IMS media plane makes this project a probable service that could run on the IMS if could be extended as later shown in section 7.2.4.

1.7 Contributions

This project is a novel approach to session mobility in HTTP by using SIP as the carrier of session data. It introduces a new service in web browsing context, namely content sharing and session handoff among web browsers. The service is derived from SIP Mobility mechanisms -

Third Party Call Control and Session Handoff mode flow for transfer to a single device. This service could work in a Peer-to-Peer (P2P) environment or a client-server model, where a SIP proxy server acts as a server to the web browsers or is required because the UACs are behind Network Address Translation (NAT) boxes.

This project proposes and implements a new web browser architecture that integrates SIP to achieve HTTP session mobility and provide SIP functionalities. In the implementation, a new extension, called TransferHTTP, is developed for a web browser, and the extension makes the web browser acts as a SIP client. That is, the extension integrates a SIP stack into the web browser in order to make voice calls. In addition, the web browser, with the extension installed, can transfer web sessions to another web browser.

These contributions are contained in the author's paper listed below.

Peer-Reviewed Conference Publications

1. M. Adeyeye and N. Ventura, "Extending Web Browsers Architectures to Support HTTP Session Mobility," Proceedings of the 3rd Annual CoNEXT Conference, New York, U.S.A., Dec. 2007.
2. M. Adeyeye and N. Ventura, "Achieving Web Session Hand-off using SIP," Proceedings of the 12th Annual IEEE International Symposium on Consumer Electronics, Algarve, Portugal, April 14-16, 2008.
3. 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 (WEBIST '08), Funchal, Madeira, Portugal, May. 4-7, 2008.
4. M. Adeyeye and N. Ventura, "Performance and Possible Deployment of HTTP Session

Mobility Service using SIP," Proceedings of the 11th Annual Southern Africa Telecommunication Networks and Applications Conference, Durban, Sept. 11-13, 2008.

1.8 Thesis Scope and Limitations

Although a Hybrid-based Architectural Scheme that requires a SIP proxy is proposed, the SIP proxy is not extended. Hence, no discussion on the SIP proxy is made. The implementation, however, supports Transport Layer Security (TLS). A TLS-supported SIP proxy is used in a client-server model during the project's test. A SIP proxy server is also required in a client-server environment when the UACs are behind NAT boxes. In this thesis, emphasis is on the web browser architecture that is extended.

Some of the features in other web session migration approaches include session registration and tracking and web history tracking. These features are not found in this project. The reason is that while other projects focus on the same person having access to his or her migrated web session, this project, most notably content sharing service, focuses on the interaction between two people with some degree of privacy.

The web browser extension (TransferHTTP) is developed for Mozilla Firefox version 2.0. In addition, session data are transferred in plain text format between web browsers, though recommendations are made on the security of session data.

1.9 Thesis Structure

Chapter 2, background and related work, explains HTTP, SIP and Types of SIP Mobility. In addition, it discusses related work carried out in the academic environment and the Industry. It also identifies weaknesses of each related work.

In Chapter 3, a hybrid-based architectural scheme is proposed. In addition, Third Party Call Control and Session Handoff are mapped to Content Sharing and Session Handoff in web browsing. How the existing web browser architecture is extended is also described in this chapter.

Chapter 4 presents the design and implementation of the service. The design entails using Data Flow Diagrams to describe the flow of session data and the interaction of the new web browser extension with the web browser. The interfaces of the web browser used in the implementation are also identified in this chapter. In addition, technologies required in the implementation are explained.

Chapter 5, results and evaluation, presents the user interface of the extension, the HTTP Session Mobility Performance, the web browser performance and evaluation of the project with other related work. The web browser performance entails memory consumption test as well as upload and download speeds test.

Chapter 6 discusses ideas resulting from the integration of a SIP stack into a web browser. These ideas are improving access to multimodal and multi-channel applications and

modifying how click-to-dial works. In addition, the possible deployment and commercialization of this project are discussed.

In Chapter 7, conclusions and recommendations, a summary of the thesis is given. In addition, the contributions made in this research work are highlighted and future work is suggested.

Chapter 2

2 Background and Related Work

This chapter provides information on HTTP, SIP, SIP Session Mobility types, related academic work and industry work. In the related academic work section, three architectures used to migrate web session are discussed. The three architectures are Client-based, Proxy-based and Server-based Architectural Schemes. Related SIP projects are also discussed under the related academic work.

The strengths and weaknesses of each related work are identified, and in some cases, the relevance of a work is mentioned. The strengths and weaknesses are based on user-client interaction, HTTP 1.1 security rules and delay, to mention a few.

2.1 HTTP and SIP

Hypertext Transfer Protocol (HTTP) is an Open Systems Interconnection (OSI) application-layer protocol. It is used in distributed, collaborative and hypermedia information systems. It is a text-based and stateless protocol, which through the extension of its Request methods, error codes and headers can be used for name servers and distributed object management systems [9].

Session Initiation Protocol (SIP) is also an application-layer protocol. It can establish, modify and terminate multimedia sessions [8]. SIP media sessions include multimedia conferences and Internet telephone calls. Session Description Protocol (SDP) is used to convey

information about media streams in multimedia sessions to allow the recipients of a session description to participate in the session.

```
REGISTER sip:137.158.125.230 SIP/2.0
Via: SIP/2.0/UDP 137.158.125.230:5061;rport
 ;branch=z9hG4bKPjnUgDPIjzH1sk78.qqavSxP2v23DguDj5
Max-Forwards: 70
From: <sip:micadeyeye@137.158.125.230>
 ;tag=i13ZQyZtei0iwQbepvC32H5gScUKFm-M
To: <sip:micadeyeye@137.158.125.230>
Call-ID: dbCfTE0SqxW92S53qbNPYQRoX.GqBfhr
CSeq: 44378 REGISTER
Contact: <sip:micadeyeye@137.158.125.230:5061>
Expires: 3600
Content-Length: 0
```

Figure 2-1. SIP Request Message

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP
137.158.125.230:5061;rport=5061
 ;branch=z9hG4bKPjnUgDPIjzH1sk78.qqavSxP2v23DguDj5
From: <sip:micadeyeye@137.158.125.230>
 ;tag=i13ZQyZtei0iwQbepvC32H5gScUKFm-M
To: <sip:micadeyeye@137.158.125.230>
 ;tag=3989610a8b0968380faf33990ce2dc12.fdbf
Call-ID: dbCfTE0SqxW92S53qbNPYQRoX.GqBfhr
CSeq: 44378 REGISTER
Contact: <sip:micadeyeye@137.158.125.230:5061>;expires=3600
Content-Length: 0
```

Figure 2-2. SIP Response Message

```
http://code.google.com/css/codesite.pack.01312008.css

GET /css/codesite.pack.01312008.css HTTP/1.1
Host: code.google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.1)
            Gecko/2008070208 Firefox/3.0.1
Accept: text/css,*/*;q=0.1
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://code.google.com/
Cookie: PREF=ID=dd2140b96652c153;TM=1220671711;LM=1220684063:GM=1
```

Figure 2-3. HTTP Request Header

