Performance Evaluation of Various Open source Projects that provide SIP functionality

Viswavardhan Reddy K, Shiva Chaitanya Nallapati, Vishnuvardhana Reddy K

Abstract— In recent times the usage of VoIP services has increased tremendously. There are many signaling protocols such as Bearer Independent Call Control (BICC), H.323, Media Gateway Control Protocol (MGCP), Session Initiation protocol (SIP) etc., that are used for establishment of connections and to carry out voice and video data services. SIP has become popular because of its easy implementation, flexibility and good scalability. Choice of open source SIP server software is important when deploying in a VoIP based network. So we need to evaluate the performance of open source SIP server software's in an ideal condition before it is deployed to real environment. In this paper we evaluate and compare the performance of three open sources SIP server software's which are quite popular. A SIPp traffic generator tool is used to generate scenario's namely Registration with authentication, Registration without authentication, Session establishment, and Session establishment with response delay at User Agent Server (UAS) side. Using the scenario's mentioned above, open source SIP server software's performance can be evaluated based on the parameters such as Registrations per second, Calls per second, Response delay, and Percentage of successful calls and registrations. From the obtained results, we observed that there is a significant performance difference among the SIP server software's. OpenSIP Server is the best open source SIP server software for the scenario's Registration with authentication, Registration without authentication and Session establishment with response delay. Asterisk server is the best open source SIP server software when compared with the other two servers for the scenario Session establishment.

Index Terms— Authentication, Delay, Registration, RTP, Server Performance, SIP, SIP servers, SIPp, Throughput, VOIP.

———————————— ◆ ————————————

1 Introduction

Voice and Video over Internet Protocol (VoIP) is one of the rapidly growing real-time internet applications in this modern world. VoIP refers to the technology that sends the voice and video data over IP networks to the end users [1]. Session Initiation Protocol (SIP) is one of the protocols that were widely used in the field of research. It is popular because it establishes media sessions and instant messaging [2, 3]. There are several IP soft phones [4, 5] and hard phones [6] available in the market that are capable of providing SIP services.

SIP [7-10] protocol specifications are available in several Request for Comments (RFC's); the first proposed standard version (SIP 2.0) was defined by RFC 2543. This version of the protocol was further refined and clarified in RFC 3261.It is an application layer signaling protocol that can establish, modify and terminate the multimedia sessions between two or more participants [11]. Multimedia sessions include voice, video, chat, interactive games and internet telephony calls [12]. The main purpose of the SIP is to establish the signaling path between the user agents. For media transportation there are two protocols that are often used with SIP, they are

Real Time Transport Protocol (RTP) [13-14]: it is used to carry media streams (voice, video) over the internet.

Real Time Transport Control Protocol (RTCP) [13-14]: it is used to transmit control packets to the participants in the session.

SIP runs on transport layer protocols [12] [15] [16] such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Transport Layer Security (TLS) and Stream Control Transmission Protocol (SCTP). SIP has network elements like User Agents (UAs) and SIP servers. UAs are classified into user agent client and user agent server.

SIP server acts as Proxy server by forwarding SIP messages (request and response messages) between User Agent Client (UAS) and User Agent Server (UAS), Register server by registering UAC and UAS, Redirect server by updating the user location and redirecting the messages or calls to the users who go to another location.

Previous researches [16-21] which were carried out have found that factors like hardware resources, cpu/memory usage, network usage, call setup time, call arrival rates, call hold time and delay in call setup affects the performance of open source SIP server software's. we have found that server performance can be evaluated by the response delay of both open source SIP server software and User Agent Server (UAS) while establishing a call . Response delay affects the number of sessions that are maintained in an open source SIP server. To the end so far no attempt has been made on the comparison of various open source SIP server software's based on parameters like number of successful calls, successful registrations, total number of calls created, percentage of successful calls and how the response delay affects the open source SIP server software's.

2 Motivation

Nowadays there are several open source projects available in the market, providing SIP services like session establishment and transferring of video and audio streams to the end users. Even though many open source projects have been used for performance evaluation, still confusion prevails in choosing the right open source SIP server software's.

The time to setup a call and tear down a call depends on response delay of both open source SIP server software's and UAS. A session is the combination of all SIP request messages and response messages between the end users. The number of sessions maintained in an open source SIP server software's are mostly affected by the response delay at UAS between "180 Ringing" and "200 Ok". Increase in the response delay may degrade the open source SIP server software's performance during the call setup time. During the response delay, the server has to handle the messages which are sent to UAS from UAC.

In this paper a study has been made on how the user response delay affects the open source SIP server software's performance and the comparison of three open source SIP server software's. We also recommend the most appropriate Open source SIP server software for a specific application scenario.

3 SIP OVERVIEW

SIP is an application layer signaling protocol which is designed and developed by the Internet Engineering Task Force (IETF). SIP is a client-server protocol used to establish, modify and terminate the multimedia sessions between two or more participants [11]. Multimedia sessions include voice, video, chat, interactive games and internet telephony calls [12].

3.1 SIP ARCHITECTURE

It contains two basic SIP entities. They are SIP User Agent (UA) and SIP servers.

UA represents end point entity, and is usually a software application running on end users systems. UA initiate and terminate sessions by exchanging requests and responses. UA is a logical entity that can act as both User Agent Client (UAC) and User Agent Server (UAS). Some of the examples of UA are soft phones, hard phones, web phones, messaging clients etc.

User Agent Client (UAC):
It is a client application in the SIP systems that allows to initiate SIP request messages that are sent to UAS and they receive response messages from UAS.

User Agent Server (UAS):
It is a server application in the SIP system that allows receiving request messages from UAC and generating the response messages to UAC.

SIP servers:
There are three logical entities [12]. They are Proxy server, Redirect server, Register server. The functions of each server are explained below

Proxy server:
The main role of proxy server is to forward the SIP request messages and response messages to other SIP servers or to intended user.

Redirect server:
The duty of redirect server is to accept the request messages and if there is no user at previous location, it will re-direct the request message to the same user who goes to another location.

Register server:
It is a database that contains the list of all registered users. Users can register and update their location information at register server.

