MiniSIP as a Plug-in

ARUN ARUMUGAM MATHIVANAN

KTH Information and
Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

MiniSIP as a Plug-in

Arun Arumugam Mathivanan

arunam@kth.se

2012.11.08

Master's Thesis

Supervisor & Examiner: Prof. Gerald Q. Maguire Jr.

School of Information and Communication Technologies
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

Internet telephony has rapidly becoming an integral part of life. Due to its low incremental cost and the wide availability of voice over IP (VoIP) based services these services being used by nearly everyone. Today there are many VoIP applications available in the market, but most of them lack basic security features. Because people use VoIP services via public hotspots and shared local area networks these VoIP applications are vulnerable to attacks, such as eavesdropping. Today, there is a great need for VoIP applications with high quality security.

MiniSIP is an open-source VoIP application platform, initially developed at KTH. High quality security has been a major focus of MiniSIP developments by several students, including the first public implementations of the secure real-time protocol (SRTP) and the Multimedia Key Exchange (MIKEY) protocol. MiniSIP implements secure end-to-end VoIP services. In addition, MiniSIP implements features such as dynamically choosing the most appropriate CODEC during a call, implementing calling policies, etc. However, it suffers from having a complicated GUI that requires the use of many libraries, rendering it both hard to build and hard support – both of which make it unsuitable for commercial purposes.

Web browser plug-ins are shared libraries that users install to extend the functionality of their browser. For example, a plug-in can be used to display content that the browser itself cannot display natively. For example, Adobe's reader plugin displays PDF files directly within the web browser. Real Network’s Streaming video player utilizes a browser plug-in to provide support for live video streaming within a web page. Adobe’s Flash player plugin is required to load or view any Flash contents – such as video or animations.

The goal of this thesis project is remove the problem of the existing MiniSIP GUIs by developing a Firefox browser plug-in for the MiniSIP application that will utilize a web-browser based GUI. The prototype that will be designed, implemented, and evaluated will implement an open-source VoIP application that is easy for a Firefox browser user to install and will be easy to use via a web interface. The long term goal is to facilitate an ordinary user to utilize VoIP communication via their web browser. A secondary goal is to re-use the code within MiniSIP, while using the web-browser to provide the GUI.

Keywords: VoIP, MiniSIP, Firefox plugin, web-browser

Sammanfattning

Internettelefoni har snabbt blivit en integrerad del av livet. På grund av dess låga marginalkostnaden och den breda tillgången på Röst över IP (VoIP) tjänster dessa tjänster används av nästan alla. Idag finns det många VoIP-applikationer som finns på marknaden, men de flesta av dem saknar grundläggande säkerhetsfunktioner. Eftersom människor använder VoIP tjänster via offentliga hotspots och delade lokala nätverk dessa VoIP-applikationer är sårbara för attacker, såsom avlyssning. Idag finns det ett stort behov av VoIP-applikationer med hög kvalitet säkerhet.

MiniSIP är ett open-source VoIP-program plattform, ursprungligen utvecklats vid KTH. Hög kvalitet säkerhet har varit ett stort fokus på MiniSIP utvecklingen genom att flera studenter, däribland de första offentliga implementeringar av den säkra realtid protokoll (SRTP) och Multimedia Key Exchange (MIKEY) protokollet. MiniSIP implementerar säker början till slut VoIP tjänster. Dessutom genomför MiniSIP funktioner som dynamiskt välja den lämpligaste CODEC under ett samtal, genomföra samtalsstrategier, osv. Men lider den från att ha en komplicerad GUI som kräver användning av många bibliotek, vilket gör det både svårt att bygga och hård stöd - som båda gör det olämpligt för kommersiella ändamål.

Webbläsare plug-ins delas bibliotek som användare installerar för att utöka funktionerna i sin webbläsare. Till exempel kan en plug-in kan användas för att visa innehåll som webbläsaren inte själv kan visa inföding. Till exempel visar Adobes Reader plugin PDF-filer direkt i webbläsaren. Real Networks strömmande videospelare använder en plugin-att ge stöd för levande video strömning i en webbsida. Adobe Flash Player plugin krävs för att ladda eller visa en Flash innehåll - såsom video eller animationer.

Målet med denna avhandling projektet är bort problemet med befintliga MiniSIP GUI genom att utveckla en Firefox webbläsare plug-in för att MiniSIP programmet som kommer att använda en webbläsare baserad GUI. Prototypen som kommer att utformas, genomföras och utvärderas kommer att genomföra en öppen källkod VoIP-program som är lätt för en Firefox webbläsare användaren att installera och kommer att vara lätt att använda via ett webbgränssnitt. Det långsiktiga målet är att underlätta en vanlig användare att använda VoIP-kommunikation via sin webbläsare. En sekundär målsättning är att återanvända kod i MiniSIP, medan du använder webbläsare för att ge det grafiska gränssnittet.

Nyckelord: VoIP, MiniSIP, Firefox plugin, webbläsare

Acknowledgement

Foremost, I would like to thank my thesis supervisor Prof. Gerald Q. “Chip” Maguire Jr. (School of Information and Communication Technologies, KTH Royal Institute of Technology, Stockholm, Sweden) for his valuable suggestions and continuous motivation. I am very grateful for his valuable time in encouraging me and getting back to me in a short span of time whenever approached.

I am very grateful to the Almighty for bestowing me with the opportunity and will to pursue my higher education in Sweden and giving me courage and strength to overcome all the obstacles that I encountered during this period.

My sincere gratitude to Ms. May-Britt Eklund Larsson for the continuous support she provided during my course of study at KTH.

Furthermore, I want to thank all my friends and family for their constant moral support and bestowed with their kind gestures throughout my life.

My gratitude will never be enough to my Mom, Dad, and My sister who stood in staunch support helping to make my dreams come true.

8 November 2012, Stockholm, Sweden

Table of Contents

Abstract i

Sammanfattning iii

Acknowledgement v

Table of Contents vii

List of Figures ix

List of Tables xi

List of Acronyms and Abbreviations xiii

1 Introduction 1

1.1 Problem Statement 1
1.2 Approach 1
1.3 Structure of this thesis 2

2 Background and Related Work 3

2.1 Firefox Plugin 3
2.1.1 Sample Plugin framework 3
2.1.2 Difference between Plugins and Extensions 4
2.2 Session Initiation Protocol 4
2.3 FireBreath Plugin framework 9
2.3.1 Plugin Design 9
2.4 MiniSIP libraries 10
2.5 Other VoIP plugins for Firefox 11
2.5.1 Zoep (also known as OpenZoep) 11
2.5.2 Abbeynet’s Voice Over Web (VOW) 11
2.5.3 VoIPfone™ 11
2.5.4 sipgateFFX 12
2.5.5 WebX 12
2.5.6 Mizutech’s Mizu Webphone 12
2.5.7 Greasemonkey extension for the Firefox browser and Ralf Muehlen's VoIP_dial_user.js script 13
2.5.8 OpenWengo 13
2.5.9 Google Talk 13
2.5.10 MozIAX 14
2.5.11 8x8's Virtual Office Pro VoIP and Unified Communications Solution 14
2.5.12 Switchvox Fire Dialer Extension 14
2.5.13 Skype Extension for Firefox 14
2.6 Comparison between the various VoIP plugins 15

3 Design and Implementation 17

3.1 Creating plugin using FireBreath 17
3.2 Adding library dependencies to the plugin project in FireBreath 19

3.3 Design of MiniSIP Firefox plugin and its web-based GUI 20
3.3.1 Register Dialog 21
3.3.2 Dialpad and Call Dialog 22
3.4 MiniSIP Firefox Plugin data-flow diagram: 23
4 Analysis 24
4.1 Comparison between the plugin GUI and the GTK+ GUI MiniSIP 24
4.2 Comparison between Mizutech's Mizu webphone and MiniSIP Firefox Plugin 25
4.3 Analysis and comparison of the call setup-delay between the MiniSIP application and the MiniSIP Firefox plugin 26
4.4 Analysis and compare the one-way RTP delay between MiniSIP application and the MiniSIP Firefox plugin 29
4.5 Comparison of installing and initiation time of MiniSIP as an application versus the new plugin 32
5 Conclusions and Future work 33
5.1 Conclusions 33
5.2 Future work 33
5.3 Required reflections 34
References 35
Appendix A: Sample Javascript functions 39
Appendix B: Sample HTML code 41
Appendix C: Call set up delay measurements 43
Appendix D: RTP delay measurements 45

List of Figures

Figure 2-1: Sample plugin framework 3
Figure 2-2: SIP REGISTER request/response 5
Figure 2-3: Request and Response made through proxy server 7
Figure 2-4: Request made through Re-direct server 8
Figure 2-5: Design of VoIP plugin 10
Figure 3-1: About Plugins Window in Firefox browser 18
Figure-3-2: FireBreath test page to check installed plugin 19
Figure-3-3: Making Javascript calls on plugin using Firebug console 19
Figure-3-4: JSP graphical user interface of MiniSIP Firefox plugin 20
Figure 3-5: Sample JSP code: HTML code 21
Figure 3-6: Sample JSP code: JavaScript function 21
Figure-3-7: User Register Dialog 21
Figure 3-8: Dialpad of MiniSIP Firefox plugin 22
Figure 3-9: Call Dilaog of MiniSIP Firefox plugin 23
Figure 3-10: MiniSIP Firefox Plugin data-flow diagram 23
Figure 4-1: GTK graphical user interface of MiniSIP 24
Figure 4-2: Graphical user interface of Mizu-Webphone 25
Figure 4-3: SIP Protocol messages exchanged during a call setup 26
Figure 4-4: Call setup delay of MiniSIP VoIP application 27
Figure 4-5: Call setup delay of MiniSIP Firefox plugin 28
Figure 4-6: Comparison between call setup delay of MiniSIP and Firefox Plugin 28
Figure 4-7: Box Plot - Comparison between call setup delay of MiniSIP and Firefox Plugin 29
Figure 4-8: One way RTP delay of MiniSIP VoIP application 30
Figure 4-9: One-way RTP delay of MiniSIP Firefox plugin 30
Figure 4-10: Comparison of one way RTP delay between MiniSIP and Firefox plugin 31
Figure 4-11: Box Plot - Comparison of one way RTP delay between MiniSIP and Firefox plugin 31

List of Tables

Table 2-1: Comparison between the various VoIP plugins 15
Table 3-1: FireBreath Plugin project files 17
Table C-1: PDD measurements of MiniSIP 43
Table C-2: PDD measurements of MiniSIP Firefox Plugin 44
Table D-1: One way RTP delay Measurements 45

List of Acronyms and Abbreviations

API	Application programming interface
CODEC	Coder/Decoder
CSS	Cascading Style Sheets
DND	Do not disturb
GUI	Graphical user interface
HTML	Hyper Text Markup Language
JSP	Java Server Pages
MIKEY	Multimedia Key Exchange
NPAPI	Netscape Plugin Application Programming Interface
PDD	Post Dial Delay
PSTN	Public Switched Telephony Network
RAD	Rapid application development
RTP	Real-time Transport Protocol
SCTP	Stream Control Transmission Protocol
SDP	Session Description protocol
SIP	Session Initiation Protocol
SMS	Short message service
SRTP	Secure Real-time Transport Protocol
UI	User interface
VoIP	Voice over IP
XMPP	Extensible Messaging and Presence Protocol

1 Introduction

Internet telephony and voice over IP (VoIP) based services have rapidly grown in popularity in recent years. Many VoIP based application have been developed and introduced in the market. MiniSIP[1] is a highly secure open-source VoIP application initially developed at KTH Royal Institute of Technology. The main focus when developing MiniSIP was high quality security. It featured one of the first public implementations of both the secure real-time transport protocol (SRTP) and the Multimedia Key Exchange (MIKEY) protocol. Additionally, MiniSIP has various features such as dynamically selecting an appropriate coder/decoder (CODEC) and support for calling policies. MiniSIP was mainly targeted for execution in a Linux environment, although implementations have also been done for Microsoft's Windows and Windows Mobile platforms. However, under Linux due to its use of a complicated graphical user interface (GUI) which has a lot of dependencies on other libraries, MiniSIP has not been widely used. Furthermore, there are few simple but user friendly VoIP based applications hence there was a need for such an application.

Mozilla's Firefox web browser is one of the most widely used internet browsers. One of the main features of Firefox is its support for plugins. Browser plugins are shared libraries that users can install to extend the functionality of the browser[2]. For example, Adobe's reader plugin allows the user to view PDF files directly in the browser. Real Networks' streaming video player, and Adobe's Flash player plugin. Adobe's Flash player allows users to download Adobe Flash contents directly from a web-page and have the contents rendered in a browser window. Plugins are easy to install and do not affect the browser in the same manner that an extension or an add-on does. Later we will describe in detail the differences between browser plugins, extension, and add-ons.

1.1 Problem Statement

In MiniSIP, the current GUI is rather complicated and requires the use of many libraries, making it hard to build and hard to support. The current GUI was build using the GTK+ library[3]. In addition, the Glade design interface tool was used to draw GUI contents and render them as a XML files that together with the code realize the GUI contents in order to add suitable controls to the user interface. Using the GTK+ library and the Glade tool makes the MiniSIP GUI too complicated for commercial use and too time consuming to build for a casual user.

1.2 Approach

This thesis project will remove the problem of the existing MiniSIP GUI by developing a Mozilla Firefox Plugin and a web-based GUI to support this plugin. The goal is to develop a prototype stand-alone VoIP application as a Firefox plugin for the Linux platform. The prototype that will be designed, implemented, and evaluated will implement an open-source VoIP application that is easy for a Firefox browser user to install and will be easy to use via a web interface. To support the Firefox plugin development, we plan to use the FireBreath plugin framework. FireBreath is an open source cross-browser plugin framework. A secondary goal is to re-use the session initiation protocol (SIP), SRTP, and MIKEY code within MiniSIP, while using the web-browser to provide the GUI. The long term goal is to facilitate an easily installed and ready-to-use VoIP communication system that can be installed and used by ordinary users.

1.3 Structure of this thesis

Following this introduction, Chapter 2 provides some background about Firefox plugins and the MiniSIP libraries that will be utilized. Section 2.5 describes some of the existing Firefox VoIP plugins that exist. Chapter 3 describes the design and implementation of a prototype MiniSIP plugin. Chapter 4 compares this solution to a native C++ implementation of MiniSIP running on the same computer. The thesis concludes with some conclusions and suggestions for future work in Chapter 5.

2 Background and Related Work

This chapter explains the underlying concepts needed to understand what a Firefox plugin is and how it work. This chapter will also review the basic concepts necessary to understand the rest of the thesis and will use related work to put the thesis in a proper context.

2.1 Firefox Plugin

A plugin (or plug-in) is a shared library that can be easily installed and maintained as part of a web browser. The main purposes of browser plugins are to render contents natively in the browser when the web application cannot do so natively. This rendering is typically a transformation of the content into a format that can be displayed in the web browser's window. The rendering could also generate audio output or even accept audio input. There are many different plugins currently available for Firefox, but the most common ones are Adobe's PDF reader plugin, Real Network's streaming video player, and Adobe's Flash player plugin.

Firefox plugins are written to use the Netscape Plugin Application Programming Interface (NPAPI)[4], a cross-browser application programming interface (API) for Firefox plugins[2]. Documentation for the API can be found in the Gecko Plugin API Reference[5]. The methods in the plug-in API that are available from the plugin object all start with the string "NPP_"[5]. These methods can be called by the plugin to perform a function *within* the browser.

2.1.1 Sample Plugin framework

Logically we can view the plugin as an independent piece of code that has a plugin interface that enables the web browser invoke it and a service interface that enables the plugin to interact with the web browser. This is shown schematically in Figure 2-1. In this figure we note that there is a "Plugin Manager" inside the web browser that loads and controls all of the various plugins that a user has configured.

