
# Characterization of SIP Signaling-Messages Over OpenSIPS Running On Multicore Server

**By:**

Naser Saeed Awan

**Industrial Supervisor**

Dr. Iyad Al Khatib

**Examiner:**
Professor Ahmed Hemani

**Department of Information and Communication Technology**
**Royal Institute of Technology (KTH)**

Stockholm, Sweden.

## Abstract

Over the course of last decade, the demand for VoIP (Voice over Internet Protocol) applications has increased significantly among enterprises and individuals due to its low cost. This increasing demand resulted in a significant increase in users who require reliable VoIP communication systems. QoS (Quality of Service) is a major issue in VoIP implementation and is a method to impel the development of real-time multimedia services like VoIP and videoconferencing. However, there are certain challenges in achieving QoS for VoIP application, which need special attentions; like latency and packet loss.

The VoIP servers which are functioning on single core software/hardware model have high latency and packet loss issues due to their limited processing bandwidth. A multicore software/hardware model is the solution to cope up with the increasing demands of VoIP and yet an active research area in telecommunication. Using a multicore software/hardware model for VoIP has several challenges, one of the challenges is to design and implement QoS Benchmarking module for VoIP client and server on multicore.

In this thesis the focus is on latency and packet loss of SIP messages on OpenSIPS server. This is done by performing stress testing for QoS benchmarking, where delay and call drop rate is calculated for SIP (Session Initiation Protocol) signaling messages on parallel VoIP client server model. The model is built in C for multicore and is used as a simulation tool. SIP is widely deployed protocol for call establishment, maintenance and termination in VoIP.

**Keywords:** *Delay, Latency, OpenSIPS, Packet Loss, QoS, QoS Benchmarking, SIP Signaling Messages, Videoconferencing, VoIP*

## Dedications

I owe my humble thanks to ALLAH for the strength to keep me standing and for the hope to believe that this project is possible. I present my heartiest gratitude to everyone who supported me in making this project a glorious experience, especially to Muhammad Badawi a Phd student at KTH (The Royal Institute of Technology).

I present my special thanks to my supervisor Dr. Iyad Al Khatib for his advice and support during this project. Special thanks to my thesis examiner Professor Ahmed Heimani for his support and patience.

Last but not the least I pay many thanks to my parents and family who gave me not only financial, but also moral and spiritual support whenever I needed.

## Glossary

**D**
**DBMS:** Database Management System
**DNS:** Domain Name System
**DPKG:** Debian Package

**H**
**HTTP:** Hypertext Transfer Protocol

**I**
**IRTP:** Internet Reliable Transaction Protocol
**IP:** Internet Protocol
**IP PBXs:** Internet Protocol Private Branch Exchanges

**L**
**LAMP:** Linux, Apache, Mysql and PHP

**O**
**ODBC:** Open Database Connectivity
**OS:** Operating System
**OpenSIPS CP:** OpenSIPS Control Panel

**P**
**PSTN:** Public Switched Telephone Network

**Q**
**QoS:** Quality of Service

**R**
**RAS:** Registration, Admission and Status
**RDBMS:** Rational Database Management System
**RTP:** Real Time Transport Protocol

**S**
**SCTP:** Stream Control Transmission Protocol
**SIP:** Session Initiation Protocol
**SLA:** Service Level Agreement
**SMTP:** Simple Mail Transfer Protocol

**T**
**TCP:** Transport Layer Protocol

**U**
**UDP:** User Datagram Protocol
**UAC:** User Agent Client
**UAS:** User Agent Server

**V**
**VoIP**: Voice over Internet Protocol

## Table of Contents

## List of Tables



## List of Figures

# 1. Introduction

The basic problem in VoIP is that the VoIP server gets populated very quickly, which increase latency and call drop rate. This increase in latency and call drop is due to the VoIP servers which are based on single-core. The problem arrives when the host server is busy with the client requests and more requests arrive from new clients. At such a stage the server starts to drop the calls [1]. There are processors available in the market having clock speed up to 3.0GHz i.e.; 3 billion clock ticks per second. On the other hand its utilization is far more less than its capability, therefore, we propose VoIP server utilizing multi core processors as a solution. Another main concern in VoIP is the bandwidth, which is not only required to connect the call but it is also a way to provide good quality of service.

The goal of this project is the analysis of SIP signaling messages for VoIP streams in a network. The focus of the work is on the following main issues:

i. Testing QoS parameters on the existing model, developed by other teams of the project [2]
ii. Testing QoS parameters for SIP signal messages
iii. Designing of clients to function with a multicore apparatus.

The result is a working client prototype that interacts with the server and other clients for whom we need to understand the following issues:

i. Understanding VoIP and QoS for IP networks
ii. Designing and implementing the QoS test modules, packets, and scenarios to work on multicore
iii. Implementing the test modules using C/C++ over ubuntu
iv. Verification of the benchmark module

The aim is to make analysis of SIP signaling message to find out bottlenecks. The focus is on multicore and tried to find out ways by which call delays/drops can be decreased. Comprehensive tests are performed during network idle state and by making 50% of network bandwidth busy by using network utility tool Iperf.

## 1.1 Overview

Quality of Service is an elusive term which has no set definition; every user has his/her own QoS requirements. VoIP users expect that the services provided by the VoIP should be equivalent to traditional public switched telephone network (PSTN). Technically, video and voice quality over IP network depends on minimum delay and loss of packets. QoS is a collective effect of service performance which determines the degree of satisfaction by the users of the service [3]. Therefore basic components of QoS are the requirements of the users, which are offered by the provider [4].

VoIP encompasses protocols, technologies and techniques designed for the transmission of multimedia sessions and voice communication by IP network. VoIP data like audio, video can be transferred over Internet between clients and servers by using different VoIP protocols. The VoIP protocol stacks namely H.323, SIP, MEGACO and MGCP are derived by standard bodies like ITU-T, IETF, and some vendors [5].

In this thesis SIP (Session Initiation Protocol) is used, which is an application-layer protocol and the basic functionality of this protocol is handling signaling mechanism; like creating, modifying and terminating sessions between users/hosts or clients. SIP is designed as an independent protocol which can run on TCP, UDP or SCTP.

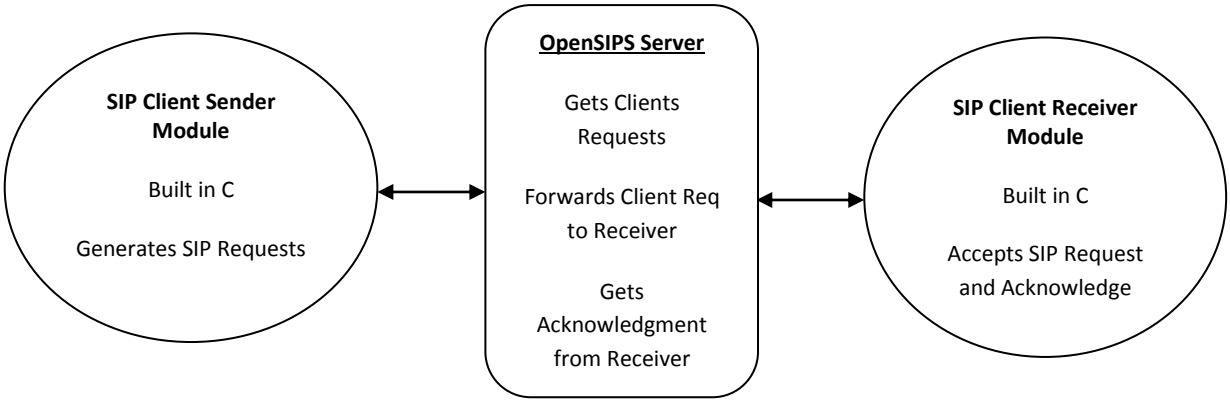



Figure 1: SIP Based Request Response Scenario

**This thesis is in continuation with research work for development of VoIP application over multicore. There are several There are several teams working in this research work, as elaborated in**

Figure 1 the client sender and receiver module is built in C developed by co-teams. The main contribution in this thesis is installation, configuration of OpenSIPS server and analysis of sender/receiver requests on OpenSIPS server. This thesis is performed to set benchmark of QoS for SIP signaling messages over multicore, so there is a need of repetitive tests to make analysis. The test pattern of user selection to perform tests is entirely different from the pattern of users other groups selected.

## 1.2 Organization of thesis

Chapter 1 briefly describes about background, problems addressed, the purpose, and the goals of this thesis. Chapter 2 covers in detail the thesis background and topics which are required to understand for designing of test environment for QoS Benchmarking. Chapter 3 presents requirements and Implementation of the work, which elaborates the basic requirements for the thesis and explains way to configure OpenSIPS server in order to perform tests. Chapter 4 depicts experimental setup, limitations and results which we obtained from tests. Chapter 5 concludes the research and suggests future work.

## 2. Technical Background

IP (Internet Protocol) itself doesn't provide any guarantee for QoS, so network providers face performance issues that bring call delay and drop. Service providers need real-time call quality statistics, which is very hard to achieve because Internet services are not constant.

In order to work on QoS in VoIP applications that can run on multicore servers, there is a need to know about QoS Benchmarking, VoIP protocol suite and parallel methods. These issues are explained in sections below:

### 2.1 QoS Benchmarking

Benchmarking is a standard, or a set of standards, used as a point of reference for evaluating performance or level of quality [7].

In computers; benchmarking is a standard test to measure performance of hardware or software. The purpose of a benchmark is to evaluate the performance and functionality of different systems.

QoS Benchmarking is helpful to analyze the performance and find out where improvements are required.

### 2.2 VoIP Protocol Suite

VoIP facilitates real time data for communication (like audio/video) between two or more user over Internet using Internet Protocol. There are number of VoIP protocol stacks namely H.323, BICC, SIP, MEGACO and MGCP are derived by standard organizations like ITU-T, IETF, and some vendors [5]. These signaling protocols are involved at different layers to establish connection for VoIP users.

The Table 1 shows the number of VoIP protocols recommended by different standard bodies [5].

| Standard Bodies and Vendors | Protocols |
|---|---|
| **ITU-T** | H.323: Packet-based multimedia communications(VoIP) architecture<br>H.225: Call Signaling and RAS in H.323 VOIP Architecture<br>H.235: Security for H.323 based systems and Communications<br>H.245: Control Protocol for Multimedia Communication<br>T.120: Multipoint Data Conferencing Protocol Suite |
| **IETF** | Megaco / H.248: Media Gateway Control protocol<br>MGCP: Media Gateway Control Protocol<br>RTSP: Real Time Streaming Protocol<br>SIP: Session Initiation Protocol<br>SDP: Session Description Protocol<br>SAP: Session Announcement Protocol |
| **Cisco Skinny** | SCCP: Skinny Client Control Protocol |
| **Media/CODEC** | G.7xx: Audio (Voice) Compression Protocols<br>(G.711, G.721, G.722, G.723, G.726, G.727, G.728, G.729)<br>H.261: Video Coding and Decoding (CODEC)<br>H.263: Video Coding and Decoding (CODEC)<br>RTP: Real Time Transport Protocol<br>RTCP: Real Time Transport Control Protocol |
| **Others** | COPS: Common Open Policy Service<br>SCTP: Stream Control Transmission Protocol<br>TRIP: Telephony Routing Over IP |

**Table 1: VoIP Protocols recommended by standard bodies**

The widely used protocol due to its popularity among IP community is SIP along-with H.323. SIP protocol uses SDP (Session Description Protocol) for call control mechanism, while the same task is handled by two different protocols in three different stacks by H.323. Due to this simplicity and ease in call control mechanism in SIP, it is considered an important protocol. SIP is recommended by IETF while H.323 protocol is recommended by ITU-T [8], [9]. SIP is designed to provide VoIP call setup protocol of IP network. Just like HTTP and SMTP protocols, SIP is a text based protocol designed to manage multimedia sessions over the internet.

In this thesis tests are performed by using SIP protocol, as explained above SIP uses Session Description Protocol for call control mechanism, while the same is handled by two different protocols in three different stacks by H.323. Figure depicts how these two protocols work in VoIP Protocol Suite [9].

