ISSN No. 0976-5697

Volume 3, No. 3, May-June 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Development of SIP UDP Transport Process under Linux Environment

K.Venkateswarlu*, Swarnalatha.P, L. Ramanathan and D. Ganesh Gopal
M.Tech- C.S.E*, Asst. Professor (S.G), Asst. Professor (Sr), Asst. Professor
SCSE, VIT University Vellore-14
venkateswarlu.vit19@yahoo.com*, pswarnalatha@vit.ac.in, lramanathan@vit.ac.in, ganeshgopal@svit.ac.in

Abstract: SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls. The Objective is to relinquish the SIP UDP transport process from the SIP gateway architecture (Voice application) running on IOS (Internetworking OS) and make it as an independent process running on Linux OS. The paper deals with the UDP (User Datagram Protocol) Socket Handler for SIP Service Provider under Linux Environment and the functionality of SIP UDP process which will handle the incoming messages from Network Layer (NL) and placing those messages on SIP Stack, followed by forwarding incoming messages will be sent to Application layer(AL) and Converse with the methods of Linux System Calls,Socket(),Setsockopt(),connect(),Bind(),Accept(),close(), Select() [16] and Using POSIX Message Queue System calls [15].

Keywords: IOS, UDP, SIP, Linux, NL, AL, SIP Stack.

I. INTRODUCTION

A. *SIP Part:*

Sip is the generic IETF session establishment protocol [10]. Sip is a peer-to-peer protocol [4]. Sip is human readable, extensible [11]. Sip is only one piece of the puzzle [4]:

SDP (Session Description Protocol): media negotiation
RTP/RTCP: Media
DNS: name resolution
NAT/Firewalls/SIP Security
HTTP 1.1: message formatting
MIME: application body encoding.

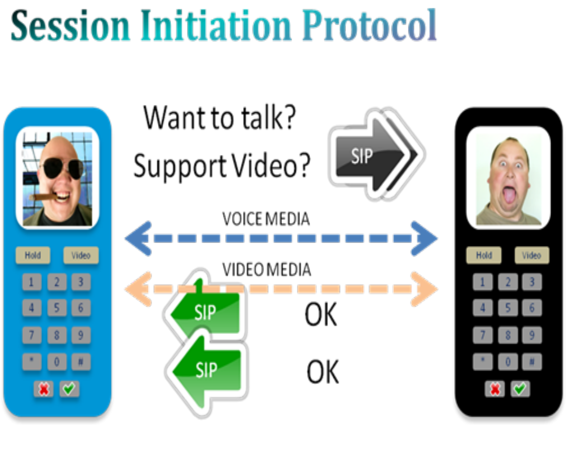


Figure1. SIP Protocol Descriptions.

There are many applications of the internet that require the creation and management of a session, where a session is treated as exchange of data between associations of Participants [6].

The basic SIP message format [7] and conversation as follows,

a. ***Received:***

```
INVITE sip:123@9.44.29.12:5060 SIP/2.0
Via: SIP/2.0/UDP 10.104.45.90:5080;
branch=z9hG4bK-14159-1-0
From :  sipp
<sip:sipp@10.104.45.90:5080>;tag=14159SIPpTag001
To :  sut     <sip:123@9.44.29.12:5060>
Call-ID :     1-14159@10.104.45.90
CSeq : 1      INVITE
Contact :     sip:sipp@10.104.45.90:5080
Max-Forwards : 70
Subject :          Performance Test
Content-Type :  application/sdp
Content-Length : 0 [1]
```

b. ***Sent:***

```
SIP/2.0                    200                    OK
Via:  SIP/2.0/UDP 10.104.45.90:5080; branch=z9hG4bK-14159-1-0; received=9.45.33.1
From :  sipp
<sip:sipp@10.104.45.90:5080>;tag=14159SIPpTag001
T o : sut <sip:123@9.44.29.12:5060>;tag=4E8720A4-2395
Date :      Wed, 15 Feb 2012 13:31:09 GMT
Call-ID :  1-14159@10.104.45.90
CSeq :      1 INVITE
Allow :   INVITE, OPTIONS, BYE, CANCEL, ACK,
PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY,
INFO, REGISTER
Allow-Events :  telephone-event
Remote-Party-ID : <sip:123@9.44.29.12>;party=called;
screen=no; privacy=off
Contact : <sip:123@9.44.29.12:5060>
Supported :   replaces
Supported :    sdp-anat
Server :        Cisco-SIP Gateway/IOS-12.x
Supported: timer
Content-Type: application/sdp
Content-Disposition: session; handling=required
Content-Length: 208
V = 0
o = CiscoSystemsSIP-GW-User Agent 227 4705 IN IP4
9.44.29.12
s = SIP Call
c = IN IP4 9.44.29.12
t = 0 0
m=audio 18498 RTP/AVP 8 19
c=IN IP4 9.44.29.12
a=rtpmap: 8 PCMA/8000
```

a=rtpmap: 19 CN/8000
a=ptime: 20 [5]

c. Received:

ACK sip:123@9.44.29.12:5060 SIP/2.0 [8]
Via: SIP/2.0/UDP 10.104.45.90:5080;branch=z9hG4bK-14159-1-5
From : sipp <sip:sipp@10.104.45.90:5080>;tag=14159SIPpTag001
To : sut <sip:123@9.44.29.12:5060>;tag=4E8720A4-2395
Call-ID : 1-14159@10.104.45.90
CSeq : 1 ACK
Contact : sip:sipp@10.104.45.90:5080
Max-Forwards : 70
Subject : Performance Test
Content-Type : application/sdp
Content-Disposition : session; handling=required
Content-Length: 371

v=0
o=user1 53655765 2353687637 IN IP4 10.104.45.90
s=SIP Call
c=IN IP4 10.104.45.90
t=0 0
m=audio 6000 RTP/AVP 8 96 101
a=rtpmap: 8 PCMA/8000
a=rtpmap: 96 mpeg4-generic/48000
a=fmtp: 96 profile-level-id=16; streamtype=5; mode=AAC-hbr; Config=11B0; sizeLength=13; indexLength=3; indexDeltaLength=3; constantDuration=480
a=rtpmap: 101 telephone-event/8000
a=fmtp: 101 0-15 [2]
<============= Voice Path Established and the two phones are in Conversation ================>

d. Sent:

BYE sip:sipp@10.104.45.90:5080 SIP/2.0
Via: SIP/2.0/UDP 9.44.29.12:5060;branch=z9hG4bK0727
From : sut <sip:123@9.44.29.12:5060>;tag=4E8720A4-2395
To: sipp <sip:sipp@10.104.45.90:5080>;tag=14159SIPpTag001

Date : Wed, 15 Feb 2012 13:31:24 GMT
Call-ID : 1-14159@10.104.45.90
User-Agent : Cisco-SIP Gateway /IOS-12.x
Max-Forwards : 70
Timestamp : 1329312704
CSeq : 101 BYE
Reason : Q.850;cause=16
P-RTP-Stat:
PS=228,OS=36480,PR=0,OR=0,PL=0,JI=1,LA=0,DU=19
Content-Length: 0 [1]

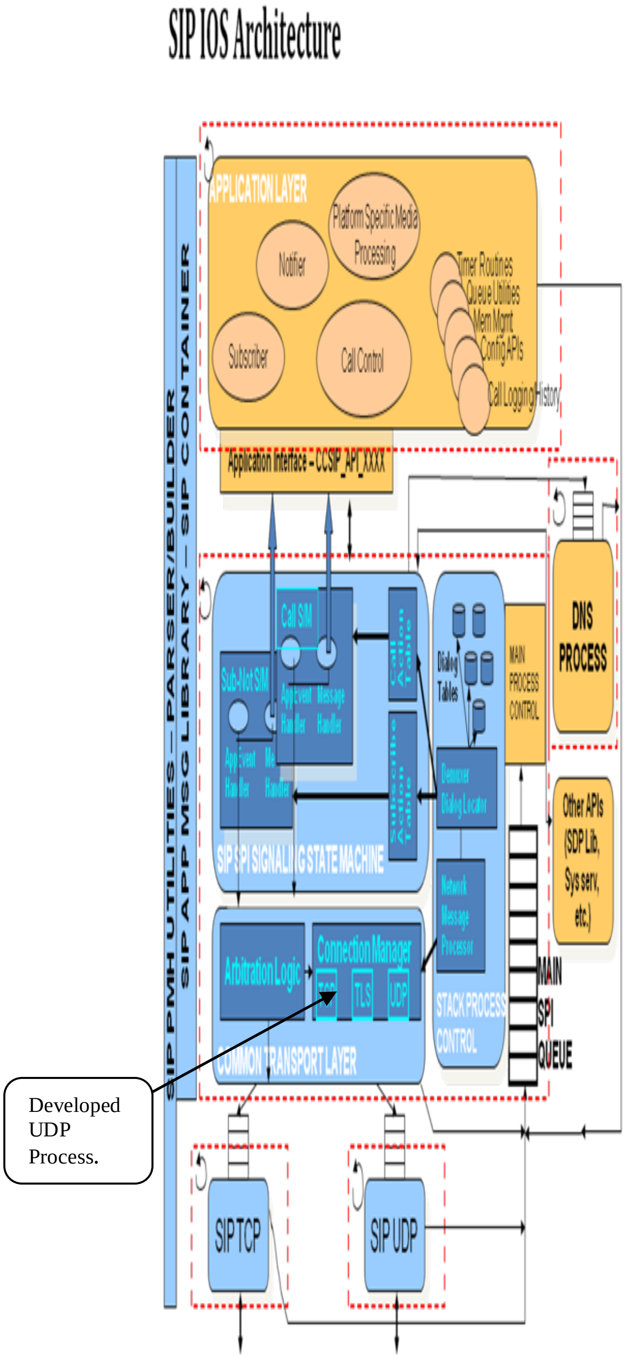
*Feb 15 13:31:44.767: //5/27E927458013/SIP/Msg/ccsipDisplayMsg:

e. Received:

SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.44.29.12:5060; branch=z9hG4bK0727
From: sut <sip:123@9.44.29.12:5060>; tag=4E8720A4-2395
To: sipp <sip:sipp@10.104.45.90:5080>; tag=14159SIPpTag001
Call-ID: 1-14159@10.104.45.90
CSeq: 101 BYE
Contact: <sip: 10.104.45.90:5080; transport=UDP>
Content-Length: 0 [3]

<============= Call Terminates =============>

B. SIP IOS Architecture:



Developed UDP Process.

Figure2. SIP IOS Architecture Diagram

In the above SIP IOS Architecture, under Connection manager UDP Transport process we are moving that part to Linux Environment for the purpose of performance improvement using Socket Programming and Linux Message queues Using C programming.

C. SIPPPart:

Sipp is a third party tool which allows for the crafting and transmission/receipt of SIP messages [14]. It is a very beneficial tool to reproduce behavior seen from a third party SIP endpoint [14]. It will run on Linux and win32 [14]. It also has the capacity to replay RTP packets from a packet capture with the media replay function [14].

a. How to run Sipp:

Home /... /sipp 10.105.35.65 –sf sample_uac.xml –m 2 –s 123

II. SYSTEM DESIGN

A. Design Methodology:

The project proposes top-down approach. A top-down development process starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts. It starts with creating classes for the general concepts. Then it specialize the class by creating some of its subclasses. Fig-1 describes Top-Down development process for implementation of sip transport layer UDP process under Linux environment.

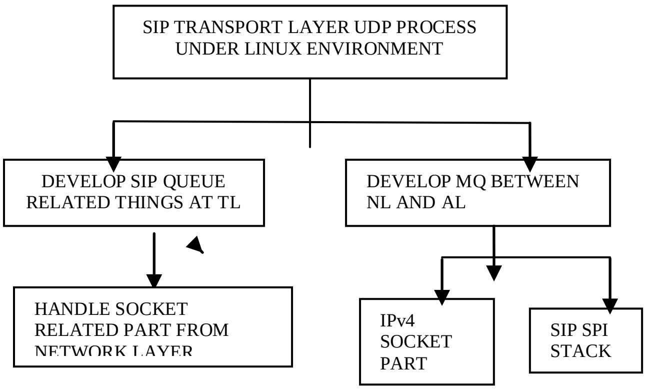

SIP TRANSPORT LAYER UDP PROCESS UNDER LINUX ENVIRONMENT
DEVELOP SIP QUEUE RELATED THINGS AT TL
DEVELOP MQ BETWEEN NL AND AL
HANDLE SOCKET RELATED PART FROM NETWORK LAYER
IPv4 SOCKET PART
SIP SPI STACK

Figure3. Top-Down development process for SIP Transport Layer UDP Process.

In the above top-down approach, indicates that implementation of sip transport layer UDP [12] process under Linux environment contains three modules.

First, the UDP [12] process takes the care about whatever the incoming messages that are coming from network layer, received on either IPv4 or IPv6 sockets. To develop these we have used the Linux system calls such as socket (), bind (), SETSOCKOPT () and select () System call (). The select allow a program to monitor multiple file descriptors, waiting until one or more of the file descriptors become "ready" for some class of I/O operation. A file descriptor is considered ready if it is possible to perform the corresponding I/O operation without blocking.

Second, the UDP Process takes care about the SIP Queue related things such as,

```
Void *dequeue (void *queue_ptr)
Bool enqueue (queuetype *qptr, void *eaddr)
Void *peekqueuehead (const queuetype* q)
Void *sip_peekqueuehead (sip_queue_t* queue)
Void *sip_peekqueuenext (sip_queue_t* queue, void
*element)
sip_queue_init (sip_queue_t *sip_q, int size)
```

Enqueue - add an element to a FIFO queue.
Dequeue - remove first element of a FIFO queue.
Peekqueuehead - Return address of element at head of Queue.

Third, The main Functionality of SIP UDP Process, After Receiving Messages on socket, using sip_udp_read_sock () it reads the message and it processes message and it forwards that message to SIPSPI on application layer.

These, functionality can be implemented in Linux Using Message Queue Concept. We are using the POSIX Linux Message queue System calls [15] such as mq_create (), mq_send (), mq_receive (), mq_notify (), mq_close (), mq_unlink ()...Etc.

B. Module-Wise Algorithmic Approach:

The project is divided into three modules:

a. Handle the Sockets (NL) Related Parts.
b. Develop SIP Queue Related Parts.
c. Implement Message Queue Between TL and AL.

MODULE-1 HANDLE RECEIVED MESSAGES ON NETWORK LAYER

These, Module Functionality can be handled using the following functions.

a. Bool process_watch_socket_event ():

The API to call sockets, and ask it watches for events you want. This Function initializes os structure, watched item and puts u on the notification queue. RECURRING -> every time packet arrives.

LINUX:
Using, Message queues.

b. Bool process_get_socket_event ():

The API call to get the socket off the event_list.
This function dequeues the socket from the event_list and gives it to the application if there are interesting events it wants.
Also, takes itself of the run queue, providing fairness if necessary.
Input Parameters:
FD -> File descriptor for which to retrieve events.
event_mask-> the mask for the event.
LINUX:
Using Select API, We can do the same functionality.
SYNTAX OF SELECT:
Int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout)[16];
select(): It allow a program to monitor multiple file descriptors, waiting until one or more of the file descriptors become "ready" for some class of I/O operation[16] .

A file descriptor is considered ready if it is possible to perform the corresponding I/O operation without blocking [16].

Parameters:
Nfds -> is the highest-numbered descriptor in any of the three sets, plus 3.
fd_set -> FD_SET and FD_CLR add or remove a given descriptor from a set.
fd_zero -> FD_ZERO will clear a set.
fd_isset -> FD_ISSET tests to see if a descriptor is part of the set; this is useful after select returns.
Timeout -> is an upper bound on the amount of time elapsed before select returns. It may be zero, causing select to return immediately. (This is useful for polling.) If timeout is NULL (no timeout), select can block indefinitely.
INPUT: Send the Multiple Messages to Network Layer (Sockets) From Data Link Layer.
OUTPUT: After, Receiving the Message on the Different Sockets, Which are created by the SIP UDP Transport Process read those messages and print those messages.

MODULE-2 HANDLE SIP QUEUE RELATED PARTS

The aim of this module is to implement the sip queue functionalities**.**

The Functions that are developed are as follows**,**

a. Void *dequeue ():

Dequeue - remove first element of a FIFO queue.

LINUX:

These is normal queue operation

b. Bool enqueue ():

Enqueue - add an element to a fifo queue.

LINUX:

This is normal queue operation**.**

c. Void *peekqueuehead ():

Peekqueuehead -- Return address of element at head of queue.

MODULE-3 DEVELOP MESSAGE QUEUE BETWEEN TRANSPORT LAYER AND APPLICATION LAYER

The main functionality of this module is the SIP UDP Process received the messages from Network Layer; it takes that message send to the AL via Message Queue using message queue id and converse.

III. DETAILED DESIGN

A. Design Diagram for UDP Process:

In the below diagram, IOS Scheduler is Non Preemptive**,** The task scheduler is responsible for scheduling and executing kernel processes on a CPU [9]. Because the scheduler is run-to-completion, all tasks must voluntarily relinquish control to the scheduler [9]. In general, a task should perform a small amount of work, relinquish the processor, and then continue working the next time it receives the CPU from the scheduler [9].