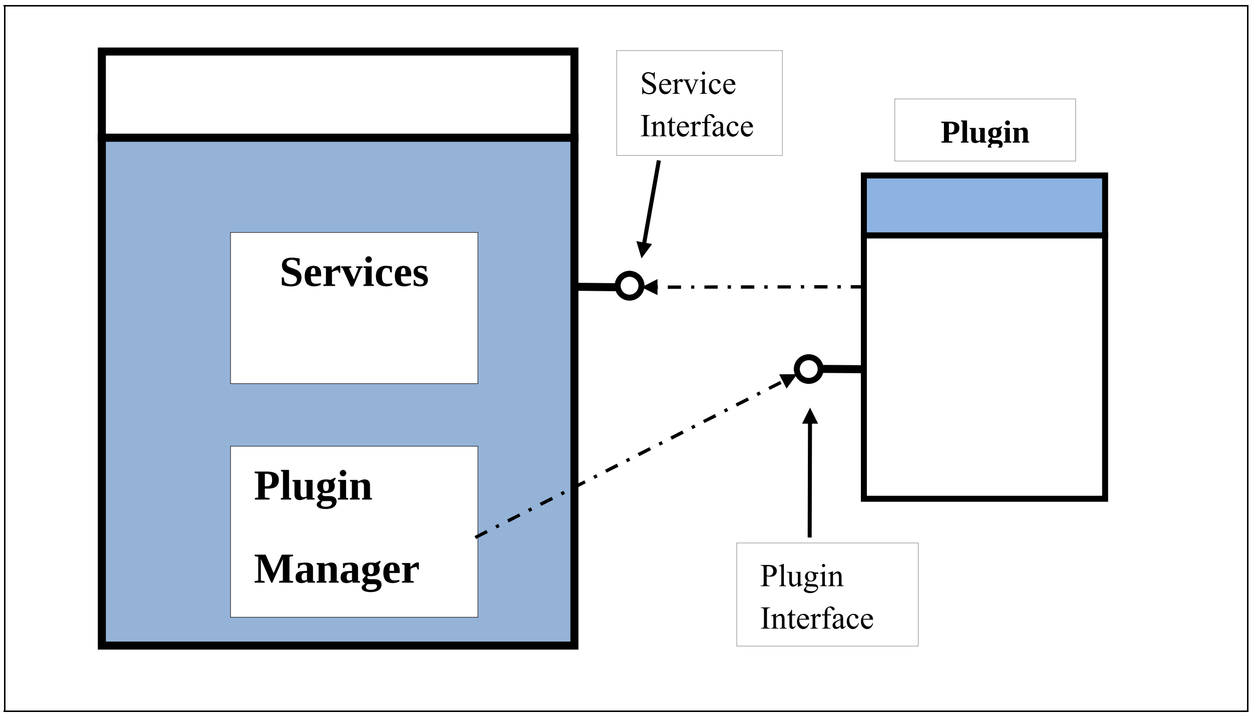


Figure 2-1: Sample plugin framework

2.1.2 Difference between Plugins and Extensions

A plugin is, simply, a third party library that "plugs in" to the browser. A plugin can be embedded inside a web page using an <embed> tag or a <object> tag. Plugins can be called through JavaScript[4]. A plugin affects only a specific page in which it is placed. Examples of common plugins include Macromedia Flash, Microsoft Silverlight, Apple QuickTime, and Adobe Reader. Some plugins respond to a Multipurpose Internet Mail Extensions (MIME) type and can run in place of the page, such as Adobe Reader, which allows you to view PDF files in your web browser. As noted earlier the plugin affects only this page, and no others. In Firefox, these plugins are usually called "NPAPI plugins", since they are written using the Netscape Plugin Application Programming Interface (NPAPI) [4]. Plugins do not know about tabs or even about other pages in the same browser process.

Extensions or Add-ons are programs that add new functionality to Firefox. They can be used to create toolbars, browser menus, or add to the browser's user interface (UI). These programs can even process the page the browser loads. These extensions can be customized to fit the personal needs of each user, while keeping the size of the applications that must be downloaded small. Unlike plugins, they affect the browser *itself* rather than only a downloaded page. In Firefox, we can even put a plug-in *inside an extension*. An examples of an extension is Firebug for Firefox, which is a powerful web development tool that integrates with Firefox for editing, viewing, and debugging CSS, Java Script, and HTML live in any webpage[6]. Another example is the Zotero extension for Firefox, which provides a bibliographic reference system - including the ability to provide information to a separate word processing program enabling it to insert citations and create a bibliography for a document.

The difference between using an extension and a plugin in the context of this thesis project concerns **where** the functionality is placed: (1) as a plugin – the code would be in a library associated with processing a page or (2) as an extension – into the browser itself. As we want to remove the existing Minisip GUI and utilize a web-based GUI we could require that the user install a GUI-less MiniSIP application that would use a Firefox extension to provide a web-based GUI. This option was view as being more complicated for ordinary users, since they would have to download and install a local program and download and install the extension into their browser. Thus we have taken the alternative approach and decided to develop a Firefox VoIP plugin. This plugin will incorporate some of the MiniSIP code – so as to exploit the SIP and CODEC functionality implemented in this code base. This alternative should make MiniSIP's VoIP features available for any user of Firefox in a Linux environment.

2.2 Session Initiation Protocol

The Session Initiation Protocol (SIP) [7] is the underlying protocol for most VoIP applications. SIP is an application layer control protocol for creating, modifying, and terminating sessions with one or more participants. These sessions can include internet telephone calls, multimedia distribution, and multimedia conferences [8]. SIP employs design elements similar to the HTTP request/response transaction model. SIP uses TCP, SCTP, or UDP transport protocols on port numbers 5060 or 5061. SIP uses the Session Description protocol (SDP) to negotiate parameters such as port numbers, type of media streams between the clients. These sessions can use the Real-time Transport Protocol (RTP) to carry media streams (video, audio, timed text, etc.) between the end-points.

The four major SIP network components are: User Agent (UA), Registrar server, Proxy server, and Redirect server. Each of these is explained in further detail below.

User Agent (UA)	A SIP UA is an end-point that can send and/or receive SIP messages. The UAs are responsible for maintaining SIP sessions. A given UA could be a User Agent Client (UAC) which *originates* a SIP message or a User Agent Server (UAS) which listens for *incoming* SIP messages. UAs can be realized as softphones, hard phones, messaging clients, etc.
SIP Registrar server	The SIP registrar server keeps tracks of the UAs that wish to be reachable by others. It does this by processing SIP REGISTER request messages. A UA sends a REGISTER request to the registrar server. If the registrar server accepts this request it will create a mapping between the SIP URI in the requests and the IP address (or a fully qualified domain name) provided in the request. The registrar server stores this information in a "location" database and is frequently co-located with an incoming SIP proxy server **[7]**, as the incoming proxy server will query the location database to learn where it should forward SIP messages to. Figure 2-2 shows the registration request and response messages that pass between the UA and the SIP registrar server.

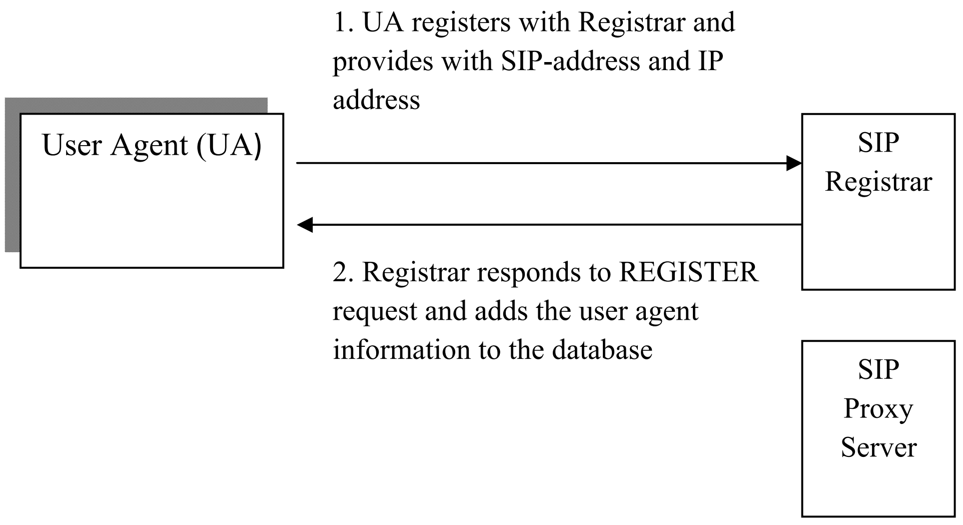

1. UA registers with Registrar and provides with SIP-address and IP address
User Agent (UA)
SIP Registrar
2. Registrar responds to REGISTER request and adds the user agent information to the database
SIP Proxy Server

Figure 2-2: SIP REGISTER request/response

SIP Proxy Server

The SIP proxy servers are the central entities in a SIP network. SIP UAs (such as phones and gateways) generally connect to a proxy server so that this proxy can appropriately forward the SIP message to its destination. While the main role of a SIP proxy server is routing requests to appropriate destination, these servers can also be used for enforcing security policies (for example, blocking calls from certain users) or to implement services to deal with missed calls, perform selective call forwarding, etc. The use of a SIP proxy is shown in Figure 2-3.

A SIP proxy can fork a SIP INVITE (to initiate a session) sending a copy of the INVITE to several different locations[8]. If a call is being made to a particular user and this user has registered their SIP URL from multiple UAs, then each of these UAs would receive a copy of the INVITE request and hence would “ring” at the same time. The first of these UAs that responds to the INVITE would receive the call and proxy would send messages to the other UAs to stop ringing.

A SIP proxy server can act as both a UAS and a UAC in order to make requests on behalf of a UAC **[7]**. Such a device is often called a back-to-back user agent.

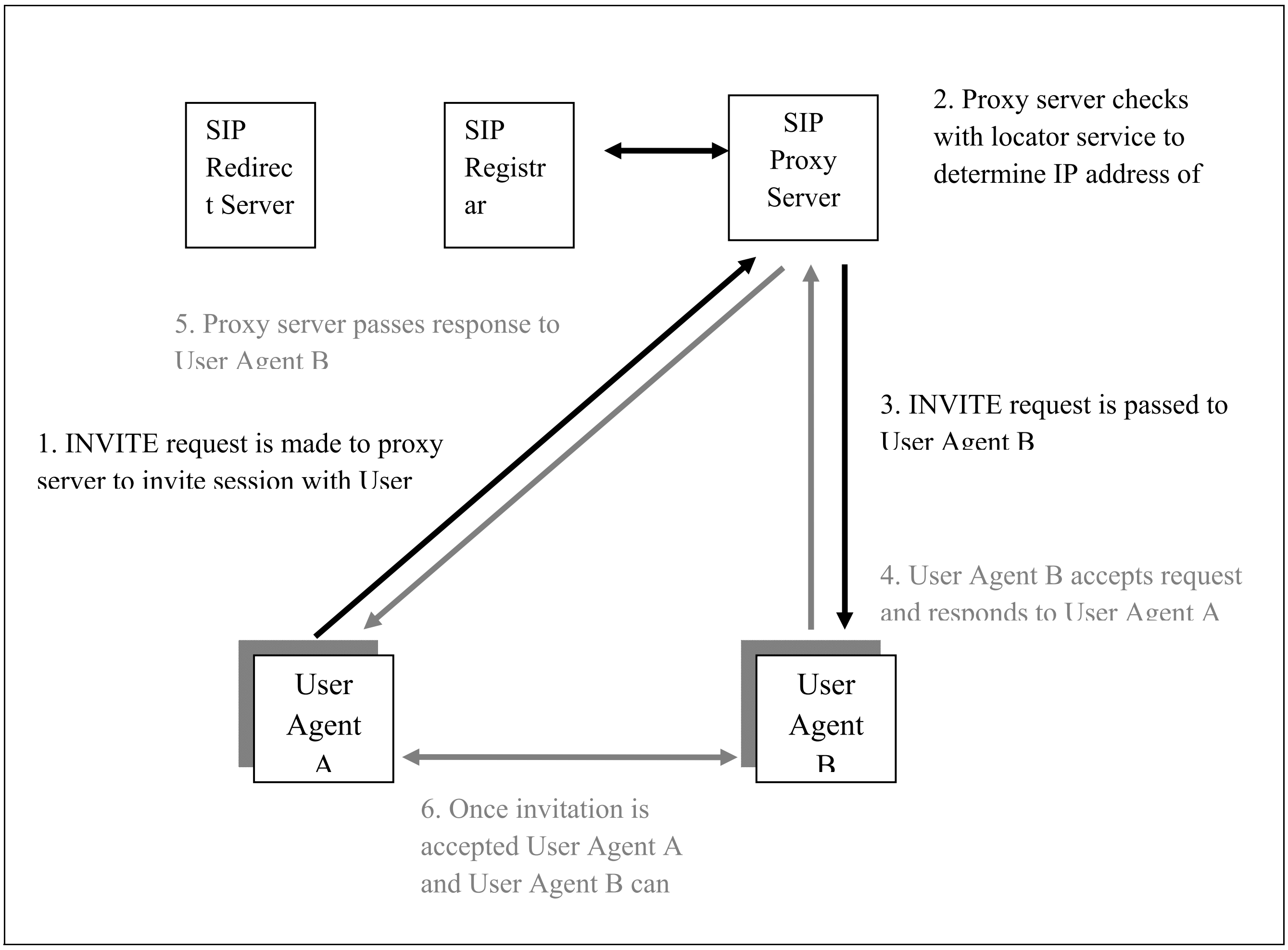

SIP Redirec t Server
SIP Registr ar
SIP Proxy Server
2. Proxy server checks with locator service to determine IP address of
5. Proxy server passes response to User Agent B
3. INVITE request is passed to User Agent B
1. INVITE request is made to proxy server to invite session with User
4. User Agent B accepts request and responds to User Agent A
User Agent A
User Agent B
6. Once invitation is accepted User Agent A and User Agent B can

Figure 2-3: Request and Response made through proxy server

SIP Redirect server A redirect server is used by SIP to redirect clients to the UA they are attempting to contact. If a UA makes a request, the redirect server can respond with the IP address of the desired UA. This is different from a proxy server, which forwards the request on the UA's behalf, while the redirect server redirects the SIP request. The re-direct server can allow a proxy server to direct SIP session invitations to external domains. Figure 2-4 shows the operation of a redirect server.

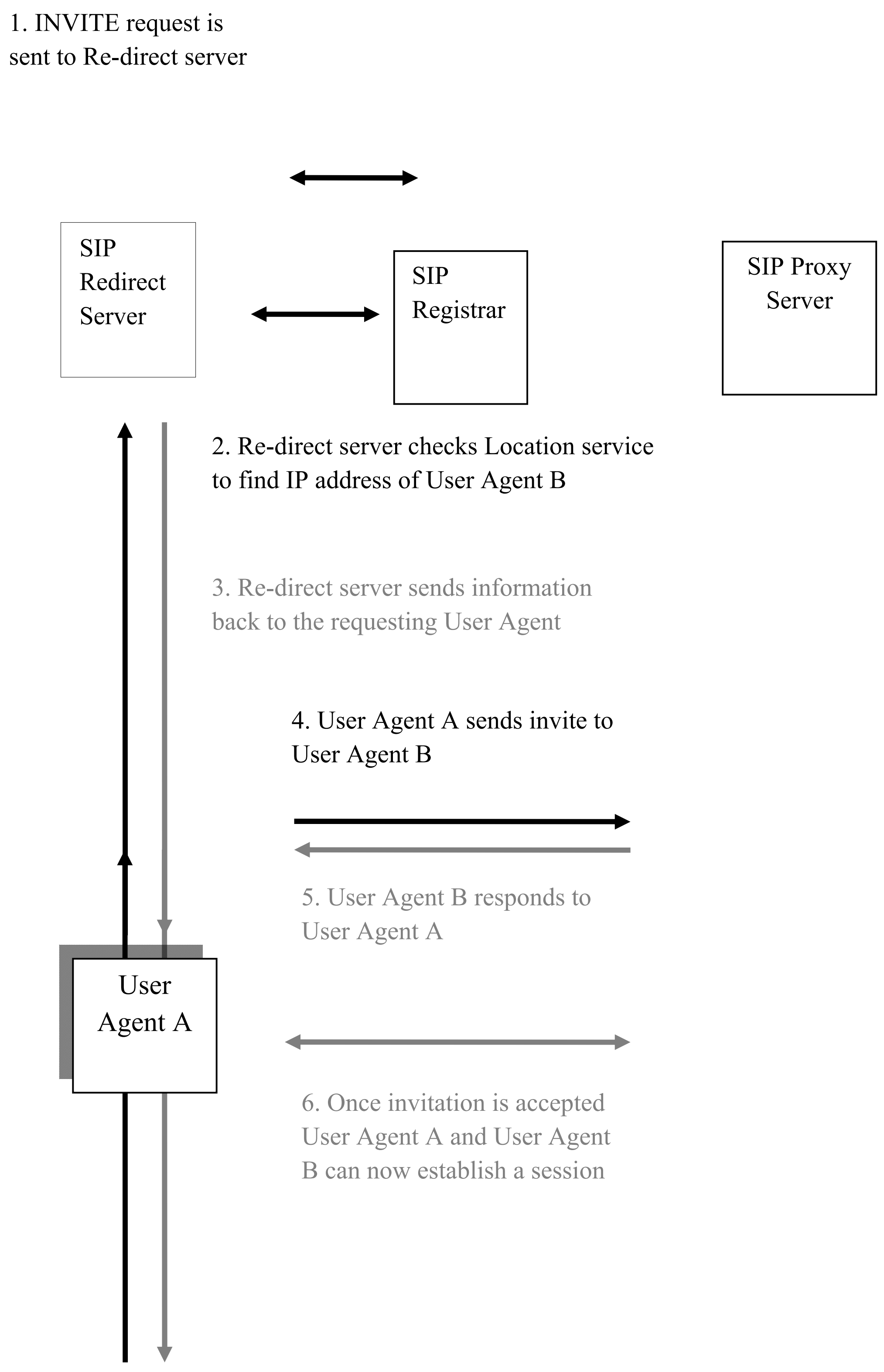

1. INVITE request is sent to Re-direct server
SIP Redirect Server
SIP Registrar
SIP Proxy Server
2. Re-direct server checks Location service to find IP address of User Agent B
3. Re-direct server sends information back to the requesting User Agent
4. User Agent A sends invite to User Agent B
5. User Agent B responds to User Agent A
User Agent A
6. Once invitation is accepted User Agent A and User Agent B can now establish a session

Figure 2-4: Request made through Re-direct server

2.3 FireBreath Plugin framework

FireBreath [9] is a cross-platform browser plugin framework written in C++. FireBreath supports many different platforms including NPAPI browsers on Windows, MacOS, and Linux, for example:

- Gecko/ Firefox
- Google Chrome
- Apple Safari
- Opera (usually)
- Activex Control hosts
- Microsoft Internet Explorer 6, 7, and 8

Supported development environment include Visual Studio on Microsoft's Windows operating systems, XCode on Apple's MacOS, and GCC on Linux [10]. All these platforms rely on CMake for easy cross-platform building. Feature highlights include thread safety checks, Unicode support (with std::wstring), ActiveX support, built-in drawing model negotiation for MacOS, automatic type conversion (including JavaScript Arrays and Objects), advanced security features, abstraction for Mouse and Keyboard events on MacOS and Windows, automatic project creation, JavaScript interface, DOM access and manipulation abstraction, an automatic install generator for Windows to create a plugin installer, and more[11].