SIP Addressing Scheme:
The users in the SIP are identified using Uniform Resource Identifier (URI). Using SIP URI users can communicate with each other. The syntax of the SIP URI [12] is given as

"sip:user:password@host:port;Uri-parameters?headers"

Description of SIP URI is:
Sip: it is the protocol that we are using. User: it indicates the user-name in the domain. Password: it is for the user. Sending passwords in the SIP URI is not recommended. Host: this is the mandatory field that represents the domain name like "open-ims.test" or IP address like 192.168.1.100. Port: it represents the port number. Requests are sent and received through the specified port. It is an optional field and the default port number is 5060 [12]. URI-parameters: it is the form of parameter-name and parameter-value which can affect the request. Header: it is a part of the request that is made from SIP URI. The following are the examples of SIP URI:

sip:alice@open-ims.test
sip:alice@192.168.1.100
sip:alice:alice@open-ims.test; transport=tcp
sip:+46760880641@sudan.se;user=phone

SIP Messages:
SIP is text-based protocol similar to Hypertext Transfer Protocol (HTTP). The context of the message can be clearly visible and readable by human beings, and can be changed easily. There are two types of SIP messages namely SIP Request Messages and SIP Response Messages [12].

SIP Request Messages:
There are six request messages defined in RFC 3261. They are REGISTER request message: it is used to register the UA. INVITE request message: it is the first request message to establish a session. ACK request message: this is the acknowledgment sent by the UA to confirm that he had received the final response for the invite request. BYE request message: it is used to tear down the sessions. OPTIONS request message: querying open source SIP server software's about the UA. CANCEL request message: it is used to cancel the pending sessions.

SIP Responses Messages:
UAS generates SIP response messages for the SIP request messages sent by a UAC. SIP response messages are classified in to 6 classes like 1XX, 2XX, 3XX, 4XX, 5XX and 6XX. Each class of SIP response messages are represented in integer numbers. For instance, class 1XX ranges from 100 to 199 SIP response messages and is similar to rest of all classes. SIP response message contains response code and reason phrase

like “100 Trying”. Where 100 represent response code and trying indicate reason phrase.

The following are 6 classes of SIP response messages:

1xx are Provisional response messages: it comes from the server and indicates that the invite request message progress is continuing. UAC stops re-transmission of invite packets as soon as 1xx response message is received by him.
2xx are Success response messages: it indicates that the SIP request messages are successfully completed.
3xx are Redirection responses messages: it is used to retrieve the present UA information when the UA moves from one location to another location.
4xx are Client error response messages: it indicates error in the SIP request message.
5xx are Server error responses messages: it is generated, when the server fails to fulfill the valid request message.
6xx are Global error response messages: it is sent by the open source SIP server software, if the request message is not fulfilled at any server.

SIP message body:
SIP message body comprises of series of messages. Each message is divided into three sections. They are, method used to send SIP messages (request and response messages), Header fields of a SIP message and SDP message description. To understand the body of a SIP message, we illustrate with the following example.
Example: SIP message body for an invite request message.
Section 1: first field of a SIP message is INVITE, which represents the method to send SIP request message, followed with SIP URI, port number to communicate and version of SIP.

INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0

Section 2: it explains about header fields of a SIP message body.

```
Via:
SIP/2.0/[transport][local_ip]:[local_port];branch=[branch]
From:
bob<sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[cal
l_number]
To:alice<sip:[service]@[remote_ip]:[remote_port]>
Call-ID: [call_id]
CSeq: 1 INVITE
Contact: sip:sipp@[local_ip]:[local_port]
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: [len]
```

Via: this field tells about the reply of a SIP request message needs to be sent back to original initiator (UAC) who initiates the SIP request message.
SIP/x.x/ : it is the version number of SIP. Example: SIP/2.0/
[transport]: it indicates the protocol used for sending SIP messages. Example: UDP or TCP or TLS.
[local_ip]: it indicates the SIP URI of an original initiator. Example: sipp@192.168.1.99, alice@client.server.com

[local_port]: it represents port number of an original initiator from which the message is sent. Example: 5060, 4085, etc.
[branch]: it indicates the branch number of the message. Example: 4kjDjik54JK, UOKhjd34ikm, etc.

From: this is used to identify the initiator of a request.
[sname]: name of the sender. Example: bob, alice, etc.
[pid]: process id. Example: 10402, 20674, etc.
[call_number]: it starts from 1 when the request message is initiated, and is incremented by 1for each successful call.

To: this field indicates the details of a receiver (UAS).
[name]: name of the receiver. Example: bob, alice, etc.
[service]: indicates service name. Example: alice.
[remote_ip]: it is the URI of a receiver. Example: alice@client.server.com.
[remote_port]: it indicates the port number of a receiver, to which the message has to be sent. Example: 5060, 4085, etc.

Call-ID: it is a unique identification of initiator (UAC) who is in global network.
Cseq: it is the command sequence number starts with 1 and is incremented by 1 for each new request message.

Contact: it contains SIP URI and port number of UAC. If UAS wants to communicate back with the UAC, UAS uses UAC contact details.

Max-Forwards: it limits the number of open source SIP servers that can forward the SIP request messages.

Content-Type: it indicates description of the SIP message body.

Content-Length: it represents the length of a SIP message.

4 Methodology

The main aim is to compare the performance of various open source SIP server software’s and to decide the best open source SIP server software for a specific application scenario. In this, we present the test-bed that has been designed, hardware and software components, configuration procedures, and the experimental scenario’s used for comparison purpose.

4.1 Test-Bed Description:

The Test-Bed consists of 3 PCs connected with a 10/100 Mbits/sec Fast Ethernet Switch. The experimental environment is schematically shown in Fig. 1. The experiments were conducted in Linux operating system by using three different open sources SIP server software’s namely Asterisk, OpenIMSCore and Open SIPS. The traffic is generated via SIPp traffic generator tool [22]. The entire SIP signaling between the two clients and server is monitored by Wireshark [23] and ‘tcp dump’.

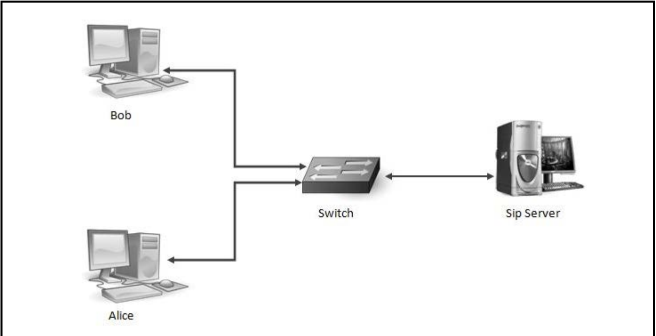


Fig. 1.Test-Bed for experiments.

4.2 Hardware Components:

Three Desktops are employed for performing experiments. Two of them works as SIP UAs, and the rest one acts as the open source SIP servers. Their specifications are reported as follows:

Three straight data cables with RJ45 connectors are used to connect UAs and open source SIP server to switch.