```
HTTP/1.x 200 OK
Via: 1.1 SRVWINISA005
Connection: Keep-Alive
Proxy-Connection: Keep-Alive
Transfer-Encoding: chunked
Expires: Fri, 19 Sep 2008 17:33:42 GMT
Date: Fri, 19 Sep 2008 16:33:42 GMT
Content-Type: text/css; charset=UTF-8
Etag: "76152fd8436a5f02623d3255fb0a9f60"
Server: codesite_static_content/-1
Last-Modified: Thu, 04 Sep 2008 21:40:46 GMT
Cache-Control: private, x-gzip-ok=""
```

Figure 2-4. HTTP Response Header

While HTTP is a stateless protocol, SIP can be a stateful or stateless protocol. HTTP as a stateless protocol makes multiple and unique HTTP Requests for every component of a web page. In order to make HTTP stateful, RFC 2965 defines the use of set-cookie and set-cookie2 [25]. Alternative approaches to making HTTP stateful include the use of Hidden Elements and Universal Resource Locator (URL) rewriting [26]. Both HTTP and SIP are request-response and text-based application protocols.

SIP supports most of the security measures attributed to HTTP, such as Secure Sockets Layer (SSL), Transport Layer Security (TLS), Secure Multipurpose Internet Mail Extension (S/MIME) and HTTP-Authentication [8]. SIP is a Session Layer Protocol but referred to as an Application Layer Protocol due to the absence of Session Layer in the TCP/IP Model [15]. Figures 2-1 and 2-2 show examples of SIP Request and Response Headers, and Figures 2-3 and 2-4 show examples of HTTP Request and Response Headers.

2.2 Types of SIP Mobility

Although Session Handoff described in Shacham et al. [27] 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 thesis. This is because session handoff in this project is transferred to only one device. Figures 2-5 and 2-6 show Third Party Call Control and Session Handoff mode flow for transfer to a single device, respectively [27, 28]. These two types of SIP Mobility are mapped to Content Sharing and Session Handoff in the web-browsing context in Chapter 3. Figure 2-5 demonstrates Third Party Call Control (3pcc) in SIP [27, 28].

A, B and Controller are three mobile phones, and the controller and B are already in a call session. When the controller wants the session transferred between A and B, a SIP INVITE with no Session Description Protocol (SDP) is sent to A. A 200 OK reply containing A's SDP is sent to the controller from A. The controller sends another SIP INVITE, using A's SDP, to B. B replies with a 200 OK containing its SDP and the controller acknowledges it with a SIP ACK message. Thereafter, the controller sends an acknowledgment, SIP ACK, containing B's SDP to

A. A new call session that replaces the earlier one now exists. At the end of the session, when a SIP BYE from A is sent to the controller, the controller replies with a 200 OK to end the session. Thereafter, a new SIP BYE is sent to B from the controller and B replies with a 200 OK.

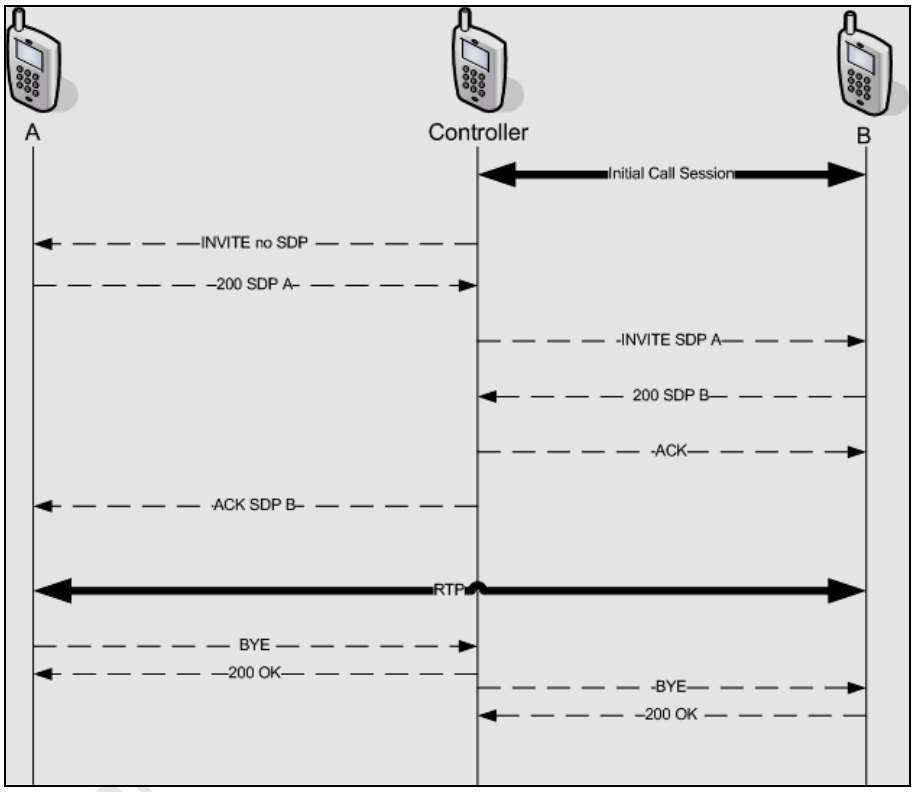


Figure 2-5. Third Party Call Control in SIP

Figure 2-6 shows Session Handoff mode flow for transfer to a single device [27]. Initially, there exists a call session between the controller and B. When a session handoff is required, a SIP REFER is sent to A. A replies the controller with a SIP 202 Accepted. The SIP REFER message contains "Refer-To" and "Referred-By" header lines. It also contains S/MIME

authentication body. A SIP INVITE with “Replaces” header line is directly sent to B from A. The SIP INVITE contains S/MIME authentication body. This is used to keep a malicious user from indiscriminately replacing another user’s session. As a new session, the controller subscribes to the Dialog Event Package of A. When a normal session now ensues between A and B, a SIP NOTIFY is sent to the controller from A [27]. A receives a 200 OK and the session between the controller and B is torn down. The controller is no longer in the communication path, and a SIP BYE can be sent between A and B when the session has to be terminated.

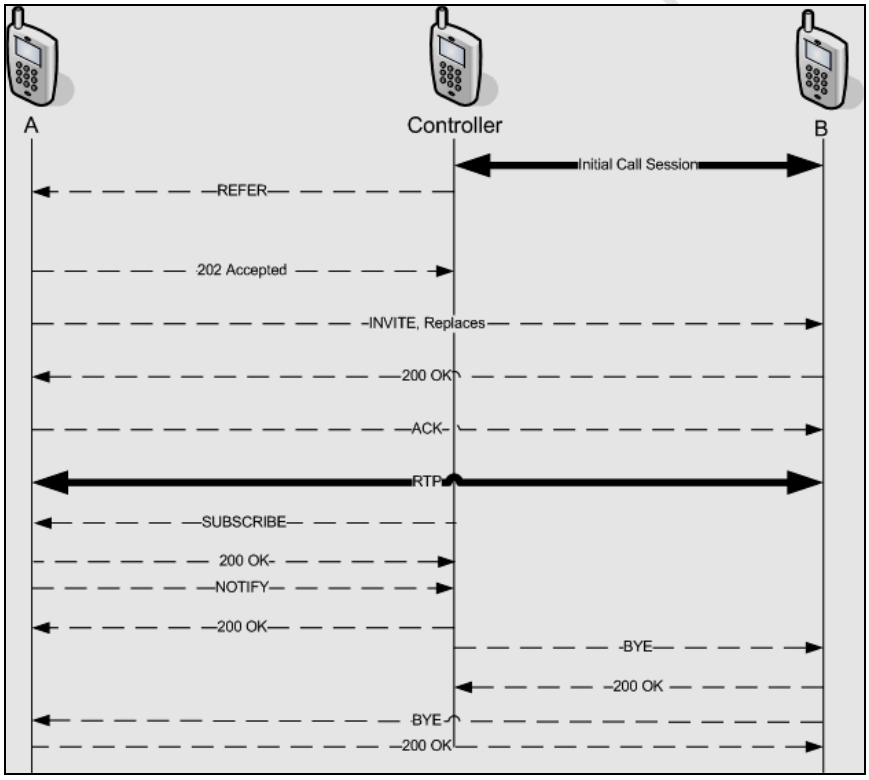


Figure 2-6. Session Handoff mode flow for transfer to a single device

2.3 Related Academic Work

In the literature, web session handoff has been carried out using different schemes, namely Proxy-based, Client-based, and Server-based Architectural Schemes [15, 16, 29]. A Client-based Architectural Scheme requires modifying the web browser, and a Server-based Architectural Scheme requires modifying the web server. In some cases, a Proxy can be placed along the communication path of the client and the server. This is called a Proxy-based Architectural Scheme. Figures 2-7, 2-8 and 2-9 show Server-based, Client-based and Proxy-based Architectural Schemes, respectively. Session Handoff Component (SHOC) in the figures refers to the software responsible for the web session migration.

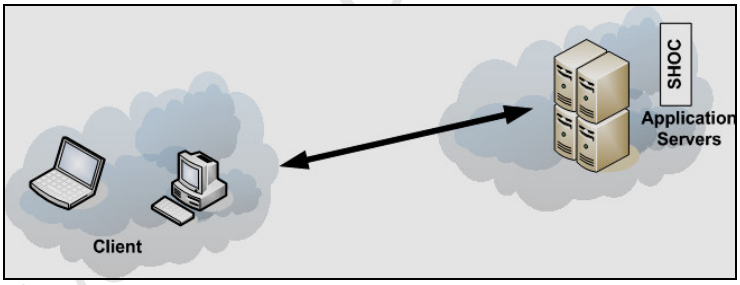


Figure 2-7. Server-based Architectural Scheme

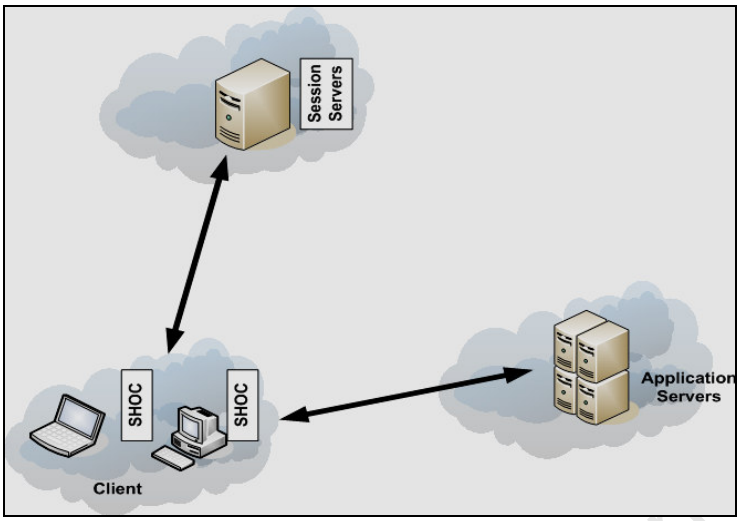


Figure 2-8. Client-based Architectural Scheme

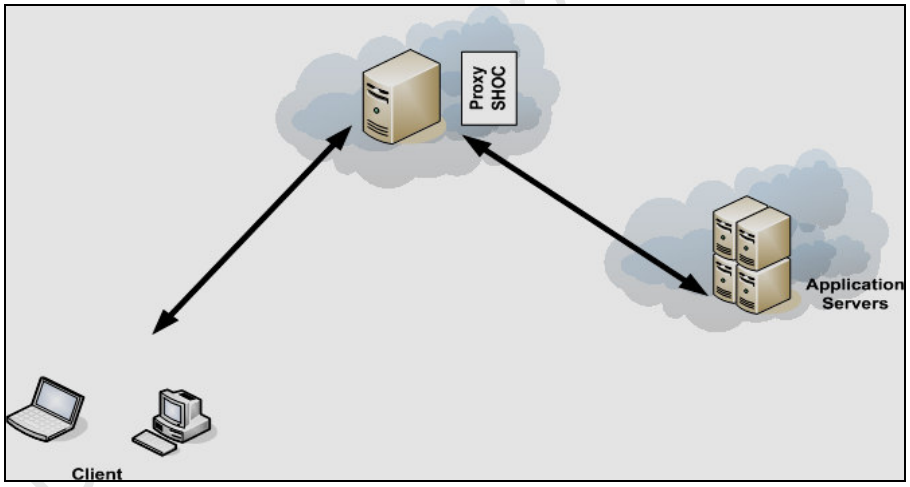


Figure 2-9. Proxy-based Architectural Scheme

It is worth mentioning that most research work on mobility focuses on terminal or host mobility. Examples include Jan et al. [30], Snoeren et al. [31], Atiquzzaman et al. [32, 34] and Stewart et al. [33]. In terms of Personal Mobility, approaches that have been used include Stefano et al. [35], Roussopoulos, et al. [36], Appenzeller et al. [37], R. Liscano et al. [38], C. H.

Herman et al. [39], K. Raatikainen [40] and Bagrodia et al. [41]. These projects address problems related to Mobile IP and using middleware for mobile networks, to mention a few. What distinguishes this project from those mentioned above is that it addresses HTTP session mobility, while those mentioned above address terminal mobility. This project addresses HTTP session mobility at the application layer, while those mentioned above address terminal mobility not only in the application layer, but also other OSI layers. Similar research work that addresses HTTP session mobility at the application layer is discussed below.

Browser State Preservation and Migration (BSPM) Service [29] is one way of handing over web sessions between user devices. This is a Client-based Architectural Scheme. Its implementation requires a BSPM repository server and a BSPM plug-in for Microsoft Internet Explorer (IE) 5.0. The BSPM plug-in of the first device takes a web session snapshot of its browser state and saves the snapshot at a BSPM repository server. Figure 2-10 shows the BSPM Plug-in Bar before and after Sign-on. The web session snapshot captures the browser's running state, including the last page that appears on the browser, values of document objects, values of scripting objects, values that a user enters in forms on the last page, browser history for back and forward pages and cookies. Thereafter, a browser on the second device can retrieve the saved snapshot to resume the saved state and session. It is required that a user signs on to the BSPM proxy before he can take a web session snapshot. BSPM also provides an optional lock mechanism whereby a user can protect a saved web session snapshot with a session password.

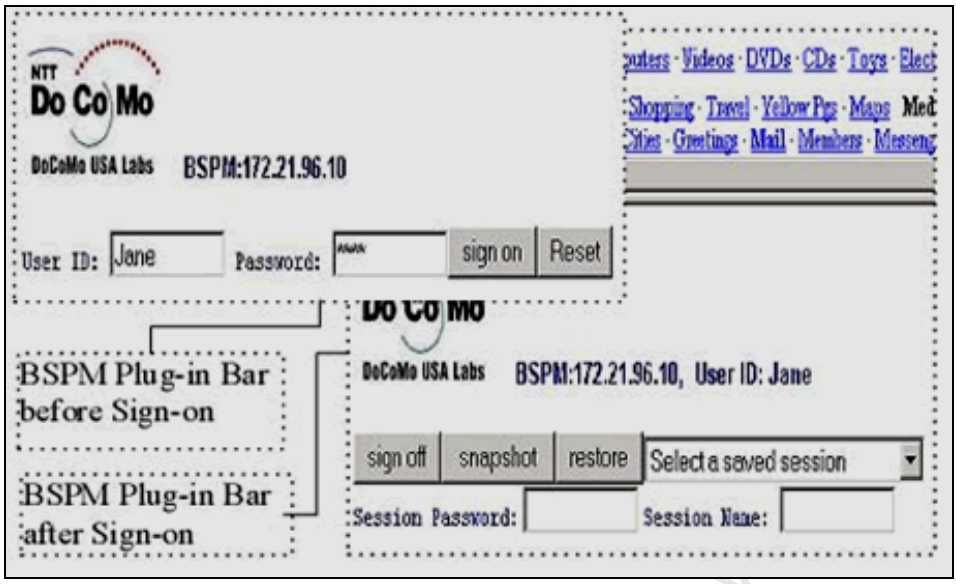


Figure 2-10. BSPM Plug-in Bar before/after Sign-on

In a work by Canfora et al. [15], a Proxy-based Architectural Scheme is implemented. Its building blocks are a User Authenticator, a Server Authenticator, a Cache Manager and a Cookie Manager. The implementation is based on extending a filtering system called Muffin [42]. Muffin is freely available under the GNU General Public License and supports HTTP/0.9, HTTP/1.0, HTTP/1.1 and SSL. In the implementation, the Proxy is referred to as MuffinSH. The main interfaces implemented by MuffinSH are *RequestFilter, ReplyFilter, ContentFilter, HttpFilter* and *RedirectFilter.* By using the filters, the proxy is able to store, for each user and each domain, web content received from a server before the user hands-off the session.

Another Proxy Based Architectural Scheme is proposed and implemented by Hsieh et al. [16]. In the work, a session is registered and tracked at a User Agent Proxy (UAP). Processing of a proxy session is classified into three stages, namely registration, browsing and tracking and session handoff. The UAP is a special purpose web proxy that replaces a normal web proxy. It is

able to perform handoff related functionalities in addition to acting as a web proxy. The UAP adds user profile as CC/PPex headers [43] in HTTP Requests sent by browsers not supporting CC/PPex. Figure 2-11 shows the PC Client Program. This work is implemented in the Microsoft Windows environment. The client program is implemented as a plug-in in Internet Explorer (IE). It is the rectangular toolbar under the address bar in Figure 2-11.

With the client program, user-client interaction is gained, and a user could perform session registration and handoff directly in the IE's main window. User-client interaction refers to ability to continue what a user is doing during a session transfer. For example, a user filling a web form will be able to continue filling the form after the session transfer instead of starting afresh. However, web session handoff could work without the client program. The configuration window of the client is also shown in Figure 11. It enables a user to input his username, password, IP address of a UAP, port number on the UAP for accessing the UAP's functionalities, basic user preferences for adaptive browsing and URL of the user profile. The client program for the Microsoft Pocket PC PDA is a standalone program and offers the same functionalities as a PC's client program.

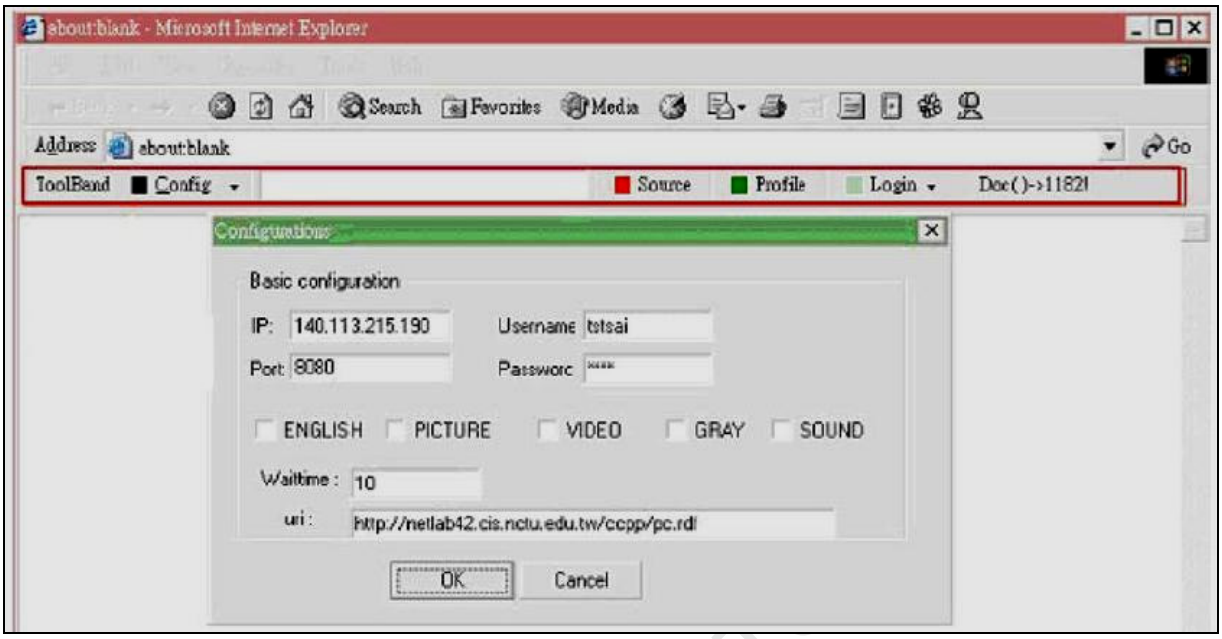


Figure 2-11. PC Client Program

2.3.1 Weakness of the related academic work

In Song [29], BSPM repository server remains idle most of the time. That is, it does not act as a web cache or a session tracker. In addition, it only works when a web session has not expired. It is not a pure client-based architectural scheme because it still requires a repository server during its operation. Hsieh et al. [16] is similar to Canfora et al. [15] because both are Proxy-based Architectural Schemes. It was found that Song [29] lacks some features available in Hsieh et al. [16]. Examples are session registration and tracking session information. This is because Song [29] is a client-based architectural scheme, while Hsieh et al. [26] is a proxy-based architectural scheme. However, both Hsieh et al. [16] and Canfora et al. [15] violate the HTTP 1.1 security during their implementations.

In Hsieh et al [16], the User Agent Proxy (UAP) could store responses that are specified as non-cachable. This storing operation violates the specifications in RFC 2616 [9] and RFC 2965 [25]. Also in Canfora et al [15], the HTTP 1.1 security is broken by storing users' credentials in the proxy in order to make the proxy manage authentication on behalf of the client. It poses some confidentiality issues, such as making users aware that their credentials are stored in the proxy. The proxy carries out user or client authentication using HTTP Basic Authentication, where username and password are sent in an unencrypted form. In addition, the implementation loses user-client interaction.