2.3.1 Plugin Design

The overall design for the proposed VoIP plugin is shown in Figure 2-5. The proposed plugin will be developed using the FireBreath plugin framework. The includes a simple user friendly GUI, support for SIP user registration (however, it will be limited to supporting existing SIP users – thus there will not be provisions for creating new SIP user accounts as this lies outside the scope of this project), and making VoIP calls to other SIP users. Additionally, this plugin will include support for several audio CODECs (specifically those currently supported in MiniSIP).

This plugin will get the user's configuration information from a web server. The plugin will send a registration message for this user to the SIP registrar in order to register the plugin's address as a UA for this user. When the user wants to initiate a call, the "Dial" module will send the appropriate SIP INVITE message to an outgoing SIP proxy server which will forward this INVITE toward the "dialed" SIP user (i.e., the callee). The sessions will stream audio to and from the plugin using the CODECs currently implemented by MiniSIP.

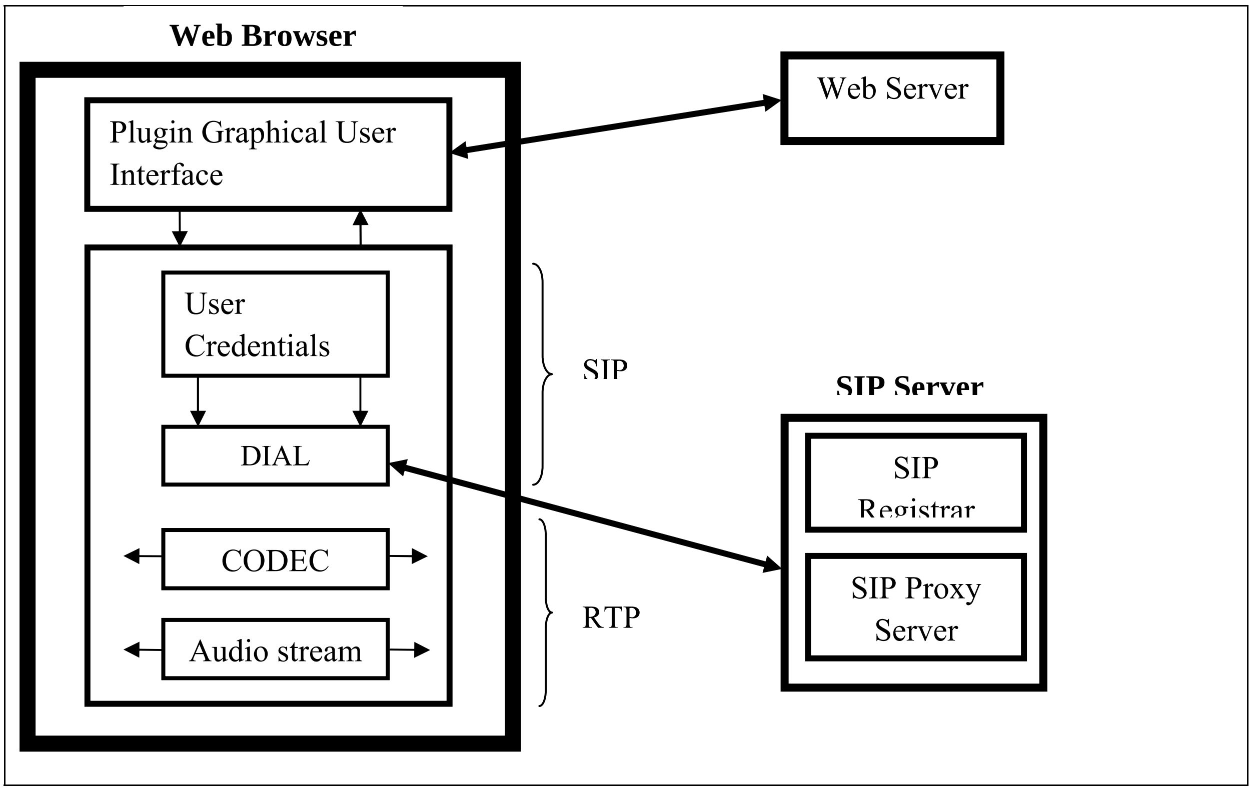


Figure 2-5: Design of VoIP plugin

2.4 MiniSIP libraries

MiniSIP is implemented as a number of libraries and a user interface. The set of libraries that we are concerned with include:

libmsip	This library implements the functions to support SIP.
libminisip	This library implements the core of MiniSIP and deals with the RTP and media streams.
libmutil	This library contains cross-platform support for threads, mutexes, semaphores, and other utility classes.
libmnetutil	This library contains various network related utility functions.
libmstun	This library supports STUN in order to deal with network address translation (NAT) devices.
libmikey	This library implements MIKEY.
libmcrypto	This library implements all of the cryptographic functions used by the other libraries – it makes use of the OpenSSL libraries.

2.5 Other VoIP plugins for Firefox

Since 2006 quite a number of VoIP plugins have been made for Firefox [12]. One of the earliest of these is Zoep. In this section we review some of the existing Firefox plugins for VoIP and described their advantages and disadvantages.

2.5.1 Zoep (also known as OpenZoep)

The OpenZoep Foundation developed an open source VoIP client named Zoep [13]. It was a Mozilla Firefox add-on that allows free PC-to-PC VoIP calls between the Zoep clients as well as a pre-pay PC-to-Phone option and instant messaging service also available. The OpenZoep Foundation's OpenZoep communications engine is based on the XMPP Extensible Messaging and Presence Protocol (used by both Jabber and Google Talk). Their main goal is to create "Open Directories" in order to find users more easily. Since OpenZoep uses Jabber for its presence and instant message handling, it is easier to search for a user on other jabber networks. This ensures that applications that are based on OpenZoep will benefit from a shared and global telephone directory.

In the future OpenZoep plans to introduce an initiative to establish a global service provider discovery server. This server allows end-users to pick and choose a service provider that offers basic telephony services such as public switched telephony network (PSTN) interworking with PSTN-in and PSTN-out, voicemail, and other services from thousands of Internet telephony service providers. The main features of this program are free calls, cheap VoIP calls to landline and mobile phones, and instant messaging support. The main advantage of using OpenZoep is that user can select any service provider. Additionally, with more and more service providers starting to use the OpenZoep communication engine there is a shared/global directory for search and it is easy to invite friends to join in this service. Currently Zoep is only available for Microsft's Windows platform, although Linux and OS-X version are currently being ported.

2.5.2 Abbeynet's Voice Over Web (VOW)

Abbynet's VOW [14] turns Firefox into a VoIP client, which implies that one can have a full featured SIP user agent directly in Firefox. VOW was developed as a Firefox Extension. The main advantage of VOW was it allows users to configure their favorite VoIP service provider; they are not restricted only to Abbeynet's VoIP service. There is no need for any other software, as all the VoIP functionality was built inside the extension itself. Internet calls are free as well as cheap PSTN call rates provided by Abbeynet. Another interesting feature is that a user can dial a phone number directly from the web page they are browsing with a single-click to call. They are also adding video support as well as English and Italian language support. VOW can accessed from the bottom of the Firefox browser window, an edit box with the call buttons allows user to call any number typed into the edit box directly from the web browser. Additionally, it also possible to send a short message service (SMS) message.

2.5.3 VoIPfone™

VoIPfone [15] is a browser plugin from iNet Telecoms Ltd. for Microsoft's Internet Explorer, Firefox, and Google Chrome. They are offering a wide range of hosted VoIP phone services and products to both the business and residential customers. The VoIPfone browser dialer plugin recognizes telephone numbers in web pages, hence the user can simply click to call such a number, regardless of what telephone the user is using. When the user clicks on a number their own phone will rings. After the user answers this call, there is a pause for a moment indicating that the user is connected to respective number they clicked. Now the user will hear the ringing tone of the telephone they just called, after the process continues just as a normal call. Currently, VoIPfone is only available for users in United Kingdom. The VoIPfone plugin is free to download and calls are charged as per the usual fixed telephony rate.

2.5.4 sipgateFFX

Sipgate is a VoIP service provider. SipgateFFX [16] is an add-on for Firefox and Thunderbird developed by Sipgate. The extension provides users of Sipgate's VoIP service with lots of interesting features, such as practical overview of their account, direct call list notifications, high speed access to several parts of sipgate's website, initiating click2dial calls, and sending messages and faxes directly from the user's browser. Another major feature is that users can send PDF documents via fax directly from their browser. Users can also disable sipgateFFX in specific websites to avoid providing their account numbers or phone number in click2dial entries. This feature could be helpful when using online banking websites. The latest version of the extension adds setting the do not disturb (DND) status for the customer's phone(s). This feature mutes all the user's phones. This feature is useful when a user does not want to receive calls during a particular time – they can simply activate this DND feature, then all callers will get a busy signal. Currently Sipgate's VoIP service is available only for customers in the United Kingdom or the United States.

2.5.5 WebX

Next Solution's WebX [17] is a JavaScript based conventional softphone working in a browser. It can be accessed from anywhere with the same functionalities and quality as a softphone. The user needs to have an Asterisk PBX or a SIP account for use with WebX. WebX only works with Microsoft's Internet Explorer due its use of ActiveX. Additionally, Javascript needs to be enabled. WebX allows users to choose which CODEC they want to use and to receive calls through a selected SIP account. Apart from acting as a free software phone (softphone) which has ads enabled, there are two other modes of operation: ads free mode and white label mode. In the ads free mode users can use Webx without any advertisements; but this requires that the user purchase this mode from NXT through their store. In white label mode users can replace all the images and create a new WebX with their own brand for use by their own customers, thus creating a customized WebX for commercial use. A license for white mode can be purchased via the NXT store. As of June 2012, the company's website http://www.nxtsolution.net/ is no longer in operation.

2.5.6 Mizutech's Mizu Webphone

Mizutech's Mizu webphone [18] is a lightweight VoIP softphone that runs in a web browser and is embeddable in any webpage. This software is platform independent, as the phone is implemented as a java applet or application that can be run in any java enabled browser. With Mizu Webphone users can add VoIP capabilities to their websites. Internet calls are free of charge, but the user can also call mobile phones or landlines using the VoIP provider of their choice. A few important features of Mizu webphone are compatibility with all standard VoIP servers or devices, it is a standard java applet, NAT/Firewall support, conference calls, instant messaging, voice recording, click to call, customizable GUI, and support in multiple languages. Mizu webphone is easy to use and deploy, customizable, platform independent, and based on telecommunication standards. This software can be deployed to provide click-to-call functionalities in a webpage, as a communication tool between company employees, as a browser phone plugin, etc. The full version license for the Mizu Webphone can be purchased for a reasonable price that varies from a license for a single user to a multi-level business entity.

2.5.7 Greasemonkey extension for the Firefox browser and Ralf Muehlen's VoIP_dial_user.js script

Greasemonkey is an extension for the Firefox browser that lets users customize the way a web page is displayed or behaves using small bits of JavaScript known as "userscripts" [19]. Already there are a wide variety of userscripts available on the internet to solve many problems. For example, the `linkify` script allows a user to make all the plain text URLs in a webpage into clickable URLs. Users can also write their own customizable userscripts to solve their own problems. Ralf Muehlen's `VoIP_dial_user.js` script was one of these userscripts. This userscript enables VoIP functionality for a web browser. Users can dynamically link every phone number in their webpage into a custom click-to-dial script with the Greasemonkey extension for the Firefox browser and Ralf Muehlen's `VoIP_dial_user.js script` [20]. The user can also modify this java script to suit their usage. Note that the actual work is done by SunRocket as the script simply turns phone numbers into a link with arguments of the form:

https://www.sunrocket.com/members/contacts/clickToCall.do?phoneToCall=.

2.5.8 OpenWengo

WengoPhone [21] is an open source application for voice, text, and video conferencing developed by OpenWengo, an open source project supported by the French telecommunications company Wengo. OpenWengo enables VOIP based communication using SIP standards; this ensures compatibility with other service providers using the same standards. The Firefox extension is only available for Microsoft's Windows, Linux, and Apple's MacOs X users. The extension adds a button to the Firefox toolbar. Clicking on this button opens the WengoPhone sidebar. This sidebar allows users to create and manage their Wengo accounts as well as to make VoIP calls, send instant messaging, etc. The user can create a Wengo contact list. This contact list is stored locally in the Firefox browser instead of on the Wengo server. As usual internet calls are free and the calls to mobile phones and landline phones can be made with call-out credits purchased from Wengo.

2.5.9 Google Talk

Google Talk [22] is an instant messaging service provided by Google. They provide both voice and text services through Google Talk. Google also provides voice and video chat in Google Talk through a browser plugin. This plugin is available for all platforms. A potential user simply downloads and installs the plugin. This plugin works seamlessly with the Gmail interface in the browser itself. This plugin is different from the stand-alone Google Talk application, which also supports both voice and video communication. It is also possible to make phone calls through Google Talk. Google talk servers communicate with the client using an open protocol called the Extensible Messaging and Presence Protocol (XMPP)[23].

Since XMPP is an open standard, Google Talk can be accessed through other clients that support the XMPP standard, such as Pidgin and Psi. Google talk interoperates with any service provider who uses the XMPP protocol.

2.5.10 MozIAX

MozIAX [24] is a Firefox VoIP extension. MozIAX implements a cross platform open source Softphone (specifically an IAX2 phone) to be used with Asterisk, an open source PBX. Users of the MozIAX Firefox extensions can select any phone number from a webpage, then right-click to call. MozIAX supports text messages and Firefox or Mozilla themes. This extension displays the URL received from IAX2, supports many of the major human languages, and also defines the prefix "tel:" to initiate a call from the webpage. MozIAX also connects to an Asterisk manager to display busy channels and queue status. MozIAX supports Yealink USB devices[25] only on Microsoft's Windows systems. MozIAX has a wide range of functionalities that are well suited for building a call center using open technologies.

2.5.11 8x8's Virtual Office Pro VoIP and Unified Communications Solution

8x8, Inc. offers integrated voice, video, and mobile unified solutions for small to medium-sized business and distributed enterprise customers. These solutions leverage existing broadband Internet connections and cellular networks to deliver advanced features and high definition voice service at a fraction of the cost of legacy alternatives. 8x8's Virtual Office Pro [26] VoIP is its flagship hosted PBX business phone service. This is an economical alternative to traditional PBX systems. 8x8's Virtual Office Pro is a full suite of cloud-based, unified communications services featuring automated attendants, conference bridges, extension-to-extension dialing, and ring groups. In addition, this product offers a rich variety of business class features normally found in dedicated PBX equipment. These services are accessible from a centralized web-based client and include: Virtual Meeting web conferencing with video, Internet fax, call recording, a mobile iPhone and Andriod application, click to dial, presence management, and more.

2.5.12 Switchvox Fire Dialer Extension

Switchvox Fire Dialer Extension [27] is a Firefox extension that works with a web browser and the Swithvox PBX to allow a user to click on phone numbers on any web page and to set up a call between that number and their extension. The general features of the Switchvox Fire Dialer extension are that users are allowed to dial numbers from any web page and they can import their Rapid Dial list to allow calls from the browser itself. The software also features a highlight feature to make finding a phone number easy in the web browser and conveniently keeps track of the user's previously dialed numbers.

2.5.13 Skype Extension for Firefox

The Skype Toolbar for Firefox is an extension that detects phone numbers in web pages, and renders them as a clickable button that can be used to dial the number using the Skype desktop application. This extension is normally bundled with the Skype application, and is installed into Firefox by default when Skype is installed or, in some circumstances when Skype is separately updated. To call the highlighted telephone numbers users need to purchase

Skype credits or have a Skype subscription [28]. The Skype click-to-call extension is also available for Microsoft's Internet Explorer and Google's Chrome browsers.

2.6 *Comparison between the various VoIP plugins*

The following table compares the VoIP plugins that have been presented in the previous section.

Table 2-1: Comparison between the various VoIP plugins

Plugin / Extension	Platform Support	Protocol Used	Features
Zoep	Microsoft Windows, Linux and Mac OS	XMPP	VoIP calls, Instant messaging, Presence support, User preferred VoIP service provider
Abbeynet's Voice Over Web (VOW)	Microsoft Windows, Linux and Mac OS	SIP	VoIP calls, SMS and call directly from webpage, video support , English and Italian language support
VoIPfone	Microsoft's Internet explorer, Firefox and Google chrome browsers	SIP	VoIP calls , click-to-dial and available only for UK customers
sipgateFFX	As a Firefox Extension for all the platforms	SIP	VoIP calls, click2dial calls, Sending messages and faxes, send PDF documents as fax and currently available only for UK and US customers
WebX	Microsoft internet explorer	SIP	VoIP calls, Ability to receive VoIP calls and select usr-preferred CODECs
Mizutech's Mizu Webphone	Any java enabled browser and Platform independent	SIP	Embeddable in any webpage, conference calls, IM, voice recording, click to call, customizable GUI

Plugin / Extension	**Platform Support**	**Protocol Used**	**Features**
Greasemonkey extension and Ralf Muehlen's VoIP_dial_user.js script	As a Firefox extension for all the platforms	SIP	VoIP functionality using userscripts, Click-to-dial
OpenWengo	Microsoft Windows, Linux and Mac OS	SIP	VoIP calls. Instant messaging and contact management in web-browser
Google Talk	As a browser plugin for all the platforms	XMPP	Voice and video chat using Google account and interoperable with any service provider or client supports XMPP protocol
MozIAX	As an open-source Firefox extension and cross-platform	SIP	Uses Asterisk to make VoIP calls, Click-to-dial, uses Asterisk manager to display busy channel and queue status
8x8's Virtual Office Pro VoIP and Unified communications solution	PC and Mobile platforms	SIP	Voice, Video and mobile solutions for businesses, Virtual Meeting web conferencing with video, Internet fax, call recording, iPhone and Andriod application, click to dial, presence management
Switchvox Fire Dialer Extension	As a Firefox extension for all platforms	SIP	Click-to-dial and supports user call logs.
Skype Extension	As a Firefox extension	SIP	Click-to-dial using Skype calling credits.