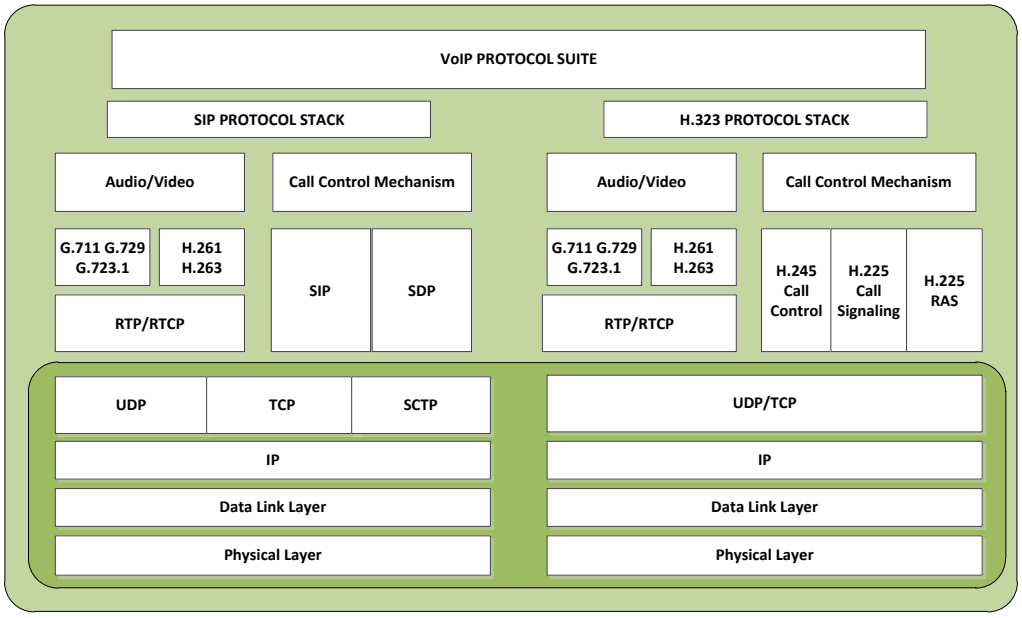



Figure 2: VoIP Protocol Suite

Figure 2 shows that there are two main parts in VoIP Protocol Suite. The upper part elaborates signaling mechanism, that is, how Call Set up is done while the lower part shows the communication stream area or media stream, that is, once a call is established by using signaling protocol then how real time data like audio, video or other media stream starts transmission.

In SIP protocol stack call control mechanism is handled by SIP and SDP, on the other hand in H323 protocol stack, call control is done by H.245, signaling mechanism is handled by H.255 and RAs by H.255. For multimedia stream audio video codec are same in both protocols.

The main focus in this thesis is on signaling mechanism as other teams are still working on communication of real time data streams by using IRTP.

## 2.3 SIP (Session Initiation Protocol)

SIP is an application layer control protocol for creating, modifying and terminating sessions with one or more participants. SIP is one of the most well known standard ways of establishing Internet telephone calls, multimedia distribution and multimedia conferences between two parties, multi-party or multicast sessions. SIP is an independent protocol and can function with TCP (*Transmission Control Protocol*), UDP *(User Datagram Protocol)* or SCTP (*Stream Control Transmission Protocol*) [10].

SIP uses port number "5060" behind which there is a server that processes all the SIP requests so that the clients can be connected by establishing a session. This session can be established by exchanging the address information through a referenced address required by the protocol.

SIP is a text based protocol and uses the following six messages.

1. ***INVITE***: Caller sends an invite message to callee
2. ***ACKNOWLEDGE***: Callee on receiving Invitation sends ACK message for confirmation
3. ***BYE***: This message terminates a session
4. ***OPTIONS:*** This message checks the capability of machine by sending query message
5. ***CANCEL:*** The cancel message terminates an initialized process
6. ***REGISTER:*** This message makes a connection when the callee is not available

**How does SIP find addresses**? Like in PSTN (*Public Switched Telephone Network*) a telephone number is basic identity from which one can call at other numbers. SIP is a flexible protocol and it can identify users by a SIP address, IP address, a telephone number or any type of address. However, there is a basic format of SIP by which this address should be written like as shown in Figure 3

| Sip: *username*@192.168.0.10 | Sip: *username* @gmail.com | Sip: *username* @467-00-312 |
|---|---|---|
| IPv4 address | Email address | Phone number |

Figure 3 SIP Addressing Schemes

SIP functionality is limited and is used to setup, modify and terminate a session. There are four major purposes for which SIP is used. These are: [11]

1. To establish a connection in a way that SIP translates the user's ID into the network address currently in use
2. SIP takes care of managing sessions among the participants, so that they can acknowledge the features to be backed up among them
3. SIP derives out the way to manage the calls in all respect. It may be call adding, dropping or transferring services among participants
4. SIP can also change the features of session where a call may be in progress

These features show that SIP is neither a conference control protocol nor a session description protocol. It is not a resource reservation protocol and it has nothing to do with QoS, it is only used for session setup, modification and termination.

There are two components of SIP.

1. UAC (*User Agent Client*): It generates requests and sends these requests to servers.
2. UAS (*User Agent Server*): It gets requests, processes those requests and generates responses. There are three different servers that perform number of activities involved in handling requests generated by UAC. The servers are: Proxy server which can be state full or state less, Register server and Redirect server.

### 2.3.1 Basic Communication Method of SIP

SIP follows client-server paradigm as used widely in Internet by protocols like HTTP or SMTP. Figure 4 shows typical exchange of request and response. This scenario discusses the typical call functionality of SIP and this thesis ignores all other possible scenarios like "call failed", "user busy" or "dropped" etc.

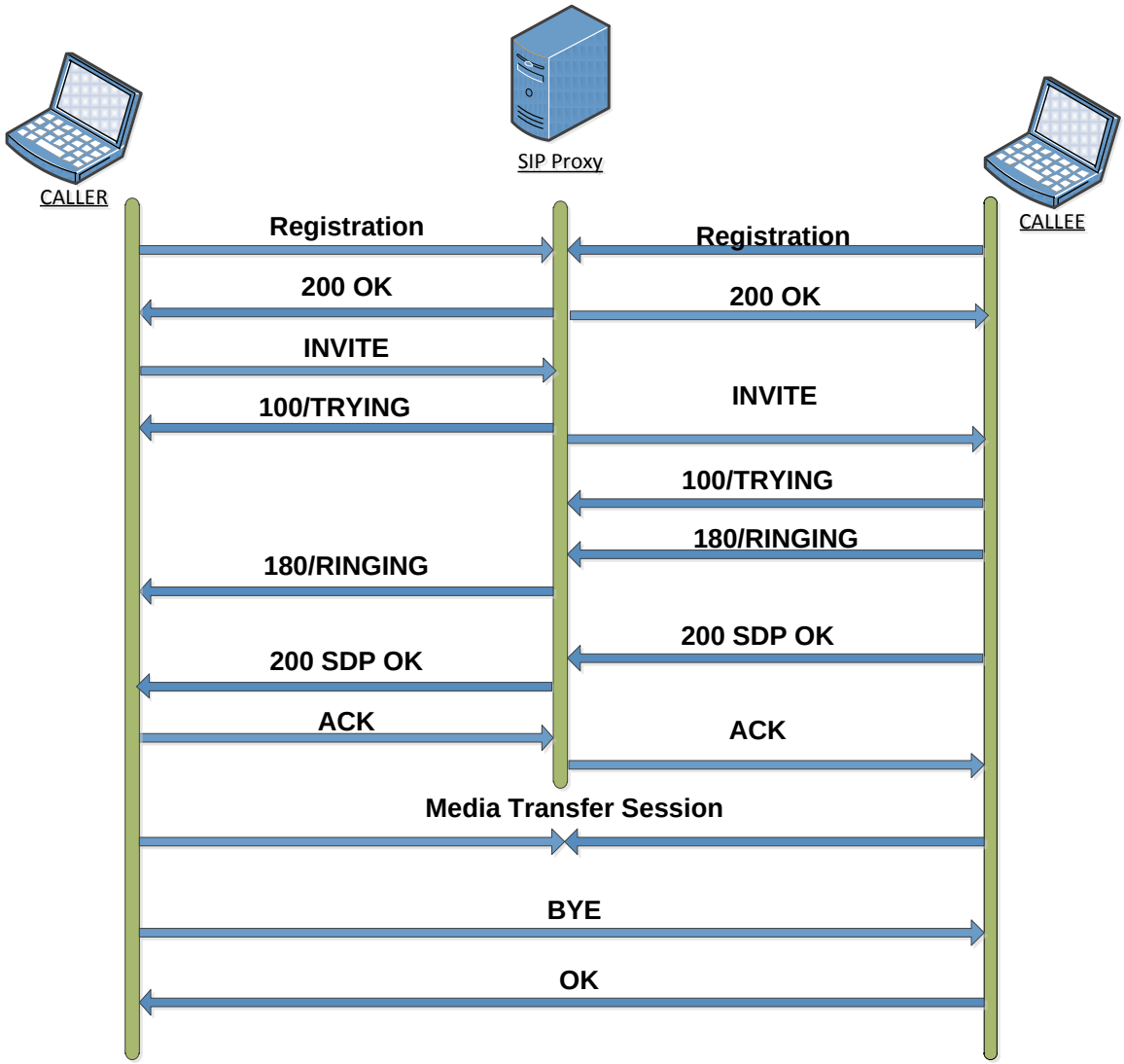



Figure 4: SIP Message Types and Flow

User Registration process is pre-requisite in VoIP communication in which a VoIP User Agent (UA), which is usually a VoIP client software, sends a registration request to VoIP server. The server registers the user and stores the entry in database. On successful registration the server will send 200 OK messages in response.

A User Agent starts call setup by sending INVITE message containing the target User Agent username and SIP proxy server IP address. If the target UA is many hops away, INVITE may be routed through a number of proxy servers until it reaches to its destination. Destination UA sends a 100 TRYING and 180 RINGING message, meanwhile it bundles the multimedia support information in SDP header and sends it back to source UA in 200 OK with SDP message. The source UA checks the SDP parameters and acknowledges it by sending ACK message. On receiving ACK by destination UA, the session is established and voice/video session is initiated by RTP or IRTP.

The session will end with a BYE message which can be sent by any party (either Caller or Callee) to terminate the session. On receiving BYE message at the end point 200 OK message will be passed on to confirm session termination.

## 2.4 Open Source SIP Server

SIP uses client/server model, on client side the user applications like soft phones are used as SIP client e.g. Skype, Ovoo, xlite, Ekiga etc. Other teams are working on the development of soft phone instead of using already available soft phones.

On server side there are number of open source implementations of SIP but two of them are most important that gained worldwide acceptance and growth. These are Asterisk [12] and OpenSIPS [13].

### 2.4.1 Asterisk

Asterisk [12] is a well known open source server which includes IP PBXs and VoIP gateways. Asterisk is released under the GNU General Public License (GPL).

Asterisk provides all features offered by a classical PBX system, such as making simple telephone calls, call transfer, voice mail, call conferencing and voice menu and other features also [12].

### 2.4.2 OpenSIPS

OpenSIPS is an open source proxy server [13] compliant with the IETF RFC3261 SIP protocol. OpenSIPS is very fast in forwarding requests and can handle thousands of users with a single server. OpenSIPS is not simply a SIP proxy server as it can have more application level functionalities. It is being used by both large VoIP providers and embedded IP PBXs with very low processing power.

OpenSIPS is a multi-functional, multi-purpose SIP server, Sip proxy, access point and registrar [18]. OpenSIPS is installed and configured on Ubuntu operating system for the project. OpenSIPS server is managed by OpenSIPS-CP (OpenSIPS Control Panel) which is a graphical user interface based tool.

## 2.5 Apache Server

The Apache [23] HTTP Server referred to as Apache is a web server. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently known as Sun Java System Web Server), and has since evolved to rival other Unix-based web servers in terms of functionality and performance.

In our project Apache is configured for OpenSIPS control Panel, but due to time constraint, management of SIP users is left for future work.

## 2.6 Iperf

Iperf [15] measures TCP and UDP bandwidth performance and runs in a client-server mode in a network. Iperf allow users to set many parameters like udp maximum size segment, tcp window size, time, bidirectional traffic, port and interval. Iperf application is used in this thesis as a helping tool to generate network traffic while performing SIP stress tests.

## 2.7 Mysql

OpenSIPS uses Mysql to store client account information. Mysql (“My S-Q-L” OR “My sequel” is a relational database management system (RDBMS) which has more than 6 million installations. Mysql stands for "My Structured Query Language". The program runs as a server providing multi-user access to a number of databases [17].