The Linux Scheduler is Preemptive, Means the process is forcibly interrupted by the Linux scheduler, and because of this behavior every process will get the chance to execute their tasks periodically. Ultimately this increases the performance of the SIP UDP Transport Process by decreasing the socket depth and message queue depth

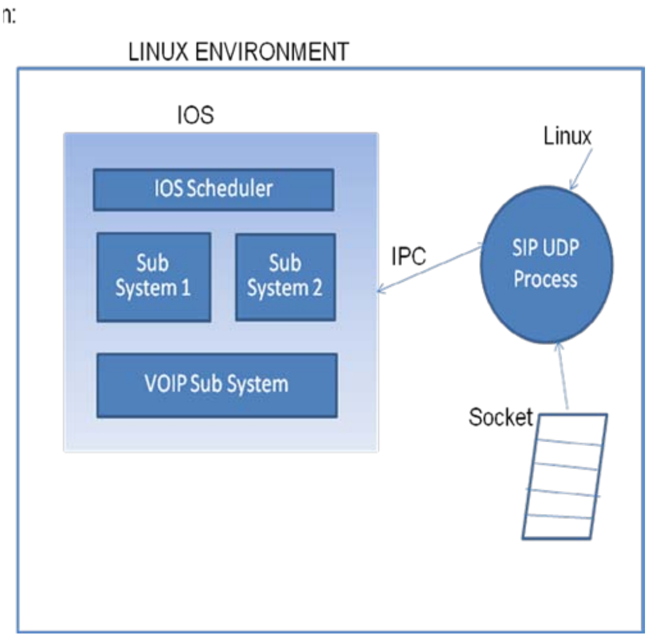


Figure 4. SIP UDP Transport Process under Linux Environment.

B. Design Explanation:

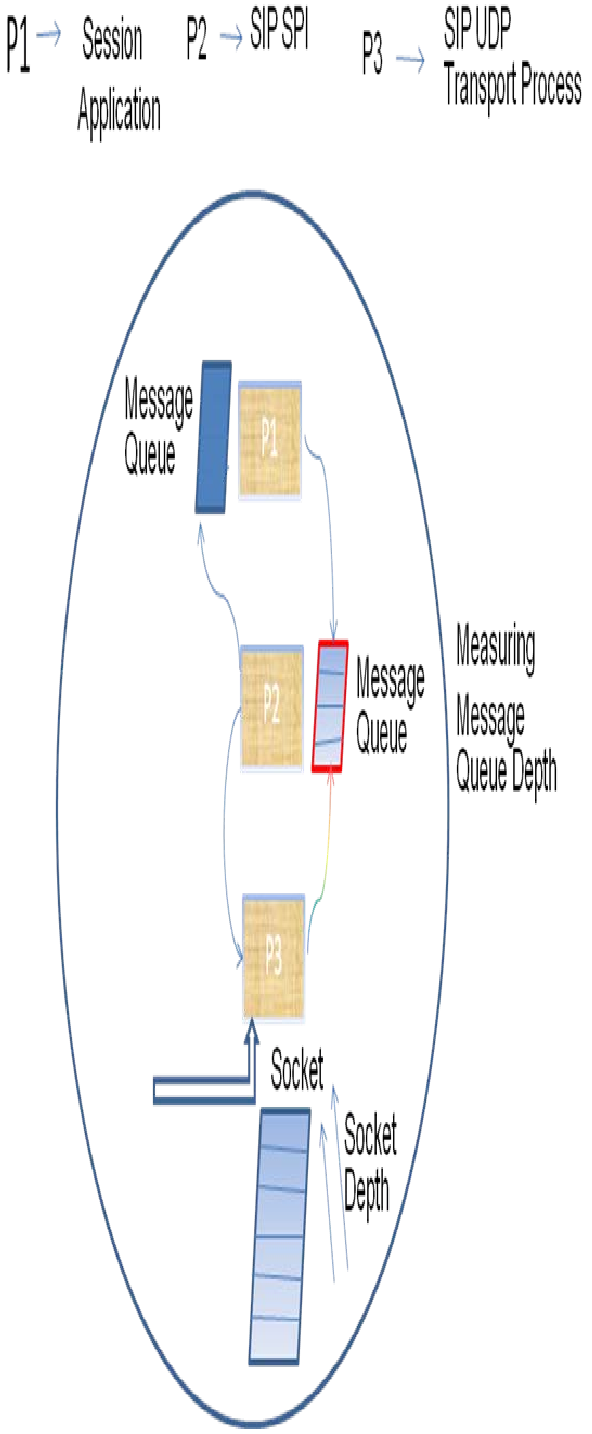
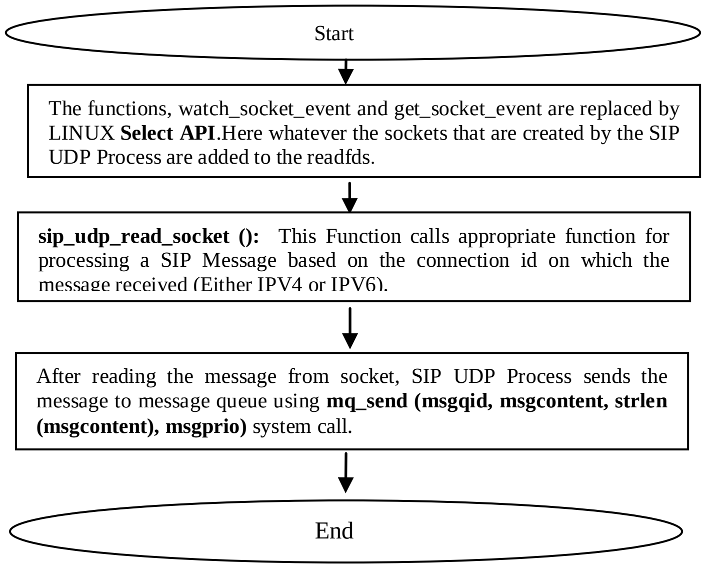


Figure 5. Detail Description about the SIP UDP Transport Process.

In the above diagram, P3 receives the Messages from the Network Layer and it performs some processing, forwards that message to P1 via P2. Assume the process P1 has the control in case of IOS, at that time so many messages received by the P3 but P1 has to give control voluntarily to process P3 until the socket depth increases. This is the main reason why we want to port to Linux Environment. In case of Linux, the scheduler is preemptive so after some time scheduler forcibly gives control to the P3. So these behaviors increase the performance of the overall process.

C. *Code Flow Diagram @Socket Part:*

LINUX:



Start

The functions, watch_socket_event and get_socket_event are replaced by LINUX **Select API**.Here whatever the sockets that are created by the SIP UDP Process are added to the readfds.

sip_udp_read_socket (): This Function calls appropriate function for processing a SIP Message based on the connection id on which the message received (Either IPV4 or IPV6).

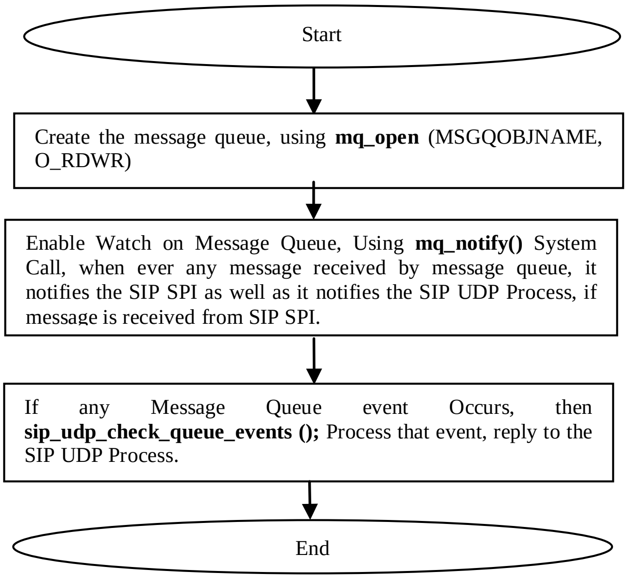
After reading the message from socket, SIP UDP Process sends the message to message queue using **mq_send (msgqid, msgcontent, strlen (msgcontent), msgprio)** system call.

End

Figure 6. Code Flow Diagram for Sockets Handling at Network Layer.