3 Design and Implementation

This chapter explains the basic procedure for developing a Firefox plugin using the FireBreath plugin framework. This chapter also describes the design and implementation of the MiniSIP Firefox plugin.

3.1 *Creating plugin using FireBreath*

The MiniSIP Firefox plugin is developed using the FireBreath plugin framework [9]. In the FireBreath plugin framework a blank plugin project is created using the fbgen.py tool. The fbgen.py tool is the fastest way to create a blank project with the user's given parameters. To run this tool on a Linux machine, the only pre-requisite is Python (version 2.6 or 2.7). If the user does not want to install Python, there is also way to create a user customized plugin template by copying the fbgen template and changing various parameters, the instructions for this can be seen at the FireBreath website under the topic "creating a plugin project without fbgen".

In order to run the fbgen.py tool we will assume that the Linux machine has Python version 2.6 or 2.7, FireBreath, and CMake 2.8 or greater installed (as per the instruction in [29]).

Connect to the firebreath directory. The following command prompts the user to enter a series of project related parameters such as Plugin name, Plugin identifier, Plugin mime type, and Plugin description etc.:

```
python fbgen.py
```

These parameters need to be filled in according to the user's project requirements. Upon completing all of these details, the template is processed to generate the project related files. These generated files are kept inside the project's folder in the FireBreath directory under the plugin name.

The plugin project directory contains a number of important files that are required for the development of the Mozilla plugin. These files are shown in Table 3-1.

Table 3-1: FireBreath Plugin project files

PluginConfig.cmake	This is the most important file as it contains ¨90% of the configuration. Configuration options from this file are used to generate other files that are critical to the plugin's configuration.
CMakeLists.txt	This is the main configuration file for generating the project and it also contains the entries for the cross platform files needed for the project.
Factory.cpp	This is a method called by FireBreath to create the main plugin object and it also contains the projects global init and deinit functions.
Pluginname.(cpp | h)	This is the main entry for the project and contains the plugin object. It has an API which handles the JavaScript calls.
PluginnameAPI.(cpp | h)	This is the default API class and it's where the users add functions, methods and JavaScript objects related to their plugin project.
Linux/projectDef.cmake	This contains the other half of the project definition started in 'CmakeLists.txt' and that specifically deals with Linux.
Win/np_winmain.cpp Win/dllmain.cpp	These files deal with Windows platform that has entry points for NPAPI and the plugin DLL as well.

The next step is to generate the project files to make them usable for developing. To generate project files, the user needs to run the platform specific 'prep' script. For Linux the script prepmake.sh script needs to be run in order to generate the project files. Execute the script file in the Linux terminal as follows.

Login as a root user and run the command:

```
./prepmake.sh
```

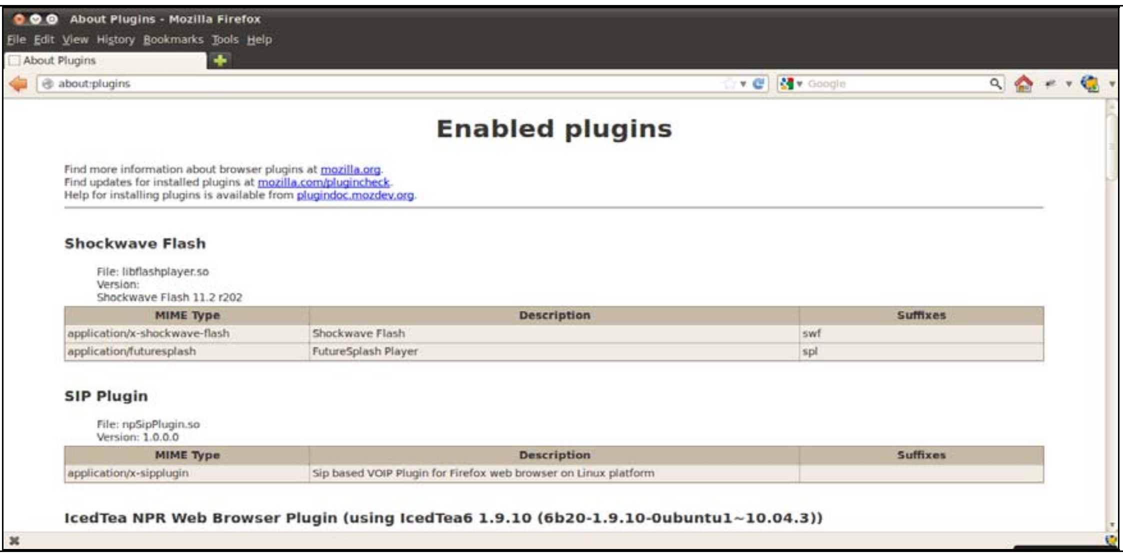
This command generates all the project build files under the build directory in the project root folder.

In Linux, we need to build the plugin in order to generate the NPAPI plugin files. We simply invoke the `make` in the build folder of FireBreath to create the plugin.

To make the plugin accessible, it needs to be installed in the *mozilla/ plugins* directory. In Linux, browsers such as (Firefox, Chrome) look for the NPAPI plugins in the "*/usr/lib/mozilla/plugins"* folder [30]. The plugin's shared object file is copied to the above mentioned location with the command:

```
cp bin/SipPlugin/npSipPlugin.so ~/usr/lib/mozilla/plugins
```

After installing the plugin, open the Firefox web browser and type *about:plugins* in the address bar. The browser will display the currently enabled plugins as shown in Figure 3-1.


About Plugins - Mozilla Firefox
File Edit View History Bookmarks Tools Help
About Plugins
about:plugins
Google

Enabled plugins

Find more information about browser plugins at mozilla.org.
Find updates for installed plugins at mozilla.com/plugincheck.
Help for installing plugins is available from plugindoc.mozdev.org.

Shockwave Flash

File: libflashplayer.so
Version:
Shockwave Flash 11.2 r202

MIME Type	Description	Suffixes
application/x-shockwave-flash	Shockwave Flash	swf
application/futuresplash	FutureSplash Player	spl

SIP Plugin

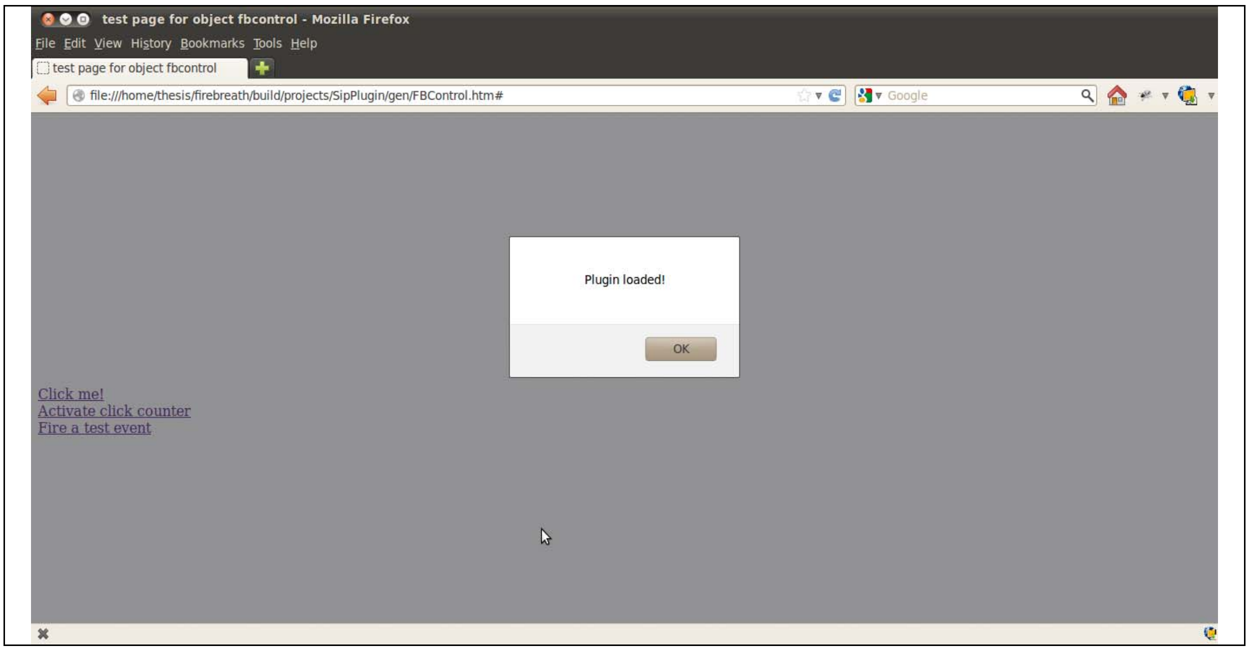
File: npSipPlugin.so
Version: 1.0.0.0

MIME Type	Description	Suffixes
application/x-sipplugin	Sip based VOIP Plugin for Firefox web browser on Linux platform	

IcedTea NPR Web Browser Plugin (using IcedTea6 1.9.10 (6b20-1.9.10-0ubuntu1~10.04.3))

Figure 3-1: About Plugins Window in Firefox browser

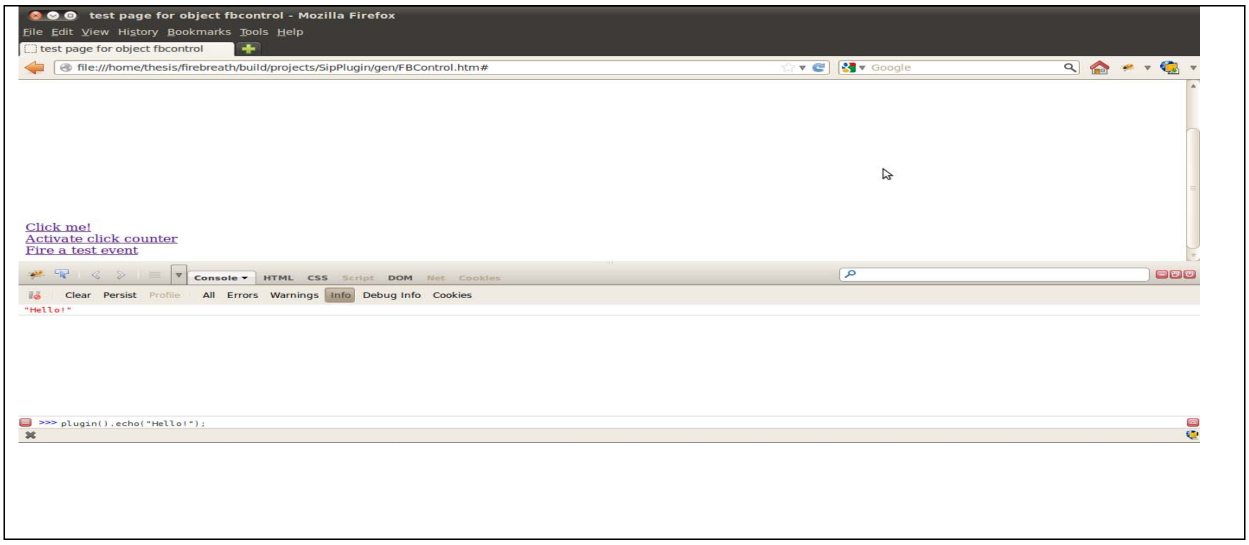
Now open the file *build/projects/SipPlugin/gen/FBControl.htm* in the Firefox browser. If the plugin installed correctly it displays an alert box with "Plugin Loaded" message. The *FBControl.htm* is a test page generated when creating a plugin to test few basics functionalities of the plugin. This page is shown in Figure-3-2.


test page for object fbcontrol - Mozilla Firefox
File Edit View History Bookmarks Tools Help
test page for object fbcontrol
file:///home/thesis/firebreath/build/projects/SipPlugin/gen/FBControl.htm#
Google
OK
Click me!
Activate click counter
Fire a test event

```
Plugin loaded!
```

Figure-3-2: FireBreath test page to check installed plugin

If the browser is installed with Firebug [6] or with the use of the web console in the browser, JavaScript calls can be made to the plugin. For example, Figure-3-3 shows the output of the JavaScript echo command.


test page for object fbcontrol - Mozilla Firefox
File Edit View History Bookmarks Tools Help
test page for object fbcontrol
file:///home/thesis/firebreath/build/projects/SipPlugin/gen/FBControl.htm#
Google
Click me!
Activate click counter
Fire a test event
Console HTML CSS Script DOM Net Cookies
Clear Persist Profile All Errors Warnings Info Debug Info Cookies

```
"Hello!"
>>> plugin().echo("Hello!");
```

Figure-3-3: Making Javascript calls on plugin using Firebug console

The MiniSIP Plugin was developed using the fbgen.py tool as described above. The resulting NPAPI plugin is called *npSipPlugin.so*, the installed plugin was shown in Figure 3-1.

3.2 Adding library dependencies to the plugin project in FireBreath

The development of the MiniSIP Firefox Plugin depends on the host having the MiniSIP libraries listed in section 2.4 installed. The FireBreath plugin framework can add these library dependencies to the plugin project. The library dependencies are added in the platform specific projectDef.cmake file [31]. For the Linux platform, the projectDef.cmake file is located in the X11 folder inside the plugin project folder. For example, the library dependencies for our plugin project can be added in the projectdef.cmake file under the option as shown below:

```
# add library dependencies here; leave ${PLUGIN_INTERNAL_DEPS}
target_link_libraries(${PROJECT_NAME}
${PLUGIN_INTERNAL_DEPS}
"/home/thesis/projects/SipPlugin/build/bin/SipPlugin/libmcrypto.so.0.0.0"
"/home/thesis/projects/SipPlugin/build/bin/SipPlugin/libmikey.so.0.0.0"
"/home/thesis/projects/SipPlugin/build/bin/SipPlugin/libmutil.so.0.0.0"
"/home/thesis/projects/SipPlugin/build/bin/SipPlugin/libmnetutil.so.0.0.0"
"/home/thesis/projects/SipPlugin/build/bin/SipPlugin/libmsip.so.0.0.0"
"/home/thesis/projects/SipPlugin/build/bin/SipPlugin/libmstun.so.0.0.0"
"/home/thesis/projects/SipPlugin/build/bin/SipPlugin/libminisip.so.0.0.0"
```

3.3 Design of MiniSIP Firefox plugin and its web-based GUI

A simple Firefox plugin was created using the FireBreath plugin framework as described in section 3.1. The MiniSIP Firefox Plugin added the MiniSIP as described in section 3.2. The MiniSIP Firefox plugin implements the basic SIP functionalities, such as registering a SIP user agent and making calls using the user's SIP accounts from a web-based GUI. To support this MiniSIP Firefox plugin, a web based GUI was developed using JavaServer pages (JSP). JavaServer Pages (JSP) technology provides a simplified, rapid way to create dynamic web content. JSP enables the rapid development of web-based applications that are both server and platform independent [32]. The web-based JSP graphical user interface of the MiniSIP Firefox plugin is shown in Figure-3-4.

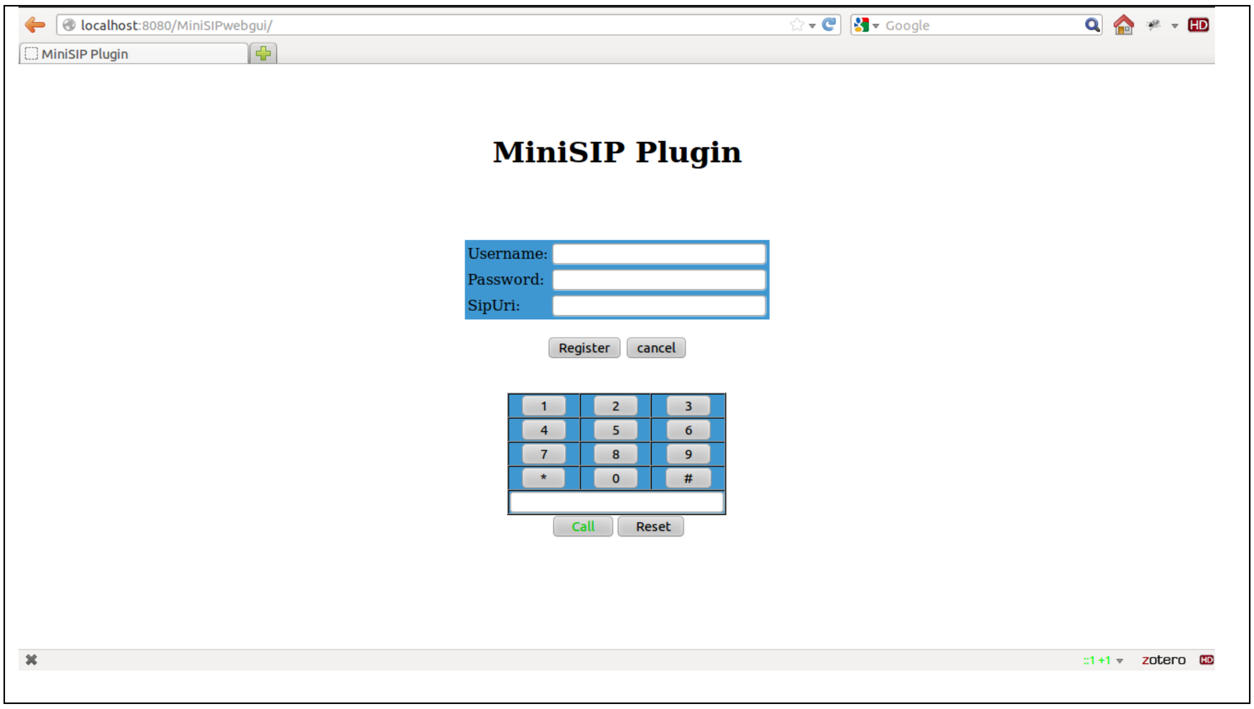


Figure-3-4: JSP graphical user interface of MiniSIP Firefox plugin

The Web-based JSP GUI is running as a service inside a web server (such as Apache Tomcat) or in an application server (such as GlassFish [33] – further described below). In Figure-3-4 we can see that this GUI was running as a service page (*MiniSIPwebgui*) inside the web server, which runs on the default 8080 port of the local Linux machine (as this is the port commonly used by such local web servers). To provide the plugin's JSP GUI, the GlassFish application server was used. GlassFish is a cross platform open source application server for Java EE platform developed by Oracle [34]. The web-based JSP GUI was written as a combination of HTML and JavaScript functions. HTML is used to create the graphical components of the interface and JavaScript functions are used to assign and control the features of the graphical components of the interface.