Mysql is commonly used by free software projects which require a full-featured database management system, such as Word Press, phpBB and other software built on the LAMP (Linux, Apache, Mysql and PHP) software stack. It is also used in high-scale social World Wide Web products including Face book and Google [17].

## 2.8 Wireshark

Wireshark [18] is a free packet sniffer; it is used for network troubleshooting and analysis of communication protocols. Originally it was named as Ethereal, but due to trademark issues the name was changed later to Wireshark.

Wireshark understands the structure of different networking protocols. Thus, it is able to display the encapsulation and fields along-with their meanings of different packets specified by different networking protocols. Wireshark uses pcap to capture packets, so it can only capture packets on the networks supported by pcap [18].

Wireshark is used to trace and capture SIP packets that were transmitted between server and clients.

# 3. Requirements and Implementation

The goal of this project is the Analysis of VoIP streams over access point at signal level. The focus of the work is on the following main issues:

1. Testing QoS parameters on OpenSIPS server
2. Testing QoS parameters on access points
3. Design for the client to function with a multicore server

Extensive tests in order to find out delays and bottlenecks in project. There is a need to configure OpenSIPS server and connected it with an access point through LAN so that stress testing can be performed.

Below are the basic requirements for the project:

## 3.1 Requirements

The basic software and hardware which are required in this project are as follows:

1. **OS:** Ubuntu is the main operating system which is used in this project
2. **Database**: MYSQL 5
3. **OpenSIPS**: 1.6.4.0
4. **Apache:** 2
5. **CPU:** Dual Core 2.1 processor
6. **RAM:** 2GB
7. **LAN:** Wide LAN 100Mbps
8. **Sender Receiver:** SIP traffic Generator
9. **Wireshark:** For SIP traffic capture and monitoring

OpenSIPS server needs to be configured on a system with multicore processor, two more computers are required that can send and receive SIP Invites to each other.

## 3.2 OpenSIPS Server Configuration

There are two ways for installation of OpenSIPS server:

*(i)* Installation with Binary File
*(ii)* Installation of OpenSIPS by Source Code.

Figure 5 shows basic process flow of OpenSIPS server installation process. Regardless if it is installed from binary or source code, steps in *Figure 5* must be followed.

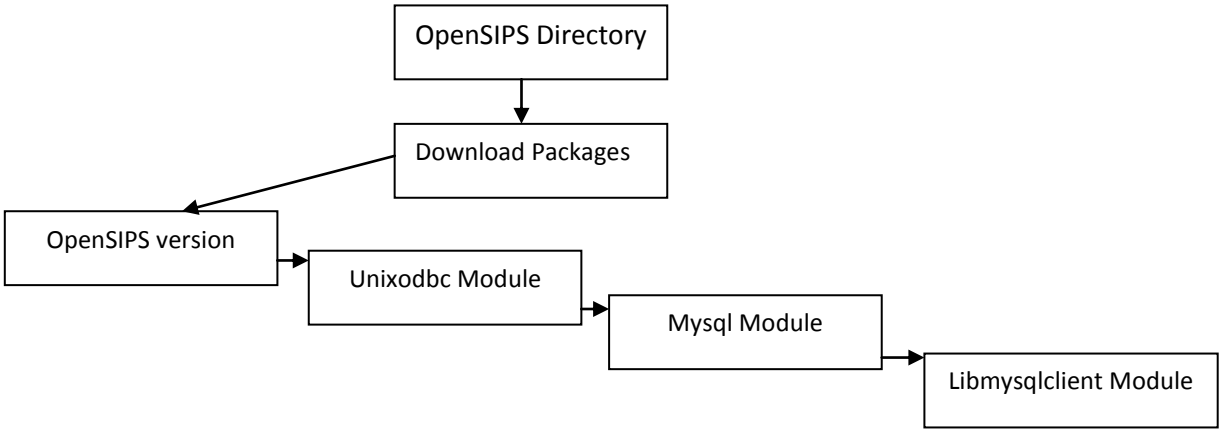



Figure 5: Stepwise Installation of OpenSIPS server

The following sections explain installation process of OpenSIPS from binary files and by compiling the source code.

### 3.2.1 Installing OpenSIPS with Binary Files

To install OpenSIPS download necessary packages; these packages are free and easily available on different websites. Before installation check version and compatibility of the system. There are different packages for different machines depending on either 32 bit or 64bit.

It is recommended to install OpenSIPS with DNS, so that making changes manually to move the system from a machine to another is not needed [19]. In this document there are simple steps for installation of OpenSIPS server without DNS support.

**Create a Directory**

The first step is creating a directory called "OpenSIPS" it will help you to easily access and find maintained files. One can create directory by using command prompt also known as terminal, open the terminal and type the following command [20].

```
# mkdir OpenSIPS
```

This will create a directory on system with name OpenSIPS; now enter into the directory by typing following command.

```
# cd OpenSIPS
```

**Download the Packages**

After creating directory, download packages which are required to install OpenSIPS. Use the GNU Wget utility to download files from the web. GNU Wget is a free software package for retrieving files using HTTP, HTTPS and FTP, the most widely-used Internet protocols. It is a non interactive command line tool, so it may easily be called from scripts, terminals without graphical window support. Type the following command as presented in order.

```
#wget http://debian.leurent.eu/debian/pool/main/OpenSIPS_1.6.4-0_amd64.deb

#     wget     http://debian.leurent.eu/debian/pool/main/OpenSIPS-unixodbc-
module_1.6.4-0_amd64.deb

#      wget      http://debian.leurent.eu/debian/pool/main/OpenSIPS-mysql-
module_1.6.4-0_amd64.deb
```

First of all download OpenSIPS, there are many versions of OpenSIPS so whenever you download be sure to about versions and compatibility of your system with that version, so that OpenSIPS works perfectly. After installing OpenSIPS download OpenSIPS-unixodbc-module and then download Mysql module.

After downloading these packages download one of the two mirrors in the following list.

```
#       wget       http://mirror.pnl.gov/ubuntu//pool/main/m/mysql-dfsg-
5.0/libmysqlclient15off_5.0.51a-3ubuntu5.8_amd64.deb
```

Or

```
#   wget   http://security.ubuntu.com/ubuntu/pool/main/m/mysql-dfsg-5.0/
libmysqlclient15off_ 5.0.51a-3ubuntu5.8_amd64.deb
```

*Note:* Always check specification of your machine before installation whether it is i386 or amd64 bit machine and then accordingly downloaded the package. In this case the machine is amd64 bit, so 64bit version of OpenSIPS and its dependency packages are used.

**Installation of Mysql**

To install Mysql go to Ubuntu software repository and search for Mysql, when repository show the result it will give a button which is Mysql, to install Mysql just click on it and it will start installation of Mysql [21].

Mysql can be installed by using command prompt, for that open command prompt and type the following command:

```
#sudo apt-get install mysql-server mysql-client"
```

Sudo is the command which enables a system to work with administrative privileges. Apt-get is a powerful command-line tool used to work with Ubuntu's Advanced Packaging Tool (APT) performing such functions as installing of new software packages, updating of the package list index and upgrading of existing software packages and the entire Ubuntu system.

Mysql works in a networked environment using client/server architecture. In other words, a central program acts as a server, and various client programs connect to the server to make requests.

**Verify files on disk**

On completion of installation, check whether the packages are downloaded successfully or not. This can be done by using the "ls" command as shown in Figure 6

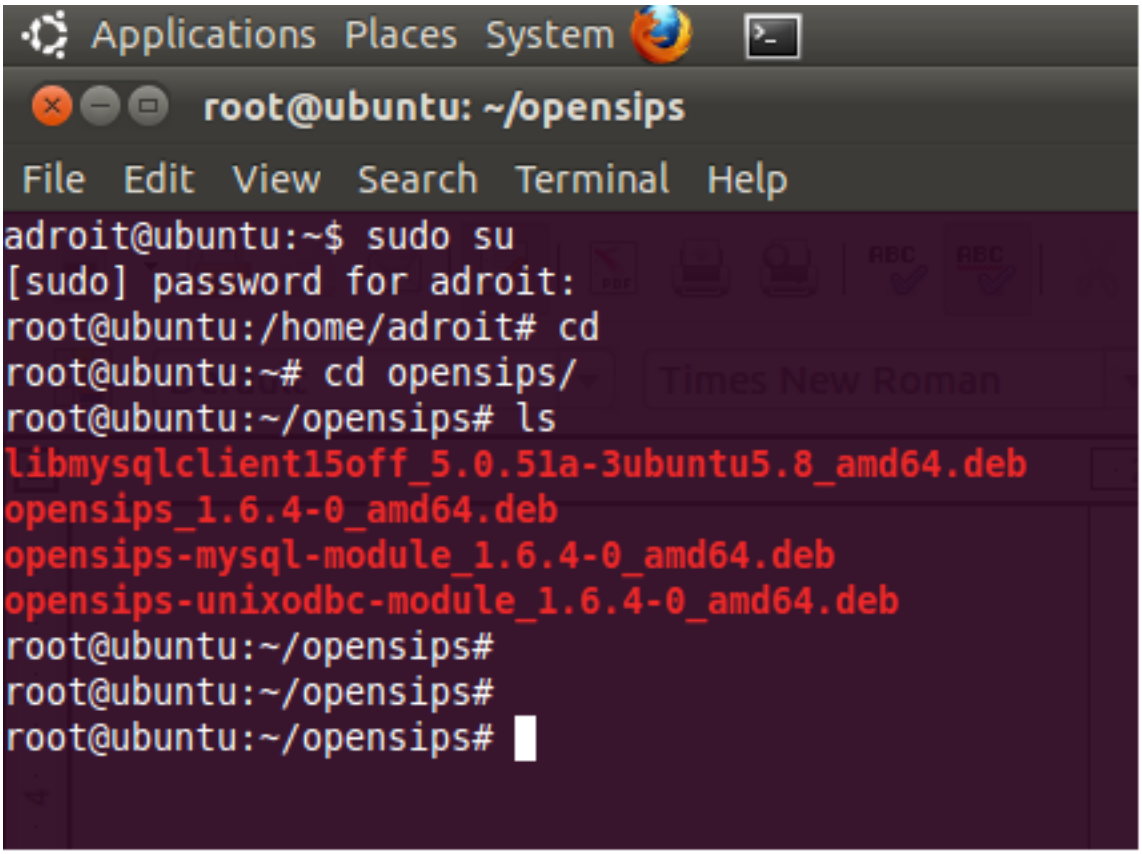



Figure 6: List of downloaded packages

**Configuring OpenSIPS and its Dependency packages**

There are certain dependency packages which are necessary to configure so that OpenSIPS works perfectly fine. Dpkg (Debian Package) is a medium-level Ubuntu package manager to install, build,

remove and manage Ubuntu packages, controlled via command line parameters. The parameter tells dpkg what to do and options control the behavior of the action in some way.

Dpkg package utility is used to download and install packages of OpenSIPS. To solve dependency relationship it is necessary to follow the following order.

```
#dpkg -i OpenSIPS_1.6.4-0_amd64.deb
#dpkg -i libmysqlclient15off_5.0.51a-3ubuntu5.8_amd64.deb
```

In case you do not have unixodbc installed, then you should install it with its dependencies from the Synaptic Packages, or with the following command by command prompt:

```
# apt-get installs unixodbc
```

Open Database Connectivity (*ODBC*) is a standard software API specification for using database management system (DBMS). ODBC is independent of programming language, database system and operating system. It enables applications to connect with database and execute SQL commands and retrieve results [22]. After installation of unixodbc run following commands to complete installation:

```
#dpkg -i OpenSIPS-unixodbc-module_1.6.4-0_amd64.deb
#dpkg -i OpenSIPS-mysql-module_1.6.4-0_amd64.deb
```

### 3.2.2 Installing OpenSIPS through Compilation:

Another way of installation of OpenSIPS is by compilation method; the compilation is bit difficult than the Binary method. The stepwise procedure of Installation is given as below, one can install and configure by following these steps [24]:

**Installing dependency packages**

The step wise Installation and compilation are as under:

```
1.  sudo apt-get install bison bison++ bisonc++
2.  sudo apt-get install flex
3.  sudo apt-get install libsctp1
4.  sudo apt-get install libmysqlclient-dev
5.  sudo apt-get install libxml2-dev
6.  sudo apt-get install libexpat1-dev
7.  sudo apt-get install libradius-ng2 libradius-ng-dev
8.  sudo apt-get install libcurl3-dev
9.  sudo apt-get install libxmlrpc-c3 libxmlrpc-c3-dev
10. sudo apt-get install libperl-dev
11. sudo apt-get install libsnmp-dev
12. sudo apt-get install libconfuse0 libconfuse-dev
13. sudo apt-get install build-essential
14. sudo apt-get install mysql-server
```

***Note***: *Line 14* is for mysql-server, it is not a dependency package for OpenSIPS; but there is a need of database for storage of OpenSIPS records like users, accounts, billing etc.