TABLE 1
HARDWARE SPECIFICATIONS OF SIP UAS AND OPEN SOURCE SIP SERVERS

HOST	HARDWARE	SOFTWARE
Client PC1	Intel®Core ™2CPU T6600, 2.4GHz, 3GB RAM	Ubuntu 10.04(lucid)
Server PC	Intel®Core ™2CPU P7450, 2.13GHz, 4GB RAM	Ubuntu 10.04(lucid)
Client PC2	Intel®Core ™2CPU T6600, 2.4GHz, 3GB RAM	Ubuntu 10.04(lucid)
Switch	D-Link Model: DES-1008D, 8 Port 10/100 Fast Ethernet switch	

Parameters that affect the performance of open source SIP server software's are described below:
Registrations per second: it is defined as number of registrations per second
Calls per second: it is defined as number of calls completed with "200 OK" SIP response message.
Percentage of successful calls: it is defined as ratio of number of successful calls per second to total number of calls per second. And it is similar with percentage of successful registrations.
Total number of calls created: it represents total number of calls created for each test.
Response delay: It represents total number of seconds that a open source SIP server software can handle all the incoming SIP request messages from UAC to UAS and SIP response messages from UAS to UAC.

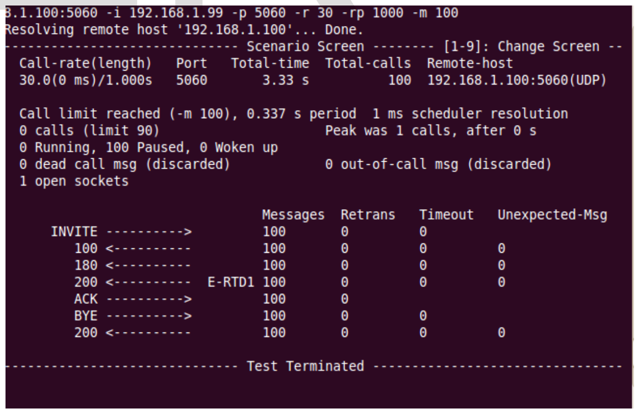
4.3 Software Components:

In this section software's that are used in experiments are described in detail

Ubuntu:
This is an open source operating system available on the web for free, where anyone can download, burn it in to a CD, or even order a CD for free. The installation is easy, just to put the CD in the drive, reboot the PC and then follow the installation wizard. The version of the operating system used for the SIP server and for User Agents is:
UBUNTU Release 10.04(lucid) with Kernel Linux version: 2.6.32-29-generic

Client software: SIPp
SIPp [22] is a free open source test tool, traffic generator and performance testing tool for the SIP protocol. SIPp has few scenarios (UAC & UAS) which were written in XML (Extensible Markup Language). With the help UAC and UAS scenarios SIPp is able to establish and release multiple calls. SIPp runs as both SIP UAC and UAS in the same computer with different port numbers or in different computers with the same port number.

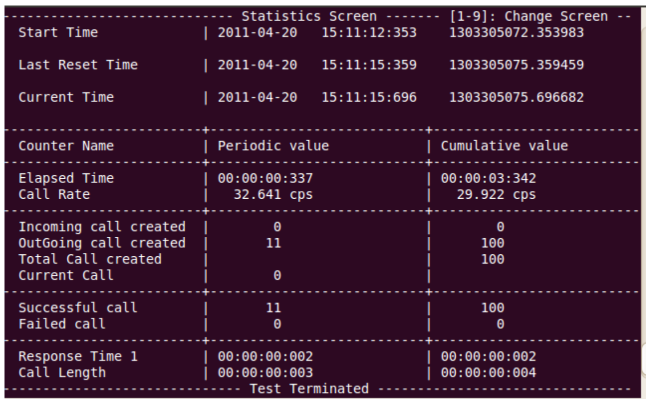
A screen shot representing the scenario which displays call flow, call rate, port number, remote host, port number and the protocol used etc is shown in Fig 2.



```
8.1.100:5060 -i 192.168.1.99 -p 5060 -r 30 -rp 1000 -m 100
Resolving remote host '192.168.1.100'... Done.
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
  30.0(0 ms)/1.000s   5060      3.33 s          100  192.168.1.100:5060(UDP)

  Call limit reached (-m 100), 0.337 s period  1 ms scheduler resolution
  0 calls (limit 90)                      Peak was 1 calls, after 0 s
  0 Running, 100 Paused, 0 Woken up
  0 dead call msg (discarded)             0 out-of-call msg (discarded)
  1 open sockets

                                 Messages  Retrans   Timeout   Unexpected-Msg
      INVITE ---------->         100       0         0
         100 <----------         100       0         0         0
         180 <----------         100       0         0         0
         200 <----------  E-RTD1 100       0         0         0
         ACK ---------->         100       0
         BYE ---------->         100       0         0
         200 <----------         100       0         0         0

------------------------------ Test Terminated --------------------------------
```

Fig. 2. Establishing a connection.

A screenshot representing the statistics screen is shown in Fig. 3. In SIPp the available statistics are: incoming call represents total number of incoming calls from UAS to UAC, outgoing call is defined as total number of outgoing calls from UAC to UAS, total call created is number of calls created from UAC, current call represents number of current calls ongoing, successful call and failed call are defined as total number of successful calls and failed calls, response time is the time duration to get a response from UAS, call length is the time taken to complete the call.



```
----------------------------- Statistics Screen ------- [1-9]: Change Screen --
  Start Time             | 2011-04-20   15:11:12:353    1303305072.353983

  Last Reset Time        | 2011-04-20   15:11:15:359    1303305075.359459

  Current Time           | 2011-04-20   15:11:15:696    1303305075.696682

-------------------------+---------------------------+--------------------------
  Counter Name           | Periodic value            | Cumulative value
-------------------------+---------------------------+--------------------------
  Elapsed Time           | 00:00:00:337              | 00:00:03:342
  Call Rate              |   32.641 cps              |   29.922 cps
-------------------------+---------------------------+--------------------------
  Incoming call created  |        0                  |        0
  OutGoing call created  |       11                  |      100
  Total Call created     |                           |      100
  Current Call           |        0                  |
-------------------------+---------------------------+--------------------------
  Successful call        |       11                  |      100
  Failed call            |        0                  |        0
-------------------------+---------------------------+--------------------------
  Response Time 1        | 00:00:00:002              | 00:00:00:002
  Call Length            | 00:00:00:003              | 00:00:00:004
------------------------------ Test Terminated --------------------------------
```

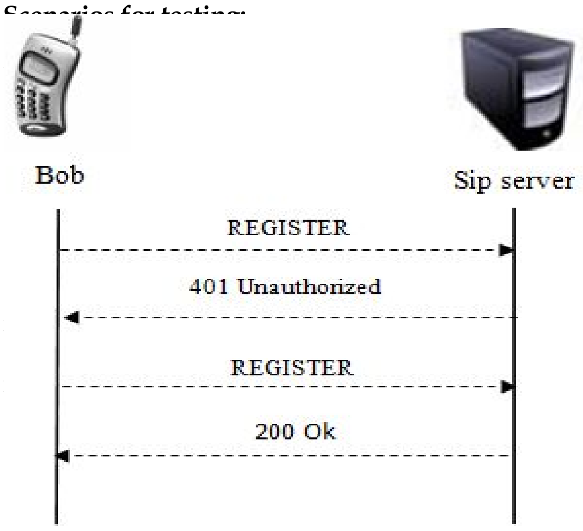
Fig. 3. Representing the cumulative and periodic values.