In summary, both proxy-based implementations do not only break HTTP Security, but also modify all HTTP Request and Response Headers. Hsieh et al. [16] loses user-client interaction when the optional client program is not used, and Canfora et al [15] has no support for optional client program. Hence, Canfora et al [15] loses user-client interaction. Conversely, Song [29], a client-based architectural scheme, provides user-client interaction but does not offer web cache or session tracking found in Hsieh et al [16].

2.3.2 Related SIP Projects

Projects that take advantage of SIP extensibility include the Akogrimo Project [44]. It involves embedding web service data in a Session Description Protocol (SDP). In addition, in an expired IETF Internet draft [45] by Wu and Schulzrinne, two approaches are identified for transferring URL between two web browsers. The first approach is by sending the URL via a SIP MESSAGE method. This research work is similar to this first approach. Furthermore, in this

research work, the session data attributed to the URL are also transferred. The second approach is by using SIP NOTIFY method. This approach is 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. The shortcoming of the project is that it is discontinued neither are the approaches standardized. However, the approaches show different ways of achieving web share using SIP MESSAGE and SIP NOTIFY methods.

Another project that exploits the SIP extensibility and very similar to this research work was carried out by Munkongpitakhun et al [46]. In the project, a SIP stack is also integrated into a web browser, and a SIP MESSAGE method is used to transfer session data as well. The project was discovered at the time of writing of this thesis. Although it is very similar to this research work, the project, like other related academic work, only addresses session handoff. In terms of the signalling, two web browsers have to establish a call session using a SIP INVITE method before a session handoff can take place. This approach introduces unnecessary overheads in the signalling because it is not stated if users need to be involved in a multimedia session, such as voice call, before session handoff can take place. Although the work is also an attempt to solve the session handoff problem, the implementation details are limited and no software is available to confirm their findings.

2.4 Related Industry Work

Two closely related software packages, which are related to this research work, are discussed here. These are the Google Browser Sync and the Mozilla Weave. The differences

between the two are also discussed here. In addition, their strengths and weaknesses are identified.

2.4.1 Google Browser Sync

Google Browser Sync is a product of Google Corporation. It is an extension for the Mozilla Firefox, which makes it possible for a user to synchronize his/her currently opened tabs and windows across different PCs, where the extension must have been installed. Identification from the user is required before the synchronization process is carried out. Other browser user-specific data that can be continuously synchronized between PCs are cookies, saved passwords, bookmarks and history [13].

2.4.2 Mozilla Weave

Google Browser Sync is not installable on Mozilla Firefox version 3.0. The reason is that Mozilla Weave replaces it in Mozilla Firefox version 3.0, though there is support for it in older versions of the web browser. Mozilla Weave is a project under development by Mozilla Corporation. It aims to create a repository where each user's bookmarks, history, privacy settings and preferences can be stored and retrieved as he/she moves across PCs [14]. At the time of writing, a prototype already exists and is installable as an extension on the Mozilla Firefox version 3.0. Another related Mozilla Corporation Product is the Firefox Crash Recovery. This is a built-in recovery mechanism in the web browser. With this mechanism, a user can recover all

tabs and windows earlier opened after an abrupt browser crash. While the Mozilla Weave works between two or more PCs, the Mozilla Crash Recovery Project only works on a single PC.

2.4.3 Extensible Messaging and Presence Protocol

Extensible Messaging and Presence Protocol (XMPP), formerly known as Jabber Protocol, is also widely used to achieve interaction between UACs [47]. It is commonly used for instant messaging and developing online games between two or more UACs. Although XMPP could be integrated into a web browser to achieve content sharing and session handoff between web browsers, using SIP in this project offers the advantage of having an adaptive UAC in which a web browser could act as a SIP client for voice call and possibly be extended to support other SIP related functionalities.

2.4.4 Weaknesses of the related industry work

These industry solutions are distinctively different from this research work, though all of them tend to improve the web browsing experience. Although Google Browser Sync transfers session data between PCs used by the same person, it suffers a triangular delay. That is, session data are first pushed to the Google web server and then pulled by the extension at another end after user's identification. Similarly, Mozilla Weave also suffers the same triangular delay found in Google Browser Sync, though it gives users more control over their data and personal information than Google Browser Sync [13].

Both Mozilla Weave and Google Browser Sync are based on conventional HTTP, which does not provide support for Peer-to-Peer (P2P) interaction, pure asynchronous events (in the case of SIP SUBSCRIBE/NOTIFY, though AJAX does) and multimedia sessions. XMPP, however, could be used in place of SIP to move the web session data, but it is not capable of providing multimedia services that SIP offers.

2.5 Conclusion

In summary, this chapter has explored the strengths and weaknesses of related projects to this research work. These projects are classified into related academic work and industry work. The related academic work also includes related SIP projects. While one of the related SIP projects is discontinued, most of the other related academic work have weaknesses, such as lack of user-client interaction and violation of HTTP 1.1 security rules. The next chapter describes the HTTP mobility service by mapping it to the SIP mobility types. It also explains modifications made to the existing web browser architecture.

Chapter 3

3 Service Description

The Hybrid-based Architectural Scheme, which this project is based on, requires extending a web browser to support SIP and setting up a SIP proxy server for the UAC registration. UAC in this thesis refers to the SIP User Agent in the web browser. This chapter presents the Hybrid-based Architectural Scheme, mapping Third Party Call Control and Session Handoff mode flow to Content Sharing and Session Handoff in web browsing, respectively. It also presents the extended web browser architecture.

3.1 The Hybrid-based Architectural Scheme

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 3-1 shows two scenarios. First, User A migrates a web session from his Notebook PC at work to his Desktop PC at home. In the second scenario, User A shares the current web page he is viewing with a friend, User B. The first scenario is referred to as session transfer, and the second is referred to as content sharing. Figure 3-1 also shows the client interacting with the SIP proxy server. The SIP proxy server is also referred to as a User Agent Server (UAS). In this

project, it is responsible for the UAC registration and could be made to track all interactions between the UACs.

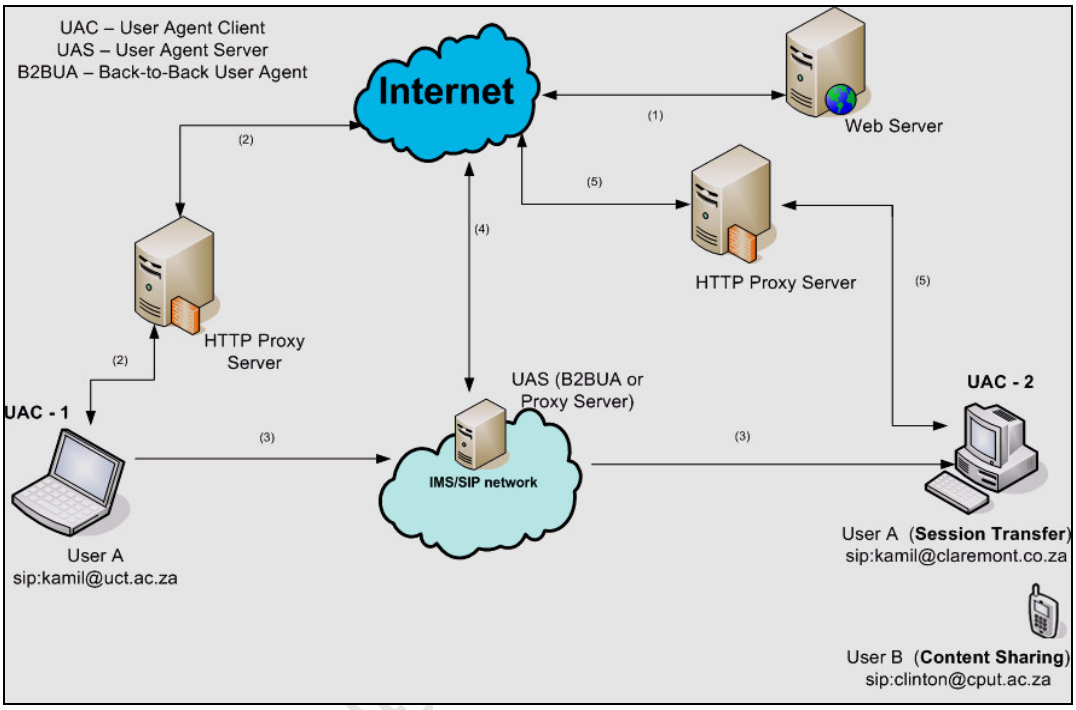


Figure 3-1. The Hybrid-based Architectural Scheme

3.2 Mapping Third Party Call Control in SIP Mobility to Content Sharing in Web Browsing

Third Party Call Control (3pcc) in SIP involves a controller transferring a session between it and its correspondent UAC to another UAC (Figure 2-5). 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 3-2, UAC 1 and the HTTP web server represent two UACs 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.

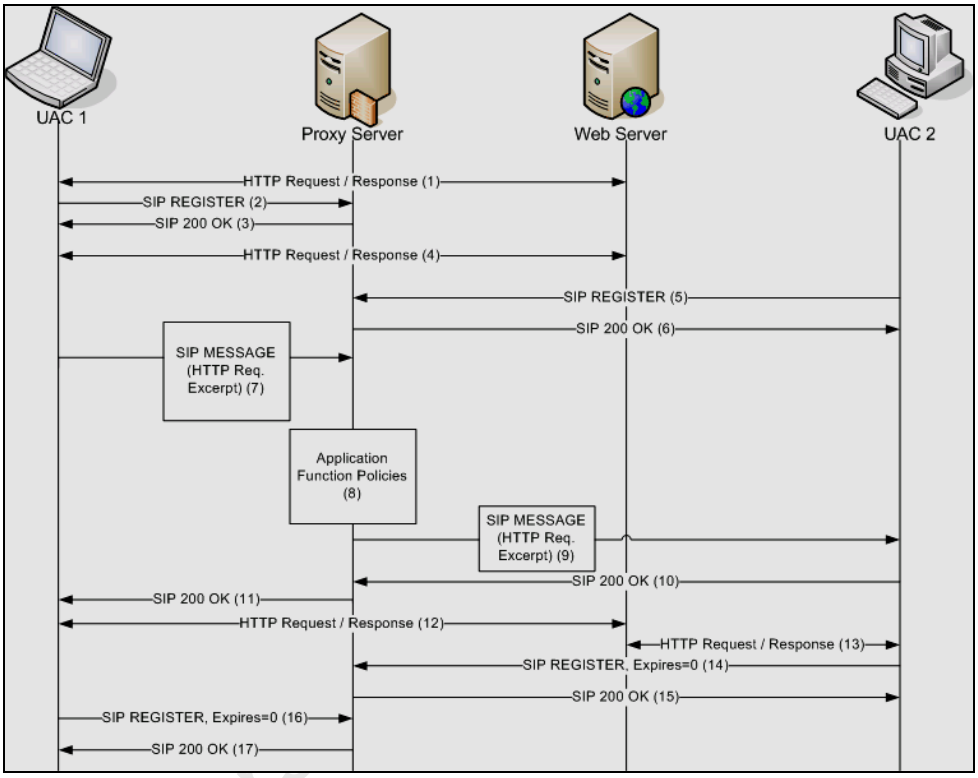


Figure 3-2. The Content Sharing Process Flow

When UAC 2 has successfully registered with the SIP proxy server and the user at UAC 1 makes a content sharing request to UAC 2, 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, like filter criteria in the IMS [48], be met, the SIP proxy server would 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 3-2 represent the controller and UAC 2 in 3pcc, respectively, successful mapping of 3pcc in SIP (Figure 2-5) to content sharing in web browsing (Figure 3-2) 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 Figure 2-5, the controller wants its call session with UAC 2 transferred to UAC 1 and still wants to have access to the signalling. In Figure 3-2, UAC 1 wants its web session with the HTTP web server transferred to UAC 2 and still wants to maintain access to the signalling; that is, it wants 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 web 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. Although the service is specifically meant for web pages that requires no identification or authentication, the user at a destination UAC might be required to sign in before accessing a 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 is 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. In addition, 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.

3.3 Mapping Session Handoff mode flow in SIP Mobility to Session Handoff in Web Browsing

Session Handoff also involves a controller transferring a session between it and its correspondent UAC to another UAC (Figure 2-6). 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 3-3, 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 to the SIP proxy server, and the user continues with his/her web browsing. As the user continues to browse the web, an excerpt of each HTTP Request Header from UAC 1 could be forwarded to the SIP proxy server using the SIP MESSAGE method. With this, all HTTP Requests could 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 are sent to the destination SIP Universal Resource Identifier (URI) via the SIP proxy server. Unlike the Content Sharing earlier described, session data alongside the URL are sent to the destination SIP URI. Should necessary application function policies be met, the SIP MESSAGE would be forwarded by the SIP proxy server to UAC 2.

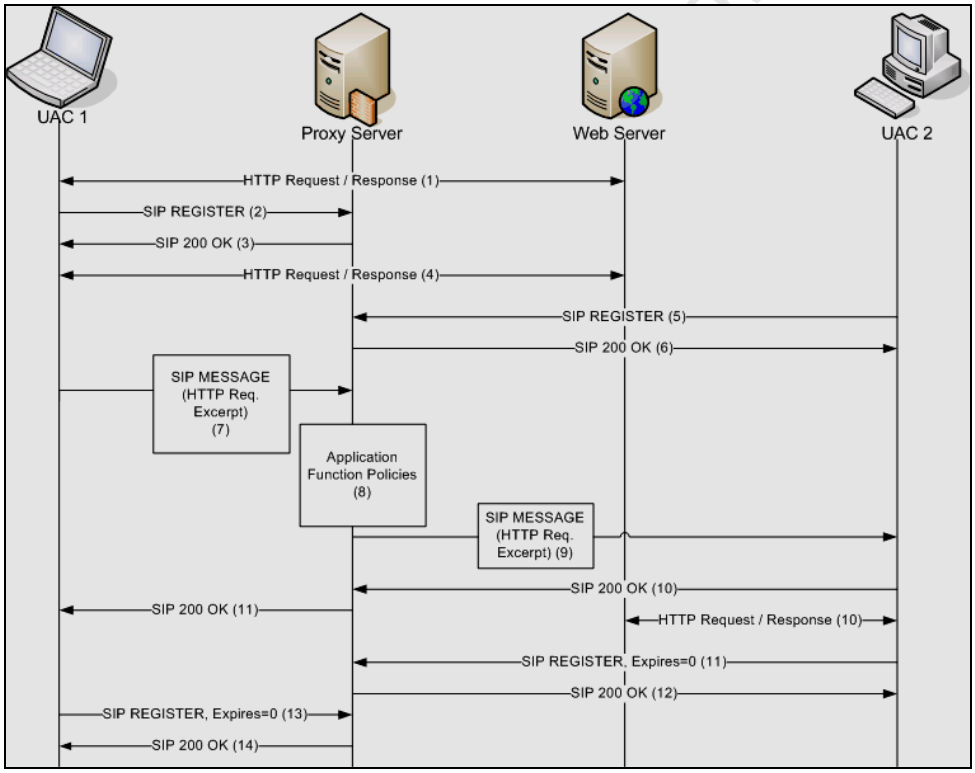


Figure 3-3. The Session Handoff Process Flow

Similarly, where UAC 1 and the web server in Figure 2-6 represent the controller and the UAC 2 in Session Handoff in SIP, respectively, successful mapping of Session Handoff in SIP

(Figure 2-6) to Session Handoff in web browsing (Figure 3-3) 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 Figure 2-6, the controller wants its call session with the UAC 2 transferred to UAC 1, but thereafter, does not have access to the signalling. In Figure 3-3, 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, it does not have a 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 web browsers have the same session data at the same time. Hence, when session data are transferred to another web browser, they are deleted from the source web browser. At this point, the source loses a stateful HTTP connection with the HTTP web server. As earlier stated, session data could be in the form of cookies, hidden HTML elements and rewritten or encoded URL. These session data are transferred between UACs (SIP User Agents in web browsers) using the SIP MESSAGE method.