**Download package**

Download OpenSIPS_1.6.4-tls_src.tar.gz from www.OpenSIPS.org

```
cd /usr/src
wget http://OpenSIPS.org/pub/OpenSIPS/1.6.4/src/OpenSIPS-1.6.4-
tls_src.tar.gz
```

Now untar file:

```
tar -xzvf OpenSIPS-1.6.x-tls_src.tar.gz to extract the files into
directory /usr/src
```

**Compile and install core modules**

Include the *db_mysql*
Go to the directory where you unzip all the files and type on command line:

```
cd OpenSIPS-1.6.4.tls
make prefix=/ all include_modules="db_mysql"
make prefix=/ install include_modules="db_mysql"
```

This will install OpenSIPS into your system unless you get an error. It works fine with mostly computers.

**Configuring and Running OpenSIPS:**

After Installation of OpenSIPS we need to edit opensips file and turn on OpenSIPS, this can be done by the following commands.

```
sudo gedit /etc/default/opensips
RUN_OPENSIPS=YES
```

Now edit opensipsctlrc file in order to configure domain for OpenSIPS; this can be done by the following command:

```
sudo gedit /etc/opensips/opensipsctlrc
```

Uncomment the following line and replace domain name with correct SIP domain

```
SIP_DOMAIN=domain.com (or IP address)
```

Run OpenSIPS server and test installation with the help of following commands

```
sudo /etc/init.d/opensips start
sudo netstat –tulp | grep opensips
```
(to check opensips status)

**Configuring OpenSIPS with Mysql module:**

Mysql is required to configure in the OpenSIPS server so that it interact and function properly with OpenSIPS server; for that follow the following steps:

Configure and create database OpenSIPS: [19]
Edit the file /etc/OpenSIPS/OpenSIPSctlrc

```
#gedit /etc/OpenSIPS/OpenSIPSctlrc
```

Make the following change in the lines below and uncomment if any required:

```
SIP_DOMAIN=localhost
DBENGINE=MYSQL
DBHOST=localhost
DBNAME=OpenSIPS
DBRWUSER=OpenSIPS
DBRWPW="OpenSIPSrw"
DBROUSER=OpenSIPSro
DBROPW=OpenSIPSro
DBROOTUSER="root"
USERCOL="username"
INSTALL_EXTRA_TABLES=ask
INSTALL_PRESENCE_TABLES=ask
INSTALL_SERWEB_TABLES=ask
CTLENGINE="FIFO"
OSIPS_FIFO="/tmp/OpenSIPS_fifo"
PID_FILE=/var/run/OpenSIPS/OpenSIPS.pid
```

Now create the database:

```
OpenSIPSdbctl create
```

This command will ask for the root mysql password.

**Configuring Mysql support:**

Make the following changes in the OpenSIPS file configuration /etc/OpenSIPS/OpenSIPS.cfg/:

```
vim /usr/local/etc/OpenSIPS/OpenSIPS.cfg
loadmodule "db_mysql.so"
loadmodule "auth.so"
loadmodule "auth_db.so"

modparam("usrloc", "db_mode", 0);#THIS LINE MUST TO BE COMMENTED

modparam("usrloc", "db_mode", 2)
modparam("usrloc", "db_url",
"mysql://OpenSIPS:OpenSIPSrw@localhost/OpenSIPS")
```

----- auth_db params -----

```
/* uncomment the following lines if you want to enable the DB based authentication */
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "db_url",
"mysql://OpenSIPS:OpenSIPSrw@localhost/OpenSIPS")
```

**Running OpenSIPS**

Now run application by using the “OpenSIPS start” command as shown in Figure 7. Same can be done by the following command:

```
OpenSIPSctl start|stop|restart
```

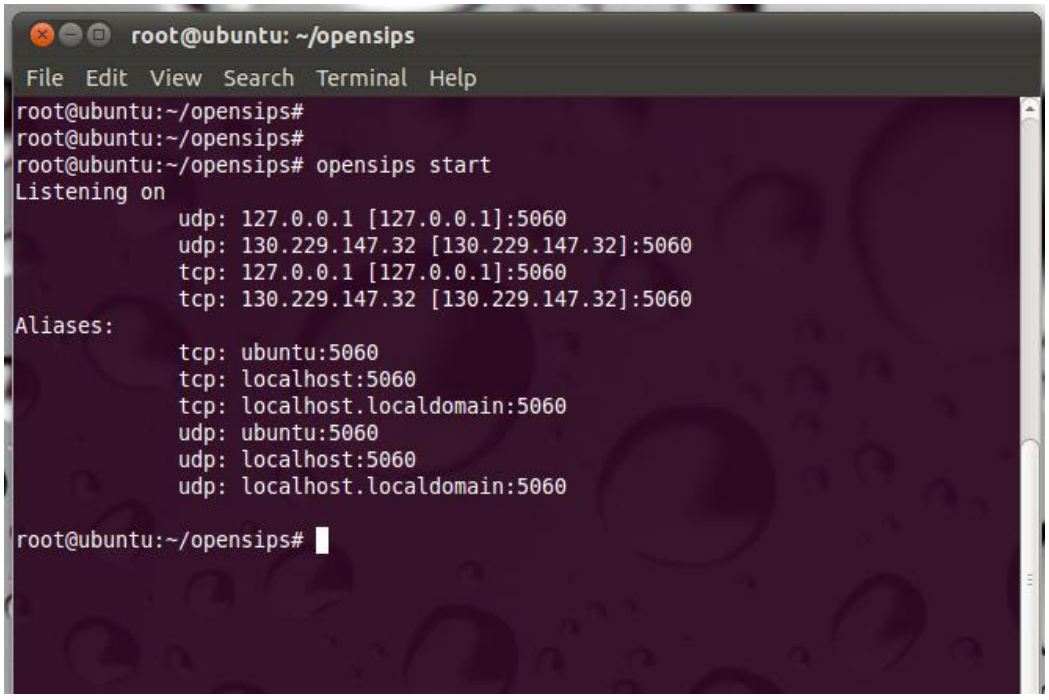



Figure 7: Start OpenSIPS

Figure 7 shows OpenSIPS is running in the “130.229.147.32” at port 5060.

**Authentications with Mysql:**

**Step 1:**

Uncomment the following lines in ***/etc/OpenSIPS/OpenSIPS.cfg*** file

loadmodule "db_mysql.so"

```
#loadmodule "auth.so"
#loadmodule "auth_db.so"
```

The “auth.so” and “auth_db.so” modules provide authentication. These modules must be enabled in OpenSIPS.cfg file.

**Step 2:**

Authenticate “REGISTER” Requests:

When a user/soft phone tries to login, it sends “REGISTER” request to be registered with the server. To enable authentication check uncomment the following piece of code in OpenSIPS.cfg file:

```
if (is_method("REGISTER"))
    {
    # authenticate the REGISTER requests (uncomment to enable auth)
    if (!www_authorize("", "subscriber"))
    {
    www_challenge("", "0");
    exit;
    }
    if (!db_check_to())
    {
    sl_send_reply("403","Forbidden auth ID");
    exit;
    }
    if (!save("location"))
    sl_reply_error();
    exit;
    }
```

**Step 3:**

**Testing Authentication:**

Add two users with OpenSIPSctl utility with the command

OpenSIPSctl username password

For example:

OpenSIPSctl `naser naser`

OpenSIPSctl `arshad arshad`

OpenSIPSctl `adnan adnan`

**Step 4:**

Check these users in “OpenSIPS” database. Users are stored in the “subscriber” table, and one can check them by loging in to mysql database. The command to check whether the user exists in database or not is as under:

```
mysql> select * from subscriber;
```

The pattern of information which the command will bring would be as under:

```
+----+----------+-----------------+----------+---------------+----------------------------------+----------+---------+
| id | username | domain          | password | email_address | ha1                              | ha1b     | rpid    |
+----+----------+-----------------+----------+---------------+----------------------------------+----------+---------+
| 17 | naser    | 130.229.141.195 | naser    |               | 3f303f8c9906eb0b9bd5602d21dd5c26 | 05e9a9c36b4e31efa9aab97bc039fe07 | NULL    |
| 18 | arshad   | 130.229.141.195 | arshad   |               | 65f1f7b7a2570b3a7b78c2db3b7062f5 | b4c284ba084b94c0ad0fef41acee4ea4 | NULL    |
| 19 | adnan    | 130.229.141.195 | adnan    |               | 4ae249c47888d56c34a92f73a6277a12 | 2f34023fce27d8624f221d246be26c64 | NULL    |
+----+----------+-----------------+----------+---------------+----------------------------------+----------+---------+
```

**Step 5:**

**Restart the OpenSIPS**

After completion of above steps, it is the time to restart OpenSIPS server. The server can be restarted by command as given below:

***OpenSIPSctl restart***

This command will restart OpenSIPS server.

**Step 6:**

**Login to User Agents**

Now try to login with the usernames and passwords on different clients/computers. Now you will be prompted to provide correct username and password, otherwise you will receive an error message that authentication “failed”.

## 3.3 SIP Traffic Generation

The function of SIP traffic generator is to generate large number of SIP users, in order to monitor and test OpenSIPS server performance. This module is developed by another team [16] who participated in the same project. The traffic generator is designed and coded in independent modules in order to utilize the capacity of multicore platform in future, e.g, registration, invite, session description etc. to be processed on separate cores. This is the first step towards multicore or parallel programming. This thesis SIP Traffic generator is designed on modular base and elaborated in Figure 8; which shows that it comprises two modules ***sender*** and ***receiver***.

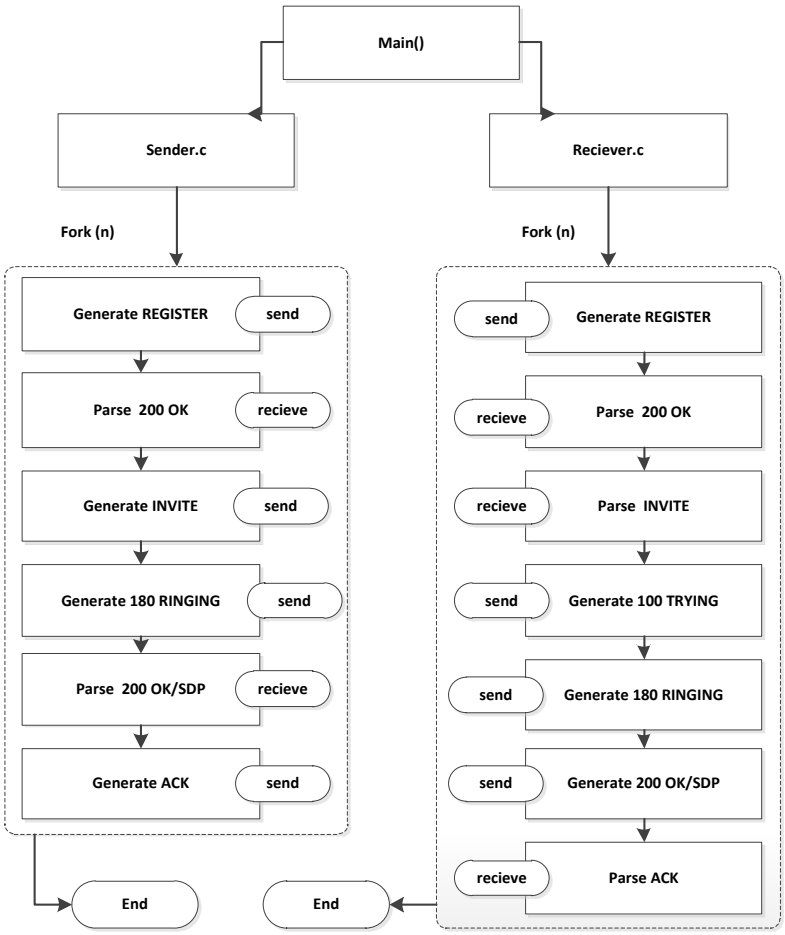



Figure 8: Sender Receiver Module [16]

### 3.3.1 Sender Module

The sender module generates the desired number of SIP users to establish SIP sessions. The idea is taken from RFC 3261 section 4. The sender module registers unique SIP users with OpenSIPS to establish session by creating a Linux child process for every user [16]. This is the way the sender module can create the desired number of SIP users for testing.