D. *Code Flow Diagram @Message Queue Module:*

LINUX:



Start

Create the message queue, using **mq_open** (MSGQOBJNAME, O_RDWR)

Enable Watch on Message Queue, Using **mq_notify()** System Call, when ever any message received by message queue, it notifies the SIP SPI as well as it notifies the SIP UDP Process, if message is received from SIP SPI.

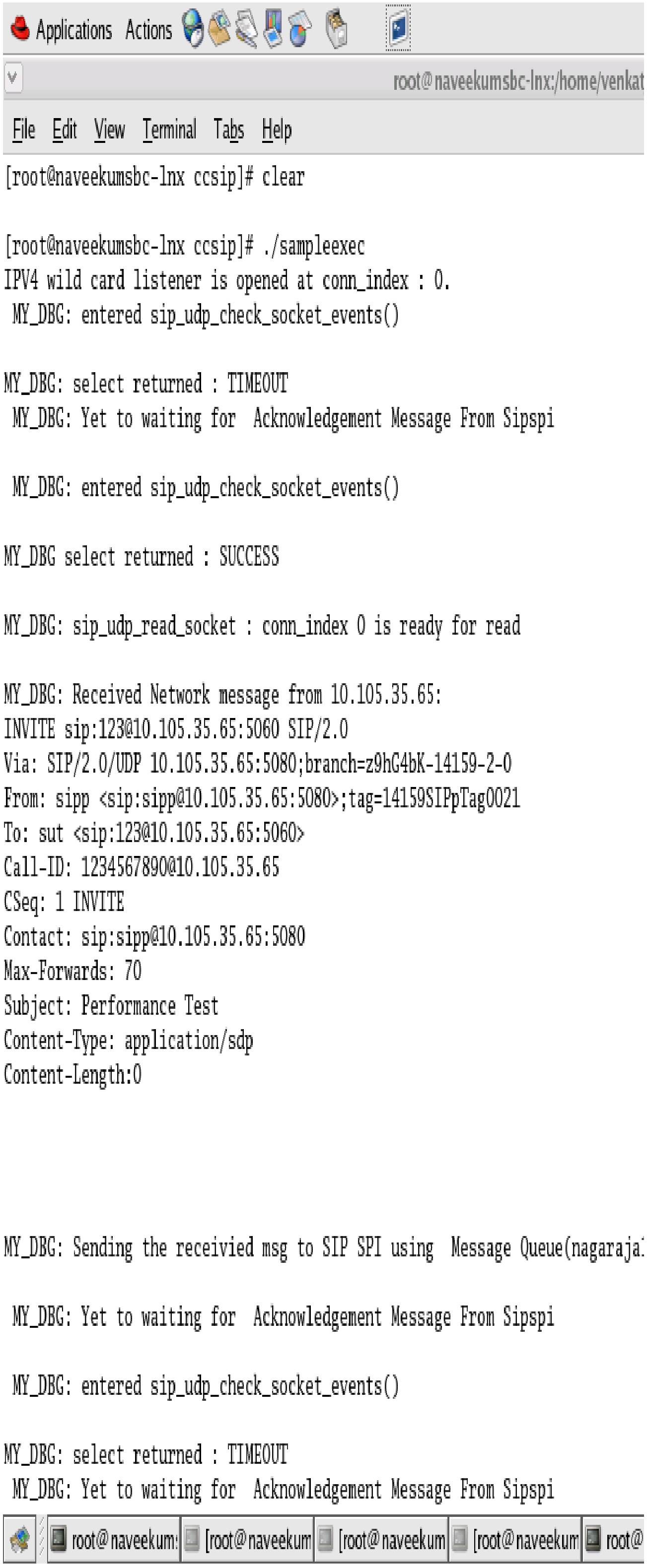
If any Message Queue event Occurs, then **sip_udp_check_queue_events ();** Process that event, reply to the SIP UDP Process.

End

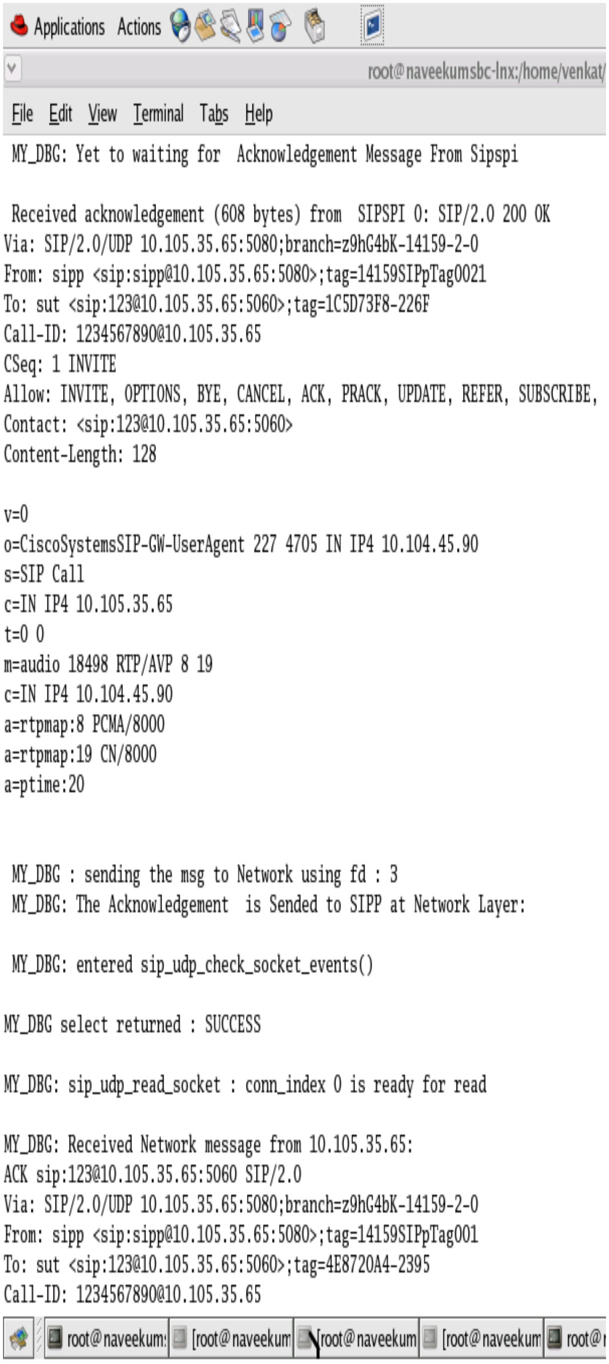
Figure7. Code Flow Diagram for Message Queue Module