4.3 Test-Bed Validation:

The experiments are conducted based on the test-bed described in Fig. 1. The major objective of the established test-bed is to help us in studying and analyzing the performance of three different open sources SIP server software's. To do this four different scenarios are adopted and are described as follows.

[1]. Registration with authentication
[2]. Registration without authentication
[3]. Session establishment and
[4]. Session establishment with response delay.

The open source SIP server software's performance is evaluated based on four different parameters namely Registrations per second, Number of calls per second, Percentage of successful calls and how the response delay affects SIP servers.



d with the
n with the
r instance,
the open
r request
vith a 401
with the
s again the
server by
ion. Once
source SIP
ob into its
e message

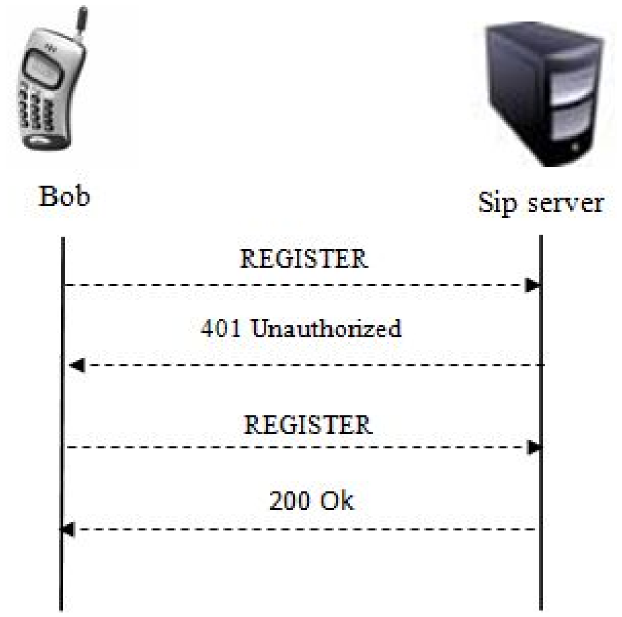


Fig. 4. Bob registering at open source SIP server.

Registration without authentication:
This is a registration scenario where UA get registered with the open source SIP server without the username and password as shown in Fig 5. For instance, Bob sends a SIP Register request message to the open source SIP server. After receiving the SIP register request message, open source SIP server sends a response message of "100 Trying" to bob. Meanwhile open source SIP server processes the request and sends a response message of "200 OK" to the bob. Now, bob is registered without username and password.

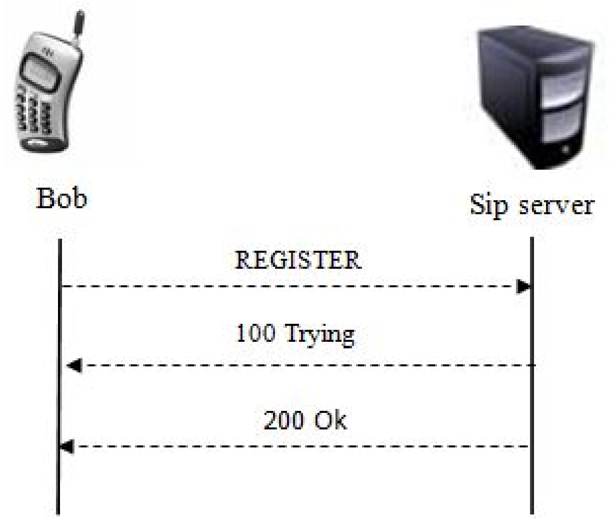


Fig. 5. Registration without authentication

Session Establishment:
Session establishment takes place through three-way handshake among Bob, open source SIP server and Alice. In Fig 6 Bob (UAC) sends an invite request message to Alice (UAS) through open source SIP server. The open source SIP server receives the invite request message and sends "100 Trying" response message to Bob. Then the open source SIP server forwards the invite request message from Bob to Alice. Once invite request message reaches Alice, he sends "180 Ringing" response message to open source SIP server. The open source SIP server receives "180 Ringing" response message and forwards it to Bob. Now Alice sends "200 OK" response message to the open source SIP server and is forwarded to the Bob. Bob sends an "ACK" request message to the open source SIP server and is forwarded to Alice indicating that the session from Bob is accepted by Alice. After the conversation, if Bob wants to end the session, he sends a "BYE" request message to the open source SIP server. The open source SIP server receives "BYE" request and forward to Alice. Alice receives bye request message and teardown the session. Alice sends "200 OK" response message to the open sources SIP server confirming the disconnection of session. Bob receives "200 OK" response message from the open source SIP server and releases the session.