### 3.3.2 Receiver Module

The function of receiver module is pretty same as sender module. The sender and receiver module generates their users and then sender users send an INVITE to the receiver. Both modules generate the specific number of SIP users in order to establish sessions

These models are very efficient and open UDP port separately for each user, as there is a need to create different number of users ranging from 1 to 1000 for testing.

# 4. Experimental Setup and Results

This chapter describes experimental setup, what are the limitations and results obtained by performing tests.

## 4.1 Testing Procedure

To achieve goals of project the following are the requirements:

1. OpenSIPS server configured on a system having multicore processor
2. Senders Receivers simulation code which can generate user defined Invites to open session, this code is designed by the other teams of the project.
3. The SIP transactions between sender and receiver will be captured by using wireshark from server on which OpenSIPS is configured.
4. These captured transactions are then exported into a text file from wireshark, and exported text files are then extracted by using timestamp code developed by other teams of the project, it will give timestamp from sender to server and server to receiver. The receiver acknowledges and sends back the acknowledgement to sender via server.
5. The extracted files are then imported into excel sheets, where delays and loss of invites can be calculated.
6. Iperf to change network state from idle to user defined busy state. It will help to test when the behavior of a system in an environment (in case of some particular amount of usage) is going on.

### 4.1.1 Test Scenario 1

The tests are performed on the basis of two network scenarios. In scenario 1 the network is established by connecting Sender, Receiver and OpenSIPS PCs as shown in the Figure 9. There is no other connection in this network and all the network resources are utilized by sender, receiver and OpenSIPS server. OpenSIPS is configured on server pc while sender module is running on sender pc and receiver module is running on receiver pc. Wireshark is running on all the PCs, monitoring and capturing the traffic from these PCs for each experiment.

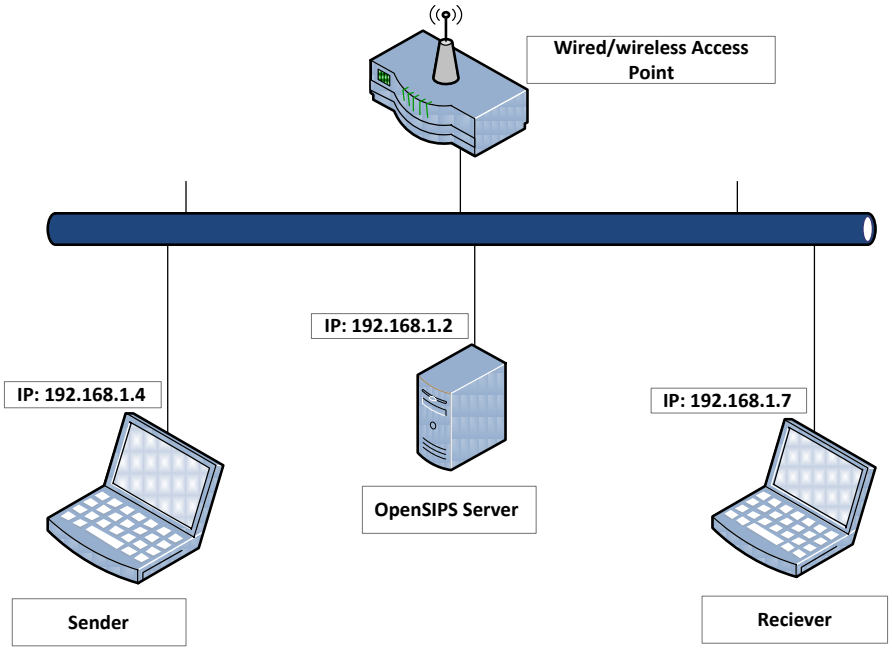



Figure 9: Test Scenario 1

In Figure 9 the Server is having UBUNTU 9.04 operating system with 2.1 GHz Intel dual core processor along-with 2GB of RAM. The server is configured with OpenSIPS Server with Mysql 5, Apache and Wireshark. Wireshark is used in our project to monitor network traffic specifically SIP messages. The IP address assigned to OpenSIPS server is 192.168.1.2.

The server is connected with an access point through Ethernet, while the access point is capable of wireless communication, as Sender and Receiver are connected with access point through wireless card. The Sender and Receiver send SIP messages to each other. The IP address assigned to sender machine is 192.168.1.4, while for receiver machine the IP address is 192.168.1.7

### 4.1.2 Test Scenario 2

In Test Scenario 2, the network is partially utilized by generating traffic before starting SIP traffic generator. In this project, the network is partially utilized by using Iperf. It is useful to know the behavior of network, when the bandwidth utilization is at its peak. In this scenario it is expected that the delay for SIP messages will increase.

The equipment and network design is similar to scenario 1 in addition with 2 PCs that are connected with the network called Iperf client and server. Iperf is used to generate the traffic in order to make the network partially busy by utilizing network resources. Figure 10 demonstrates the network design for scenario 2.

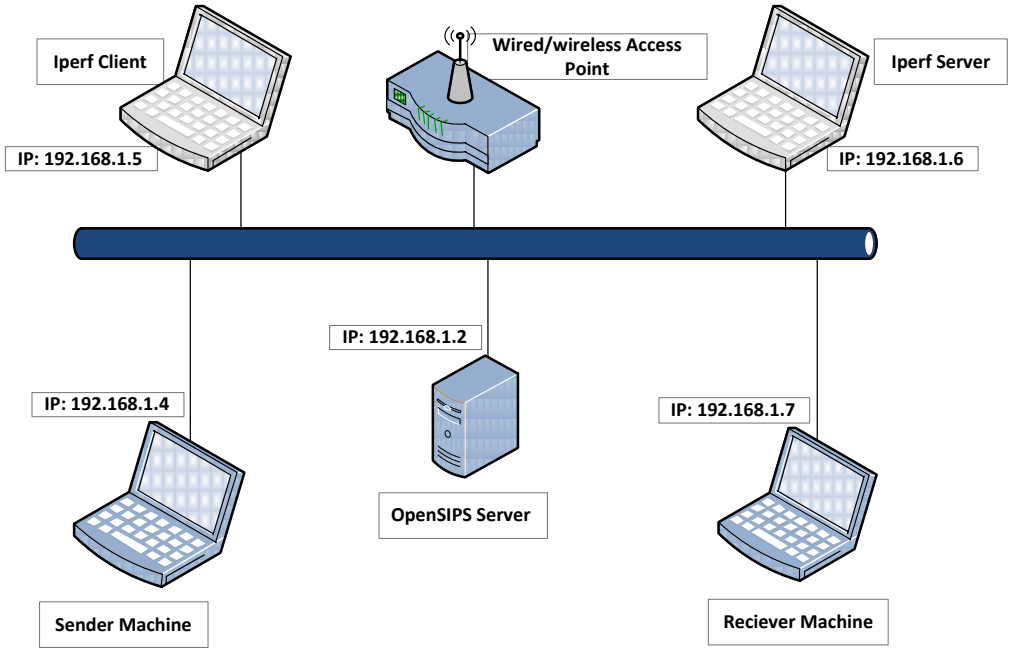



Figure 10: Test Scenario 2

In Figure 10 the network topology is same as used previously for test scenario 1. The only addition is the two Iperf systems, one is Iperf server and other is Iperf client. The Iperf server and client generate the traffic and make the network bandwidth busy. In this case 50% of network bandwidth is used by Iperf. The IPs assigned to the systems connected with the network is same as in previous case. The two additional Iperf systems are assigned with an IP of 192.168.1.5 to Iperf client and 192.168.1.6 to Iperf server.

The SIP messages are sent to each other and these messages are elaborated by the Figure 11:

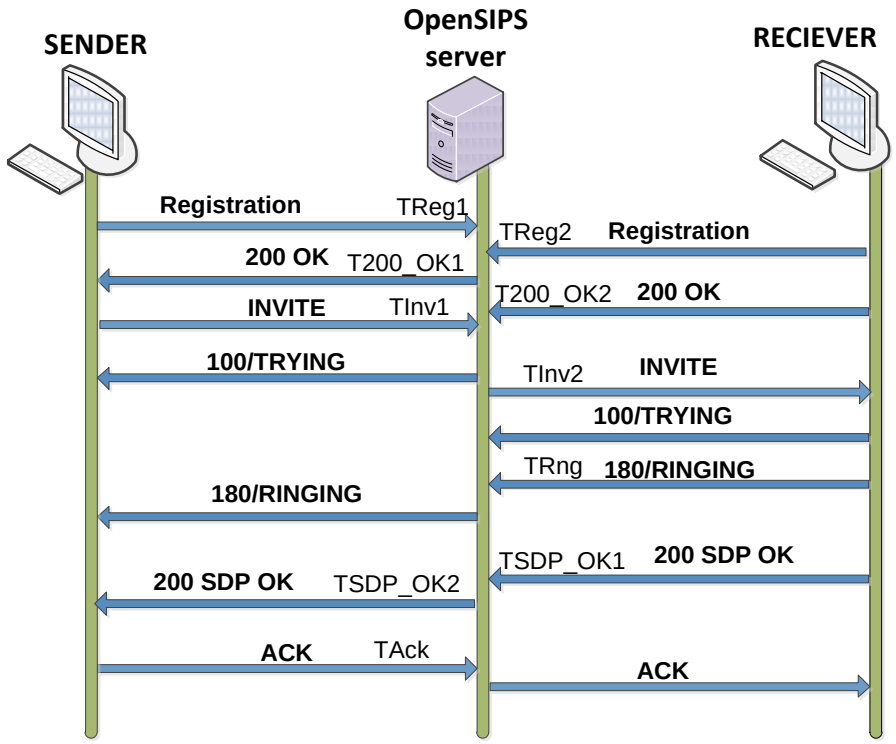



Figure 11: Sender Receiver Model Communication

Figure 11 shows how SIP messages are generated between sender and receiver in order to communicate with each other; server resides in the middle; connected with an access point through LAN. The access point is capable of wireless communication as two systems working as sender and receiver are connected with access point through wireless cards.

The following timestamps are generated during the communication process which is captured for analysis and calculation during the test for QoS benchmarking.

1. **TReg1 and T200_OK1:** this timestamp shows sender sends a SIP message for Registration to OpenSIPS server and OpenSIPS server registers it and sends registration successful message back to sender
2. **TReg2 and T200_OK2:** The same process is applied at the receiver end and receiver will send a SIP Registration message to OpenSIPS server and OpenSIPS server will acknowledge.
3. **TInv1:** This timestamp shows INVITE request from a user for a call set-up between sender and receiver
4. **TInv2:** The INVITE from OpenSIPS server towards receiver
5. **TRng:** SIP message 180 RINGING from a SIP user on a receiver to sender
6. **TSDP_OK1:** SIP message 200 OK with Session Description Protocol (SDP) response from a SIP user on a receiver who has received an INVITE from sender
7. **TSDP_OK2:** SIP message is 200 OK with SDP response being forwarded to a SIP user on sender.
8. **TACK:** SIP message ACK (acknowledgment) to SDP initiated by receiver for SIP session.

## 4.2 Limitations

VoIP companies have high performance servers that can handle huge number of calls simultaneously. As an example a high performance server can handle up to 1000 parallel calls like Mizutech [6].

In this project there were limited hardware resources, such as high performance servers. The tests are performed by using laptops as well as a desktop system to analyze QoS parameters. The specification of and role of computer system is given below:

1. ***Server***
   Processor – Intel dual core 2.1 GHz
   RAM - 2GB
   Network Card – wide LAN 100MBPS
   Operating System – UBUNTU 9.04

2. ***Clients***

   *(i)* *Sender System Specifications*
   Processor – 2.1GHz
   RAM - 3GB
   Wi-Fi – 802.11b/g/n
   Operating System – UBUNTU 9.04

   *(ii)* *Receiver System Specifications*
   Processor – Intel core i5 – 2410, CPU@2.30GHz
   RAM - 4GB
   Wi-Fi – IEEE 802.11b/g/n
   Operating System – UBUNTU 9.04

There were two Pentium-III personal computers which were used as traffic generators by using Iperf tool [15].