For example, the <*call*> button was created using HTML and assigned the call function using JavaScript to pass this request into the plugin. An example is shown in Figure 3-5 and Figure 3-6.

```
  <input style="color:#00CC00" type=button value="     Call    "onClick="pluginExecute();
window.focus()">
```

Figure 3-5: Sample JSP code: HTML code

```
JavaScript function:
  function pluginExecute()
  {     phone_num = document.board.pn.value;
  var phn= "call " + phone_num;
 plugin().pluginExecute(phn);
}
```

Figure 3-6: Sample JSP code: JavaScript function

This section elaborates the design and implementation of various graphical components of the web-based JSP graphical user interface. The JSP graphical user interface can be divided into three categories based on their design. As shown in Figure-3-4, the plugin GUI has two major components: (1) a register dialog, (2) a dialpad, and (3) the call dialog that pops-up once the call button is clicked.

3.3.1 Register Dialog

The components of the register dialog are shown in Figure-3-7. This dialog has three input boxes and two buttons.

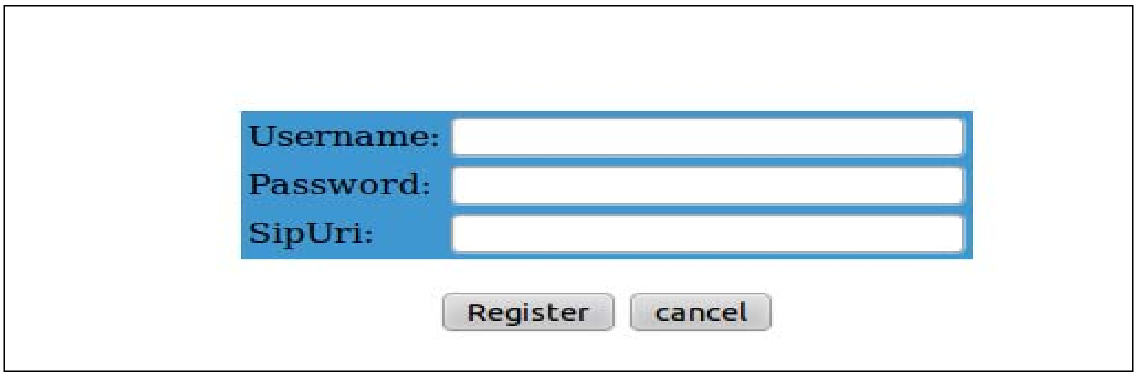


Figure-3-7: User Register Dialog

Via this dialog the user inputs three important credentials (<*Username*>, < *Password*>, <*SipUri*>) into the input boxes in the register dialog in order to register their user agent using an existing SIP account. When the user clicks the <*Register*> button, the Register dialog JavaScript function receives the *Username, Password, and SipUri* from the user and passes this information to the plugin. The MiniSIP Firefox plugin utilizes the SIP stack functions of MiniSIP in order to register this user agent with the corresponding SIP Registrar Server. If the user made any mistake or wants to reset the credentials entered in the input boxes, they simply click the <*cancel*>button. This button will reset all the values in the (<*Username*>, < *Password*>, <*SipUri*>) input boxes of the register dialog. The format for <*SipUri*> is **sip:username@sip.example.com.**

3.3.2 Dialpad and Call Dialog

The Dialpad part of the JSP graphical user interface contains the basic Dialpad component required in a SIP user agent and also contains the Dial *(SIP URI / Phone Number)* input box along with <*Call*> and <*Reset*> button. Figure 3-8 shows the Dialpad of the MiniSIP Firefox Plugin JSP GUI.

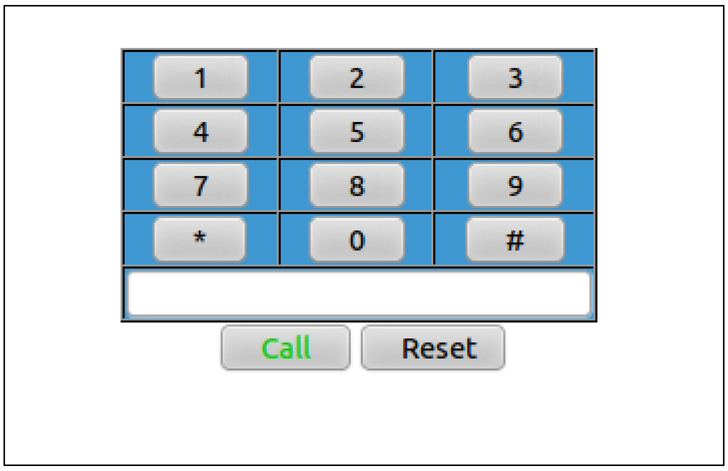


Figure 3-8: Dialpad of MiniSIP Firefox plugin

If a user clicks on any number, this number will be shown in the Dial input box. The user can even type the number or SIP URI directly into the input box. Once the user enters the phone number or SIP URI into the Dial input, they can make a call via MiniSIP Firefox plugin by clicking the <*Call*> button. The call button's JavaScript function gets the value entered in the Dial input box and passes it to the plugin. The plugin uses the call module and the SIP stack of MiniSIP to initiate the call by sending a SIP INVITE message to the remote user through the SIP Proxy server. If the user wants to cancel or reset the phone number or SIP URI entered in the Dial input box, the user simply clicks the <*Reset*> button. This will reset the all the values in the Dial input box.

When the user enters a phone number or SIP URI into the Dial input and then clicks the <*Call*> button, this action brings up the Call Dialog as shown in Figure 3-9. This dialong displays the phone number or SIP URI that was typed into the Dial input along with a <*hangup*> button to terminate the call.

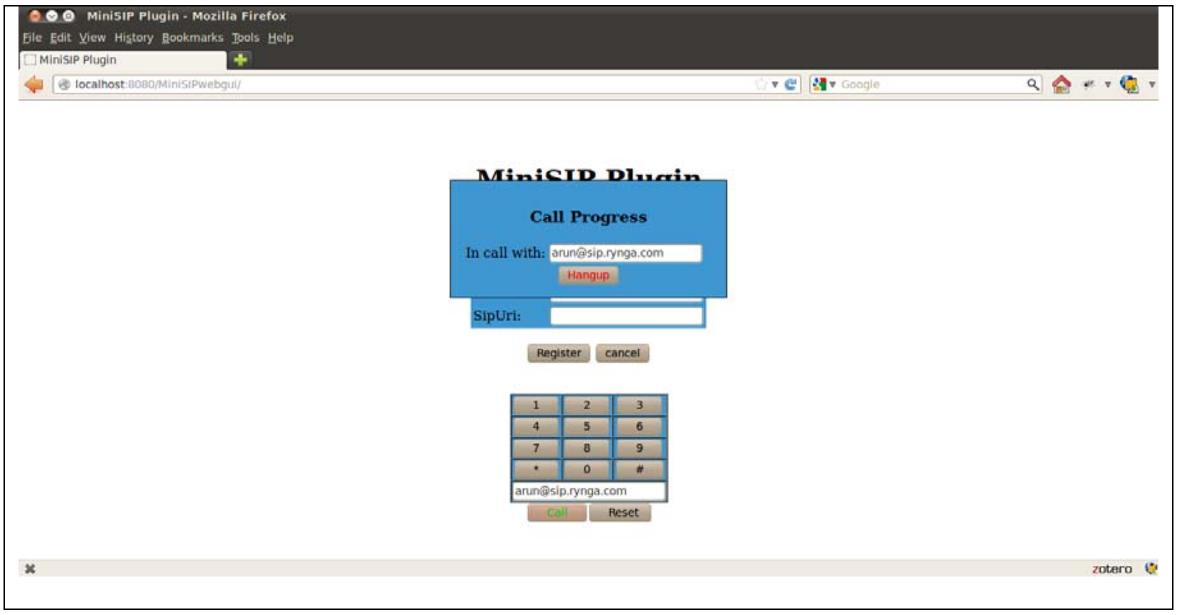

MiniSIP Plugin - Mozilla Firefox
File Edit View History Bookmarks Tools Help
MiniSIP Plugin
localhost:8080/MiniSIPwebgui/
MiniSIP Plugin
Call Progress
In call with: arun@sip.rynga.com
Hangup
SipUri:
Register cancel
1 2 3
4 5 6
7 8 9
* 0 #
arun@sip.rynga.com
Call Reset
zotero

Figure 3-9: Call Dilaog of MiniSIP Firefox plugin

The <*hangup*> button ends the call and closes the Call Dialog. The Call Dialog was designed using the modal dialog concept of HTML. This modal dialog uses a combination of CSS and JavaScript [35]. JavaScript only has three different dialog boxes, they are prompt (), alert (), and Confirm box (). Therefore, in order to design the call interface the modal dialog was used.

3.4 MiniSIP Firefox Plugin data-flow diagram:

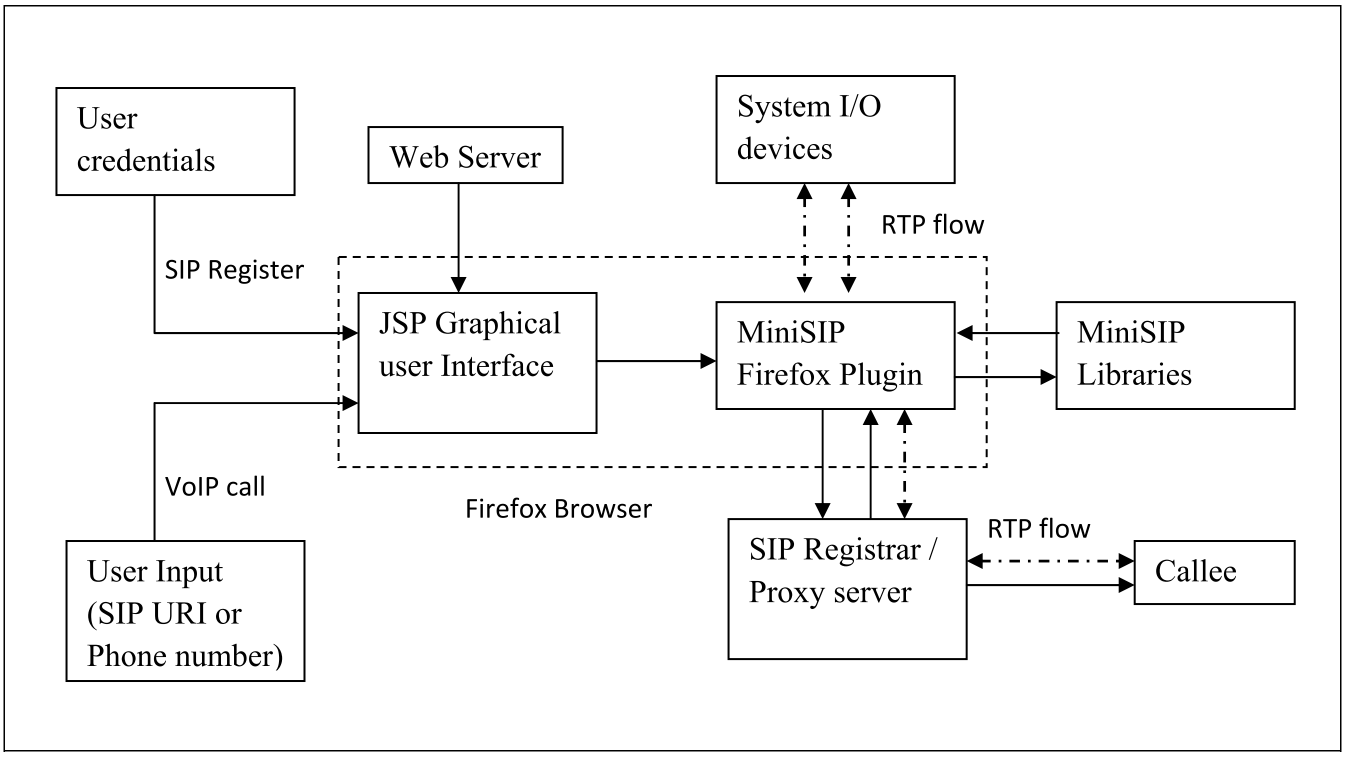


Figure 3-10: MiniSIP Firefox Plugin data-flow diagram

4 Analysis

This chapter explains the various comparisons and analyses done of the MiniSIP Firefox plugin. The chapter begins with a comparison of the plugin and GTK+ GUIs. This is followed by a comparison between Mizutech's Mizu webphone and MiniSIP Firefox Plugin. The chapter concludes with two sections that compare the call setup-delay of the MiniSIP application with the MiniSIP Firefox plugin and the one-way RTP delay of the MiniSIP application versus the one-way RTP delay of the MiniSIP Firefox plugin.

4.1 *Comparison between the plugin GUI and the GTK+ GUI MiniSIP*

As mentioned in the problem statement (section 1.1 on page 1), the MiniSIP Firefox plugin was developed mainly to overcome the problems of MiniSIP's GTK+ GUI. The GTK+ library is a cross-platform toolkit for creating a GUI. In MiniSIP, the Glade user interface designer was used to design the GUI components. Glade is a rapid application development (RAD) tool that is used to enable quickly & easily development user interfaces to be implemented with the GTK+ toolkit. The resulting interfaces are saved as XML files that are used to access the code that adds controls to the GUI [36]. The complications of developing a GUI using the GTK+ and Glade are their dependencies on gtkmm and libglademm libraries (as these libraries are required while running the GUI). Additionally, the resulting GUI is likely to crash more often than a handwritten GUI. A lot of programming is need before the user can add an interface to the GUI. Experience over the last decade has shown that using the GTK+ library and the Glade design interface tool makes the MiniSIP GUI too complicated for commercial use and too time consuming for a normal user to build from sources.

An example of the MiniSIP GTK+ GUI is shown in Figure 4-1.

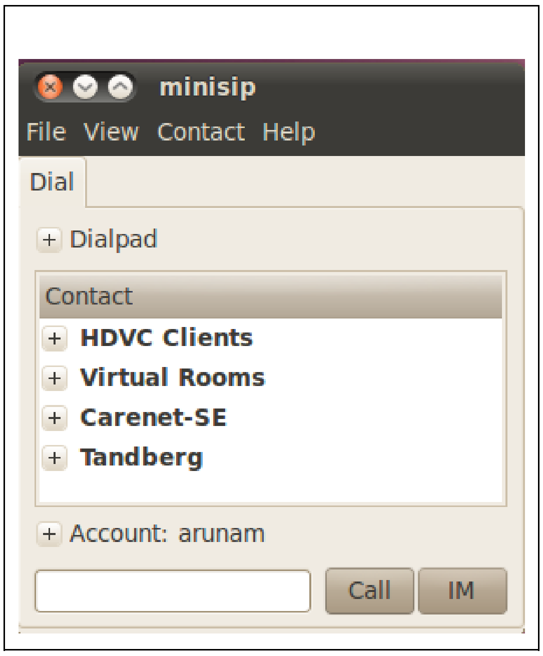


Figure 4-1: GTK graphical user interface of MiniSIP

As mentioned in the section 3.3, the MiniSIP Firefox plugin GUI was developed using JavaServer pages (JSP). The plugin JSP web-based GUI is a based upon a combination of HTML and JavaScript. HTML is used to create graphical components and JavaScript is used to add control functionality to these graphical components. The web-based JSP graphical user interface is run as service inside a web server, hence the MiniSIP Firefox plugin simply connects to port 8080 of the web server and requests the page named *MiniSIPwebgui* to invoke the user interface of the plugin. The major advantage for the user is that they simply install the MiniSIP Firefox plugin in their Linux machine and the GUI will be provided by the web server as a service, thus eliminating all of the complications of the rendering and development of the GUI components. Since all the GUI components are inside the web server; this reduces the need to load the GUI components and eliminates its dependence on GUI libraries. This is the major advantage of using a JSP web-based GUI and is expected to make the MiniSIP Firefox plugin light weight and user friendly for users. Additionally, other developers can easily create new web-based interfaces for the same underlying MiniSIP libraries.