3.4 The Extended Web Browser Architecture

A reference architecture for opened and closed source (proprietary) web browsers was described by Grosskurth and Godfrey in [49]. The development of web browsers often requires different standard libraries, and these variations have led to different architectures for the existing web browsers. They, however, share common building blocks. Grosskurth and Godfrey [49] identified the building blocks as User Interface, Browser Engine, Rendering Engine, Networking, JavaScript Interpreter, XML Parser, Display Backend and Data Persistence. These building blocks, shown in Figure 3-4, are similar to the building blocks shown in Figure 3-5. While the HTML Manager in Figure 3-5 represents the XML Parser and the JavaScript Interpreter in Figure 3-4, the Input Manager in Figure 3-5 represents the Data Persistence in Figure 3-4.

Other building blocks, namely User Interface, Browser Engine, Rendering Engine, Networking and Display Backend in Figure 3-4, could be mapped into Final Presentation, Content Manager, HTML Manager, Networking Subsystem and Protocol Handler in Figure 3-5. Figure 3-5 further gives components of each building block that are not mentioned in Grosskurth and Godfrey [49]. Regarding the modification of existing web browsers' architecture to support a SIP stack, Figure 3-5 shows the proposed modifications to the web browsers' architecture. There are three forms of rectangles, namely normal, round-edge dashed-line and round-edge solid-line rectangles, in the figure (Figure 3-5). For clarity, the round-edge solid-line rectangles have wider border line than other rectangles. These round-edge solid-line rectangles and the SIP stack (in a normal rectangle) represent parts of the architecture that will be extended and a new feature that will be integrated, respectively.

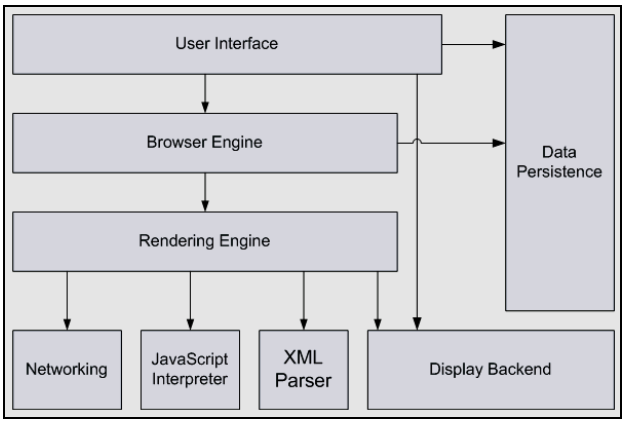


Figure 3-4. Web Browser Reference Architecture

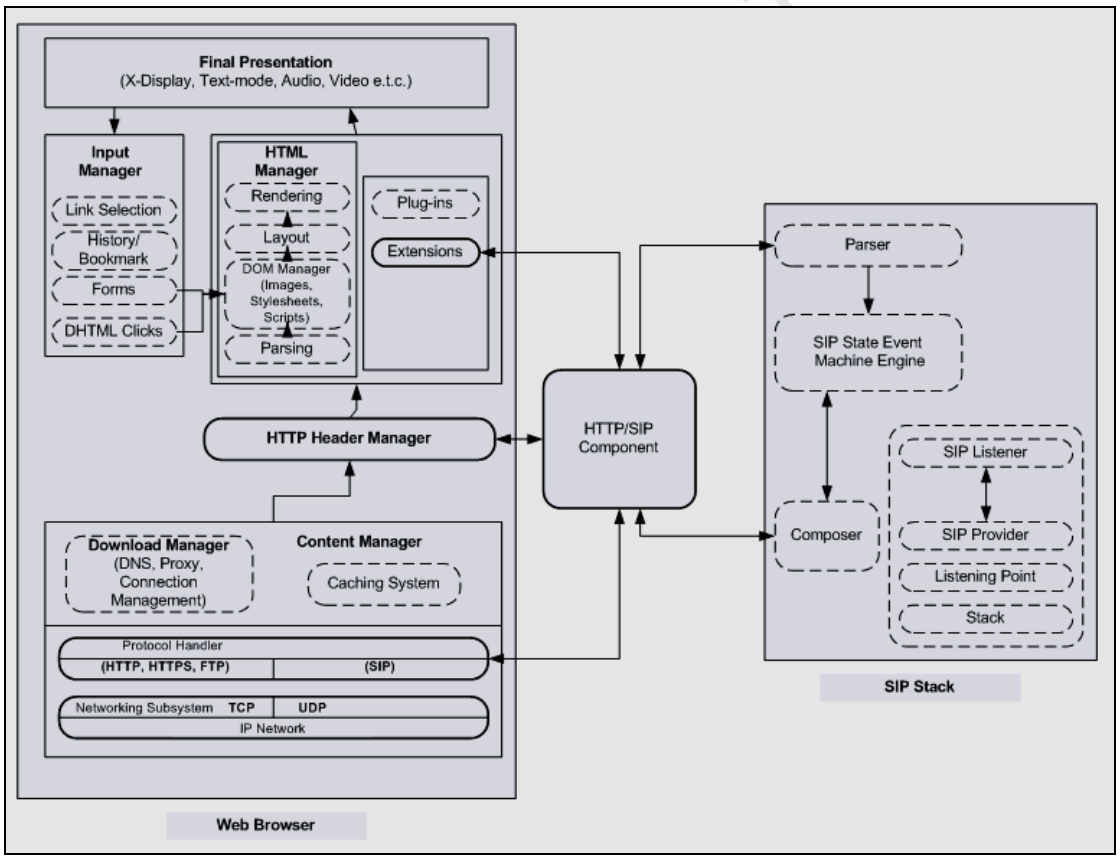


Figure 3-5. The Extended Web Browser Architecture

The round-edge solid-line rectangle called “Extension,” which is close to the HTML Manager, represents the User Interface (UI) that a user of the web browser would use to interact with the SIP stack. Extensions in web browsers often have interfaces for selecting one or more tasks. Extensions are different from plug-ins in web browsers. While a plug-in is a small runtime enhancement, an extension enables one to pass information and add functionality to existing interfaces. An example of a plug-in is the Adobe Flash Player. In this research work, an extension called “TransferHTTP” is developed for the Mozilla Firefox web browser. Other notable extensions are Firebug [50] and FireFTP [17].

When buttons on the UI are clicked in this TransferHTTP extension, they trigger events, such as registering the client and sending excerpts of the HTTP Request/Response using the SIP MESSAGE method. Although SIP MESSAGE can be sent in a session mode or a pager mode [48], a pager mode was used in this implementation. Pager mode was chosen because the SIP stack used did not support Message Session Relay Protocol (MSRP) at the time of the implementation. In addition, session mode would have added unnecessary overheads to the solution.

Processes, such as triggering SIP register and deregister events and sending HTTP Request/Response excerpts using SIP MESSAGE, take place inside the HTTP/SIP Component block in Figure 3-5. SIP transactions are handled by the SIP stack and work seamlessly with the HTTP/SIP component, which helps in building the SIP messages. The HTTP Header Manager is required to interact with the HTTP/SIP Component by providing the HTTP Request/Response excerpts, such as Request-URI and Set-Cookie. The Protocol Handler implements the SIP protocol alongside the existing protocols, such as HTTP and HTTPS. In the same manner, the Networking Subsystem integrates User Datagram Protocol (UDP) and Transmission Control

Protocol (TCP), which are required during the SIP transactions. The other blocks, such as Final Presentation, Input and Download Managers remain unmodified.

3.5 Conclusion

In summary, this chapter has presented a hybrid-based architectural scheme for web session migration, mapping 3pcc and session handoff in SIP to content sharing and session handoff in the web-browsing context and a modified web browser architecture. Content sharing and session handoff scenarios were illustrated in the hybrid-based architectural scheme. Assumptions by which 3pcc and session handoff in SIP are mapped to content sharing and session handoff in web browsing context were identified. Lastly, building blocks in the existing web browser architecture that would be modified were identified and processes that would take place in them were discussed.

The next chapter presents the service design and implementation. The implementation was based on Mozilla Firefox version 2.0, and the interfaces of the browser used in the implementation are discussed. In addition, the interface of an XPCOM, which formed an abstraction over the SIP stack, is discussed.

Chapter 4

4 Service Design and Implementation

This chapter gives details of the service design and its implementation. This project works in a server-less or Peer-to-Peer (P2P) environment, where a SIP proxy server is not required, and the UACs could interact based on their SIP addresses. The implementation of application policies function in the SIP proxy server is not discussed in this thesis because it is optional. A suitable SIP proxy or a STUN server is however required in a client-server environment, when the UACs are behind NAT boxes.

In this project, a SIP proxy server was only set-up for the UACs registrations, and the UACs were not behind NAT boxes. In addition, most tests were carried out in a P2P environment. Most tests were carried out in a P2P environment because they were specifically carried out on the web browser, which integrated a new protocol. However, a client-server model is strongly recommended, should the service be deployed in a large network, where security is of great concern. The deployment and commercialization of this service will later be discussed in Section 6.3. The implementation was carried out on Mozilla Firefox version 2.0, and the components of the browser used in the implementation are explained in detail.

4.1 Design of the TransferHTTP Extension

A Data Flow Diagram (DFD) is a graphical tool in software engineering used for modeling information-processing systems, whole organizations, business planning, and strategic planning [51]. In Figures 4-1 to 4-4, the round-edge rectangle is used to represent a Process. A *Process*, in

DFD, represents the transformation of inputs into outputs. The arrows in Figures 4-1 to 4-4 that go in or out of a process represent a *Data Flow*. A *data store*, represented by an incomplete rectangle in Figure 4-1, is a storage location in which data can be stored or retrieved. The normal rectangles in the figures represent external entities. An *External Entity* interacts with a system by either providing or receiving data. A *System* can be represented by organizing DFDs in a series of levels. Each DFD is a different level of abstraction. The highest DFD level is Level 0. As the DFD level increases, there is an increment in the level of details for representing a system or some of its components.

Figures 4-1 and 4-2 provide the highest DFD levels (Level 0) of two major functionalities in the extension, namely web session transfer and web session receipt. Figures 4-3 and 4-4 present Level 1 DFD for the same functionalities. In Figure 4-1, a user enters the destination SIP Address. The SIP client credentials and Proxy address are pulled from the Preferences of the browser into the system. The system pushes the session data with the destination SIP address to the SIP proxy. In Figure 4-2, the SIP proxy forwards the received information to the system, and the system sends a notification to the user.

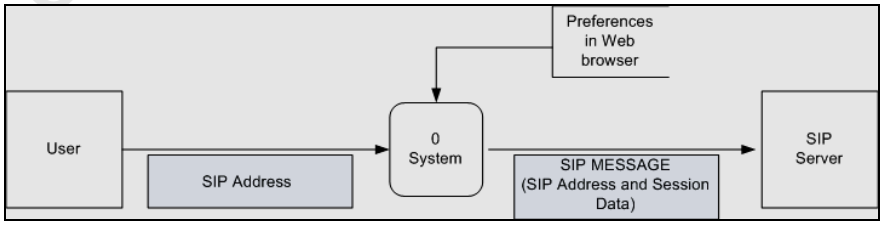


Figure 4-1. Level 0 DFD for web session transfer

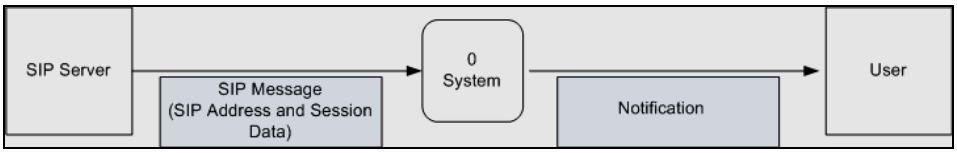


Figure 4-2. Level 0 DFD for web session receipt

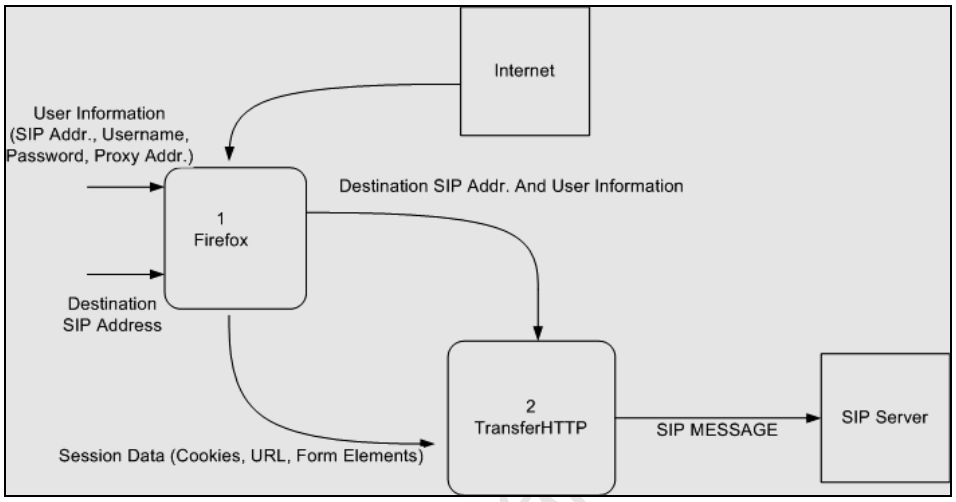


Figure 4-3. Level 1 DFD for web session transfer

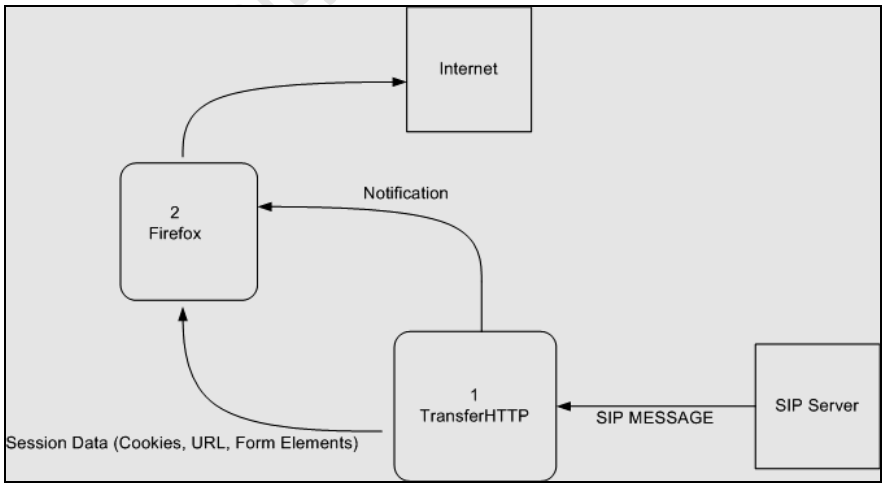


Figure 4-4. Level 1 DFD for web session receipt

Figures 4-3 and 4-4 provide diagrams that are more detailed than Figures 4-1 and 4-2. In Figure 4-3, the two arrows indicate that the SIP client credentials and the destination SIP address are entered. Although the destination address could often change, the credentials usually exist for a long period. During a web session transfer, the credentials (also referred to as the user information) with the destination SIP address are sent to the TransferHTTP extension. Simultaneously, the session data with the URL are sent to the extension.

An arrow from the browser (Firefox) to the Internet shows that the browser is accessing the Internet. TransferHTTP extension processes this information and sends it out as a SIP MESSAGE to the appropriate SIP proxy server. In Figure 4-4, the SIP MESSAGE is forwarded to the TransferHTTP extension of the destination browser, and the extension sends a notification to the browser. If the notification is accepted, the session data will be forwarded to the browser and the resource will be pulled from the Internet.

4.2 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. The packet size of the SIP stack was extended to 16kB in order to send session data at once. Session data are transferred in SIP MESSAGE method because an SDP text payload must not be more that 1kB in length [52].In addition, SIP INFO method could not be used because it is only used for communicating mid-session signalling information along the signalling path for a call. However, SIP INFO method could be used when a session already exists between two UACs, such as a voice call. Although a

SIP PUBLISH could be used in place of a SIP MESSAGE, it is better used alongside a SIP SUBSCRIBE to achieve conference model of web browsing, as explained in section 2.3.2.

After successful integration of a SIP stack, the web browser was subjected to a performance test to determine outcomes when the SIP stack was running as a background service. Here, the outcomes refer to probable changes in download and upload speeds of the browser, as well as its memory consumption. The SIP stack in the TransferHTTP extension 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 [53]. This approach provided a new layer of abstraction to the web browser in order to integrate the protocol (SIP). JavaScript was used in the implementation to pass user data to the extension's XPCOM, which was written in C++ programming language.