## 4.3 Result Generation Method

The wireshark is used to capture the streams from clients as well as from OpenSIPS. The captured file is then exported in text format from wireshark into a text file. After exporting data into text file, the file is ready to for analysis and calculations of delays by taking timestamps.

These timestamps are extracted by a code developed in C language known as timestamps.c for extraction of text data into desired output format. The data is then imported into excel sheets for calculation of timestamps in order to make analysis to find out delays, bottlenecks and the point at which the ratio of call drop increases.

There are two test scenarios in this project, as explained in section 4.1.1 for test scenario 1 and 4.1.2 for test scenario 2.

## 4.4 Test Scenario 1

In this scenario the test is made on network idle mode, as network bandwidth is majorly utilized by sender, receiver and OpenSIPS server connected with each other. The scenario topology is elaborated in Figure 9: Test Scenario 1 of section 4.1.1. Table 2 describes the pattern of test case1, which are conducted to overload the OpenSIPS server.

| Test | Number of Users | Number of Tests |
|---|---|---|
| 1 | 1 | 10 Tests for each |
| 2 | 10 | |
| 3 | 50 | |
| 4 | 100 | |
| 5 | 200 | |
| 6 | 300 | |
| 7 | 500 | |
| 8 | 700 | |
| 9 | 1000 | |

Table 2: Test Case 1 Pattern

Table 2 shows that there are 9 test cases, where every test case is performed 10 times each. The number of test cases ranges from user 1 to 1000.

A packet carrying SIP request/response can be lost during the session. While exchanging SIP messages any single packet loss leads to the loss of a session, like SIP Register, 200 OK, INVITE, 180 Ringing, 200SDP OK or ACK, any single message lost leads to the session loss. The results of registration delay, invite delay and call setup delay are represented in tabular and their behavior is explained in the respective figures, in coming sections.

### 4.4.1 Registration Delay

Table 3 shows the results showing the minimum, average and maximum time taken by the Registration process of SIP users.

| Number of users | Min_Reg | Avg_Reg | Max_Reg |
|---|---|---|---|
| 1 | 0,030538 | 0,114500 | 0,186739 |
| 10 | 0,015760 | 0,110998 | 0,203808 |
| 50 | 0,002656 | 0,113342 | 0,262559 |
| 100 | 0,002618 | 0,157476 | 0,423323 |
| 200 | 0,002364 | 0,238951 | 0,685404 |
| 300 | 0,002347 | 0,298003 | 1,133649 |
| 500 | 0,002236 | 0,793845 | 2,548653 |
| 700 | 0,003502 | 0,705701 | 2,189781 |
| 1000 | 0,002639 | 0,803402 | 2,582780 |

Table 3: Minimum, Average and Maximum Registration Delay in seconds

Table 3 shows there are 9 tests which start from 1 user to 1000 users. Each test is repeated 10 times and after that gets average values, as shown in Figure 12:

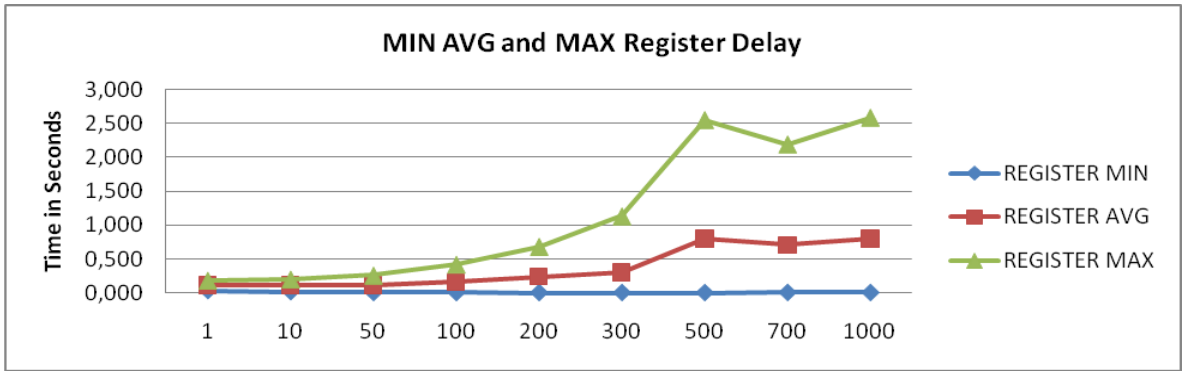



Figure 12: Minimum, Average and Maximum Registration Delay

Figure 12 shows minimum, average and maximum register delay for SIP sessions. The time is calculated in seconds. The behavior of minimum and average register delay is quite smooth, while the maximum delay rate is clearly uneven although it starts smoothly but after 300 sip session the behavior is uneven.

It is clear that when number of users increases the average and maximum registration delay increases. In the beginning of tests the delay is very smooth, but when it crosses test 6 which is for 300 users, the time required for registration increases. The uneven maximum registration delay may have many potential reasons out of which wireshark and Mysql are the two important reasons. As the network is also connected with the Internet it might be another reason as internet may lead to some unknown traffic generation which increases registration time.

### 4.4.2 Invite Delay

Table 4 shows the results showing the minimum, average and maximum time taken by the INVITE of SIP users.

| Number of users | Min_Invite | Avg_Invite | Max_ Invite |
|---|---|---|---|
| 1 | 0,011300 | 0,004382 | 0,001938 |
| 10 | 0,008000 | 0,015853 | 0,203808 |
| 50 | 0,041900 | 0,013789 | 0,045613 |
| 100 | 0,050000 | 0,016944 | 0,165818 |
| 200 | 0,027000 | 0,022085 | 0,407351 |
| 300 | 0,037900 | 0,025482 | 0,464337 |
| 500 | 0,055000 | 0,043472 | 0,346568 |
| 700 | 0,039000 | 0,047836 | 0,676668 |
| 1000 | 0,100000 | 0,070664 | 0,618663 |

Table 4: Minimum, Average and Maximum Invite Delays in seconds

Table 4 shows the time taken by invite for SIP session as the value shows after registration process the time taken by INVITE is very less. The minimum time taken by Invite is 0,008 seconds, while for Register time is 0,002 seconds.

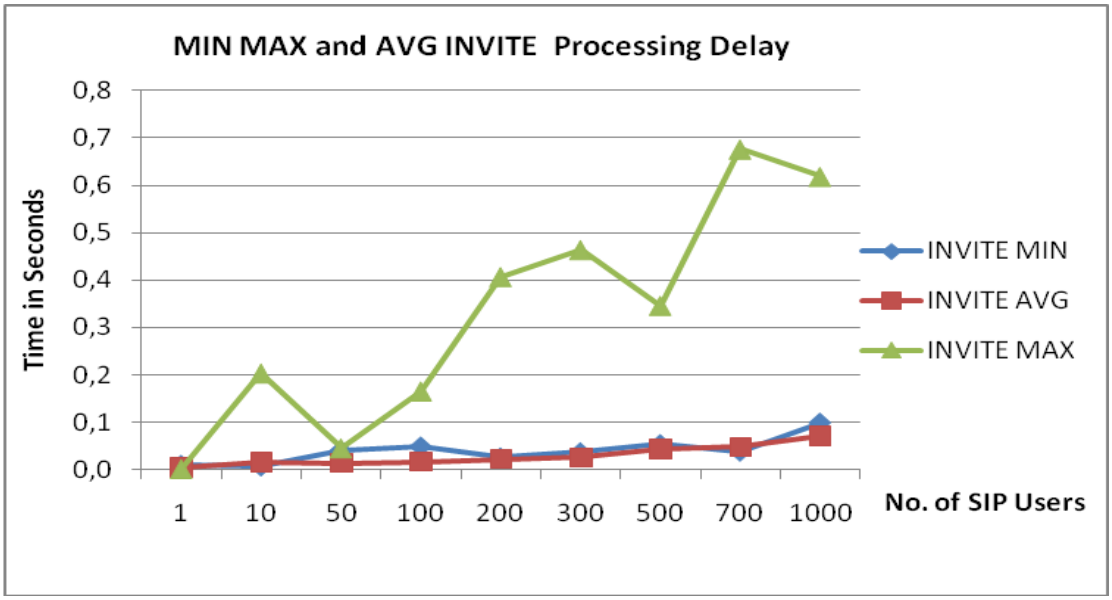



Figure 13: Minimum, Average and Maximum Invite Delays in seconds

Figure 13 shows the behavior of minimum and average Invite is smooth as in previous case of registration, while the behavior of maximum invite delay is uneven. The maximum invite delay never remained stable for all tests, and took more time. The potential reasons seem to be wireshark, Mysql and Internet load.

### 4.4.3 Call Setup Delay

Table 5 shows the results for minimum, average and maximum time taken for call setup of SIP sessions. Graphically the behavior is shown in Figure 14:

| Number of users | Min_call setup | Avg_ call setup | Max_ call setup |
|---|---|---|---|
| 1 | 0,001198 | 0,002191 | 0,006438 |
| 10 | 0,001137 | 0,004687 | 0,203808 |
| 50 | 0,001057 | 0,006188 | 0,039792 |
| 100 | 0,001142 | 0,005208 | 0,095130 |
| 200 | 0,000027 | 0,028368 | 0,210704 |
| 300 | 0,001004 | 0,071447 | 0,287457 |
| 500 | 0,000253 | 0,124813 | 0,515803 |
| 700 | 0,000953 | 0,227045 | 0,520289 |
| 1000 | 0,001011 | 0,313065 | 0,536062 |

Table 5: Minimum, Average and Maximum Call Setup Delays in seconds

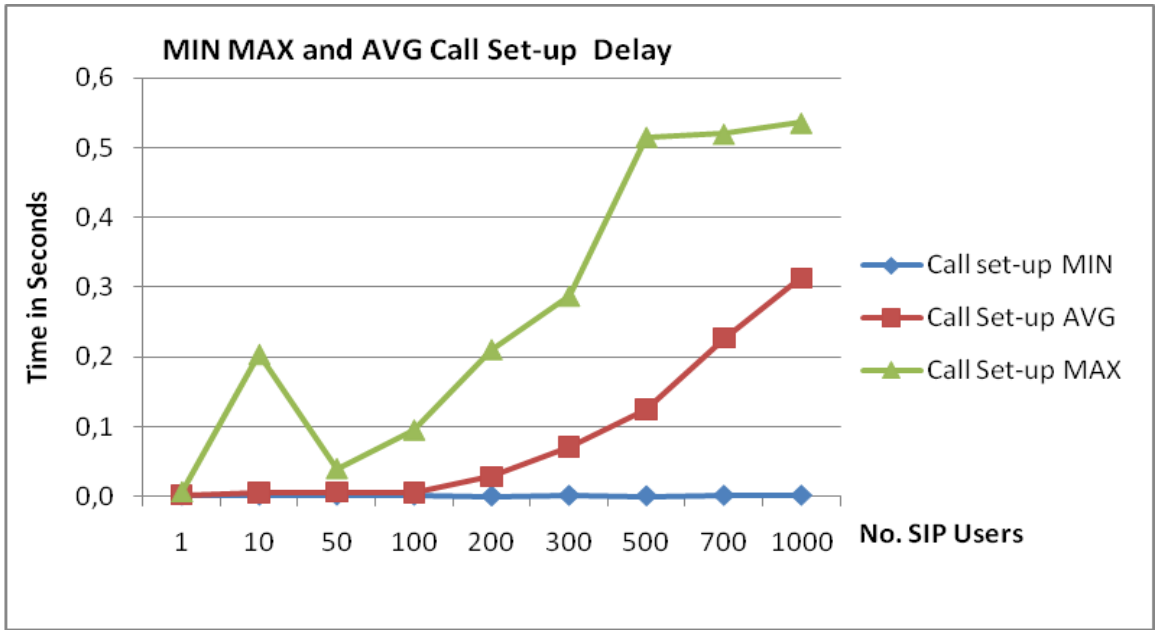



Figure 14: Minimum, Average and Maximum Call Setup Delays in seconds

Figure 14 shows the time for call setup delay, the minimum time rages around 0.001198 second. The average time starts from 0.002191 seconds and goes up to 0.3130653 seconds. Maximum delay ranges from 0.006438 seconds to 0.536062 seconds.