4.2 Comparison between Mizutech's Mizu webphone and MiniSIP Firefox Plugin

The Mizutech's Mizu webphone is a lightweight VoIP softphone, which can be embedded into any webpage. A brief description about Mizutech's Mizu webphone and its features were given in section 2.5.6 on page 12. This softphone is implemented as a Java applet and it can run in any java-enabled browser and it is platform independent. The Mizu-webphone adds VoIP capabilities to any website and is easy to deploy. The user copy-pastes the HTML code into their website and changes the VoIP server's address in the code. This enables users of the website to directly call from their browser using the webphone hosted by the web server of the host website. The user can customize their webphone by changing the applet parameters or using the JavaScript API[18]. Even the user interface can be customized fully using web-based user interface tools such as .Net, PHP, etc.

The Mizu-webphone is available for trial as a demo-package with limited features and usability, but even with this demonstration version users can make VoIP calls to other softphones, mobile phones, and the PSTN. The full featured Mizu-webphone can be purchased for a license fee of US$500. Figure 4-2 shows the GUI of the Mizu-webphone.

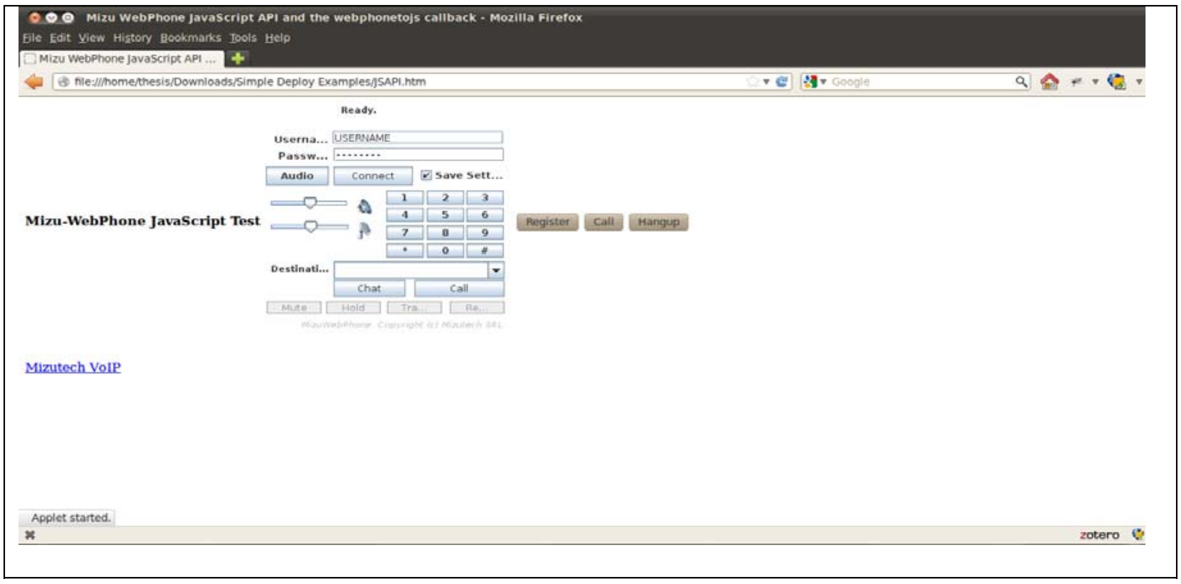

Mizu WebPhone JavaScript API and the webphonetojs callback - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Mizu WebPhone JavaScript API ...
file:///home/thesis/Downloads/Simple Deploy Examples/JSAPI.htm
Ready.
Userna... USERNAME
Passw... --------
Audio
Connect
Save Sett...
Mizu-WebPhone JavaScript Test
1 2 3
4 5 6
7 8 9
* 0 #
Register
Call
Hangup
Destinati...
Chat
Call
Mute
Hold
Tra...
Re...
MizuWebPhone Copyright (c) Mizutech SRL
Mizutech VoIP
Applet started.
zotero

Figure 4-2: Graphical user interface of Mizu-Webphone

Unlike the Mizu-webphone, the MiniSIP Firefox plugin was developed using the open-source MiniSIP libraries and its user interface was developed using JSP. Java applet has some restrictions such as dependencies on the Java Runtime Environment (JRE) and using an unsigned Java Applet can cause difficulties to accomplish some tasks [37]. The MiniSIP Firefox plugin is designed so that the users only need to install the plugin in their web-browser. Then when they wish to make a call they simply connect to the web server to utilize the GUI. The users register their SIP UA to their SIP server by giving the required information. User can utilize their incoming SIP server and they are able to make calls to any softphone, mobile phone, or PSTN connected phone from their web-browser without *any* restrictions with regard to usability, although they may have to pay charges of their calls to mobile and fixed telephones. Users can make any number calls until they close the webpage, at which they point they are de-registered with their SIP registrar. The main purpose of the MiniSIP Firefox plugin was to design a simple open source VoIP plugin with a user friendly GUI to make the MiniSIP application more suitable for commercial users.

4.3 Analysis and comparison of the call setup-delay between the MiniSIP application and the MiniSIP Firefox plugin

In this section, the call set up delay of the MiniSIP application and the MiniSIP Firefox plugin are measured and compared. The basic call setup in both utilized the SIP protocol and the same SIP messages are exchanged during the call set up. These messages are shown in Figure 4-3.

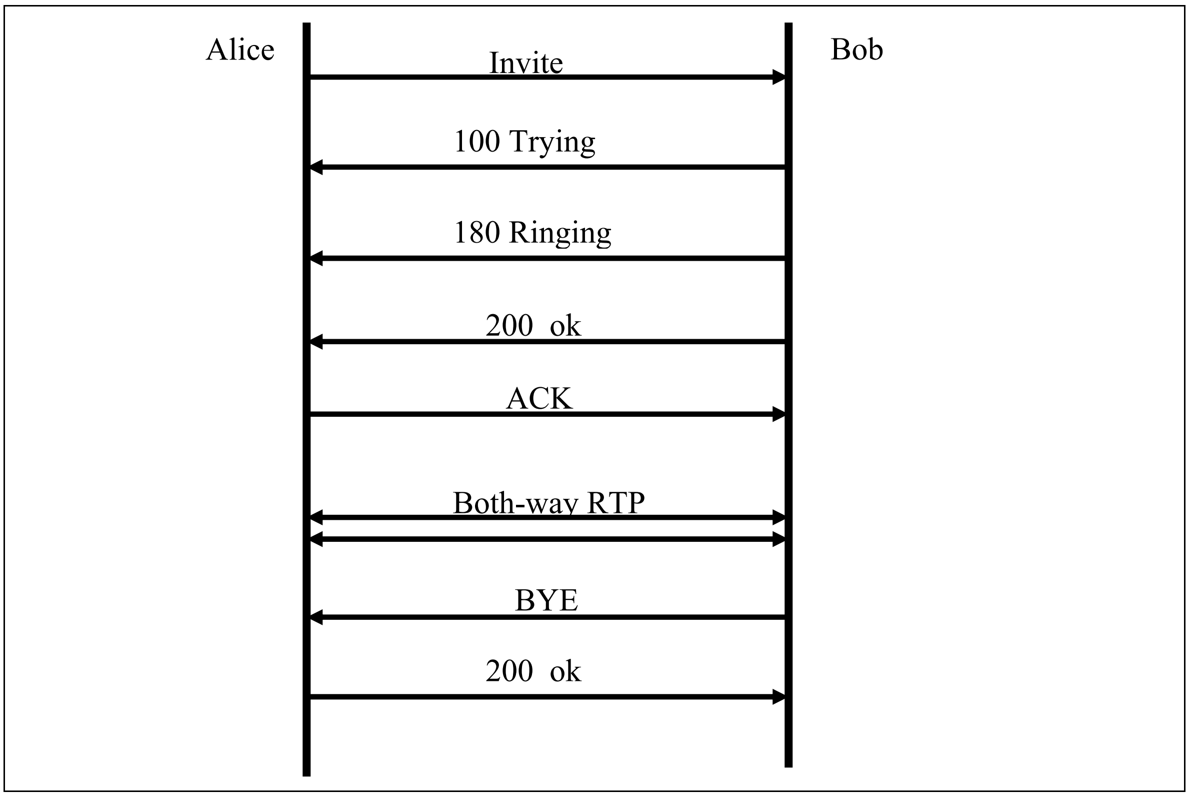


Figure 4-3: SIP Protocol messages exchanged during a call setup

In Figure 4-3 we can see that five SIP messages (Invite, 100 Trying, 180 Ringing, 200 OK, and ACK) are exchanged between the caller and the callee before the media (carried by RTP) starts flowing between the participants in the SIP session. The call set-up delay, also known as the Post Dial Delay (PDD), will be the time from sending the **Invite** message to receiving the first ringing response (e.g., **180 Ringing**) [38]. The call setup delay is the sum of the delays of the **Invite, 100 Trying,** and **180 Ringing** messages.

An experiment is conducted by making calls from the MiniSIP application and from the MiniSIP Firefox plugin both running on a Linux machine running the Ubuntu operating system version 10.04 Lucid Lynx to a mobile number by using an already registered SIP account from the VoIP service provider Rynga [39]. All of the testing was done with a Studio 1555 model Dell laptop computer. The experiment was conducted with the caller's computer connected to the KTH's eduroam wireless network. Wireshark was used to capture the traffic [40]. Forty (40) experiments are done separately of the MiniSIP application and the MiniSIP Firefox plugin.

Figure 4-4 shows the PDD or call setup delay (in seconds) of the MiniSIP application for experiment. The total delay is the sum of **Invite, 100 Trying,** and **180 Ringing** SIP messages for the calls made from the MiniSIP application to the mobile number. The maximum PDD is 0.35 seconds, the minimum PDD 0.31 seconds, and the median PDD for the MiniSIP application was 0.345 seconds. The variance and standard deviation of these measurements are 6.17E-05 seconds and 0.00785 seconds respectively.

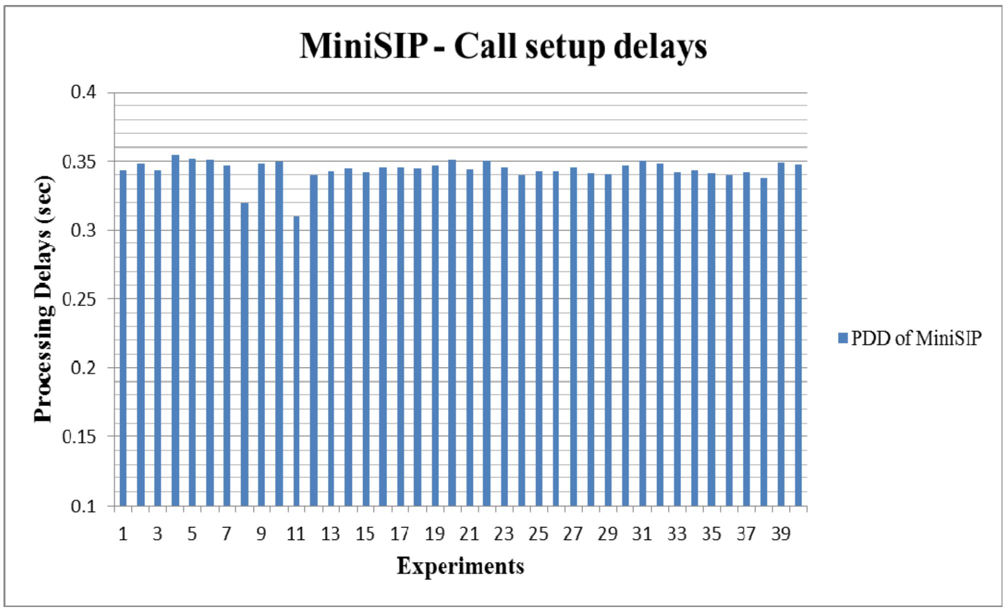


Figure 4-4: Call setup delay of MiniSIP VoIP application

In a similar way Figure 4-5 shows the PDD or call setup delay (in seconds) of the MiniSIP Firefox plugin using the same experimental setup. The maximum PDD was 0.55 second, the minimum PDD was 0.48 seconds, and the median PDD of the MiniSIP Firefox plugin was 0.535 seconds. The variance and standard deviation of these measurements are 0.00024 seconds and 0.0156 seconds respectively.

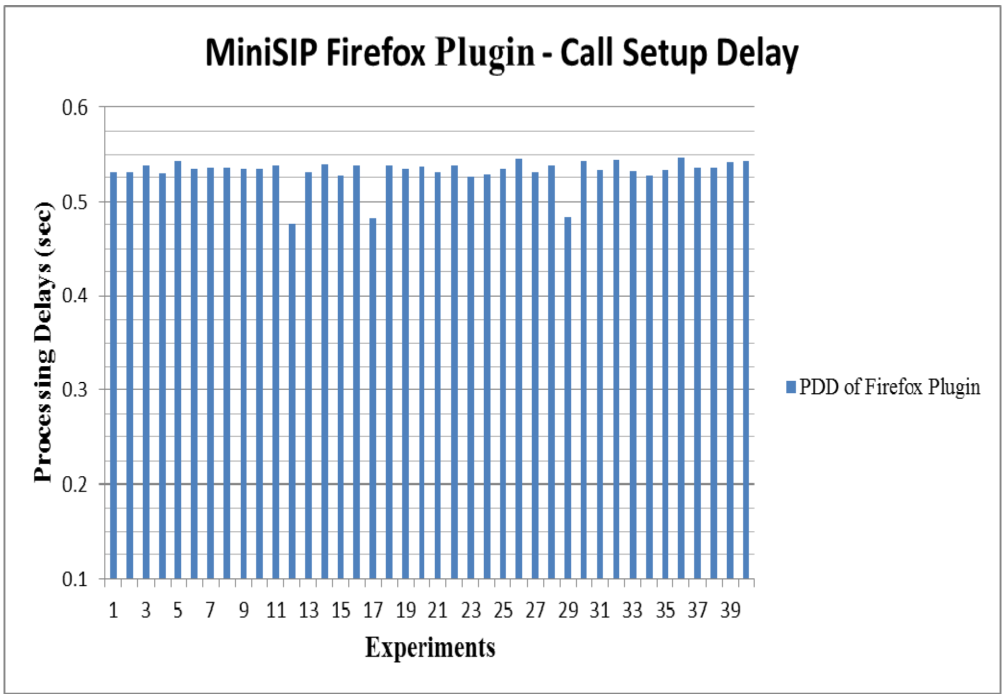


Figure 4-5: Call setup delay of MiniSIP Firefox plugin

Figure 4-6 depicts the PDD or call setup delay between the MiniSIP application and the MiniSIP Firefox plugin. Of these two sets of 40 experiments most of the time, the MiniSIP Firefox plugin has a higher delay compare to the MiniSIP application. It is evident that use of the web-front adds some additional delays to the call set up. However, the median and variance calculations show that is only 0.19 seconds of additional delay (see Figure 4-7). The actual measurements for all of these experiments are given in Appendix C.

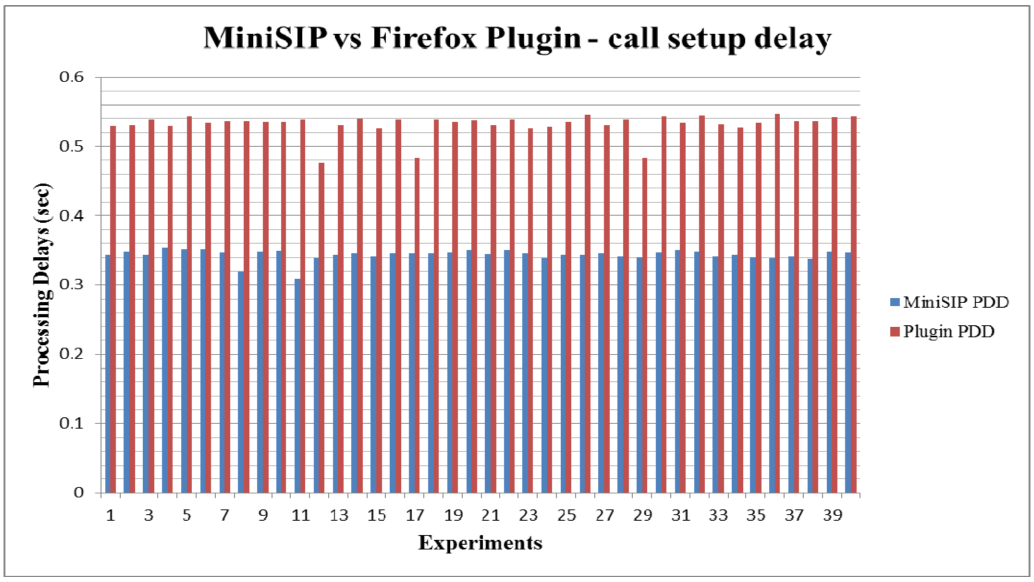


Figure 4-6: Comparison between call setup delay of MiniSIP and Firefox Plugin

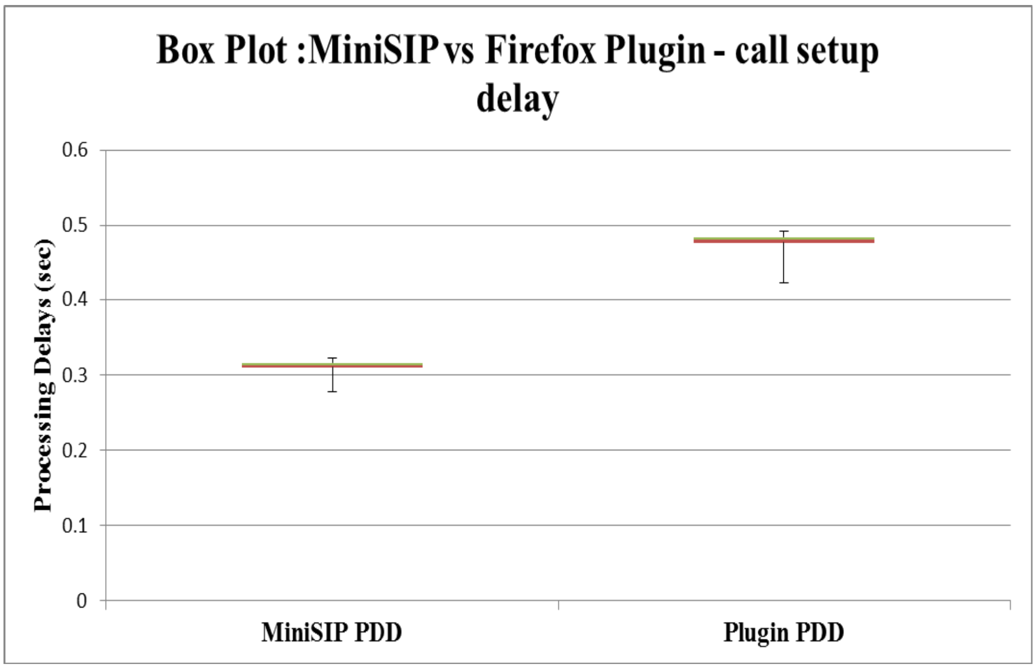


Figure 4-7: Box Plot - Comparison between call setup delay of MiniSIP and Firefox Plugin

4.4 Analysis and compare the one-way RTP delay between MiniSIP application and the MiniSIP Firefox plugin

An experiment was done using the same experimental setup as used in section 4.3 to measure and analyze the one-way RTP delay of both the MiniSIP application and the MiniSIP Firefox plugin. This analysis is done to check whether the use of web-front leads to greater delay as compare to the standalone application. Again forty (40) experiments (i.e. calls) we made using both the MiniSIP application and the MiniSIP Firefox plugin. The one way RTP delay is calculated using the difference in time between the first RTP packet and the last RTP packet of a single session send from the source IP address to the destination IP address. The RTP packets were captured using Wireshark.