IV. IMLEMENTATION

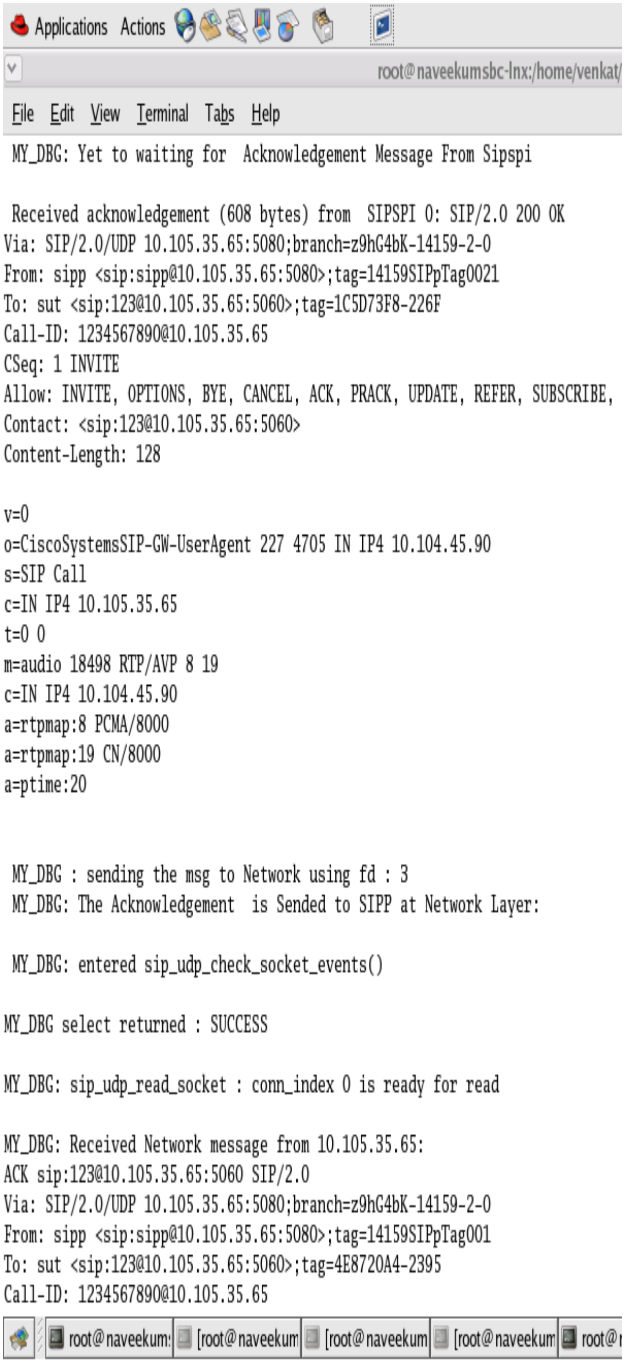
A. *Snapshots at UDP Process Side:*



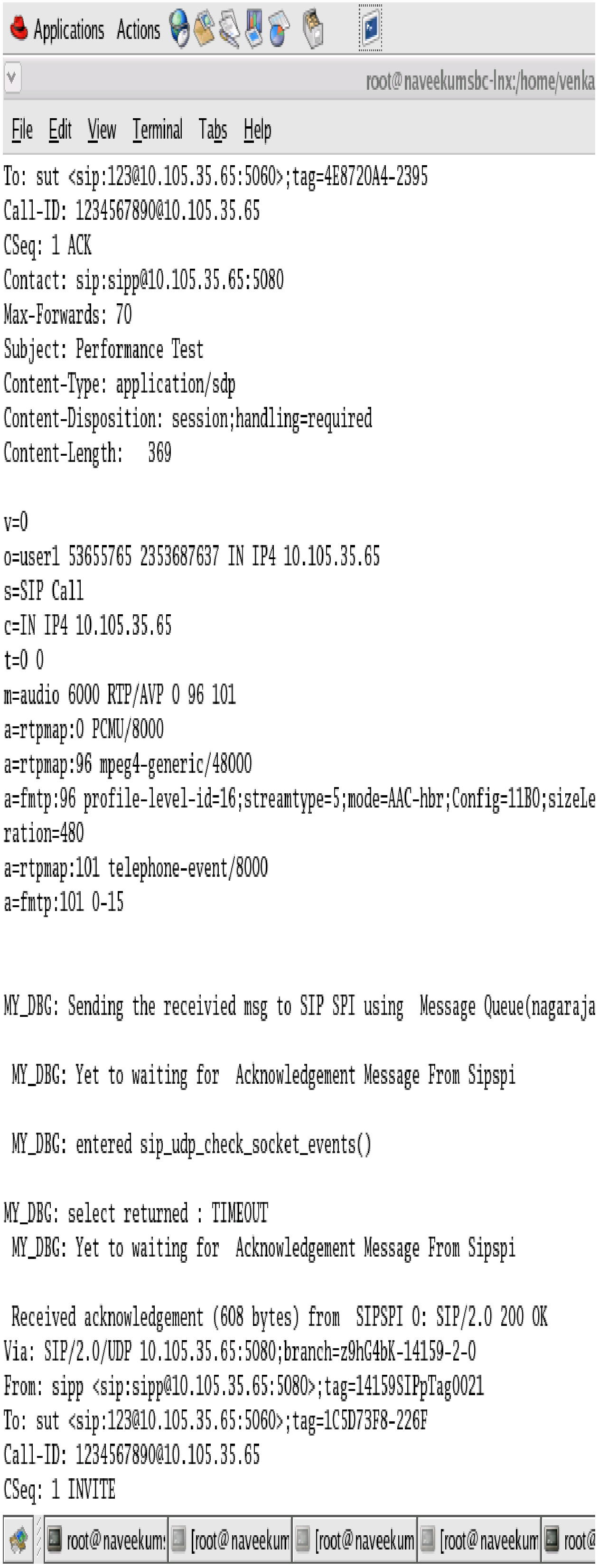
```
[root@naveekumsbc-lnx ccsip]# clear

[root@naveekumsbc-lnx ccsip]# ./sampleexec
IPV4 wild card listener is opened at conn_index : 0.
 MY_DBG: entered sip_udp_check_socket_events()

MY_DBG: select returned : TIMEOUT
 MY_DBG: Yet to waiting for  Acknowledgement Message From Sipspi

 MY_DBG: entered sip_udp_check_socket_events()

MY_DBG select returned : SUCCESS

MY_DBG: sip_udp_read_socket : conn_index 0 is ready for read

MY_DBG: Received Network message from 10.105.35.65:
INVITE sip:123@10.105.35.65:5060 SIP/2.0
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
From: sipp <sip:sipp@10.105.35.65:5080>;tag=14159SIPpTag0021
To: sut <sip:123@10.105.35.65:5060>
Call-ID: 1234567890@10.105.35.65
CSeq: 1 INVITE
Contact: sip:sipp@10.105.35.65:5080
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length:0




MY_DBG: Sending the received msg to SIP SPI using  Message Queue(nagaraja

 MY_DBG: Yet to waiting for  Acknowledgement Message From Sipspi

 MY_DBG: entered sip_udp_check_socket_events()

MY_DBG: select returned : TIMEOUT
 MY_DBG: Yet to waiting for  Acknowledgement Message From Sipspi
```


Applications Actions
root@naveekumsbc-lnx:/home/venkat/
File Edit View Terminal Tabs Help
root@naveekum: [root@naveekum [root@naveekum [root@naveekum root@r

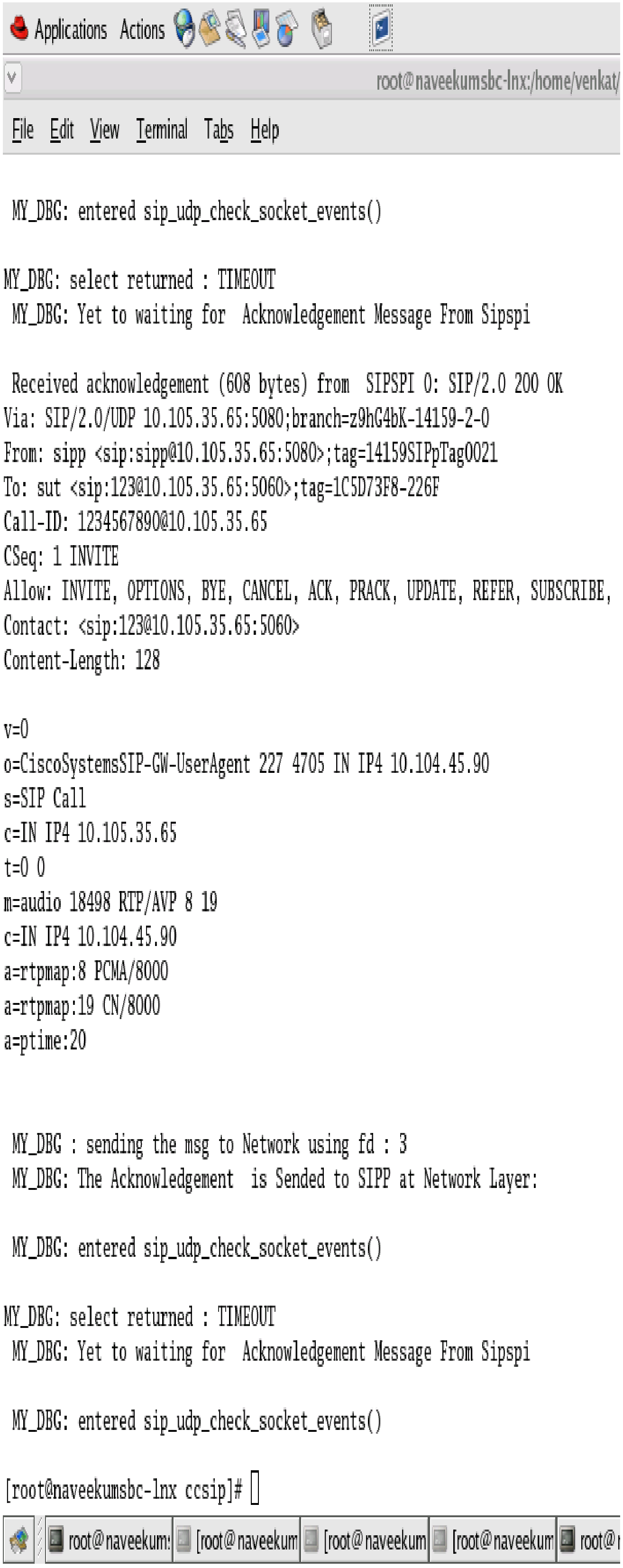
```
 MY_DBG: Yet to waiting for  Acknowledgement Message From Sipspi

 Received acknowledgement (608 bytes) from  SIPSPI 0: SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
From: sipp <sip:sipp@10.105.35.65:5080>;tag=14159SIPpTag0021
To: sut <sip:123@10.105.35.65:5060>;tag=1C5D73F8-226F
Call-ID: 1234567890@10.105.35.65
CSeq: 1 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE,
Contact: <sip:123@10.105.35.65:5060>
Content-Length: 128

v=0
o=CiscoSystemsSIP-GW-UserAgent 227 4705 IN IP4 10.104.45.90
s=SIP Call
c=IN IP4 10.105.35.65
t=0 0
m=audio 18498 RTP/AVP 8 19
c=IN IP4 10.104.45.90
a=rtpmap:8 PCMA/8000
a=rtpmap:19 CN/8000
a=ptime:20


 MY_DBG : sending the msg to Network using fd : 3
 MY_DBG: The Acknowledgement  is Sended to SIPP at Network Layer:

 MY_DBG: entered sip_udp_check_socket_events()

MY_DBG select returned : SUCCESS

MY_DBG: sip_udp_read_socket : conn_index 0 is ready for read

MY_DBG: Received Network message from 10.105.35.65:
ACK sip:123@10.105.35.65:5060 SIP/2.0
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
From: sipp <sip:sipp@10.105.35.65:5080>;tag=14159SIPpTag001
To: sut <sip:123@10.105.35.65:5060>;tag=4E8720A4-2395
Call-ID: 1234567890@10.105.35.65
```