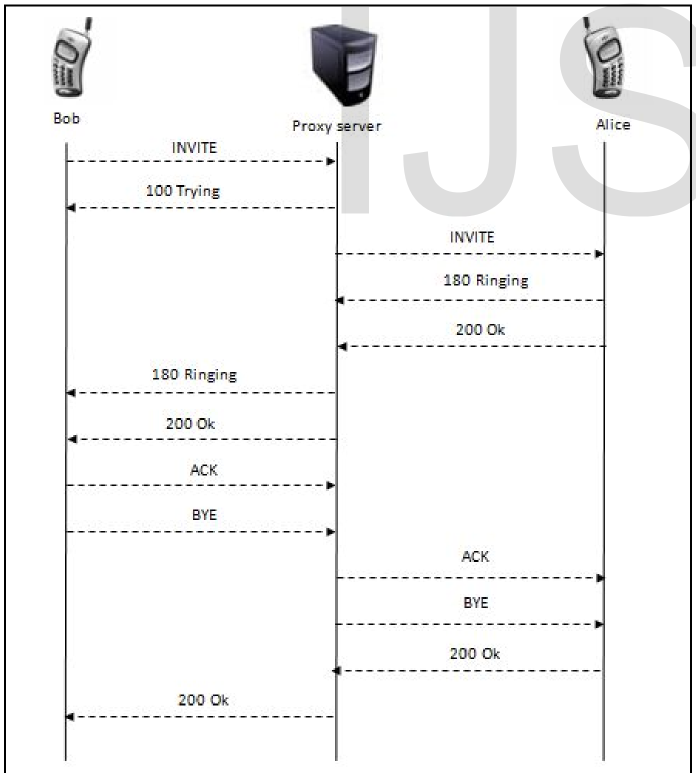


Fig. 6. Session Establishment through proxy server

Session Establishment with Response Delay:
Bob sends to Alice an "INVITE" request message through the open source SIP server. The request message is an invitation to Alice from Bob to participate in a session. The invite request message will be processed by the open source SIP server and send "100 Trying" response message to Bob. Then the open source SIP server forwards the invite request message from Bob to Alice. Once this request message reaches Alice, he generates "180 Ringing" response message to the open source SIP server and forwards the response message to Bob. In order to test the performance of an open source SIP server we introduce response delay between "180 Ringing and "200 OK". After "180 Ringing" Alice generates "200 OK" response message to the open source SIP server which in turn forwarded to Bob. Once "200 OK" response message received by Bob, he sends an "ACK" request message to the open source SIP server. The "ACK" request message from the SIP server is forwarded to Alice indicating that session is established between Bob and Alice. When Bob wants to end the session, he sends a "BYE" request message to the open source SIP server. The bye request message is then forwarded to Alice. After receiving the bye message from open source SIP server, Alice terminate the established session. Now Alice send a "200 OK" response message to the open source SIP server confirming disconnection of session. The open source SIP server then sends "200 OK" request message to Bob as shown in Fig 7.

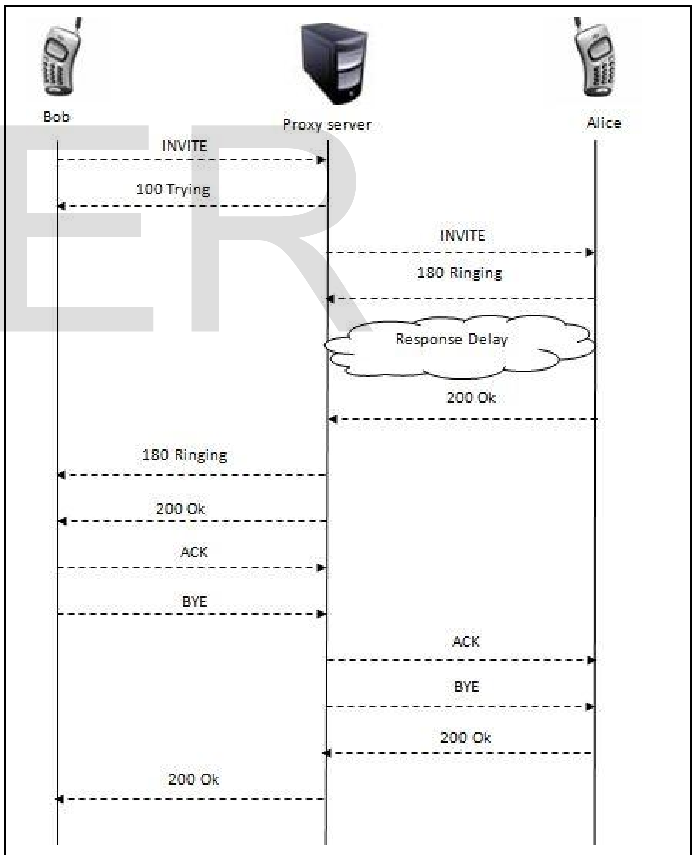


Fig. 7. Session Establishment with response delay

4.3 EXPERIMENT PROCEDURE:

We performed pilot runs of each OS-SIP Server Software, to know how many RPS that a server can handle. From the results obtained we understood that each server was able to handle only 600 RPS. Based on this pilot study we have further conducted our actual experiments on each OS-SIP Server Software (Asterisk, OpenSIPS and OpenIMSCore). Before starting the actual experiments we developed XML files for the above scenarios mentioned and injected them both at the UAC and UAS. We installed the SIPp traffic generator tool both the UAC and UAS side and using this tool we have generated traffic to the server.

For the scenarios 1, experiments are conducted on OS-SIP Server Softwares with 10000 as total number of registrations. We generated call rates starting with 100 RPS at the UAC until 10000 registrations are completed. The procedure is repeated by increasing call rate from 100 to 600 until 10000 registrations are completed. We repeated the above procedure for 2 times and have observed there were variations in the successful registrations. To overcome this we have run the scenario 1 for 10times and calculated the average results of the successful registrations.

For scenario 2, we have run the tests in the same way as for scenario 1.

For scenario 3, we have run the tests in the same way as for scenario 1 and scenario 2 but we limited the CPS to 5000. We observed there were variations in successful calls when we made pilot run for 2 times. So we have run the scenario for successful calls 10times and calculated the average results.

In the scenario 4, we have made a pilot run with 10 CPS cumulatively in- creasing the cps upto 100 CPS with the response delay varying from 5 to 25seconds with an increment of 5seconds. Initially, we have run the test with increasing the cps with 10cps and found there was no variation until 30 CPS with a response delay of 5 seconds. From 40 to 60 CPS there was huge variation with same response delay. To overcome this and to note the variations we have rerun the tests by taking 30 CPS initially and incrementing them with 30 CPS every time until 120 CPS. As there were variations in the successful calls with response delay, we have run the scenario for 10times and calculated the average result of the successful calls. Scenario 3 focuses on session establishment and scenario 4 focuses on session establishment with response delay. As both the scenarios mainly focus on session establishment and as to compare the results we have considered 5000 total number of calls to be best solution.

The SIPp generator tool installed at the UAC generates log files. The log file contains data regarding successful registrations, total number of registrations, RPS, total number of calls created, successful calls and CPS. Matlab is used to extract the data from log files. For scenario 1 and scenario 2 graphs are plotted between RPS and Percentage of successful registrations. For scenario 3 and scenario 4 graphs are plotted between CPS and Percentage of successful calls.

5. Results:

In this section, we discuss the comparision of three OS-SIP Server Softwares, namely Asterisk, OpenSIPS, OpenIMSCore.