Figure 4-8 shows the one way audio delay or RTP delay (in milliseconds – ms) of calls made between the MiniSIP application and the mobile number. The maximum one way RTP delay is 0.5 ms, the minimum one-way RTP delay was 0.4ms, and the median one-way RTP delay for the MiniSIP application was 0.4 ms. The variance and standard deviation for these measurements are 0.0016 ms and 0.044 ms. In comparison the one-way audio delay or RTP delay (in ms) of calls made between the MiniSIP Firefox plugin and the mobile number are shown in Figure 4-9. The maximum one-way RTP delay was 0.5 ms, the minimum one-way RTP delay was 0.4 ms, and the median one-way RTP delay of the MiniSIP Firefox plugin was 0.4 ms. The variance and standard deviation for these measurements are 0.0019 ms and 0.044 ms. The actual measurements for all of these experiments are given in Appendix D.

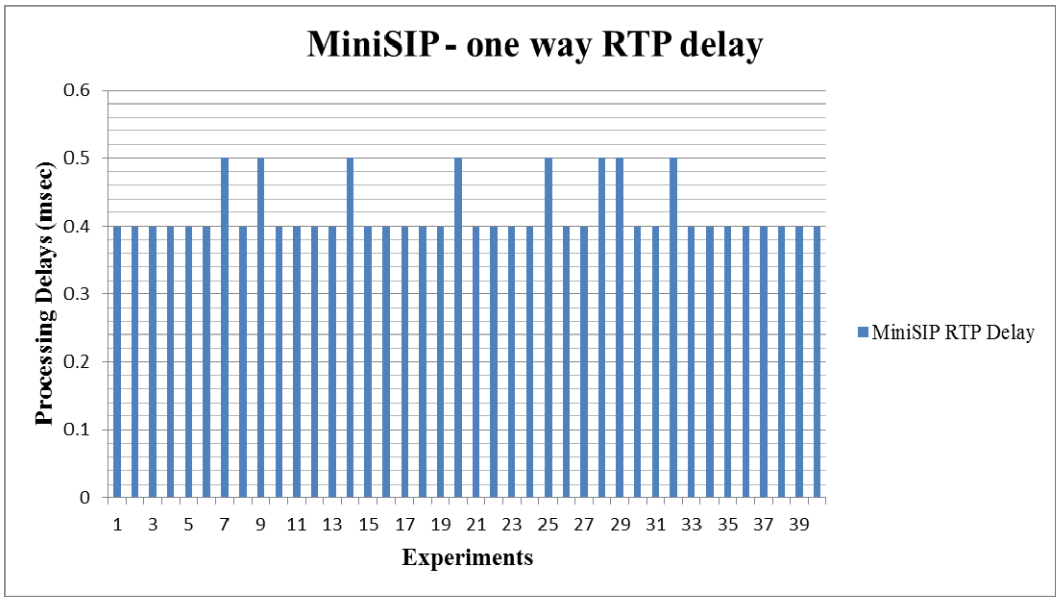


Figure 4-8: One way RTP delay of MiniSIP VoIP application

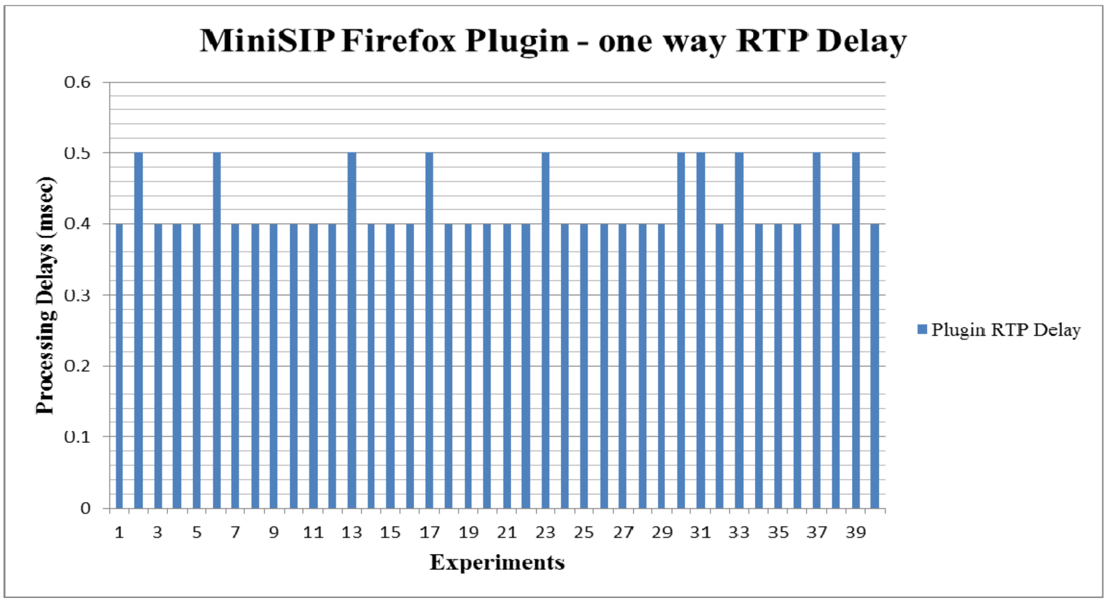


Figure 4-9: One-way RTP delay of MiniSIP Firefox plugin

A comparison between the one way audio delay or RTP delay (in msec) of the MiniSIP application and the MiniSIP Firefox plugin is shown in Figure 4-10. From this graph and the boxplot shown in Figure 4-11 it is evident that the one way audio delay (i.e. RTP delay) of the MiniSIP plugin is almost same as that of the MiniSIP application. The results from these experiments show that using the web-frontend causes no additional RTP delay compared to that of the native C++ implementation of MiniSIP.

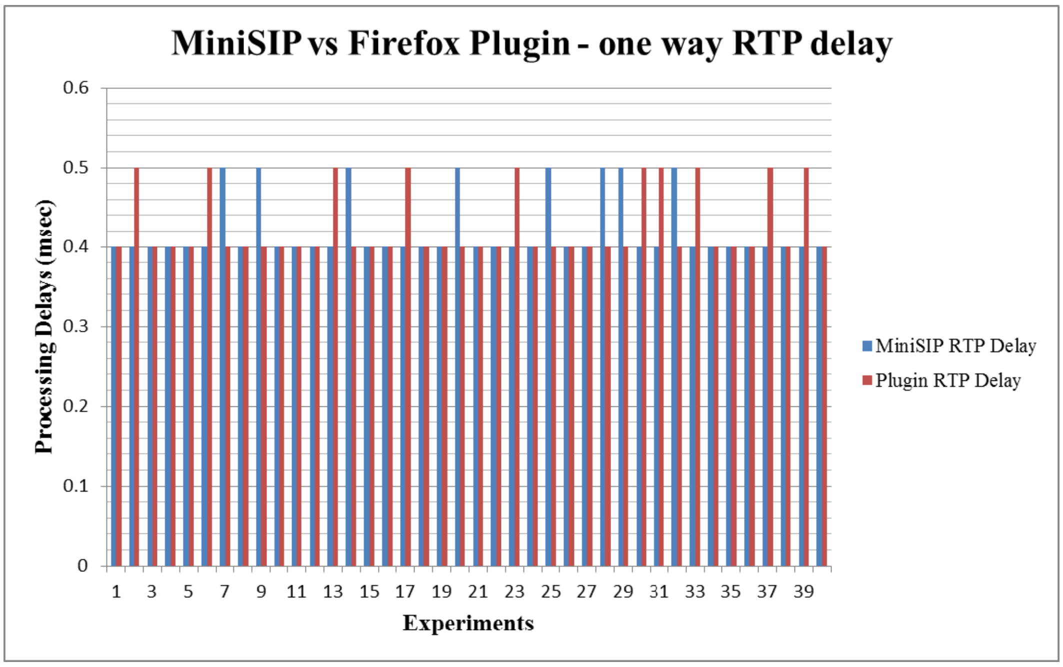


Figure 4-10: Comparison of one way RTP delay between MiniSIP and Firefox plugin

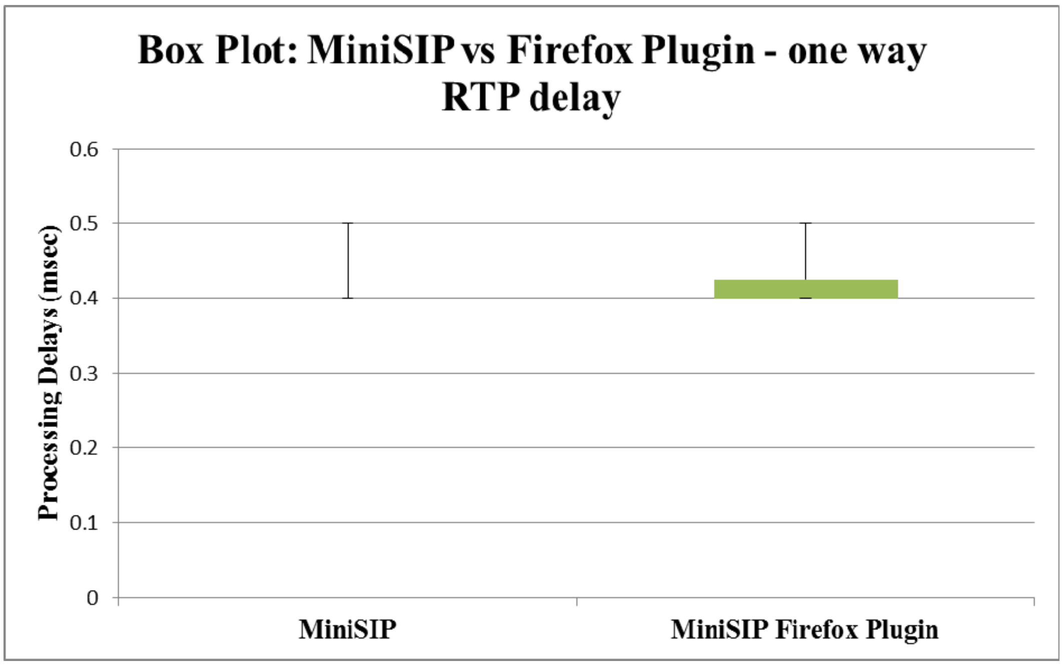


Figure 4-11: Box Plot - Comparison of one way RTP delay between MiniSIP and Firefox plugin

4.5 Comparison of installing and initiation time of MiniSIP as an application versus the new plugin

The time to install the MiniSIP application from source on the Ubuntu 10.04 Lucid Lynx system used for the testing described in the previous sections was approx. 12 minutes. The time to download and install a native binary of the application along with the browser plugin was approx. one minute. The initiation time of MiniSIP application on the Ubuntu 10.04 Lucid Lynx system was 30-40 seconds approx. The initiation time of MiniSIP Firefox plugin was 2-3 seconds, assuming that the web server is already running.

5 Conclusions and Future work

This chapter concludes the thesis offering a conclusion and suggesting some future research work. The chapter concludes with some reflections on the economic, social, and ethical aspects of this thesis project.

5.1 Conclusions

In this thesis, a new SIP based Firefox plugin was developed using the existing MiniSIP libraries to realize a web-based VoIP application. A prototype plugin was developed using the FireBreath plugin framework. The resulting plugin can be installed in any Linux machine. The MiniSIP Firefox plugin was installed and tested in several versions of the Ubuntu operating systems.

The performance of the MiniSIP Firefox plugin was tested by analyzing the call set-up delay and one-way audio delay and comparing these delays with that of native MiniSIP VoIP application. The main purpose of this thesis work is to make MiniSIP usable for commercial users. The major hindrance for such usage is the GTK+ graphical user interface of MiniSIP. A new web-based JSP graphical user interface was developed for the MiniSIP Firefox plugin. The user of the plugin does not have to face the earlier dependencies on a number of libraries in order to run the GUI. The GUI for the MiniSIP Firefox plugin is provided by the web server as a service. As a result all the complications of the GUI are removed from the user side of the application.

The MiniSIP Firefox plugin user simply installs the plugin in the Firefox browser and access the GUI by connecting to a local web server. The user's existing SIP accounts can be used to register the MiniSIP Firefox plugin as the user's SIP user agent. Users are able to make calls to other VoIP softphones and gateways to the PSTN directly from the webpage of their browser. The development of MiniSIP Firefox plugin with the web-based JSP graphical user interface makes the capabilities of MiniSIP VoIP applications readily available even for commercial users.

5.2 Future work

In the future the MiniSIP Firefox plugin could be equipped with facilities to handle user contact management. Using a web server, these services can be easily added and maintained. The plugin can readily utilize all of the security features of MiniSIP VoIP application – since it is using the same underlying MiniSIP libraries. For example it is easy to add features such as handling certificates during a call setup, using pre-shared keys, using Diffie-Hellman key agreements, etc. Also it is possible to implement MIKEY and SRTP protocols to protect media contents. The MiniSIP Firefox plugin does not currently have the ability to select the user's preferred input and output device(s), this could be added in the plugin and the JSP based GUI. Another major feature that should be added to the plugin is the ability to have an incoming call direct to the browser (this requires that the user start the web-based interface before the call arrives – but since the call will not arrive unless the plugin has registered this is not such a high barrier to implementation of this functionality). The MiniSIP libraries have support for high definition video conferencing; hence it would be great to add this feature to the MiniSIP Firefox plugin. In the future work additional work on the GUI, such as notifications of the need to re-register with the SIP registrar and error handling during sessions and SIP processing can be done. The MiniSIP Firefox plugin can equipped with the feature to re-register automatically with the SIP server using the user credentials after the expiry of the binding time period.

5.3 Required reflections

Providing an easy to install secure VoIP solution can greatly benefit society by increasing the security of the user's communication. In addition, when used together with public key certificates it is possible to authenticate the calling and the called party, thus increasing the trust that the caller and callee are who they claim to be. As this software is based on open source software the proposed solution has the advantage that it is both open (so that others can independently inspect the code and extend it) and it is freely available, hence this enables every user to have the advantage of high security for their VoIP communication – rather than this only being something that is only available to a limited number of users and even then only available at a high price. One ethical concern of this work is that it facilitates security for both honest users and for those who would use this technology in conjunction with illegal activities.

References

[1] Erik Eliasson, ‘Secure Internet telephony : design, implementation and performance measurements’, Licentiate thesis, KTH Royal Institute of Technology, School of Information and Communication Technology, Electronic, Computer and Software Systems (ECS), Stockholm, Sweden, Available at http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-4080, 2006.

[2] Steve Howard, ‘Plugins - MDN’, *Plugins - Mozilla Developer Network*, 05-December-2011. [Online]. Available: https://developer.mozilla.org/En/Plugins. [Accessed: 14-March-2012].

[3] The GTK+ Team, ‘The GTK+ Project’, *The GTK+ Project*. [Online]. Available: http://www.gtk.org/. [Accessed: 14-March-2012].

[4] Richard Day, ‘Browser Plugins vs Extensions – the difference | ColonelPanic’, 10-August-2010. [Online]. Available: http://colonelpanic.net/2010/08/browser-plugins-vs-extensions-the-difference/. [Accessed: 14-March-2012].

[5] Janet Swisher, ‘Gecko Plugin API Reference - MDN’, *Gecko Plugin API Reference - Mozilla Developer Network*, 10-August-2011. [Online]. Available: https://developer.mozilla.org/en/Gecko_Plugin_API_Reference. [Accessed: 14-March-2012].