```
<xml version="1.0">
<httpsession>
<url>http://mail.live.com/TodayLigth.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="121831">3gJeDI</cookie>
<cookie name="PHPSESSIONID">HDYK7DJ6K3</cookie>
</cookies>
</httpsession>
```

Figure 4-5. An example of an XML document that can be encapsulated in a SIP message body

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. While DHTML involves JavaScript and Cascading Stylesheet languages, Asynchronous JavaScript and XML (AJAX) and Web 2.0 are technologies similar to DHTML in purpose. In this implementation, a JavaScript code 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 as shown in Figure 4-5, XPATH was used to extract the necessary data.

4.3 Interfaces of Mozilla Firefox version 2.0

Mozilla Firefox version 2.0 has over 800 XPCOMs written in C++ and JavaScript programming languages. Its rendering engine is built on Gecko 1.8.1. Gecko is the name of the layout engine developed by the Mozilla Project to read web content, such as HTML, CSS, XUL and JavaScript, on a user's screen [54]. An XPCOM can also be defined as an object that implements an interface. The TransferHTTP extension added a new XPCOM with the contract id "@ngportal.com/SIPStack/SIPStackInit;1" into the existing XPCOMs in the browser. The extension core comprised of an XPCOM and an interface. The interface, named "ImyStack," was defined in an Interface Definition Language (IDL).

An IDL is a language-neutral specification language used to describe a software component's interface [55]. It enables software components written in different languages to communicate. Its binary type library in ".xpt" file extension and header file in ".h" file extension were generated with the aid of a command-line utility known as Cross Platform

Interface Definition Language (XPIDL) compiler [56]. Most of the built-in interfaces in Mozilla Firefox version 2.0 used in the implementation are highlighted below.

- nsIPref: This interface is meant for storing the web browser's settings, also known as preferences. In the implementation, it was used to store the SIP proxy server address, the UAC port number, the UAC SIP address and other relevant information.

- nsICookie/nsICookieManager/nsICookieManager2: These interfaces are used to manage cookies stored by the web browser. nsICookie has an enumerator to track a cookie, nsICookieManager is used to remove a cookie, and nsICookieManager2 is used to add a cookie. Cookies in Mozilla Firefox version 2.0 are stored in a text file.

- nsIThread: This interface was used to run the SIP stack in a separate thread from the main thread of the running web browser. During the implementation, it was found that the browser freezes for sometime whenever a user chooses a task, such as UAC registration. The reason is that all JavaScript code runs in the same thread with the browser's UI (User Interface). This problem was resolved by running all processes of the SIP stack in a separate thread from the UI thread.

- nsIPasswordManager: This interface is used to store passwords within the browser. In the implementation, it was used to store the UAC password. It uses a key-value mechanism slightly different from nsILoginManager in Mozilla Firefox version 3.0.

- nsIIOService: This interface is used to manage a data stream or channel in the browser. It was used to create new Universal Resource Identifier (URI) during a web session receipt.

- nsIPromptService: This interface is used to generate prompts, such as alerts, confirmations and dialogs, within the browser. In the implementation, after a notification is received and a user chooses the task "Accept Session," a confirmation pops up showing the URL of the web session (see Appendix A). On pressing OK button, the web resource is pulled from the Internet.

- nsITimer: This is a timer interface. It was used to periodically check from the browser if a message is received by the SIP stack.

- nsIObserver: This interface was used alongside the nsITimer. In the web browser, this interface is used to watch over an event and trigger an action when a change occurs. An example is when a script in a web page wants to add a cookie to the cookies list. This could lead to a confirmation in which a user is asked if he/she wants the cookie added or not. In the implementation, when the SIP stack receives a message, a notification is seen in the status bar of the web browser with the aid of the nsIObserver interface.

4.4 Conclusion

In summary, this chapter has presented the design of the TransferHTTP extension. The technologies and interfaces of Mozilla Firefox version 2.0 used in the implementation have been discussed. The next chapter provides the results and discusses the extension's evaluation, user experience, its possible deployment and commercialization.

Chapter 5

5 Results and Evaluation

This chapter presents the user interface, HTTP session mobility performance, the web browser's performance after the integration of a SIP stack and evaluation of the project with other existing web session migration approaches.

5.1 TransferHTTP User Interface

Figure 5-1 shows the installed extension in a web browser. A new menu "HTTP Mobility" is added to the menu bar. A sub-menu "Preferences" can be found under the "HTTP Mobility" menu when clicked; screenshots of the extension, when in use, are available in Appendix B. In the Preferences, a user sets, among other things, his UAC's SIP address, SIP proxy address, username and password. In addition, Figure 5-1 shows a typical notification in the status bar of the web browser. 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.

The TransferHTTP extension with its source code has been released to the public under the Mozilla Public License (MPL) and is available on the Mozilla repository for further contributions at http://transferhttp.mozdev.org.

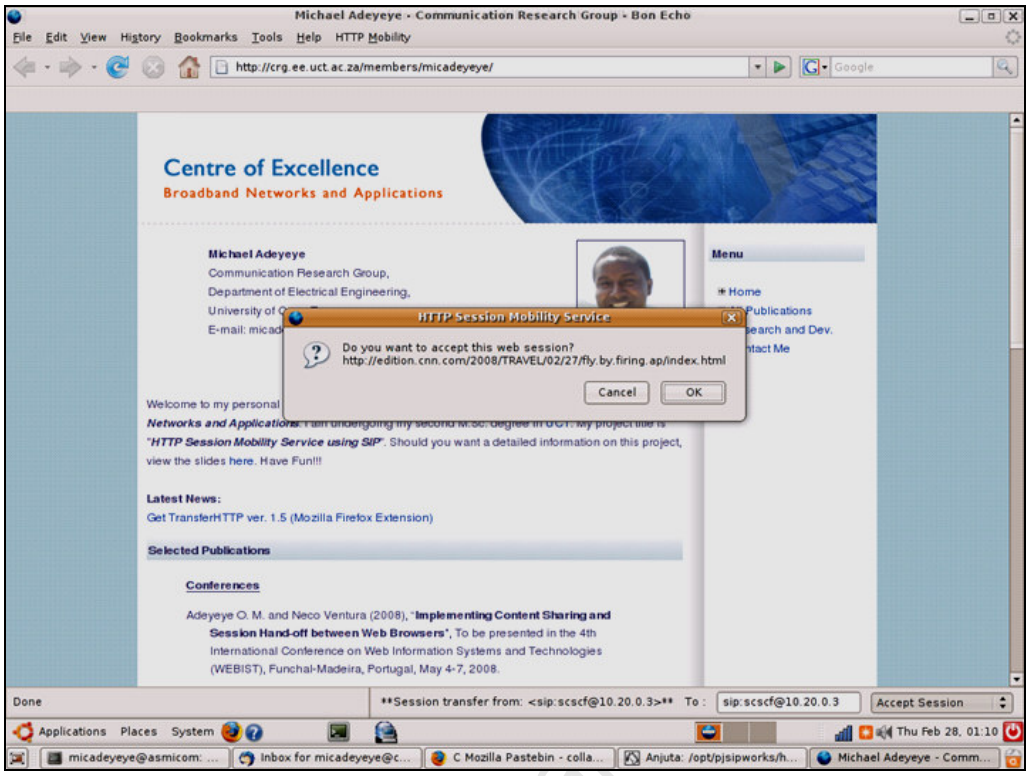


Figure 5-1. The TransferHTTP User Interface

5.2 HTTP Session Mobility Performance

Figure 5-2 shows that session data sizes vary among websites. TransferHTTP extension was tested on some of the prominent websites, namely Yahoo, Hotmail, Gmail, Facebook and Youtube. The intentions were to gather information on the size of data transferred during a session handoff and find out if session handoff could fail on some websites. As earlier explained, session handoff means the ability to move existing web session between two web browsers, and content sharing means the ability to view simultaneously the same web page on two web browsers. Content sharing requires transferring only a web page's Universal Resource Locator (URL). On the other hand, session handoff requires the transfer of a web page's URL and session

data, usually cookies, and if available, hidden input elements in the web page. The compositions of the session data can be URL, cookies and hidden input elements (Figure 4-5).

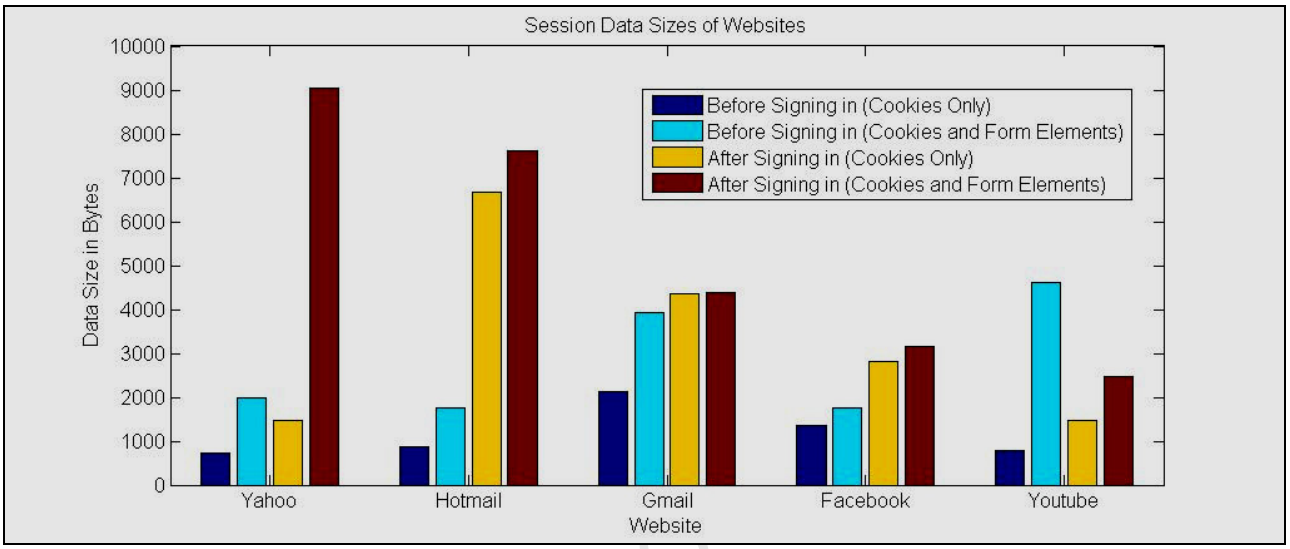


Figure 5-2. Session Data Size (Cookies and Hidden Input Elements) on prominent Websites

The packet size of the SIP MESSAGE body was measured when transferring each website's homepage before and after signing in. This packet size refers to the content-length of the SIP MESSAGE header, which is always zero when there is no data in the SIP MESSAGE body. The sampled websites already had some cookies in the cookies list of the web browser before signing in. Results obtained from analyzing the XML data showed that Yahoo and Hotmail had almost the same amount of cookies. After signing in, more cookies were generated in Gmail and Hotmail than in Yahoo. In Figure 5-2, Yahoo session data size skyrocketed because it contained many input elements, most notably button type, at the user's homepage. Hotmail had more cookies than Gmail and its user's homepage had very few input elements.

Facebook, before and after signing in, had more cookies than its hidden input elements. Session handoff was also successfully tested on websites that used SSL to encrypted submitted data. Examples were the department's private web mail service (at https://crgmail.ee.uct.ac.za) and an eCommerce website (at https://www.firstinlandonline.net). Each website had a session data size less than 500B.

Session handoff on some websites, however, was partially successful or not successful. Although session data were successfully transferred, the exact web page with it current interaction, such as a web chat session, could not be continued on some websites. An example was Meebo, which is based on AJAX. AJAX enables websites to 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. Examples of these Mash-ups were http://www.alertnet.org, http://www.3dgeomaps.com and http://www.rockstarapps.com.

This failure can also be attributed to their ability to update web page's contents without changing its URL, during a HTTP POST or GET request. Other websites that achieved partially successful session handoff were those made up of FRAME/IFRAME HTML tags. In such cases, a column or row of the FRAME-based/IFRAME-based website redirected a user to a log-in page. The reason is that only cookies of the parent FRAME/IFRAME were transferred when session handoff was tested on these websites. As a result, a prompt or log-in request appeared in a column or row of the FRAME-based /IFRAME-based website.

An example was this URL: http://www.ngportal.com/Certdumps/parent_sites.php. The web page had a frame pointing to http://www.actualtests.com/default.asp?show=exams. Both websites, NGPORTAL and ACTUALTESTS, require logging in, though the log-in processes

could be done at the parent website, NGPORTAL. During a session handoff, only cookies of NGPORTAL were transferred thereby resulting in a log-in request into ACTUALTESTS at the destination web browser.

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 [19]. RFC 2965 [25] provides information on how cookies should be selectively sent to a web server, and what modifications to cookies are not encouraged from end users. This implementation makes no changes to cookies. In addition, it also deletes cookies from the source UAC, during web session handoff in order to prevent two or more web browsers from having the same session data. The current web browser tab at the source web browser is also closed. This measure clears all hidden input elements and requires logging in again to access newly generated cookies and hidden input elements.

5.3 The Web Browser Performance