Performance evaluation of three different open sources SIP server software's:

Registration with authentication:

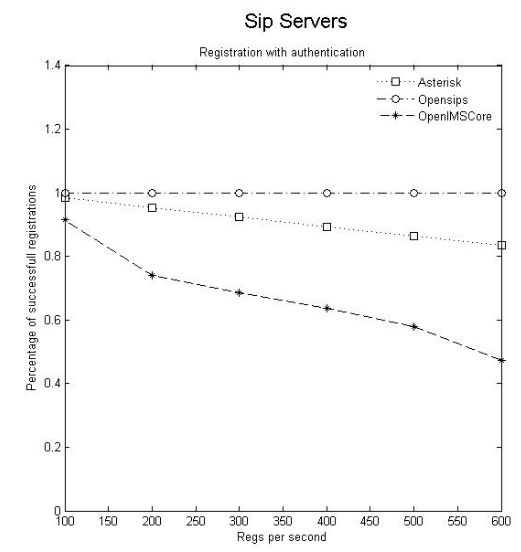


Fig. 8. Performance evaluation of three open source SIP server softwares for the scenario Registration with authentication.

Performance evaluation of three open source SIP server softwares for the scenario Registration with authentication.

For Asterisk SIP server software:

The obtained graph for Asterisk open source SIP software is mostly a straight line. From the figure 8 shown above, we found that there is a linear decrease in percentage of successful registrations with a linear increase in registrations per second (rps). So here, rps is inversely proportional to percentage of successful registrations.

For instance, percentage of successful registrations is high at 100 rps whereas from 200 – 500 rps, percentage of successful registrations decreases linearly (around 3%) and less percentage of successful registrations happens at 600 rps i.e. 83.4%.

For OpenSIPS SIP server software:

The plot between rps and the percentage of successful registrations yielded a straight line and is independent of rps. It can be seen from the fig 8, that there is no change in the percentage of successful registrations with a linearly increasing number of rps. For 100-600 rps, the percentage of successful registrations is 100%.

For OpenIMSCore SIP server software:

For OpenIMSCore, the curve is monotonically decreasing and it can be seen that from 100-200 rps there is a sudden decrease in percentage of successful registrations (from 91.4% to 73.9%), whereas from 200 rps the percentage of successful registrations decreases gradually with increasing rps.

At 100 rps, there are 91.48% successful registrations and at 200 rps percentage of successful registrations suddenly decreases to 73.98. From 200-500 rps, percentage of successful registrations decreases gradually (in between 4-6%). And at 600 rps there is less percentage of successful registrations

TABLE II
PERFORMANCE EVALUATION OF THREEOS-SIP SERVER SOFTWARES FOR THE SCENARIO REGISTRATION WITH AUTHENTICATION.

Calls Per Second (sec)	Percentage of successful calls for three different OS-SIP Server Softwares		
	Asterisk	Opensips	OpenIMSCore
100	0.9839	1	0.9148
200	0.9535	1	0.7398
300	0.9239	1	0.6847
400	0.8927	1	0.6372
500	0.8638	1	0.57847
600	0.834	1	0.473

(47.30%).

Comparison of three OS-SIP Server Softwares:

At 100 RPS, all the three curves are relatively close to each other as the percentage of successful calls for the three OS-SIP Server Softwares are above 90. At 200 RPS, OpenIMSCore handled 73.98% of successful calls. Open-SIPS and Asterisk are very high in successful calls with a difference around 5%. And the same phenomenon is observed for all the three curves from 300 to 600 RPS. Hence, from the above figure 4.1 we can observe that the OpenSIPS shows better performance than the remaining two (Asterisk and OpenIMSCore) SIP server softwares.

Registration without authentication:

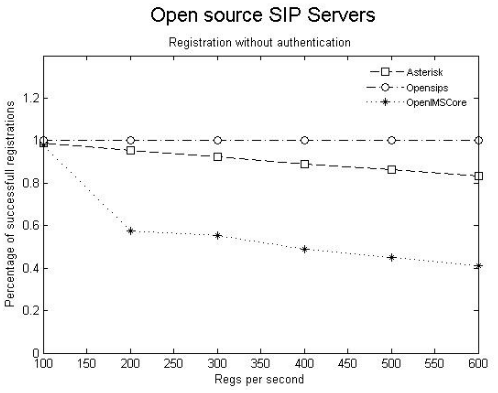


Fig.9. Performance evaluation of three open source SIP server softwares for the scenario Registration without authentication.

Performance evaluation of three open source SIP server softwares for the scenario Registration without authentication.

For Asterisk SIP server:

The plot for Asterisk SIP server is almost a straight line. From 100 rps, the percentage of successful registrations decreases from 98 to 85, at a constant rate of 3% for every 100 rps. That is, as the number of rps increases, the percentage of successful registrations decreases linearly.

In the fig 9, initially the percentage of successful registrations at 100 rps is 98.8 and thereafter at 200 rps it is 95.5% and at 300 rps the percentage of successful registrations is 92.5. Similarly decrease in the percentage of successful registrations at the rate 3% is observed up to 600 rps.

For OpenSIPS:

From the above fig 9, OpenSIPS curve is a straight line which indicates that the percentage of successful registrations remains constant for the whole scenario. The percentage of successful calls is 100% and this remains the same from 100 rps to 600 rps.

For OpenIMSCore open source software:

It can be observed from the fig 9, that the resultant plot for OpenIMSCore is a continuously decreasing curve. At 100 rps, the percentage of successful registrations is 97.9 and for 200 rps the percentage of successful registrations decreases suddenly to 57.5%. This huge difference is not observed further, and from 200 – 600 rps it decreases slightly (in between 2-7%) for every 100 rps.

TABLE III
PERFORMANCE EVALUATION OF THREE OS-SIP SERVER SOFTWARES FOR THE SCENARIO REGISTRATION WITHOUT AUTHENTICATION

Calls Per Second (sec)	Percentage of successful calls for three different OS-SIP Server Softwares		
	Asterisk	Opensips	OpenIMSCore
100	0.9888	1	0.9796
200	0.9554	1	0.5756
300	0.9253	1	0.5538
400	0.89	1	0.4888
500	0.8652	1	0.4508
600	0.8352	1	0.4105

Comparison of three OS-SIP Server softwares:

The percentage of successful calls for all the three OS-SIP Server Softwares is nearly same at 100 RPS. At 200 RPS, this percentage is high for Open- SIPS and Asterisk whereas for OpenIMSCore it is very low (57.5%) when compared to the other two OS-SIP Server Softwares. At 300 RPS, Asterisk has high percentage of successful calls (92.5%) than that of OpenIMSCore (55.3%), but both are less than that of OpenSIPS which is 100% throughout the scenario. The same is observed for 400 RPS, 500 RPS and 600 RPS but with different values. Percentage of successful calls for Asterisk and Open- IMSCore at 400, 500 and 600 RPS are 89%, 48.8%; 86.5%, 45%; 83.5% and 41% respectively. By the above graph, it is evident that the performance of OpenSIPS is higher than the remaining OS-SIP Server Softwares namely Asterisk and OpenIMSCore

Session establishment:

For Asterisk:

From the figure 10, Asterisk curve is a straight line from 100 to 500 CPS and at 500 CPS sudden fall in percentage of successful calls is observed. Percentage of successful calls is 100 at 100 CPS and with a linear increase in CPS there is no change in the percentage of successful calls until 500 CPS. For 600 CPS, the percentage of successful calls decreases to 72.12%.