[6] Mozilla, ‘What is Firebug? : Firebug’. [Online]. Available: http://getfirebug.com/whatisfirebug. [Accessed: 14-March-2012].

[7] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, ‘SIP: Session Initiation Protocol’, *Internet Request for Comments*, vol. RFC 3261 (Proposed Standard), June 2002, Available at http://www.rfc-editor.org/rfc/rfc3261.txt.

[8] Henry Sinnreich and Alan B Johnston, *Internet communications using SIP : delivering VoIP and multimedia services with Session Initiation Protocol*, Second ed. Indianapolis, IN: Wiley Pub., 2006, ISBN: 0471776572 9780471776574.

[9] Richard Bateman, ‘FireBreath Home - FireBreath - FireBreath Home Page’, *FireBreath Home*, 29-October-2011. [Online]. Available: http://www.FireBreath.org/display/documentation/FireBreath+Home. [Accessed: 15-March-2012].

[10] Mozilla Developer Network, ‘External resources for plugin creation - MDN’, *External resources for plugin creation - Mozilla Developer Network*, 01-March-2011. [Online]. Available: https://developer.mozilla.org/en/Plugins/External_resources_for_plugin_creation. [Accessed: 15-March-2012].

[11] Richard Bateman, ‘Features - FireBreath - FireBreath Home Page’, *Features*, 29-October-2011. [Online]. Available: http://www.FireBreath.org/display/documentation/Features. [Accessed: 15-March-2012].

[12] Tom Keating, ‘Firefox VoIP extension’, 08-September-2006. [Online]. Available: http://blog.tmcnet.com/blog/tom-keating/voip/firefox-voip-extension.asp. [Accessed: 15-March-2012].

[13] Tom Keating, ‘Zoep VoIP client with Firefox plugin and Jabber support’, 23-January-2006. [Online]. Available: http://blog.tmcnet.com/blog/tom-keating/voip/zoep-voip-client-with-firefox-plugin-and-jabber-support.asp. [Accessed: 15-March-2012].

[14] Luca Filigheddu, ‘Firefox VoIP Extension Open to Any VOIP Service | Tech Genial’, 13-December-2007. [Online]. Available: http://www.techgenial.com/2007/12/firefox-voip-extension-now-open-to-any-voip-service.html. [Accessed: 15-March-2012].

[15] iNet Telecoms Ltd, ‘Voipfone - Web Browser Dialler Plugin - For IE, Firefox and Google Chrome’. [Online]. Available: http://www.voipfone.co.uk/plugin_browser.php. [Accessed: 15-March-2012].

[16] sipgate, ‘sipgateFFX - the Firefox Add-on’. [Online]. Available: http://www.sipgate.com/faq/article/467/Using_sipgateFFX. [Accessed: 26-June-2012].

[17] NEXT - Solutions for a small planet, ‘WebX - your Online softphone’. [Online]. Available: http://nxtsolution.net/webxphone.php. [Accessed: 26-June-2012].

[18] Mizutech, ‘Mizutech - Mizu webphone’. [Online]. Available: http://www.mizu-voip.com/. [Accessed: 26-June-2012].

[19] Anthony Lieuallen, Aaron Boodman, and Johan Sundström, ‘Greasemonkey :: Add-ons for Firefox’. [Online]. Available: https://addons.mozilla.org/en-US/firefox/addon/greasemonkey/. [Accessed: 26-June-2012].

[20] VOIP-Info.org, ‘Asterisk click to call - voip-info.org’. [Online]. Available: http://www.voip-info.org/wiki/view/Asterisk+click+to+call. [Accessed: 01-July-2012].

[21] Mozilla Links, ‘OpenWengo extension for Firefox • Mozilla Links’, 23-February-2006. [Online]. Available: http://mozillalinks.org/2006/02/openwengo-extension-for-firefox/. [Accessed: 01-July-2012].

[22] Google, ‘Google Talk - Chat with family and friends’. [Online]. Available: http://www.google.com/talk/. [Accessed: 27-June-2012].

[23] P. Saint-Andre, Ed., ‘Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence’, *Internet Request for Comments*, vol. RFC 3921 (Proposed Standard), October 2004, Available at http://www.rfc-editor.org/rfc/rfc3921.txt.

[24] mozdev, ‘MozIAX - Firefox VoIP extension’. [Online]. Available: http://moziax.mozdev.org/. [Accessed: 27-June-2012].

[25] Yealink Network Technologies Co., LTD, ‘Yealink-Easy VoIP,SIP based IP Phone,IP Video Phone,Dect Phone,USB Phone,VoIP,Unified Communication,Skype Phone，Media Phone’. [Online]. Available: http://www.yealink.com/index.php/Products/lists/classid/3. [Accessed: 01-July-2012].

[26] 8x8, Inc., ‘Unified Communications - 8x8 Virtual Office Pro’. [Online]. Available: http://www.8x8.com/CommunicationsSolutions/ByProduct/VirtualOfficePro.aspx. [Accessed: 27-June-2012].

[27] voiptoday, ‘Switchvox Fire Dialer Extension’. [Online]. Available: http://www.voiptoday.org/index.php?option=com_content&view=article&id=390:switchvox-fire-dialer-extension&catid=40:soft-phone&Itemid=127. [Accessed: 26-June-2012].

[28] Skype, ‘Skype Click to Call’. [Online]. Available: http://www.skype.com/intl/en/get-skype/on-your-computer/click-to-call/windows/. [Accessed: 26-June-2012].

[29] Richard Bateman, “Creating a New Plugin Project - FireBreath - FireBreath Home Page,” *Getting Started*. [Online]. Available: http://www.FireBreath.org/display/documentation/Creating+a+New+Plugin+Project. [Accessed: 26-Sep-2012].

[30] Richard Bateman, “Building on Linux - FireBreath - FireBreath Home Page,” *Getting Started*. [Online]. Available: http://www.FireBreath.org/display/documentation/Building+on+Linux. [Accessed: 26-Sep-2012].
[31] Richard Bateman, “Mac Video Tutorial - FireBreath - FireBreath Home Page,” *Video tutorials*. [Online]. Available: http://www.FireBreath.org/display/documentation/Mac+Video+Tutorial. [Accessed: 02-Oct-2012].
[32]Oracle, “JavaServer Pages Technology,” *JSP*. [Online]. Available: http://www.oracle.com/technetwork/java/javaee/jsp/index.html. [Accessed: 21-Sep-2012].
[33] Oracle, “GlassFish Server,” *GlassFish Server - Overview*. [Online]. Available: http://www.oracle.com/technetwork/middleware/glassfish/overview/index.html. [Accessed: 10-Oct-2012].
[34] Oracle, “GlassFish Users and Application Developers — Java.net,” *GlassFish*. [Online]. Available: http://glassfish.java.net/public/getstarted.html. [Accessed: 27-Sep-2012].
[35] Raven Tools, “Create a Modal Dialog Using CSS and Javascript - Raven Internet Marketing Tools,” *Raven Blog*. [Online]. Available: http://raventools.com/blog/create-a-modal-dialog-using-css-and-javascript/. [Accessed: 28-Sep-2012].
[36] The Glade project, “Glade - A User Interface Designer,” *Glade*. [Online]. Available: http://glade.gnome.org/. [Accessed: 21-Sep-2012].
[37] Lapana.info, “Disadvantages of Java Applet | Lipana.info.” [Online]. Available: http://lipana.info/disadvantages-of-java-applet/. [Accessed: 02-Oct-2012].
[38] voip-info, “PDD - voip-info.org,” *Post Dial delay*. [Online]. Available: http://www.voip-info.org/wiki/view/PDD. [Accessed: 02-Oct-2012].
[39] Rynga, “Rynga | For the cheapest international calls.” [Online]. Available: http://www.rynga.com/. [Accessed: 02-Oct-2012].
[40] Gerald Combs, “Wireshark · About.” [Online]. Available: http://www.wireshark.org/about.html. [Accessed: 02-Oct-2012].

Appendix A: Sample Javascript functions

```
<script type="text/javascript">

var phone_num='';
var timerID = null;
var timerRunning = false;
function but1()
{

phone_num = phone_num + '1';
chknum();
}
function but2()
{
 phone_num = phone_num + '2';
 chknum();
}
function but3()
{
 phone_num = phone_num + '3';
 chknum();
}
function but4()
{
 phone_num = phone_num + '4';
 chknum();
}
function but5()
{
 phone_num = phone_num + '5';
 chknum();
}
function but6()
{
 phone_num = phone_num + '6';
 chknum();
}
function but7()
{
 phone_num = phone_num + '7';
 chknum();
}
function but8()
{
 phone_num = phone_num + '8';
 chknum();
}
function but9()
{
 phone_num = phone_num + '9';
 chknum();
}
function but0()
{
 phone_num = phone_num + '0';
 chknum();
}
function butasterisk()
{
 phone_num = phone_num + '*';
 chknum();
}
function buthash()
{
 phone_num = phone_num + '#';
 chknum();
}
function chknum()
{
document.board.pn.value=phone_num;
}
function plugin0()
{
  return document.getElementById('plugin0');
}
 plugin = plugin0;
function pluginExecute()
{
phone_num = document.board.pn.value;
var phn= "call " + phone_num;
plugin().pluginExecute(phn);
}
function clearit()
{
 plugin().pluginExecute('hangup');
 phone_num = "";
 document.board.pn.value=phone_num;
}
</script>
```

Appendix B: Sample HTML code

```
<body>
       <object id="plugin0" type="application/x-sipplugin" width="50" height="50"
align="center">
    <param name="onload" value="pluginLoaded" />
</object>
    </body>
       <center>
       <h1>MiniSIP Plugin</h1>
       </center>
          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
       <FORM METHOD=POST>
       <br><br><br>
       <center>
          <table BGCOLOR="#3E97D1">
<tr>      <td>   Username: </td><td>
       <INPUT TYPE=TEXT NAME=username Id="username" SIZE=20 /><br /></td>
          </tr><tr>   <td>     Password: </td>
      <td>   <INPUT TYPE=password NAME=password Id="password" SIZE=20 /><br
/></td>
      </tr><tr>  <td>   SipUri: </td>
      <td>   <INPUT TYPE=TEXT NAME=sipuri Id="sipuri" SIZE=20/><br/></td>
       </tr></table>
       <br />
       <input style="color:black" type="submit" value="Register"
onClick="register();window.focus()">
       <input style="color:black" type="reset"
value="cancel"onClick="resetregister();window.focus()">
       </center>
       </FORM>
       <br><br>
       <form name="board">
<center>
<table BGCOLOR="#3E97D1" BORDER CELLSPACING=0 CELLPADDING=0 COLS=4
WIDTH="135" >
<tr ALIGN=CENTER>
<td><input style="color:black" type=button value="    1    "onClick="but1();
window.focus()"></td>

<td><input style="color:black" type=button value="    2    "onClick="but2();
window.focus()"></td>

<td><input style="color:black" type=button value="    3    "onClick="but3();
window.focus()"></td>
</tr>
```

```
<tr ALIGN=CENTER>
<td><input style="color:black" type=button value="    4    "onClick="but4();
window.focus()"></td>

<td><input style="color:black" type=button value="    5    "onClick="but5();
window.focus()"></td>

<td><input style="color:black" type=button value="    6    "onClick="but6();
window.focus()"></td>
</tr>

<tr ALIGN=CENTER>
<td><input style="color:black" type=button value="    7    "onClick="but7();
window.focus()"></td>

<td><input style="color:black" type=button value="    8    "onClick="but8();
window.focus()"></td>

<td><input style="color:black" type=button value="    9    "onClick="but9();
window.focus()"></td>
</tr>

<tr ALIGN=CENTER>
<td><input style="color:black" type=button value="    *    "onClick="butasterisk();
window.focus()"></td>

<td><input style="color:black" type=button value="    0    "onClick="but0();
window.focus()"></td>

<td><input style="color:black" type=button value="    #    "onClick="buthash();
window.focus()"></td>
</tr>
<tr>
<td ALIGN=CENTER COLSPAN="3"><input type=text name=pn size="20"></td>
</tr>
<tr ALIGN=CENTER COLSPAN="3">
<center> <table BORDER=0 CELLSPACING=0 CELLPADDING=0 COLS=2 >
   <td ALIGN=CENTER WIDTH="50%"><input style="color:#00CC00" type=button
value="    Call    "onClick="pluginExecute(); window.focus()"></td>
   <td ALIGN=CENTER WIDTH="50%"><input style="color:#FF0000" type=button
value="    Hangup  "onClick="clearit(); window.focus()"></td>
</tr>
 </table></center>
</Form>
```

Appendix C: Call set up delay measurements

Table C-1: PDD measurements of MiniSIP

Invite	100 Trying	180 Ringing	PDD
0.0838	0.1107	0.1489	0.3434
0.0859	0.1182	0.1437	0.3478
0.0837	0.1151	0.1446	0.3434
0.0832	0.1198	0.1512	0.3542
0.0879	0.1173	0.1464	0.3516
0.0817	0.1142	0.1553	0.3512
0.0805	0.1174	0.1485	0.3464
0.0726	0.1099	0.1365	0.319
0.0842	0.1148	0.1492	0.3482
0.0876	0.1189	0.1429	0.3494
0.0725	0.1045	0.1327	0.3097
0.0875	0.1108	0.1417	0.34
0.0839	0.1163	0.1425	0.3427
0.082	0.1179	0.1454	0.3453
0.0823	0.1181	0.1419	0.3423
0.0842	0.1149	0.1467	0.3458
0.0809	0.1167	0.1482	0.3458
0.081	0.1093	0.1547	0.345
0.0832	0.1143	0.1486	0.3461
0.0841	0.1172	0.1492	0.3505
0.0854	0.1107	0.1482	0.3443
0.0848	0.1168	0.1486	0.3502
0.0872	0.1149	0.1438	0.3459
0.0877	0.1054	0.1465	0.3396
0.0822	0.1092	0.1512	0.3426
0.0884	0.1114	0.1427	0.3425
0.0846	0.1163	0.1447	0.3456
0.0852	0.1069	0.1492	0.3413
0.0851	0.1084	0.1474	0.3409
0.0886	0.1109	0.1471	0.3466
0.0894	0.1127	0.1483	0.3504
0.0809	0.1172	0.1498	0.3479
0.0827	0.1098	0.1492	0.3417
0.087	0.1115	0.1451	0.3436
0.0835	0.1089	0.1486	0.341
0.0804	0.1109	0.1484	0.3397
0.0832	0.1096	0.1491	0.3419
0.083	0.1087	0.1463	0.338
0.0881	0.1104	0.1499	0.3484
0.0892	0.1151	0.1428	0.3471

Table C-2: PDD measurements of MiniSIP Firefox Plugin

Invite	100 Trying	180 Ringing	PDD
0.1417	0.1781	0.2108	0.5306
0.1432	0.1704	0.2178	0.5314
0.1487	0.1761	0.2139	0.5387
0.1418	0.1732	0.2145	0.5295
0.1454	0.1787	0.2191	0.5432
0.1439	0.1769	0.2134	0.5342
0.1463	0.1726	0.2168	0.5357
0.1456	0.1782	0.2118	0.5356
0.1475	0.1751	0.2122	0.5348
0.1459	0.1737	0.2147	0.5343
0.1455	0.1783	0.2151	0.5389
0.1272	0.1505	0.1989	0.4766
0.1456	0.1747	0.2111	0.5314
0.1491	0.1758	0.215	0.5399
0.1404	0.1724	0.2142	0.527
0.1462	0.1794	0.2133	0.5389
0.1273	0.1578	0.1976	0.4827
0.1487	0.1763	0.2137	0.5387
0.1471	0.1756	0.2118	0.5345
0.1452	0.1798	0.2122	0.5372
0.1443	0.1721	0.2143	0.5307
0.1439	0.178	0.2162	0.5381
0.1406	0.1742	0.2117	0.5265
0.1418	0.1734	0.2138	0.529
0.1423	0.1757	0.2169	0.5349
0.1492	0.1789	0.2177	0.5458
0.1432	0.1739	0.2138	0.5309
0.1483	0.1751	0.2145	0.5379
0.1259	0.1592	0.1981	0.4832
0.1494	0.1776	0.2169	0.5439
0.1428	0.1781	0.2122	0.5331
0.1469	0.1797	0.2174	0.544
0.1428	0.1746	0.2147	0.5321
0.1419	0.1723	0.2129	0.5271
0.1437	0.1753	0.2148	0.5338
0.1482	0.1795	0.2188	0.5465
0.1425	0.1777	0.2157	0.5359
0.1473	0.1728	0.2155	0.5356
0.1486	0.1762	0.2173	0.5421
0.1457	0.1784	0.2187	0.5428

Appendix D: RTP delay measurements

Table D-1: One way RTP delay Measurements

MiniSIP	MiniSIP Firefox Plugin
0.4	0.4
0.4	0.5
0.4	0.4
0.4	0.4
0.4	0.4
0.4	0.5
0.5	0.4
0.4	0.4
0.5	0.4
0.4	0.4
0.4	0.4
0.4	0.4
0.4	0.5
0.5	0.4
0.4	0.4
0.4	0.4
0.4	0.5
0.4	0.4
0.4	0.4
0.5	0.4
0.4	0.4
0.4	0.4
0.4	0.5
0.4	0.4
0.5	0.4
0.4	0.4
0.4	0.4
0.5	0.4
0.5	0.4
0.4	0.5
0.4	0.5
0.5	0.4
0.4	0.5
0.4	0.4
0.4	0.4
0.4	0.4
0.4	0.5
0.4	0.4
0.4	0.5
0.4	0.4

TRITA-ICT-EX-2012:247