The behavior is stable for minimum call setup, while the average call setup time starts increasing after 100 numbers of sip users and keeps on increasing trend. The maximum delay behavior is similar with the registration and invite behavior, that is maximum call setup behavior is non stable.

### 4.4.4 Average Packet Loss

Table 6 shows the results for packet loss of registration, invite and call setup for SIP sessions.

| Number of users | Registration | Invite | Call setup |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 10 | 0 | 0 | 5 |
| 50 | 21 | 31 | 10 |
| 100 | 2 | 2,1 | 17 |
| 200 | 7 | 8 | 12 |
| 300 | 20 | 26 | 202 |
| 500 | 53 | 64 | 333 |
| 700 | 123 | 156 | 420 |
| 1000 | 380 | 416 | 583 |

Table 6: Loss of Registration, Invite and Calls

Graphically it can be elaborated as below in Figure 15:

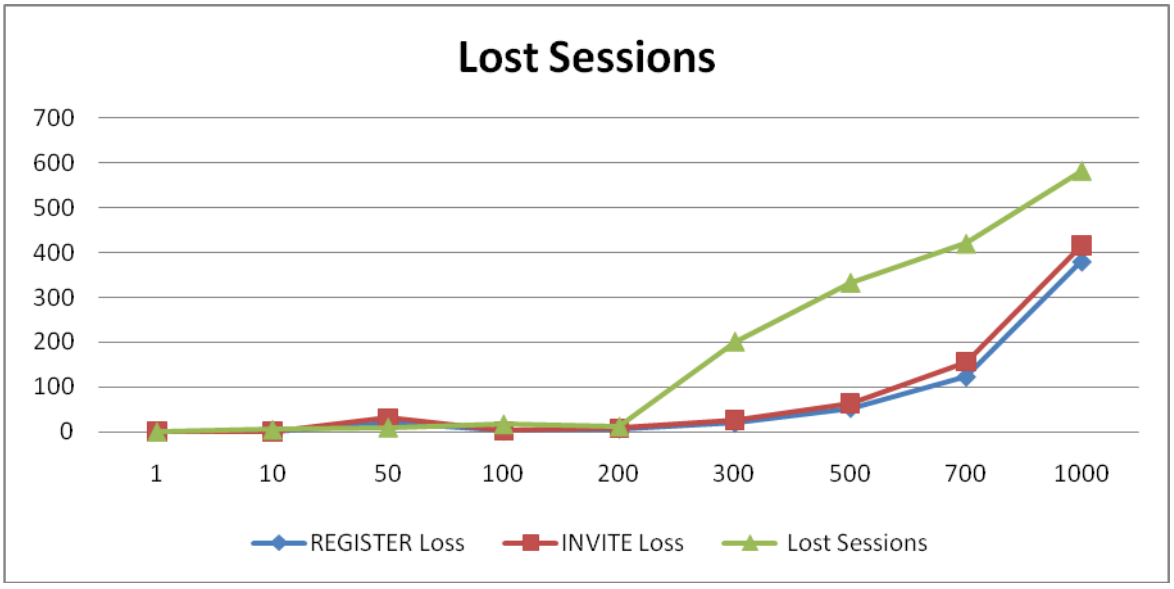



Figure 15: Loss of Registration, Invite and Calls

Figure 15 shows the number of lost session due to packet loss during network communication. The number of lost session for register and invite are negligible up to 700 user tests as it ranges within 125 to 150. While for 1000 users the ratio of lost session is high. The behavior of call setup lost session remains stable up to 200 user tests as the lost sessions are negligible. After 200 the ratio of lost sessions are on high and remain increasing with number of users.

### 4.5 Test Scenario 2

In this scenario there is a need of two extra systems and connect them with access point through Ethernet, after connection one system acts as Iperf Router and other one as Iperf Client, as explained in section 4.1.2. The Iperf client generates traffic/queries and sends that to Iperf server via access point. Iperf server on receiving query acts and generates traffic and sends back to access point. This process will work constantly and use the bandwidth up to set value, here in this project the value is set to 50%.

The number of users sequences in scenario 2 remains same as in scenario 1, but each test sequence is repeated for 5 times instead of 10 times as in scenario 1.

| Test | Number of Users | Number of Tests |
|---|---|---|
| 1 | 1 | 5 Tests for each |
| 2 | 10 | |
| 3 | 50 | |
| 4 | 100 | |
| 5 | 200 | |
| 6 | 300 | |
| 7 | 500 | |
| 8 | 700 | |
| 9 | 1000 | |

Table 7: Partially overloaded Network Scenario Tests

### 4.5.1 Registration Delay

Table 8 shows the results of registration delay for test scenario 2. The time is calculated in seconds.

| Number of Users | Min_Register | Avg_ Register | Max_ Register |
|---|---|---|---|
| 1 | 0,015371 | 0,165068 | 0,370033 |
| 10 | 0,006485 | 0,180681 | 0,456191 |
| 50 | 0,071775 | 0,277978 | 0,558802 |
| 100 | 0,004126 | 0,359814 | 0,813315 |
| 200 | 0,004772 | 1,099847 | 7,605190 |
| 300 | 0,003330 | 1,336017 | 6,682049 |
| 500 | 0,003299 | 0,990998 | 4,070590 |
| 700 | 0,177700 | 2,940312 | 9,599048 |
| 1000 | 0,198200 | 4,919223 | 11,213900 |

Table 8: Minimum, Average and Maximum Registration Delay in seconds

Table 8 shows there are 9 tests which start from 1 user and ends at 1000 users. Each test is repeated 5 times and after that gets minimum, average and maximum registration delays.

Graphically it can be shown Figure 16:

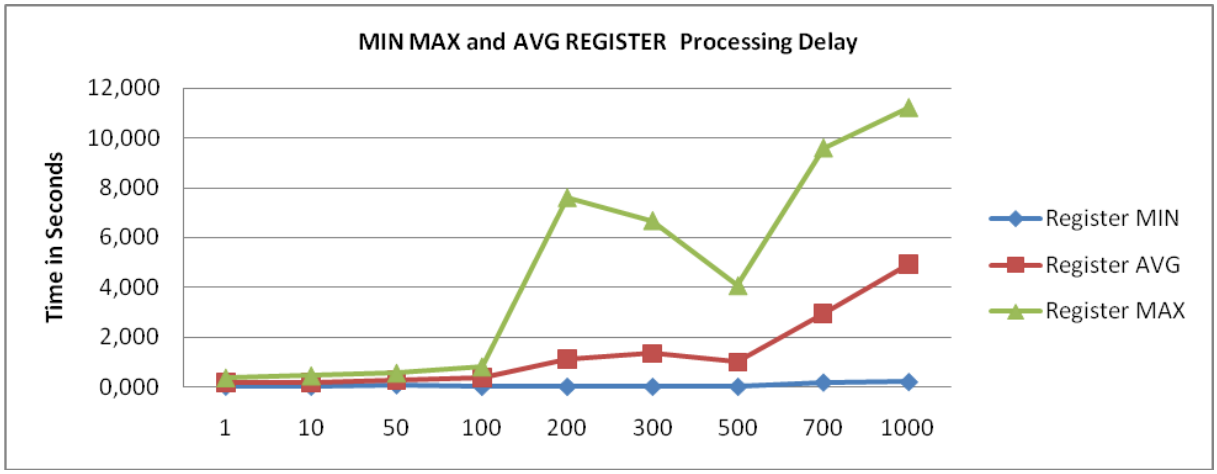



Figure 16: Average, Minimum and Maximum Register Delay in seconds

Figure 16 shows the behavior of registration delays, the minimum registration delay remains stable throughout the testing procedure in our test environment where network was partially utilized. While average and maximum registration delays remain stable up to 100 user tests, after that there are fluctuations in their time sequence. The time taken trend remain on the higher side for both average and maximum registration delays, as for average delay for 1000 users is approx. 5 seconds, while it goes around 11 seconds for maximum registration delay.

### 4.5.2 Invite Delay

Table 9 shows the results of invite delay for test scenario 2. The time is calculated in seconds.

| Number of | Min_Invite | Avg_ Invite | Max_ Invite |
|---|---|---|---|
| 1 | 0,016122 | 0,165749 | 0,370731 |
| 10 | 0,011441 | 0,283004 | 0,456820 |
| 50 | 0,073270 | 0,186819 | 0,559846 |
| 100 | 0,008558 | 0,363635 | 0,814094 |
| 200 | 0,005742 | 1,727932 | 4,606523 |
| 300 | 0,004585 | 1,363257 | 6,683516 |
| 500 | 0,258230 | 1,039738 | 4,722430 |
| 700 | 0,018563 | 3,986237 | 6,637029 |
| 1000 | 0,344318 | 6,123654 | 8,605320 |

Table 9: Minimum, Average and Maximum Invite Delay in seconds

Graphically it can be shown as in Figure 17*:*

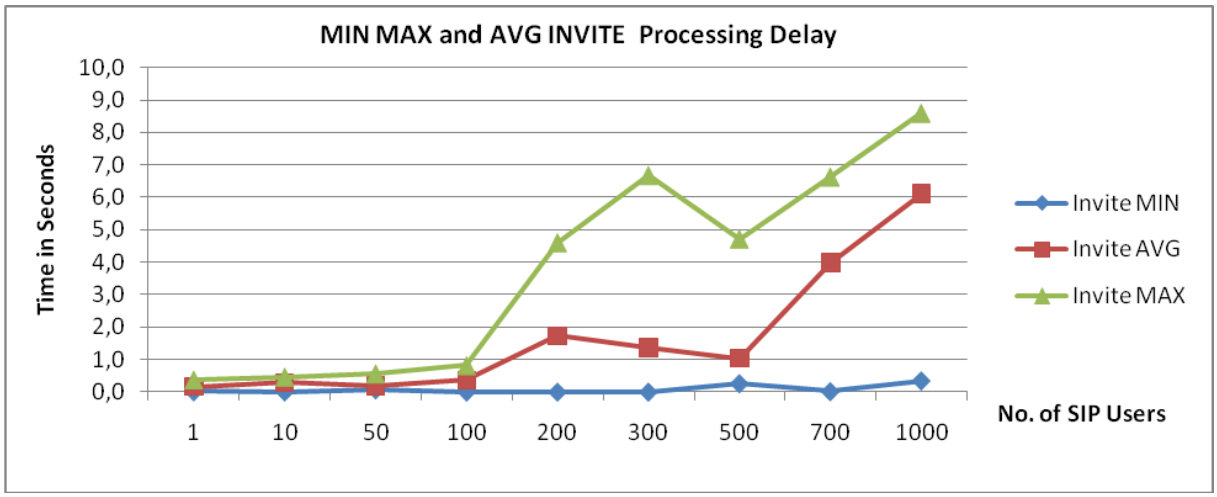



Figure 17: Minimum, Average and Maximum Invite Delay in seconds

Figure 17 shows minimum, average and maximum delays of invites sent by the sender to receiver in overloaded network. The behavior is similar to registration delay and up till 100 users it increases quite constantly, but after that the time increases little bit more than it does in the beginning and is not stable. The highest invite delay occurs at 1000 user tests where it reaches around 6 seconds for average invite and approx. 8.5 seconds for maximum delay.