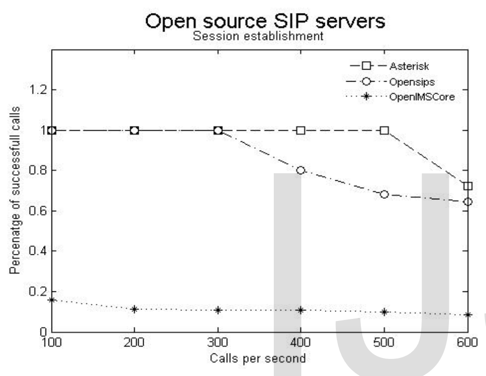


Fig.10. Performance evaluation of three open source SIP server softwares for the scenario Session Establishment.

Performance evaluation of three open source SIP server softwares for the scenario Session establishment.

For OpenIMSCore:

The figure 10 shows that OpenIMSCore curve decreases gradually with a linear increase in the number of CPS. At 100 CPS, the percentage of successful calls is 15.6 and starts with decreasing gradually, and at 600 CPS the percentage of successful calls is 8.5%.

For OpenSIPS:

It can be observed from the figure 4.3, that OpenSIPS curve is a straight line from 100 cps to 300 cps. From 300 cps, it bends down gradually which indicates the decrease in percentage of successful calls.

From 100 to 300 cps there is a slight decrease in the percentage of successful calls (99.99% at 100 cps, 99.94% at 200 cps and 99.96% at 300 cps). From 300 to 400 cps, there is a sudden change in the percentage of successful calls. At 400 cps the percentage of successful calls is 80.2, 68% of successful calls happens at 500 cps and at 600 cps it is 64.5%.

TABLE IV
PERFORMANCE EVALUATION OF THREE OS-SIP SERVER SOFTWARES FOR THE SESSION SCENARIO SESSION ESTABLISHMENT

Calls Per Second (sec)	Percentage of successful calls for three different OS-SIP Server Softwares		
	Asterisk	Opensips	OpenIMSCore
100	1	0.999	0.1566
200	1	0.9994	0.1136
300	1	0.8022	0.1082
500	1	0.68	0.0994
600	0.7212	0.6456	0.085

At 100 CPS, the percentage of successful calls for Asterisk and OpenSIPS is almost same and is very high when compared to that of OpenIMSCore value (15.66%). This relation between these three curves holds same till 300 cps but with different percentages of successful calls. At 400 cps, OpenSIPS value decreases to 80.22% while Asterisk value remains the same (100%) and OpenIMSCore value changes slightly at 300 cps. At 500 cps, Asterisk has 100 % of successful calls, OpenSIPS value is 68% and OpenIMSCore value is 9.9%. At 600 cps, Asterisk falls down from 100% to 72.12% of successful calls, OpenSIPS value falls to 64.5% and OpenIMSCore value falls to 8.5%. From the above graph, it is evident that among the three open source softwares, Asterisk has high performance and OpenIMSCore has least performance.

Session establishment with response delay:

For Asterisk:

For 30 CPS, Asterisk curve is a straight line with maximum percentage of successful calls (100%) and is constant with increasing the response delay. Whereas for 60 CPS at 5 seconds response delay, this curve drops down indicating the fall in percentage of successful calls (from 100% to 25.6%). With further increasing in response delay, the curve decreases gradually. For 90 CPS and 120 CPS, the curve behaves in a similar way as that of at 60 CPS, with some changes in the percentage of successful calls.

For OpenSIPS:

Irrespective of number of CPS and response delay, OpenSIPS curve behaves in a unique way with 100% of successful calls throughout the scenario. For all 30, 60, 90 and 120 CPS there are 100% successful calls. This curve is a straight line and as the response delay increases linearly (from 5-25 seconds), percentage of successful calls remains same (100%).

For OpenIMSCore:

For 30 CPS, OpenIMSCore curve decreases gradually with a linear increase in response delay. For 60 CPS, this curve decreases from 5-15 seconds of response delay and from 15 seconds of response delay, it increases up to 25 seconds of response delay. For 90 and 120 CPS, this curve behaves randomly. It decreases sometimes and later increases, with a linear increase in response delay.

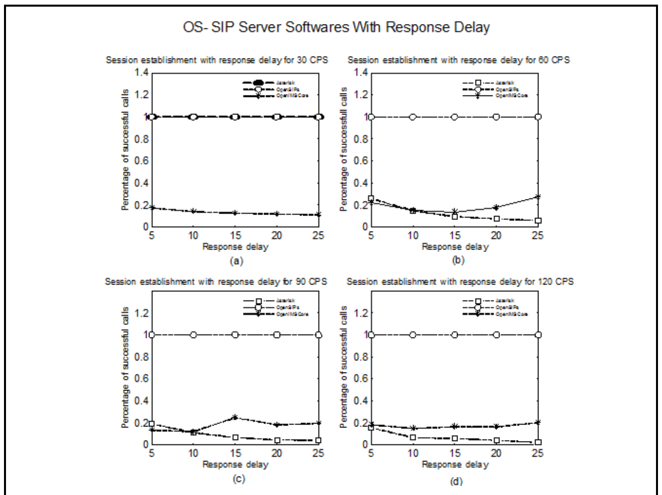


Fig.11. Performance evaluation of three open source SIP server softwares for the scenario Session Establishment with Response Delay.

Comparison of three OS-SIP Server Softwares:

The performance of OS-SIP Server Softwares is observed at 30, 60, 90 and 120 CPS. Firstly; keeping the cps constant and later by increasing the response delay linearly, the performance of all the three OS-SIP Server Softwares is observed.

Due to the limitation in the space, we are not adding the tables for the performance evaluation of three OS-SIP Server Softwares for the scenario session establishment with response delay, in this paper.