Applications Actions
root@naveekumsbc-lnx:/home/venkat/
File Edit View Terminal Tabs Help
root@naveekum: [root@naveekum [root@naveekum [root@naveekum root@r

```
 MY_DBG: Yet to waiting for  Acknowledgement Message From Sipspi

 Received acknowledgement (608 bytes) from  SIPSPI 0: SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
From: sipp <sip:sipp@10.105.35.65:5080>;tag=14159SIPpTag0021
To: sut <sip:123@10.105.35.65:5060>;tag=1C5D73F8-226F
Call-ID: 1234567890@10.105.35.65
CSeq: 1 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE,
Contact: <sip:123@10.105.35.65:5060>
Content-Length: 128

v=0
o=CiscoSystemsSIP-GW-UserAgent 227 4705 IN IP4 10.104.45.90
s=SIP Call
c=IN IP4 10.105.35.65
t=0 0
m=audio 18498 RTP/AVP 8 19
c=IN IP4 10.104.45.90
a=rtpmap:8 PCMA/8000
a=rtpmap:19 CN/8000
a=ptime:20


 MY_DBG : sending the msg to Network using fd : 3
 MY_DBG: The Acknowledgement  is Sended to SIPP at Network Layer:

 MY_DBG: entered sip_udp_check_socket_events()

MY_DBG select returned : SUCCESS

MY_DBG: sip_udp_read_socket : conn_index 0 is ready for read

MY_DBG: Received Network message from 10.105.35.65:
ACK sip:123@10.105.35.65:5060 SIP/2.0
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
From: sipp <sip:sipp@10.105.35.65:5080>;tag=14159SIPpTag001
To: sut <sip:123@10.105.35.65:5060>;tag=4E8720A4-2395
Call-ID: 1234567890@10.105.35.65
```

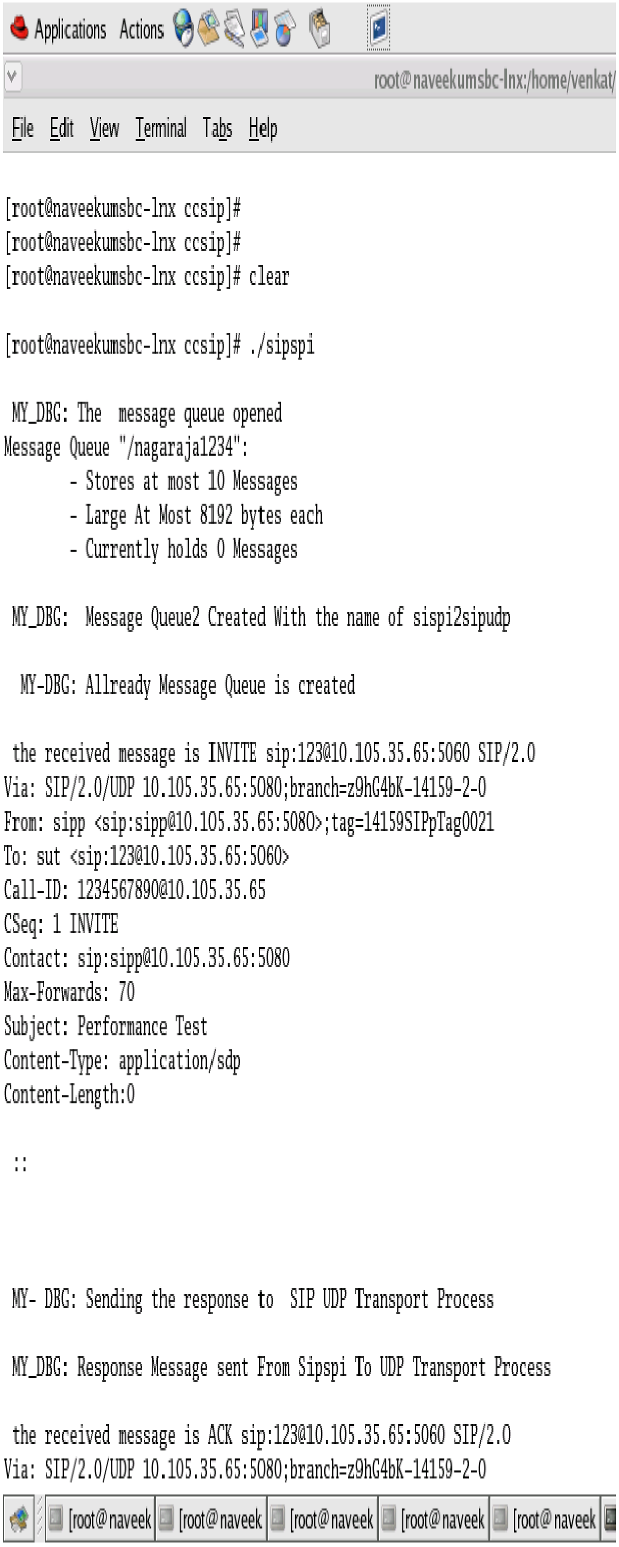


```
To: sut <sip:123@10.105.35.65:5060>;tag=4E8720A4-2395
Call-ID: 1234567890@10.105.35.65
CSeq: 1 ACK
Contact: sip:sipp@10.105.35.65:5080
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length:   369

v=0
o=user1 53655765 2353687637 IN IP4 10.105.35.65
s=SIP Call
c=IN IP4 10.105.35.65
t=0 0
m=audio 6000 RTP/AVP 0 96 101
a=rtpmap:0 PCMU/8000
a=rtpmap:96 mpeg4-generic/48000
a=fmtp:96 profile-level-id=16;streamtype=5;mode=AAC-hbr;Config=11B0;sizeLe
ration=480
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15



MY_DBG: Sending the receivied msg to SIP SPI using  Message Queue(nagaraja

 MY_DBG: Yet to waiting for  Acknowledgement Message From Sipspi

 MY_DBG: entered sip_udp_check_socket_events()

MY_DBG: select returned : TIMEOUT
 MY_DBG: Yet to waiting for  Acknowledgement Message From Sipspi

 Received acknowledgement (608 bytes) from  SIPSPI 0: SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
From: sipp <sip:sipp@10.105.35.65:5080>;tag=14159SIPpTag0021
To: sut <sip:123@10.105.35.65:5060>;tag=1C5D73F8-226F
Call-ID: 1234567890@10.105.35.65
CSeq: 1 INVITE
```