The integration of a SIP stack into the Mozilla Firefox web browser was achieved in a loosely-coupled approach. The SIP stack (written in C programming language) and an XPCOM (written in C++ programming language) were separately compiled, though the SIP stack was compiled into a shared library. The XPCOM formed an abstraction over the SIP stack so that 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 5-1 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.

In this first performance test, the browser's memory consumption, when the SIP stack was not integrated, was compared with the browser's memory consumption, when the SIP stack was running as a background service. The first column of results, in Table 5-1, 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.

Table 5-1. 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

Results gathered without the SIP stack running as a background service indicated the normal memory consumption of the browser. The memory consumption at start-up was 11.3MB and it progressively increased overtime. When the SIP stack was allowed to run as a background service, the memory consumption at start-up was 16.7MB; and the memory consumption also progressively increased. It was found out that running the SIP stack as a background service caused the browser's memory consumption to increase marginally 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 or freeze when running the SIP stack as a background service over a long period.

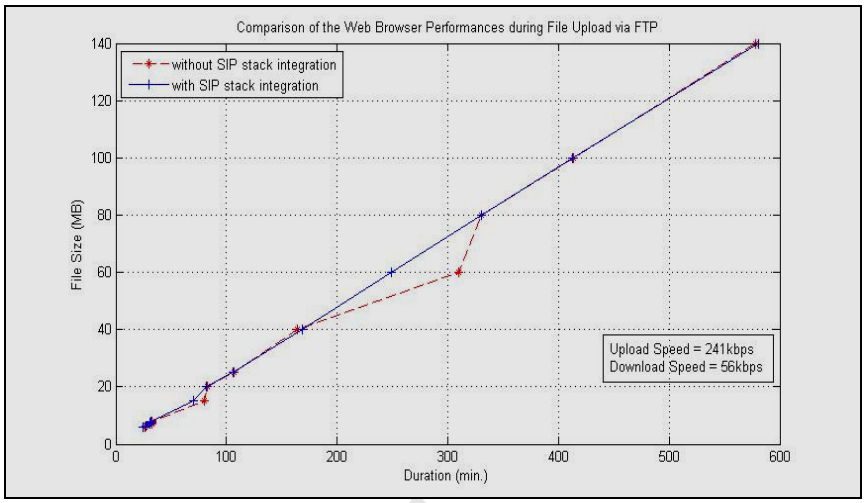


Figure 5-3. Comparison of the Web Browser Performances during File Upload via FTP

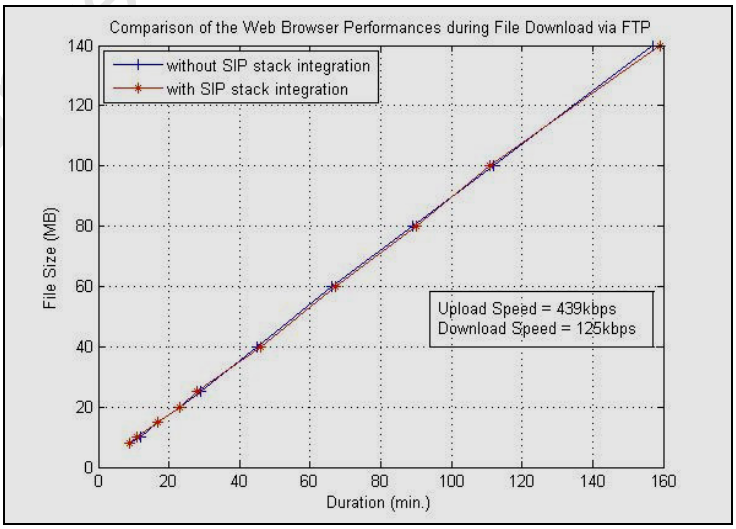


Figure 5-4. Comparison of the Web Browser Performances during File Download via FTP

The second test compared the web browser's performances. The test required using the web browser as an FTP client to upload and download files, since most web browser can act as an FTP client. Using the web browser as an FTP client is synonymous to browsing a web page; that is, it entails pulling/pushing web resources. The intention was to know if the download and upload speeds of the web browser changed when pulling or pushing web resources at the time the SIP stack was running. Figures 5-3 and 5-4 show 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 upload and download files (zipped images) of varying sizes to a web server. The average download and upload speeds of the Internet connection during the Upload Test were 56kbps and 241kbps, respectively. During the Download Test, the average download and upload speeds of the Internet Connection were 125kbps and 439kbps, respectively. The Download Test was conducted long after the Upload Test, hence the dissimilar upload and download speeds during the tests. The graphs show 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. Hence, the performance of the web browser was not impeded when the SIP stack was integrated.

5.4 Evaluation of TransferHTTP Extension

Table 5-2 shows the comparison of TransferHTTP with other existing web session migration approaches. Although Canfora et al. [15] and Hsieh et al. [16], which are proxy-based

architectural scheme, are mentioned in the related academic work, only Hsieh et al. is used in this evaluation. Hsieh et al. is chosen because it has more functionalities, such as support for optional client program, than Canfora et al.

The works compared with TransferHTTP are Browser State Preservation and Migration (BSPM) [29] and Stateful Session Handoff for mobile WWW [16]. BSPM is a client-based architectural scheme while mobile WWW (that is, Stateful Session Handoff for mobile WWW) is a proxy-based architectural scheme. This project is based on a hybrid-based architectural scheme.

Both BSMP and mobile WWW provide web session handoff between two web browsers. This research work (TransferHTTP) also provides web session handoff. Session handoff in the three works requires that all cookies are sent to the web browser, though these cookies are sent through different mechanisms, as explained in the related academic work (section 2.3). In addition to functionalities in TransferHTTP, it also offers content sharing, which entails one web browser referring another web browser to have access to the same web resource.

These works however have more differences than similarities. Regarding modifications made to web browsers architecture, BSPM is a client-based architectural scheme that requires modifying the architecture of a web browser. TransferHTTP, which is a hybrid-based architectural scheme, also requires modifying the architecture of a web browser. On the contrary, mobile WWW is a proxy-based architectural scheme that does not require modifying the architecture of a web browser. It however supports an optional client program that modifies the architecture of the web browser when the client program is installed.

Table 5-2. Comparison of TRANSFERHTTP with other existing web session migration approaches

Item	Approaches		
	BSPM	**Mobile WWW**	**TransferHTTP**
Modification to user devices	Substantially modified browser	No but an optional small client program can be installed.	A small client program is required
Session registration/tracking	Supports user authentication	By accessing UAP's web pages or by the client program	Unnecessary but supports client authentication
Tracking session information	Unnecessary	By UAP; Unable to track unsubmitted form fields when client program is not installed.	Unnecessary
Handoff action	Source saves to BSPM repository and then Target retrieves from it.	Target retrieves from UAP; UAP retrieves unsubmitted form fields from source via small client program.	Source sends session data, unsubmitted form fields inclusive, to Target via an optional SIP proxy (or UAP).
History and Cookie handoff	History is directly loaded into browser. All cookies are sent to browser.	History is sent as a list of web pages. UAP sends necessary cookies to browser during browsing and tracking step.	History is unnecessary. All cookies are sent to browser.
Web pages handoff	Content is directly loaded into browser.	UAP redirects browser to open URLs. With small client program, UAP sends URLs to client program and then browser opens the URLs.	Browser interacts with client program and opens URL.
Summary	A client-based scheme. Needs a repository that stores session data. A client is modified. No user-client interaction. Introduced session handoff.	A proxy-based scheme. A proxy is required to perform the session transfer process. An optional client program can be used to support user-client interaction. Introduced session handoff.	A hybrid-based scheme. Introduced SIP into the client and can use an optional SIP proxy. Supports user-client interaction. Introduced voice call, session handoff and content sharing.

BSPM lacks user-client interaction while mobile WWW provides user-client interaction when its optional client program is installed. TransferHTTP, like mobile WWW, also provides user-client interaction. User-client interaction offers a user the ability to continue a task, such as filling a form, rather than starting afresh at another end after a session handoff.

In terms of session registration/tracking, BSMP offers only user authentication, which is used to register a session. Mobile WWW however offers session tracking in addition to user authentication. Although TransferHTTP supports user or client authentication, it does not offer session tracking. History handoff is a feature found in BSPM and mobile WWW but not in TransferHTTP. TransferHTTP assumes a degree of privacy in its service. Since content sharing involves someone referring another person to the same web resource, it is unnecessary (and an invasion of privacy) to include the history of the source web browser. Session tracking is not provided in TransferHTTP, since a proxy is required to achieve it. It is however stated as one of the future work in section 7.2.3.

TransferHTTP could work in a P2P environment without a SIP proxy server. In addition, it introduces a new Protocol, SIP, into a web browser that makes it possible to set up voice call with the destination web browser. This offers a new way of collaboration in the web-browsing context. Lastly, unlike other approaches, it does not break the HTTP security rules. It was ensured during the implementation that no two web browsers would have the same session data, most notably cookies and hidden input elements.

5.5 Conclusion

In summary, this chapter has presented the results and discussed some findings during the implementation. It showed that session handoff was not successful on all websites. The SIP stack integration, however, was successful and did not impede the browser's performance. This chapter also emphasized the contributions of this project when compared with other existing web session migration approaches.

The next chapter, discussions, presents the possible deployment and commercialization of this service. In addition, it discusses how access to multimodal and multi-channel applications could be improved and possible modification to how click-to-dial works.

Chapter 6

6 Discussions

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. The SIP Servlet 1.0 specification was standardized through Java Specification Request (JSR) 116 [57]. The integration of a SIP stack into a web browser can improve how click-to-dial works and how multimodal and multi-channel applications are accessed. This chapter highlights the improvement that could be made and discusses the possible deployment and commercialization of this project.

6.1 Improving access to Multi-channel and Multimodal Applications

A multi-channel application presents its content to the end user based on his/her connecting device or user agent. In multi-channel access, enterprise data and applications are accessible from multiple channels. Unlike multi-channel access, multimodal access is the ability to combine multiple channels in the same interaction or session. Both multi-channel and multimodal applications require a VoiceXML gateway to interpret VXML pages and access the voice part of the applications.

The building blocks of a VoiceXML gateway include Text-to-Speech (TTS), Automatic Speech Recognition (ASR) and VXML Browser [58]. A multimodal browser, such as Opera multimodal browser [59], is always required to access a multimodal application. With more

VoiceXML gateways integrating SIP, a SIP integrated web browser could use its embedded SIP functionality to provide the required voice interaction with a VoiceXML gateway. In this case, there would be no need to install a multimodal browser. Hence, the SIP integrated web browser could be used to access a multi-channel/multimodal application. Both directed-dialog, which requires Dual-Tone Multiple Frequency (DTMF), and Interactive Voice Response (IVR) interactions could be achieved with a SIP integrated web browser, and examples of VoiceXML gateways that have already supported SIP are VOXEO [60] and OmniVox3D [61].