From the fig 11 it is shown that Asterisk is not able to handle more than 30 calls per second with the response delay ranging from 5 seconds - 25 seconds. OpenIMSCore server is not able to handle at least 30 calls per second with response delay ranging from 5 seconds - 25 seconds. As you can see from the figure 4.4 that server performance is degraded to 17% for 5 seconds. From this we can say that, OpenIMSCore server has very poor performance when compared with the remaining two OS-SIP Server Softwares. This is due to the installation of all four core modules in a single PC. We predict the performance of this server can be improved by installing core modules in different PCs. Due to the limitation of hardware resources we had to install OpenIMSCore in single PC. From these we can say that hardware resources also play an important role in affecting the performance of OS-SIP Server Softwares.

OpenSIPS is able to handle 90 CPS successfully till 15 seconds and from there increase in calls per second with linear increase in response delay, the server performance is slowly degrading as shown in figure 4.4. We can say that OpenSIPS is able to handle more number of calls per second with the response delay ranging from 5 seconds - 25 seconds. Finally we conclude that OpenSIPS and Asterisk use less number of hardware resources and provide high performance when compared with the OpenIMSCore.

Note: OpenIMSCore has three main modules. The performance of OpenIMSCore may differ from that of the performance shown in these experiments, if it is installed and experiments are conducted on at least three different computers. Here, the performance of OpenIMSCore is observed by conducting the experiments on a single computer due to the limitations of hardware resources.

6. Conclusion and Future Work:

In this thesis we have analyzed the performance comparison of three OS-SIP Server Softwares based on four scenarios described in section 3.7. Performance comparison has been done taking in to account 6 parameters which are described in section 3.3. An experimental test bed has been developed to observe the performance of the OS-SIP Server Softwares (Asterisk, Open- SIPS, and OpenIMSCore) in an ideal environment. From the analysis it is found that there was a significant performance difference among the three OS-SIP Server Softwares. In addition, we found that the OS-SIP Server Softwares performance mostly degraded with the response delay at UAS between “180 Ringing” and “200 Ok”.

For scenarios 1, 2 and 4 OpenSIPS is the best OS-SIP Server Softwares. Performance wise this acts as the best software and For scenario 3 Asterisk server acts as the best OS-SIP Server Software.

From the overall comparison of the results we found that OpenSIPS gives best performance in an ideal environment.

There is always a scope in studying the OS-SIP Server Softwares performance. In this thesis we mainly focused on the signaling part. Our future work is to test the performance of these three different OS-SIP Server Soft-wares focusing on both signaling part and on media part in real condition, and to compare the performance of OS-SIP server Softwares with that of the performance in ideal conditions.

References

[1] A. Singh, A. Acharya, P. Mahadevan, and Z.-Y. Shae, “SPLAT: a unified SIP services platform for VoIP applications: Research Articles,” International Journal of Communication Systems, vol. 19, May. 2006, p. 425–444.

[2] Campbell B, Rosenberg J, Schulzrinne H, Huitema C, Gurle D, “Session initiation protocol (SIP) extension for instant messaging,” RFC 3428, Internet Engineering Task Force, Dec. 2002.

[3] Campbell B, Mahy R, Jennings C, “The message session relay protocol,” SIMPLE Working Group Internet Draft, Feb. 2005.

[4] Pingtel SIP Soft phone. [Online]. Available: http://www.pingtel.com/solutions/instantxpressa.jsp, 2005.

[5] SJPhone. [Online]. Available: http://www.sjlabs.com, 2005.
[6] Cisco IP Phones. [Online]. Available: http://www.cisco.com/en/US/products/hw/phones/ps379, 2005.
[7] M.Handley, H. Schulzrinne, E. Schooler and J. Romberg. "Session initiation protocol:' Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.
[8] H.S. Jonathan and J. Rosenberg, "Internet Telephony: Architecture and Protocols an IETF Perspective," COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 31, 1998, p. 237–255.
[9] H. Schulzrinne and J. Rosenberg, "The IETF Internet telephony architecture and protocols," IEEE Network, vol. 13, Jun. 1999, pp. 18-23.
[10] H. Schulzrinne and J. Rosenberg, "The Session Initiation Protocol: Internet-centric signaling," Communications Magazine, IEEE, vol. 38, Oct. 2000, pp. 134-141.
[11] Benjamin Meyer, Marius Portmann, "Practical Performance Evaluation of Peer-to-Peer Internet Telephony Using SIP," citworkshops, pp.204-209, 2008 IEEE 8th International Conference on Computer and Information Technology Workshops, 2008
[12] M. Handley, H. Schulzrinne, E. Schooler and J. Romberg. "S I P session initiation protocol" Request for Comments 3261, Internet Engineering Task Force, June 2002.
[13] Henning Schulzrinne, Stephen Casner and Ron Frederick, and, Van Jacobson, "RTP: A Transport Protocol for Real-Time Applications," 2000.
[14] H.Schulzrinne S. Casner, R. Frederick, V. Jacobso, "RTP: A Transport Protocol for Real-Time Applications "Request for Comments 3550, Internet Engineering Task Force, July 2003.
[15] C. Shen, E. Nahum, H. Schulzrinne, and C. Wright, "The impact of TLS on SIP server performance," Principles, Systems and Applications of IP Telecommunications, New York, NY, USA: ACM, 2010, p. 59–70.
[16] J. Janak. SIP server proxy effectiveness. Master's thesis, CzechTechnicalUniversity Department of Computer Science, Prague, Czech Republic, May 2003.
[17] Razvan Rughinis, Cristian Iconaru "A Practical Analysis of Asterisk SIP Server Performance" online available at: http://conference.cluj.roedu.net/papers/25.pdf.
[18] Caixia Chi, Dong Wang and Ruibing Hao, , and , Wei Zhou, "Performance evaluation of SIP servers," Aug. 2008, pp. 674-679.

[19] T. Yanik, H. Hakan Kilinc, M. Sarioz, and S.S. Erdem, "Evaluating SIP proxy servers based on real performance data," Jun. 2008, pp. 324-329.
[20] Henning Schulzrinne, Sankaran Narayanan, and Jonathan Lennox. Sipstone – benchmark-ing sip server performance. Technical Report CUCS-005-02, Department of Computer Science, Columbia University, New York, March 2002.
[21] Mark A and Miller P E,"Managing Call Flows Using SIP," A technical briefing series on VoIP and Converged Networks, Vol. 5, Sept 2005.
[22] HP Invent, "SIPp." [Online]. Available at: http://www.sipp.sourceforge.net/
[23] Wireshark. "Wireshark Protocol Analyzer tool." [Online]. Available: http://www.wireshark.org/.