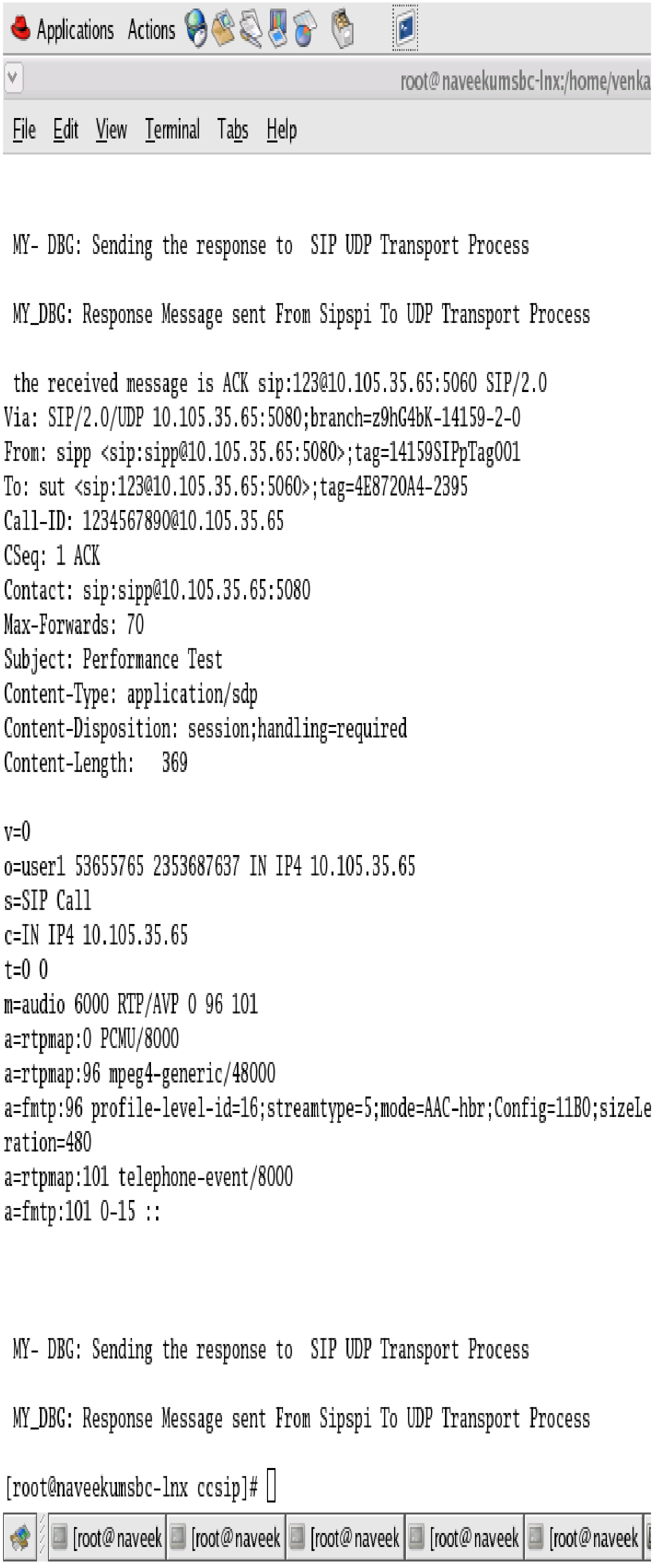


```

 MY_DBG: entered sip_udp_check_socket_events()

MY_DBG: select returned : TIMEOUT
 MY_DBG: Yet to waiting for  Acknowledgement Message From Sipspi

 Received acknowledgement (608 bytes) from  SIPSPI 0: SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
From: sipp <sip:sipp@10.105.35.65:5080>;tag=14159SIPpTag0021
To: sut <sip:123@10.105.35.65:5060>;tag=1C5D73F8-226F
Call-ID: 1234567890@10.105.35.65
CSeq: 1 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE,
Contact: <sip:123@10.105.35.65:5060>
Content-Length: 128

v=0
o=CiscoSystemsSIP-GW-UserAgent 227 4705 IN IP4 10.104.45.90
s=SIP Call
c=IN IP4 10.105.35.65
t=0 0
m=audio 18498 RTP/AVP 8 19
c=IN IP4 10.104.45.90
a=rtpmap:8 PCMA/8000
a=rtpmap:19 CN/8000
a=ptime:20


 MY_DBG : sending the msg to Network using fd : 3
 MY_DBG: The Acknowledgement  is Sended to SIPP at Network Layer:

 MY_DBG: entered sip_udp_check_socket_events()

MY_DBG: select returned : TIMEOUT
 MY_DBG: Yet to waiting for  Acknowledgement Message From Sipspi

 MY_DBG: entered sip_udp_check_socket_events()

[root@naveekumsbc-lnx ccsip]# 
```

B. *Snapshots at SIPSI Side:*

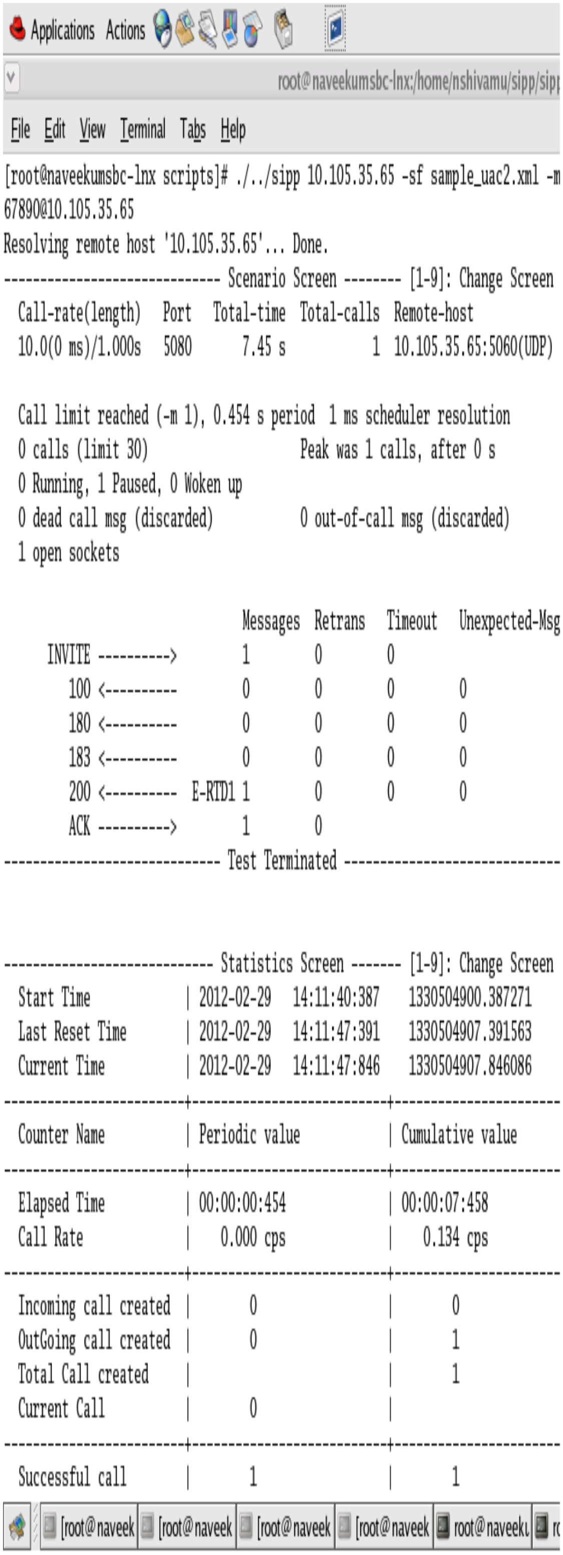


```
[root@naveekumsbc-lnx ccsip]#
[root@naveekumsbc-lnx ccsip]#
[root@naveekumsbc-lnx ccsip]# clear

[root@naveekumsbc-lnx ccsip]# ./sipspi

 MY_DBG: The  message queue opened
Message Queue "/nagaraja1234":
        - Stores at most 10 Messages
        - Large At Most 8192 bytes each
        - Currently holds 0 Messages

 MY_DBG:  Message Queue2 Created With the name of sispi2sipudp

  MY-DBG: Allready Message Queue is created

 the received message is INVITE sip:123@10.105.35.65:5060 SIP/2.0
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
From: sipp <sip:sipp@10.105.35.65:5080>;tag=14159SIPpTag0021
To: sut <sip:123@10.105.35.65:5060>
Call-ID: 1234567890@10.105.35.65
CSeq: 1 INVITE
Contact: sip:sipp@10.105.35.65:5080
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length:0

 ::



 MY- DBG: Sending the response to  SIP UDP Transport Process

 MY_DBG: Response Message sent From Sipspi To UDP Transport Process

 the received message is ACK sip:123@10.105.35.65:5060 SIP/2.0
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
```