6.2 Modifying How Click-to-dial Works

Integrating a SIP stack into a web browser could modify or extend how click-to-dial applications work [62]. When a "mailto:" command in a hyperlink is clicked, it launches a mail client in order to compose a mail to the email address. Having standardized the command "tel:" in a hyperlink [63], when such a hyperlink is clicked, it could initialize the built-in SIP stack in a web browser and set up a call between the SIP client or phone and the web browser. In click-to-dial, an application server is responsible for establishing a call session between two phones.

With a SIP stack integrated into a web browser, a call session could be established at the client end. In addition, integrating a SIP stack into a web browser can make it possible to continue a call session from a mobile phone on a PC when the caller or callee moves to an environment that has poor signal strength but a very fast Internet connection. Although a separately installed SIP client could offer this service, a SIP integrated web browser would be better, most notably when the person wants to browse the Internet at the same time. In addition, a

SIP integrated web browser encourages adaptive UAC, whereby the web browser could be used as a SIP client.

6.3 Deployment and Commercialization

This HTTP mobility service is an application layer solution that could migrate web sessions of websites, SSL-based websites inclusive. 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 advised 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.

In Figure 5-2, the data sizes in bytes represent the number of characters and could be used as the yardstick for charging users. These data sizes account for the cookies and the hidden input elements. It was observed that the session data sizes vary among web pages in the same website. An example was Youtube. Although not indicated in Figure 5-2, its session data size after signing in at the homepage was 2kB, and at a randomly selected page, it was 15kB.

An additional feature available in this service is the ability to maintain the same state of a web form during content sharing or session handoff – user-client interaction. This feature enables

a user to move a web form that he/she is filling between web browsers without starting afresh. A function, written in JavaScript Language, steps through the Document Object Model (DOM) of a web page, retrieves the names and values of its FORM elements and sends the data 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 notably during an unsuccessful web form submission.

Session data size also depends on the number of HTML Form elements in a webpage, in addition to URL, cookies and hidden HTML Form elements. The HTML Form elements here refer to Input, Checkbox, Radio, Textarea, to mention a few, and are responsible for the user-client interaction. A charging function could be integrated into the SIP proxy server and a web interface could be provided for users to monitor their usage. Regarding possible service commercialization, a flat rate could be charged periodically regardless of the varying session data sizes. In addition, users could purchase data bundles, such as 500MB and 1GB, which could be monitored via a web interface as they use the service. On the other hand, it could be rendered as a Value Added Service (VAS) to customers.

In this implementation, the SIP stack packet size was extended to 16kB in order to send session data at once. When deploying this service, the session data could be sent in chunks and merged at the destination UAC in order to propagate most SIP servers. In addition, SIP stack that supports S/MIME should be used when developing the web browser extension; besides, S/MIME is preferred to TLS. The reason is that when using 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 [48]. Hence, session data in transit could be hijacked. However, TLS-supported SIP proxies could be used when the project will be deployed within a small environment where the proxies are ensured to support TLS.

Chapter 7

7 Conclusions and Recommendations

This concluding chapter presents the summary of the chapters and points at further work that could be done to improve this project.

7.1 Summary of Chapters

While Chapter 1 provides introduction to this thesis, Chapter 2 gives the background information and related work carried out in the academic environment and the Industry. In Chapter 3, a hybrid-based architectural scheme is proposed. In addition, mapping the SIP Mobility Types to content sharing and session handoff is explained. The chapter ends with the presentation of a new architecture that supports SIP stack for web browsers. The service, designed using Data Flow Diagram, and its implementation are explained in Chapter 4.

Chapter 5 presents the results and evaluation. The evaluation of the project involves comparing it with other related work. Results showed that the performance of the web browser was not hindered when a SIP stack was integrated. Although the memory consumption increased marginally, the web browser did not crash. In addition, the upload and download rates of the web browser remained unchanged when the SIP stack was integrated. Regarding the HTTP mobility test, session handoff was successful on a large number of websites, SSL-based websites inclusive. It however failed or was partially successful on some websites using Web 2.0 technologies or IFRAME/FRAME HTML tags.

Chapter 6 discusses new ways of improving access to multimodal and multi-channel applications. In addition, it shows possible changes to how click-to-dial works. How session data could be secured during the deployment of this project and how the project could be commercialized are also discussed in Chapter 6.

7.1.1 Specific Contributions

First, a new service, referred to as content sharing and session handoff between web browsers, has been provided. This implementation has provided a fast and efficient way of referring someone else to the same web page currently viewed by the referrer rather than the slow way of copying, pasting and sending the URL in a chat session or an email. Second, integrating a SIP stack into a web browser offers the advantage of extending a web browser to act as a SIP client. In this case, web browsers can now act as SIP clients thereby setting up multimedia session between two or more users. Most notably, web browsers now have unique SIP addresses to interact with one another like PCs, which have unique Media Access Control (MAC) or IP addresses.

In summary, while session handoff has been widely explored, content sharing is a new service in the web-browsing context that could encourage collaboration or community interaction between the Internet users. In addition, having shown that the integration of a SIP stack into a web browser makes no significant change on the memory footprint or quality of experience, the inclusion of SIP in commercial web browsers is not only feasible, but also will offer new services to end users. SIP is an extensible protocol that is not only used in multimedia services

provisioning, but also in control and automation, such as smart homes. Should Free Open Source Software (FOSS) and Open Standards be widely adopted, more new and innovative solutions, like this project, would be introduced into the web browsing experience and found in this Web 2.0 era as services are rapidly converging.

7.2 Future Work

Four areas that could be looked into to improve this project are discussed below.

7.2.1 Improving HTTP Session Mobility Performance

There was a problem encountered during the HTTP session mobility performance in section 5.2. The exact web browser's state could not be reproduced when session handoff was carried out on Mash-ups, AJAX-based and FRAME/IFRAME-based websites. A probable solution to this problem is to capture the web browser's running state at the source UAC and reproduce at the destination UAC. This solution is highly technical but could be achieved by the developers of web browsers.

7.2.2 Blocking Unwanted Content Sharing Request

An address book of contacts could be integrated into the web browser extension - TransferHTTP. The integration of an address book would make it easier to retrieve the destination SIP address and could make content sharing request sent to two or more people at a

time. Furthermore, a policy control mechanism could be integrated into the SIP proxy (sections 3.2 and 3.2) to block unwanted web content sharing request. Such restrictions could be based on a domain name or a SIP address.

7.2.3 Session Passivation

While a session-based cookie expires in a short time of inactivity, a persistent cookie could provide access to a website over a long period. A session management mechanism could be integrated so that a session handoff request could be held for a long time without expiring when the destination SIP address cannot be reached or a web server uses a session-based cookie. The session management mechanism could also include a session tracking feature, since there might be more than one pending session handoff request.

7.2.4 Further Enhancement to the web browser extension

Although SIP MESSAGE method is used in this implementation to transfer the web session data, the implementation can still support instant messaging (IM) between two UACs. When the UI of the extension is extended to support IM, only messages that are not wrapped in the XML format shown in Figure 4-5 should appear in the chat box. In addition, the web browser extension could be extended to support full multimedia services including video conferencing, IP Television (IPTV), and Video on-Demand (VoD). In general, it could be extended to support features available in IMS clients. In addition, new plug-ins could be integrated into web browsers in order to support file formats that are supported in the IMS. For example, the video media type

"video/3gpp" in the IMS has a file extension ".3gp" for video files. Lastly, this project could be extended to work with other IM services, such as GTALK, AIM and Yahoo Messenger. As stated in section 7.2.2 that an address book could be integrated into this project, the address book could be linked to users' address books from those IM services, and a suitable gateway, such as SIP-XMPP gateway, could be provided so that this web browser extension (TransferHTTP) could work with those IM services.

References