### 4.5.3 Call Setup Delay

Table 10 shows the results of invite delay for test scenario 2.

| Number of Users | Min_call setup | Avg_ call setup | Max_ call setup |
|---|---|---|---|
| 1 | 0,001086 | 0,001799 | 0,003186 |
| 10 | 0,002018 | 0,033326 | 0,993150 |
| 50 | 0,006967 | 0,135100 | 0,153302 |
| 100 | 0,005625 | 0,281327 | 0,614826 |
| 200 | 0,018871 | 1,187903 | 2,403848 |
| 300 | 0,446011 | 0,581243 | 3,674348 |
| 500 | 1,455374 | 3,709702 | 6,012907 |

| | | | |
|---|---|---|---|
| 700 | 0,935404 | 3,740841 | 7,313355 |
| 1000 | 1,255374 | 7,112982 | 8,473940 |

Table 10: Minimum, Average and Maximum call setup Delays in second

Figure 18 shows the results of call setup delays:

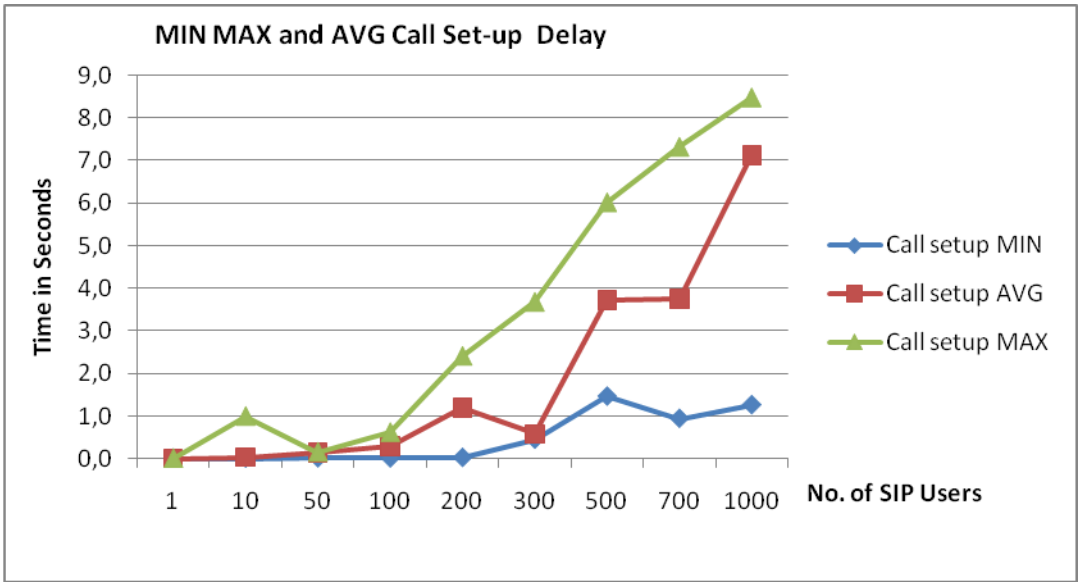



Figure 18: Minimum, Average and Maximum Call Setup Delay in seconds

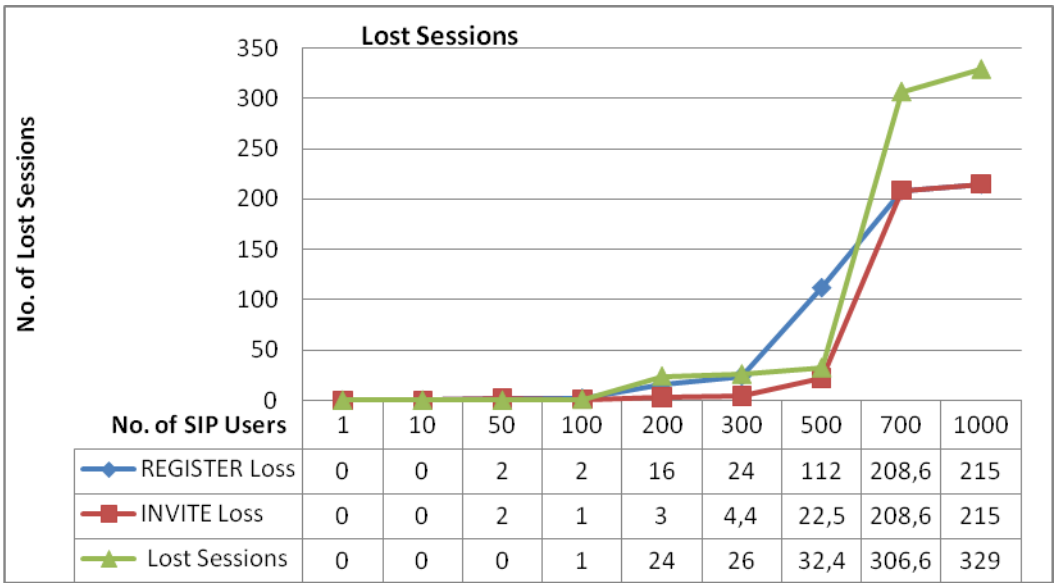
Figure 18 shows minimum, average and maximum delays in call set up. In this test the behavior is bit unpredictable and values remain variable. The delay is very little up to 300 users test after that the delay is on higher side. The delay for 1000 users test is more than 1 for minimum call setup delay, while for average and maximum call setup delay it remains in between 7 to 9 seconds respectively.

### 4.5.4 Average Packet Losses

| Number of Invites | Register | Invite | Call Setup |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 |
| 50 | 2 | 2 | 0 |
| 100 | 2 | 1 | 1 |
| 200 | 16 | 3 | 24 |
| 300 | 24 | 4,4 | 26 |
| 500 | 112 | 22,5 | 32,4 |
| 700 | 208,6 | 208,6 | 306,6 |
| 1000 | 215 | 215 | 329 |

Table 11: Loss of Registration, Invite and Calls

Table 11 shows average loss of register, invite and call setup sent by sender to receiver. Graphically it can be shown in Figure 19:




| No. of SIP Users | 1 | 10 | 50 | 100 | 200 | 300 | 500 | 700 | 1000 |
|---|---|---|---|---|---|---|---|---|---|
| REGISTER Loss | 0 | 0 | 2 | 2 | 16 | 24 | 112 | 208,6 | 215 |
| INVITE Loss | 0 | 0 | 2 | 1 | 3 | 4,4 | 22,5 | 208,6 | 215 |
| Lost Sessions | 0 | 0 | 0 | 1 | 24 | 26 | 32,4 | 306,6 | 329 |

Figure 19: Loss of Registration, Invite and Calls

Figure 19 shows average loss of invites sent by sender to receiver. The number of lost sessions was minimum until test number 6. Then the number is increasing along-with increase in users. The lost of sessions are highest for 1000 users test as it is 215 for register and average lost while for call setup the lost sessions are 329.

### 4.6 Analysis of Results

The results generated in section 4.4 and 4.5 compares the results of SIP transactions under two different scenarios. It can clearly be seen from figure number 12 to 19 that the lost sessions are more in scenario 1, when the network is in idle state. The number of lost session has decreased when the bandwidth is partially utilized. The number of lost packets in Registration, Invite and Call Setup of scenario 1 and 2 are shown in Table 12.

| Scenario | Registration Loss | Invite Loss | Call Setup Loss |
|---|---|---|---|
| 1 | 606 | 702 | 582 |
| 2 | 579 | 457 | 719 |

Table 12: Number of Lost Packets

On the other hand from time perspective, registration, invite and call setup delay in scenario 2 took more time than in scenario 1.

There are several factors which need to be considered as possible cause for this packet loss and delay. Some important factors are:

1. Ethernet link of server with access point
2. Wireless connection of sender and receiver clients with OpenSIPS server via access point
3. Background noise of wireless access point
4. Internet connection with access point as well as sender and receiver clients
5. Buffer overflow at sender, receiver and OpenSIPS server (needs to clear buffer after every test in order to remove the SIP transactions which failed to get ACK. Most importantly the buffer of OpenSIPS as it is the one where all SIP transactions move)

# 5. Conclusion and Future Work

## 5.1 Conclusion

The results show that the delay for SIP messages to reach between the sender and receiver is very low, as the highest delay is just few seconds on multicore processor. Moreover, the delay is quite minimal when the network is overloaded by using Iperf tool. A sum up can be given as follows:

1. Traffic generation for SIP by sender and receiver modules is designed for SIP stress testing. The module is tested for 1000 users successfully.

2. OpenSIPS server is configured and maintained for SIP stress testing. Where sender receiver module generates the traffic and OpenSIPS server manage these transactions. Our focus remains on delays and packet loss for SIP session establishment on idle as well as partially loaded network.

3. Results are presented to provide a simplified way for analysis and discussion. In results section timestamps are collected for SIP messages on both server and client side. This thesis remain limited with server only as it is the most important aspect in this project in order to know the basic behavior of SIP on multicore processor.

It is concluded from the test that the introduction of VoIP over multicore processors does have effect on the performance measures of QoS that is delay, jitter and frame loss. It is concluded from comprehensive tests that the delay of Invites is very small even when the users are at higher number or the network is overloaded.

## 5.2 Future Work

The usage of VoIP telephony and videoconferencing is increasing day by day. The fundamental issue in VoIP is QoS, and major factor of QoS in VoIP is delay. There are 2 types of delays which occur in a network, these are:

1. Fixed Delays
2. Variable Delays

Fixed delay depends on performance of the network nodes on the transmission path, the capacity of links and propagation delay. Variable part is the time spent in queues which depend on current network load.

This thesis focuses at SIP signaling messages level but in future there is a need to look at:

1. Delays occurred at while transferring media stream by using IRTP.

2. There is a need to work on udp protocol because when the user increases the delay at signal level increases, it is required to find out different ways and queuing schemes which can reduce this delay.

3. VoIP users are increasing day by day and it is very hard to manage huge number of users therefore, it is suggested that OpenSIPS CP (Control panel) may be configured for proper user management.

4. The security issue is not considered for this thesis. In future security may be implemented and test be conducted again to check the effect of security on packet delay and loss.

5. In results section the delay and packet loss occurred during the network idle state is more than the delay during network over-load state, deeper investigations may be carried on to find why it happened.

## 6. References


[1] iTC-MIBAX Master Thesis Project Title: VoIP QoS Benchmark Module for Multicore offered by Dr. Iyad Al Khatib

[2] QoS Analysis for Signaling in VoIP Client and Server for Multicore, by Muhammad Adnan; http://kth.diva-portal.org/smash/record.jsf?pid=diva2:523738; last visited December 18, 2012

[3] ITU-T E.800, 'Terms and Definitions Related to Quality of Service Engineering', 1994

[4] Oodan, A.P., Ward, K.E. and Mullee, A.W. 'Quality of Service in Telecommunications', the institution of Electrical Engineers, London, 1997

[5] VoIP standard bodies and vendors http://www.javvin.com/protocolVOIP.html; last visited June 30, 2012

[6] Maximum number of users by mizu VoIP company; http://www.mizu-oip.com/Products/VoIPServer.aspx; last visited: January 29, 2012

[7] QoS benchmarking; http://www.businessdictionary.com/definition/benchmark.html; last visited January 29, 2012

[8] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Application", STD 64, RFC 3350, July 2003.

[9] Zahariadis, T.; Spanos, S,; "The clear voice [SIP vs H323]," Communications Engineer, vol. 2, no.2, pp. 14-17, April-May 2004

[10] Session Initiation protocol: http://tools.ietf.org/pdf/draft-ietf-sip-rfc2543bis-09.pdf; last visited January 30, 2012

[11] Basic SIP functionality, http://www.siptutorial.net/SIP/function.html; last visited: January 30, 2012

[12] Asterisk Introduction, http://asterisks.com/an-introduction-to-asterick; last visited: March 28, 2012

[13] Building Telephony for OpenSIPS 1.6, ISBN 978-1-849510-74-5, by Flavio E. Goncalves; Opensips-cp3

[14] Introduction to OpenSIPS Server, www.Opensips.org, last visited: January, 30, 2012

[15] Iperf network bandwidth monitoring and test tool, http://www.hosting.com/support/info/using-iperf-to-test-network-bandwidth; last visited: April 03, 2012

[16] Muhammad Adnan, master thesis report QoS benchmarking in VoIP client and server communication for multicore, Industrial supervisor Dr. Iyad Al Khatib at KTH – The Royal Institute of Technology, Sweden 2012.

[17] Introduction of Mysql; http://dev.mysql.com/doc/refman/5.0/en/what-is-mysql.html; last visited July 2, 2012.

[18] Introduction of wireshark; http://www.wireshark.org/docs/wsug_html_chunked/ Chapter Introduction. html; last visited: January 30, 2012

[19] http://www.voip-info.org/wiki/view/Opensips+Installation,+How+to.; Installation and configuration of OpenSIPS with Mysql database and tables; last visited: January 30, 2012

[20] http://vidodz.wordpress.com/2009/07/28/install-opensips-on-debian-or-ubuntu/ last visited: January 30, 2012

[21] Installation of Mysql client server: http://www.mysqlfaqs.net/mysql/Client-Server-Commands; last visited; January 30, 2012

[22] Linux ODBC http://www.easysoft.com/developer/interfaces/odbc/linux.html; last visited: February 06, 2012.

[23] Introduction to Apache server: http://www.modulehosting.com/apache.html; last visited: July 2, 2012.

[24] Installation of OpenSIPS server by compilation method: http://forums.debian.net/viewtopic.php?f=30&t=47645; last visited November 14, 2012