[1] Berners-Lee T., Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor, Harper San Francisco, 1999.

[2] Alan Grosskurth and Michael W. Godfrey, "A Reference architecture for web browsers," Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM '05), pp. 661-664.

[3] J. Postel and J. Reynolds, "File Transfer Protocol," IETF RFC 959, October 1985

[4] J. Oikarinen and D. Reed, "Internet Relay Chat," IETF RFC 1459, May 1993

[5] The Pidgin Chat Engine,http://www.pidgin.im, July 13, 2008

[6] The Open ID Project, http://openid.net/what/, July 14, 2008.

[7] Sujeet 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.

[8] M. Handley, H. Schulzrinne, E. Schooler and J. Rosenberg, "SIP: Session Initiation Protocol," IETF RFC 2543, March 1999.

[9] 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.

[10] The Mozilla Firefox web browser, http://www.mozilla.org, July 12, 2008.

[11] The PJSIP Project, http://www.pjsip.org, July 12, 2008.

[12] Zamudio, R., Catarino, D., Taufer, M., Steam, B., and Karan, B. 2007. "Topaz: Extending Firefox to Accommodate the GRidFTP Protocol," Parallel and Distributed Processing Symposium, IPDPS 2007, pp. 1-8.

[13] Google Browser Sync, http://www.google.com/tools/firefox/browsersync/, June 10, 2008.

[14] A prototype of Mozilla Weave, http://labs.mozilla.com/2007/12/introducing-weave/, June 10, 2008.

[15] G. Canfora, G. Di Santo, G. Venturi, E. Zimeo and M.V. Zito, "Proxy-based Handoff of

Web Sessions for User Mobility," Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous '05), 2005.

[16] 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.

[17] The FireFTP Extension, https://addons.mozilla.org/en-US/firefox/addon/684, June 16, 2008.

[18] The Chatzilla Extension, https://addons.mozilla.org/en-US/firefox/addon/16, June 16, 2008.

[19] 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.

[20] Antonio Liotta and Ling Lin, "The Operator's Response to P2P Service Demand," IEEE Communications Magazine, July 2007.

[21] Christopher J. Pavlovski, "Service Delivery Platforms in Practice," IEEE Communications Magazine, March 2007.

[22] H. Schulzrinne and E. Wedlund, "Application-Layer mobility using SIP," ACM SIGMOBILE Mobile Computing and Communications Review, Volume 4 Issue 3, July 2000.

[23] H. Schulzrinne, "Personal mobility for multimedia services in the Internet," in European Workshop on Interactive Distributed Multimedia Systems and Services (IDMS), Berlin, Germany, March 1996.

[24] Sohel Q. Khan, Robert Gaglianello and Michael Luna, "Experiences with Blending HTTP, RTSP and IMS," IEEE Communication Magazine, March 2007.

[25] D. Kristol and L. Montulli, "HTTP State Management Mechanism," IETF RFC 2965, October 2000.

[26] Hal Berghel, “Hijacking the Web,” Communications of the ACM, Vol. 45 No. 4, April 2002.

[27] 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.

[28] 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.

[29] H. Song “Browser Session Preservation and Migration,” In Poster Session of WWW 2002, Hawai, USA, 7-11 May, 2002, pp. 2.

[30] R. Jan, et al., “Enhancing survivability of mobile Internet access using mobile IP with location registers,” IEEE INFOCOMM ’99, Vol. 1, pp 3-11.

[31] A. C. Snoeren, and H. Balakrishnan, “An End-to-End Approach to Host Mobility,” Proceedings of ACM MOBICOM 2000, Boston, USA, August 2000, pp. 155-166.

[32] M. Atiquzzaman, S. Fu, and W. Ivancic, “TraSH-SN: A Transport Layer Seamless Handoff Scheme for Space Networks,” Proceedings of Fourth Earth Science Technology Conference (ESTC), Crowne Plaza Cabana Palo Alto CA, June 2004.

[33] R. Stewart, and C. Metz, “SCTP: New Transport Protocol for TCP/IP,” IEEE Internet Computing, Vol. 5, No. 6, November/December 2001, pp. 64-69.

[34] S. Fu, and M. Attiquzzaman, “SCTP: State of the art in Research, Products, and Technical Challenges,” IEEE Communications Magazine, Vol. 42, No. 4, April 2004, pp. 64-76.

[35] A. Di Stefano, C. Santoro, “NetChaser: agent support for personal mobility,” IEEE Internet Computing, Vol. 4, Issue 2, March-April 2000, pp. 74-79.

[36] M. Roussopoulos, et al., “Person-level routing in the mobile people architecture,” Proceedings of USENIX Symposium on Internet Technologies and Systems,

1999, pp. 165-176.

[37] Appenzeller et al., "The mobile people architecture," Technical Report CSL-TR-99-777, Stanford University, 1999.

[37] R. Liscano et al., "Integrating multi-modal messages across heterogenous Networks," Proceedings of ENM-97, In conjunction with the ICC-97, 1997, pp. 45-53.

[39] C. H. Herman et al., "iMobile: a proxy-based platform for mobile services," Proceedings of the First Worshop on Wireless Mobile Internet, 2001, pp. 3-10.

[40] K. Raatikainen, "Middleware for future mobile networks," Proceedings of IEEE International Conference on 3G Wireless and Beyond, 2001, pp. 722-727.

[41] R. Bagrodia, S. Bhattacharyya, F. Cheng, S. Gerding, G. Glazer, R. Guy, Z. Ji, J. Lin, T. Phan, E. Skow, M. Varshney, and G. Zorpas, "iMASH: Interactive Mobile Application Session Handoff," Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services, San Francisco, May 2003.

[42] Muffin: "World Wide Web Filtering System," http://muffin.doit.org, May 10, 2007.

[43] Composite capability/preference profiles: requirements and architecture "CC/PP 2.0," http://www.w3.org/Mobile/CCPP/, May 10, 2008.

[44] The Akogrimo Project, http://www.mobilegrids.org, June 19, 2008.

[45] 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.

[46] W. Munkongpitakkun, S. Kamolphiwong and S. Sae-Wong, "Enhanced Web Session Mobility based on SIP," Proceedings of the 4th International Conference on Mobile Technology, Applications and Systems (Mobility 2007), Singapore, September 10-12, 2007, pp. 346-350.

[47] P. Saint-Andre, Ed., "Extensible Messaging and Presence Protocol (XMPP): Core," IETF RFC 3920, October 2004.

[48] Gonzalo Camarillo and Miguel Garcia-Martin, The 3G IP Multimedia Subsystem, Wiley Press, England, Second Edition, 2006, pp. 155-162, 345-349.

[49] Alan Grosskurth and Michael W. Godfrey, “A Reference architecture for web browsers,” Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM ‘05), pp. 661-664.

[50] The Firebug Extension, https://addons.mozilla.org/en-US/firefox/addon/1843, June 16, 2008.

[51] Yourdon, E., Modern Structure Analysis. Prentice Hall, 1989.

[52] M. Handley and V. Jacobson, “SDP: Session Description Protocol,” IETF RFC 2327, April 1998.

[53] 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.

[54] Mozilla Gecko, http://developer.mozilla.org/en/docs/Gecko, June 16, 2008.

[55] OMG IDL, http://www.omg.org/gettingstarted/omg_idl.htm, June 16, 2008.

[56] Mozilla XPIDL, http://developer.mozilla.org/en/docs/xpidl, June 16, 2008.

[57] Erik Burckart, “Session Initiation Protocol in WebSphere Application Server V6.1,” http://www.ibm.com/developerworks/websphere/techjournal/0606_burckart/0606_burckart.html, May 16, 2008.

[58] J. Larson, "VoiceXML and the W3C speech interface framework," IEEE Multimedia Magazine, Volume 10, Issue 4, Oct-Dec 2003, Page(s):91 - 93.

[59] The Opera Multimodal Browser, http://www.opera.com/products/devices/multimodal/, September 1, 2008.

[60] The Voxeo Prophecy Platform, http://www.voxeo.com/prophecy, September 1, 2008.

[61] The OmniVox3D Application Server, http://www.apexvoice.com/index.php/158/sip-application-server, September 1, 2008.

[62] 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.

[63] H. Schulzrinne, “The tel URI for Telephone Numbers,” IETF RFC 3966, December 2004.

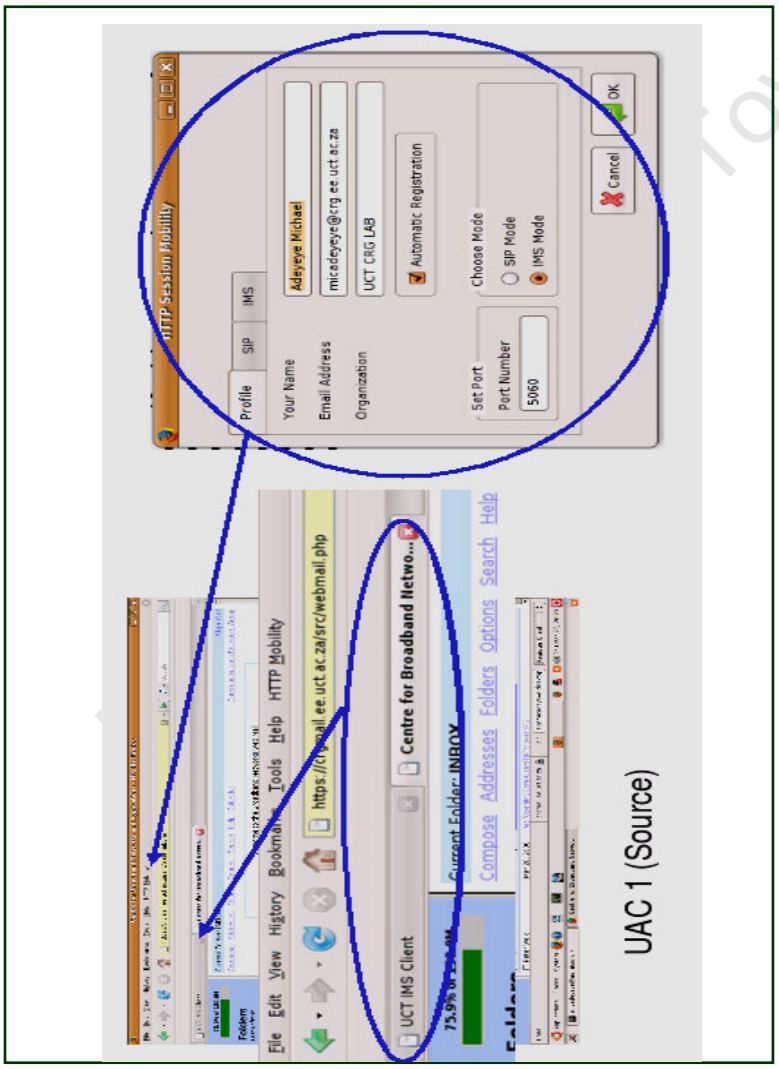
[64] Jennifer Golbeck and Michael Wasser, “SocialBrowsing: Integrating Social Networks into Web Browsing,” CHI 2007 Works-In-Progress. San Jose, California. April, 2007.

Appendix A: Screenshots of HTTP Session Mobility Service

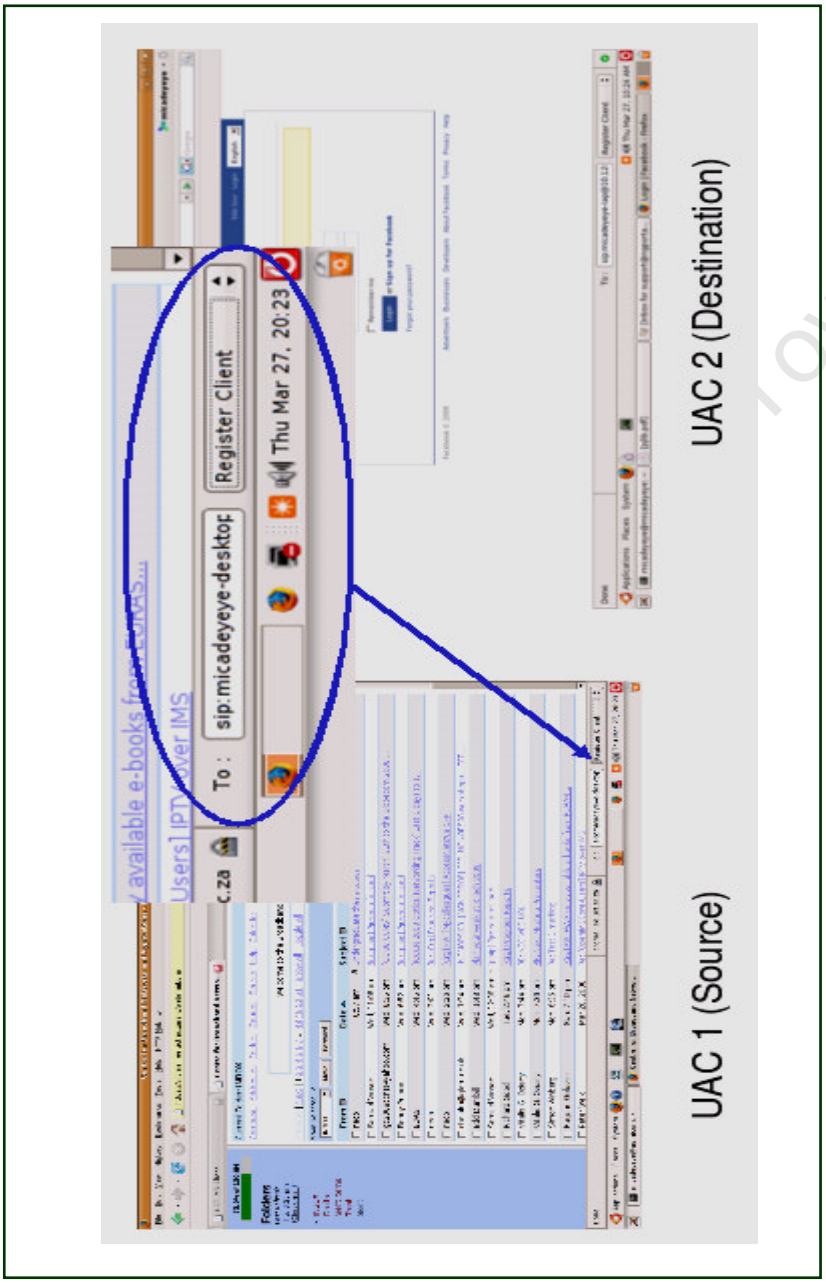
A.1: Session Handoff between two web browsers

Task: Handing over of a web mail session from UAC 1 to UAC 2

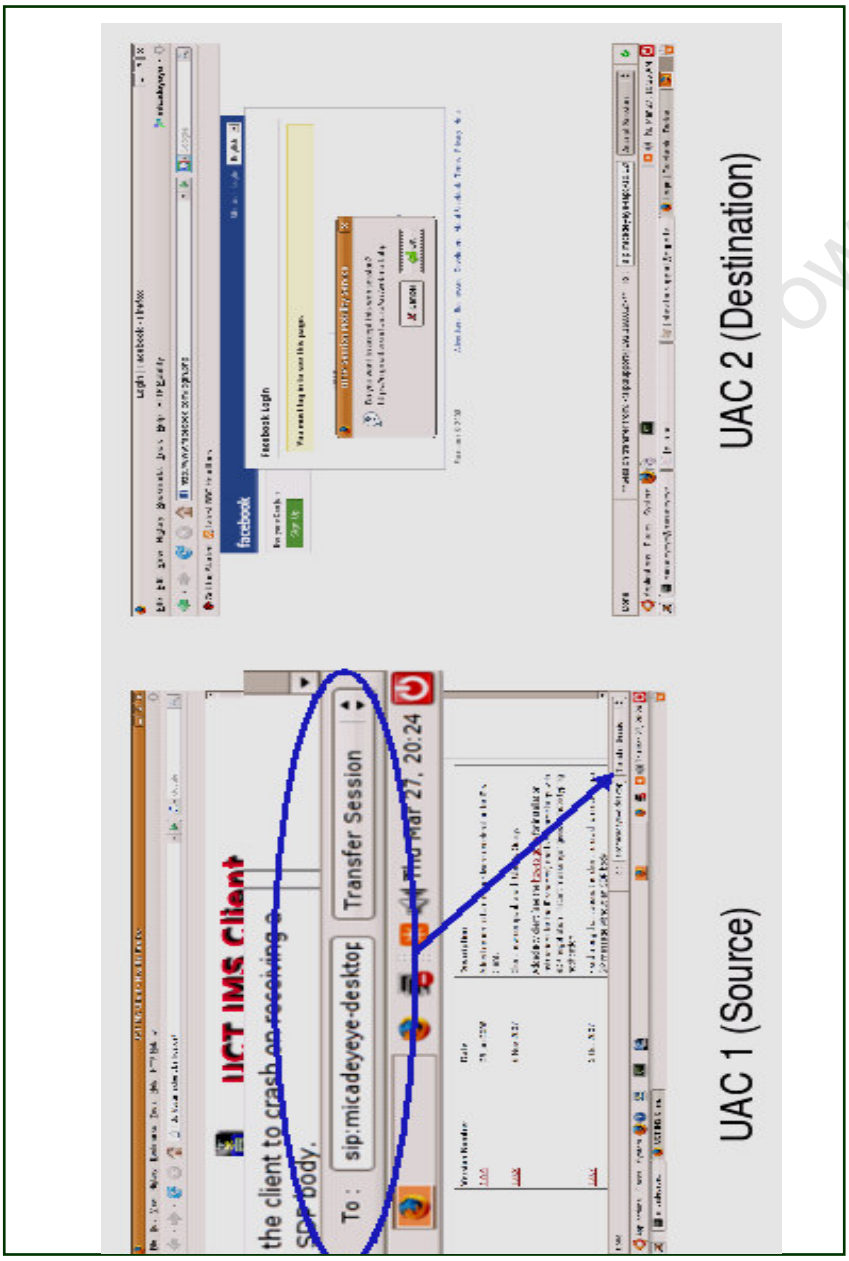
Process 1 of 5: Here, the Preferences of UAC is set. Configuration data include port number, SIP address, SIP proxy address and many more.



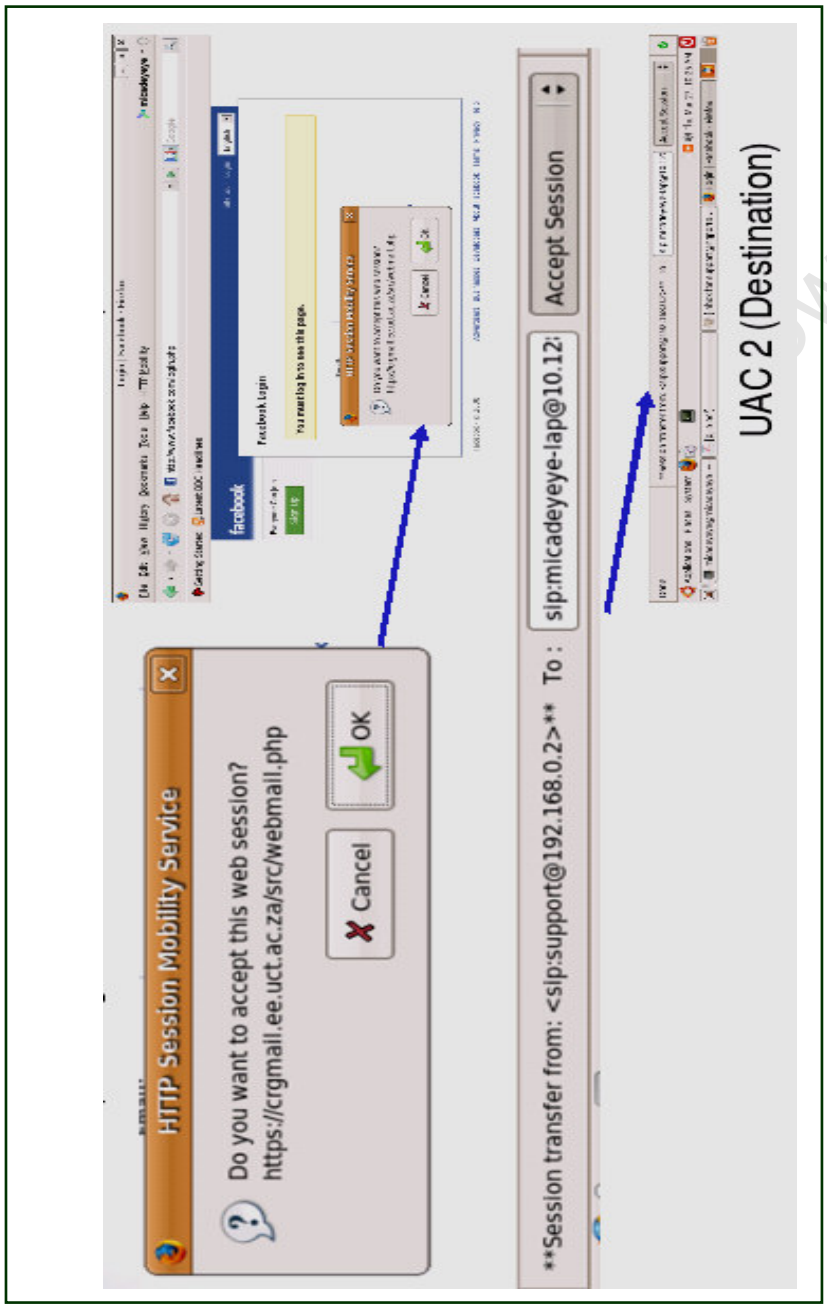
Process 2 of 5: The diagram below shows both UACs 1 and 2 registering to a SIP proxy server.



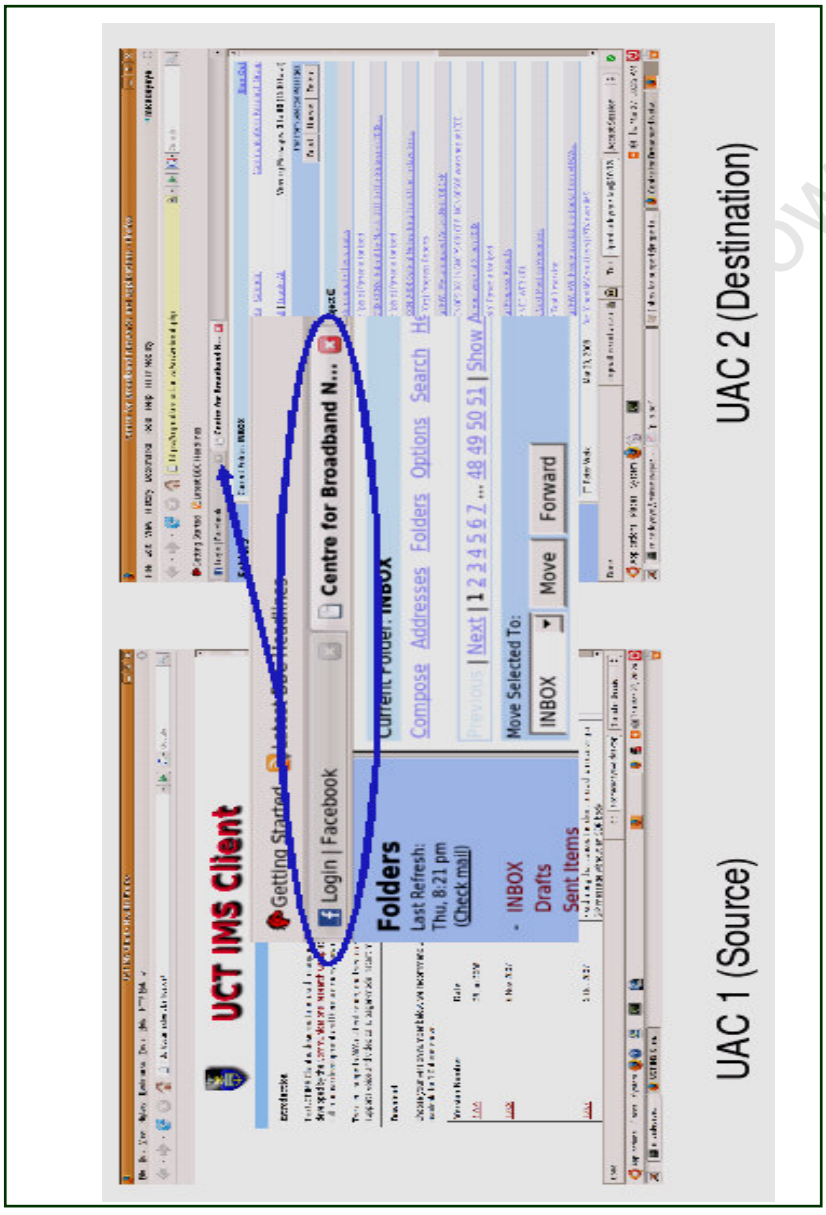
Process 3 of 5: Session data are encrypted and transferred from UAC 1 to UAC 2 at sip:micadeyeye-desktop@137.158.125.246. In addition, UAC 1 loses stateful connection to the web server and its tab is closed.



Process 4 of 5: A notification appears in the status bar of UAC 2. The user chooses "Accept Session," and a confirmation with the details of the web session pops up.



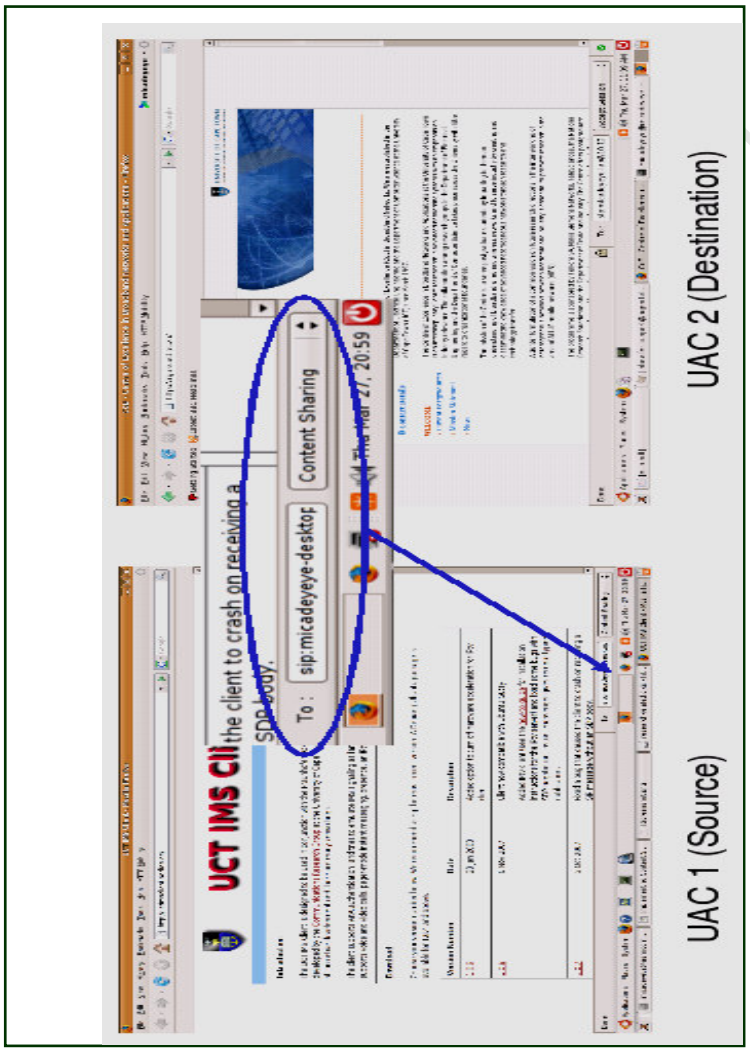
Process 5 of 5: On clicking the OK button at UAC 2, a new HTTP channel is opened, HTTP header is set and the resource is pulled from the web. In this case, the email account is automatically opened without logging in. The reason is that all session data are moved from UAC 1 to UAC 2.



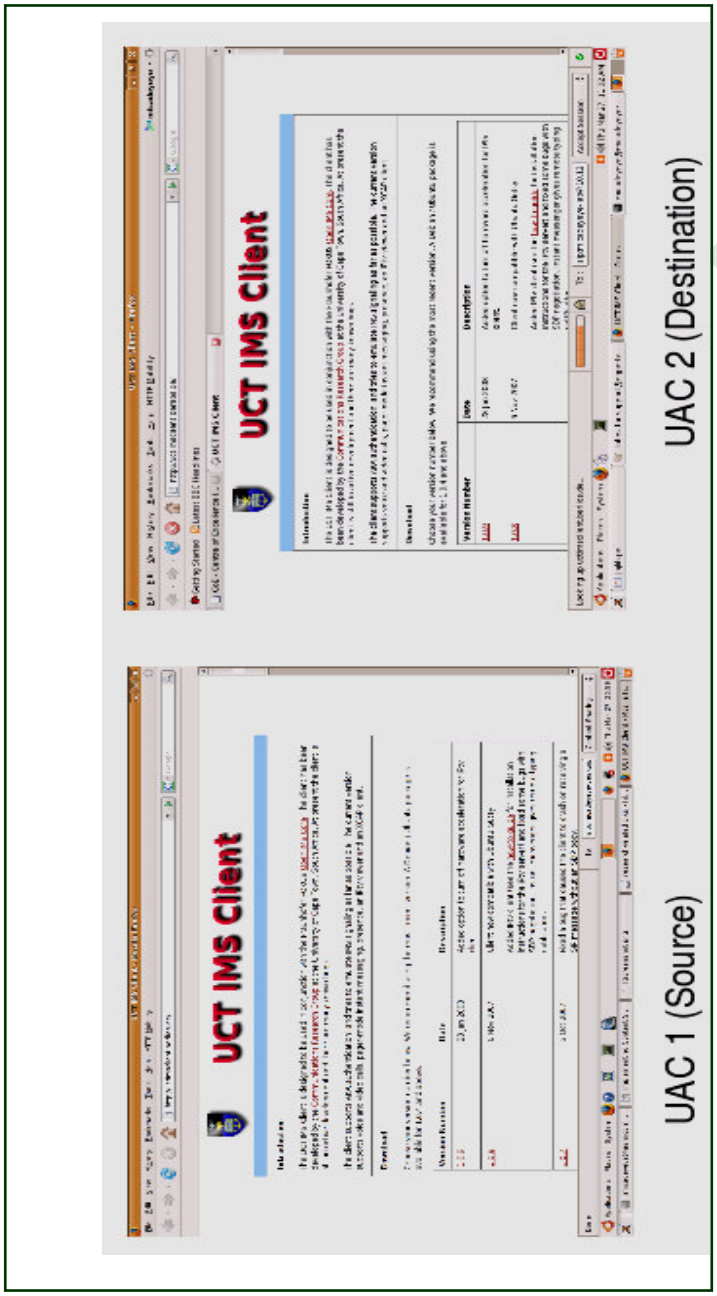
A.2: Content Sharing

Task: Sharing of a web session between two web browsers

Process 1 of 2: It is assumed that the UACs have successfully registered to a SIP proxy server. In the diagram below, content sharing process is chosen in UAC 1 and the session data, URL only, is sent to UAC 2 at sip:micadeyeye-desktop@137.158.125.246.

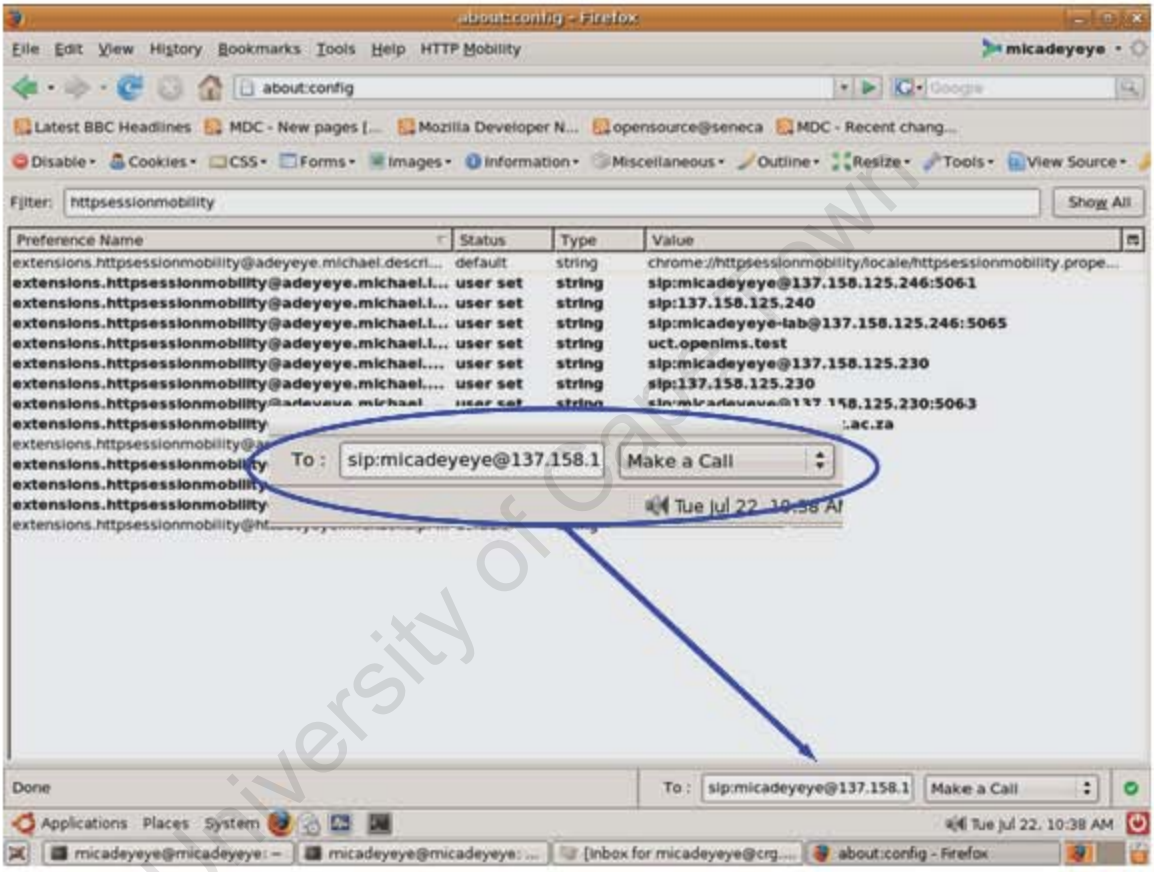


Process 2 of 2: A notification appears in the status bar of UAC 2; on choosing "Accept Session," a confirmation with the details of the web session pops up. When the OK button is clicked, a new HTTP channel is opened, HTTP header is set and the resource is pulled from the web. Here, only the URL is transferred between UACs.



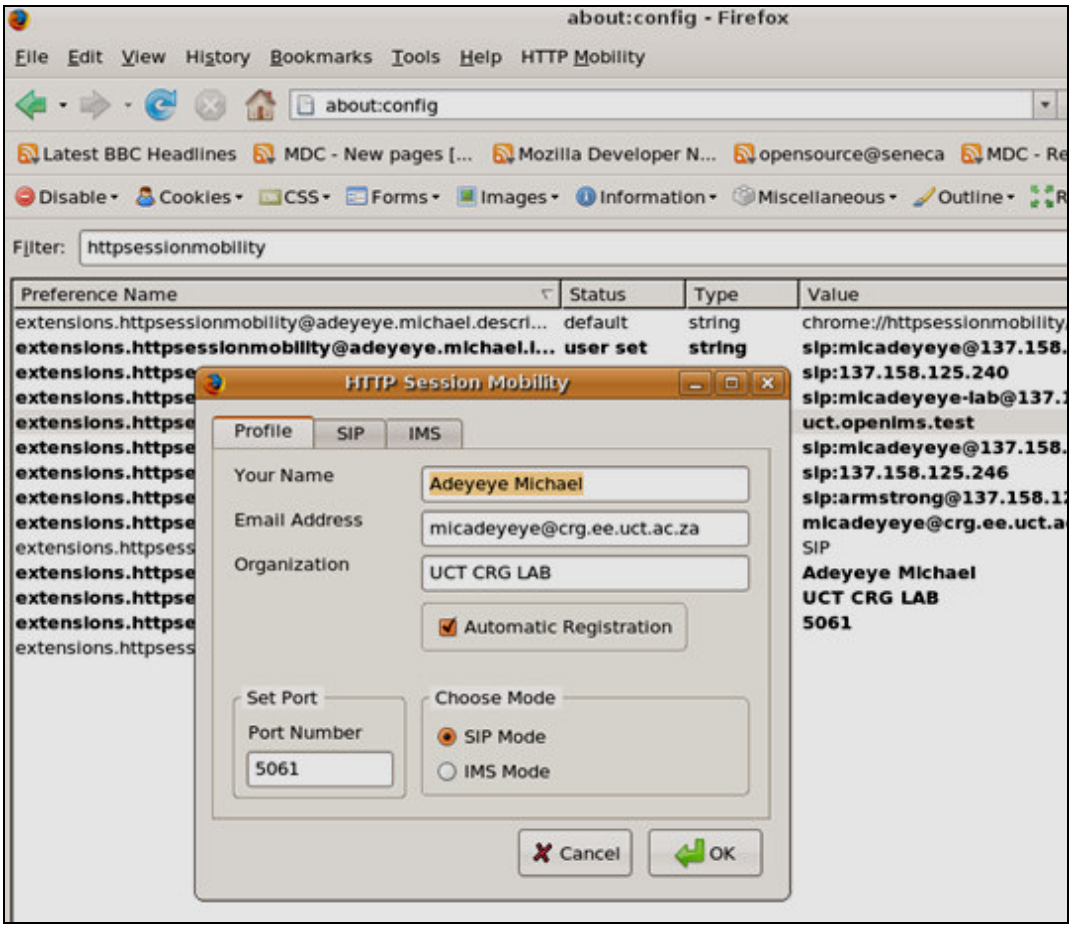
A.3: Multimedia Services

The diagram below shows how the extension can be used to make a voice call to another UAC at sip: micadeyeye@137.158.125.246:5063.



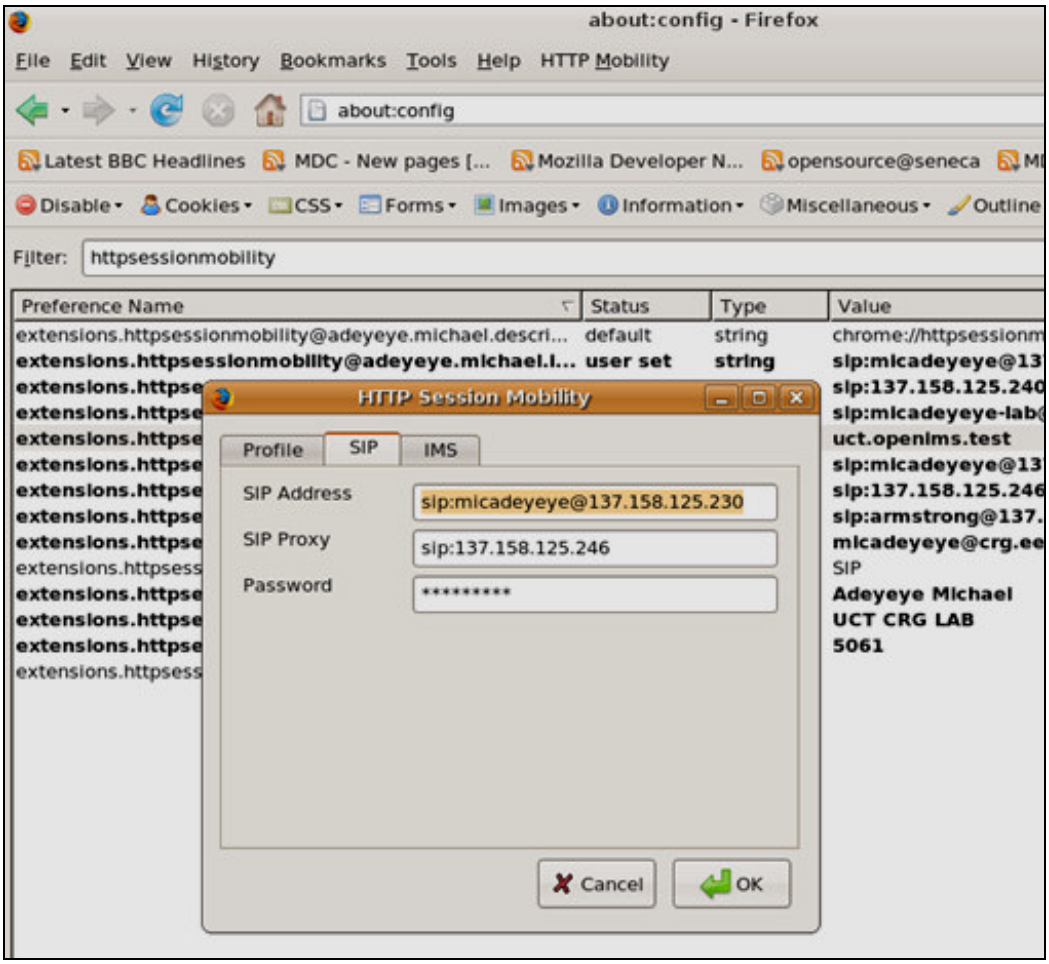
A.4: TransferHTTP Preferences

The diagram below shows the Preferences interface for setting the user information. Other settings here include the port number and the mode of operation.



TransferHTTP Preferences – cont'd

The diagram below shows the Preferences interface for setting the UAC in a SIP mode. The SIP address refers to the address of the UAC, and the SIP proxy refers to the address of the SIP proxy server. The password is required during registration to the proxy.



TransferHTTP Preferences – cont'd

The Preferences interface below shows the IMS fields required to connect to the IMS. At present, this project has not been fully extended to work in the IMS.