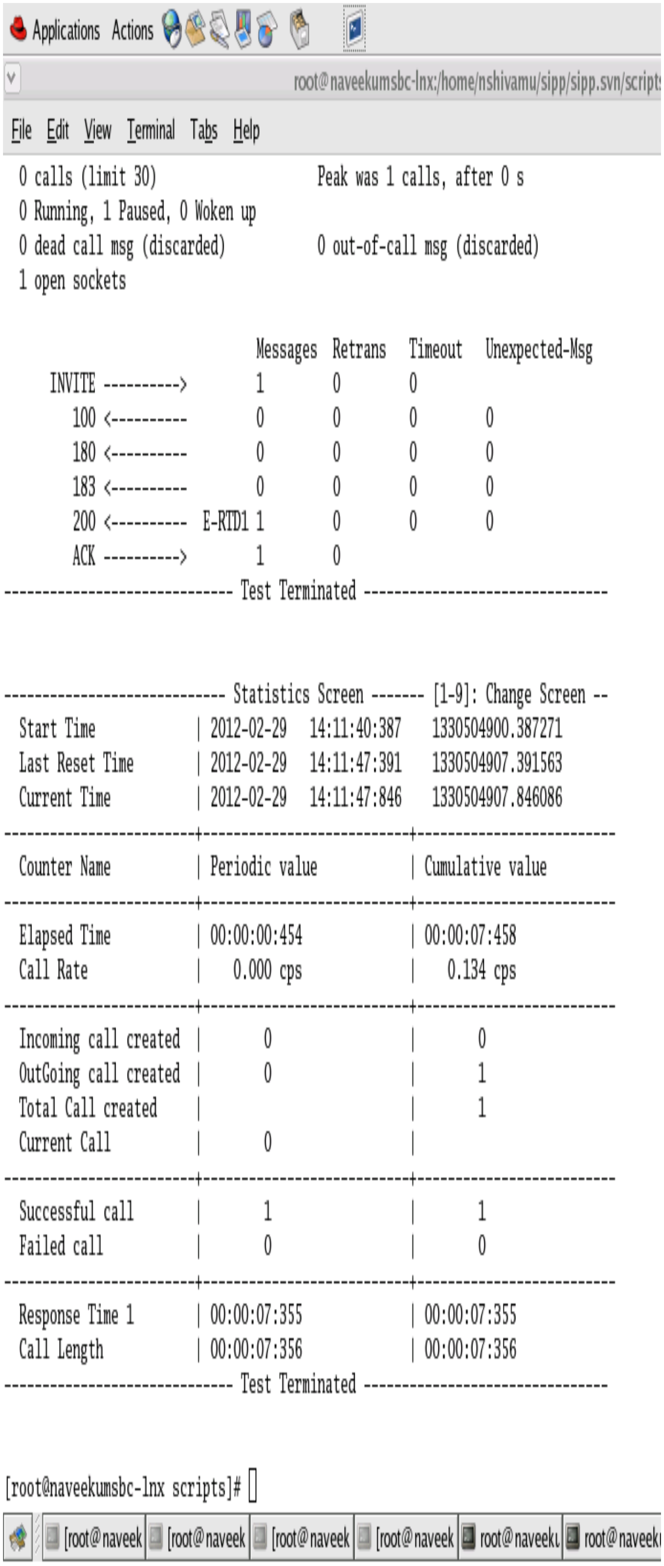


```
 MY- DBG: Sending the response to  SIP UDP Transport Process

 MY_DBG: Response Message sent From Sipspi To UDP Transport Process

 the received message is ACK sip:123@10.105.35.65:5060 SIP/2.0
Via: SIP/2.0/UDP 10.105.35.65:5080;branch=z9hG4bK-14159-2-0
From: sipp <sip:sipp@10.105.35.65:5080>;tag=14159SIPpTag001
To: sut <sip:123@10.105.35.65:5060>;tag=4E8720A4-2395
Call-ID: 1234567890@10.105.35.65
CSeq: 1 ACK
Contact: sip:sipp@10.105.35.65:5080
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length:   369

v=0
o=user1 53655765 2353687637 IN IP4 10.105.35.65
s=SIP Call
c=IN IP4 10.105.35.65
t=0 0
m=audio 6000 RTP/AVP 0 96 101
a=rtpmap:0 PCMU/8000
a=rtpmap:96 mpeg4-generic/48000
a=fmtp:96 profile-level-id=16;streamtype=5;mode=AAC-hbr;Config=11B0;sizeLe
ration=480
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15 ::



 MY- DBG: Sending the response to  SIP UDP Transport Process

 MY_DBG: Response Message sent From Sipspi To UDP Transport Process

[root@naveekumsbc-lnx ccsip]# 
```

C. *Snapshots at SIPP Side:*


Applications Actions
root@naveekumsbc-lnx:/home/nshivamu/sipp/sipp
File Edit View Terminal Tabs Help
[root@naveek [root@naveek [root@naveek [root@naveek root@naveeku r

```
[root@naveekumsbc-lnx scripts]# ./../sipp 10.105.35.65 -sf sample_uac2.xml -m
67890@10.105.35.65
Resolving remote host '10.105.35.65'... Done.
------------------------------ Scenario Screen -------- [1-9]: Change Screen
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
  10.0(0 ms)/1.000s   5080      7.45 s            1  10.105.35.65:5060(UDP)

  Call limit reached (-m 1), 0.454 s period  1 ms scheduler resolution
  0 calls (limit 30)                     Peak was 1 calls, after 0 s
  0 Running, 1 Paused, 0 Woken up
  0 dead call msg (discarded)            0 out-of-call msg (discarded)
  1 open sockets

                              Messages  Retrans   Timeout   Unexpected-Msg
      INVITE ---------->         1         0         0
         100 <----------         0         0         0         0
         180 <----------         0         0         0         0
         183 <----------         0         0         0         0
         200 <----------  E-RTD1 1         0         0         0
         ACK ---------->         1         0
------------------------------ Test Terminated ------------------------------


----------------------------- Statistics Screen ------- [1-9]: Change Screen
  Start Time             | 2012-02-29   14:11:40:387   1330504900.387271
  Last Reset Time        | 2012-02-29   14:11:47:391   1330504907.391563
  Current Time           | 2012-02-29   14:11:47:846   1330504907.846086
-------------------------+---------------------------+----------------------
  Counter Name           | Periodic value            | Cumulative value
-------------------------+---------------------------+----------------------
  Elapsed Time           | 00:00:00:454              | 00:00:07:458
  Call Rate              |    0.000 cps              |    0.134 cps
-------------------------+---------------------------+----------------------
  Incoming call created  |        0                  |        0
  OutGoing call created  |        0                  |        1
  Total Call created     |                           |        1
  Current Call           |        0                  |
-------------------------+---------------------------+----------------------
  Successful call        |        1                  |        1
```


Applications Actions
root@naveekumsbc-lnx:/home/nshivamu/sipp/sipp.svn/scripts
File Edit View Terminal Tabs Help
[root@naveek [root@naveek [root@naveek [root@naveek root@naveeku root@naveeku

```
  0 calls (limit 30)                     Peak was 1 calls, after 0 s
  0 Running, 1 Paused, 0 Woken up
  0 dead call msg (discarded)            0 out-of-call msg (discarded)
  1 open sockets

                              Messages  Retrans   Timeout   Unexpected-Msg
      INVITE ---------->         1         0         0
         100 <----------         0         0         0         0
         180 <----------         0         0         0         0
         183 <----------         0         0         0         0
         200 <----------  E-RTD1 1         0         0         0
         ACK ---------->         1         0
------------------------------ Test Terminated --------------------------------



----------------------------- Statistics Screen ------- [1-9]: Change Screen --
  Start Time             | 2012-02-29   14:11:40:387   1330504900.387271
  Last Reset Time        | 2012-02-29   14:11:47:391   1330504907.391563
  Current Time           | 2012-02-29   14:11:47:846   1330504907.846086
-------------------------+---------------------------+--------------------------
  Counter Name           | Periodic value            | Cumulative value
-------------------------+---------------------------+--------------------------
  Elapsed Time           | 00:00:00:454              | 00:00:07:458
  Call Rate              |    0.000 cps              |    0.134 cps
-------------------------+---------------------------+--------------------------
  Incoming call created  |        0                  |        0
  OutGoing call created  |        0                  |        1
  Total Call created     |                           |        1
  Current Call           |        0                  |
-------------------------+---------------------------+--------------------------
  Successful call        |        1                  |        1
  Failed call            |        0                  |        0
-------------------------+---------------------------+--------------------------
  Response Time 1        | 00:00:07:355              | 00:00:07:355
  Call Length            | 00:00:07:356              | 00:00:07:356
------------------------------ Test Terminated --------------------------------


[root@naveekumsbc-lnx scripts]# 
```

V. CONCLUSION AND FUTURE WORKS

After testing, the SIP UDP Transport Process functionality in both the operating systems such as IOS and Linux Environment, we made the conclusion as the socket Depth and Message Queue Depth decreased under Linux Environment. These, factor improves the performance of the network and response time also minimized.

The main advantage for developing the SIP UDP Transport Process under Linux Environment is to reduce the size of Socket Depth (Measure at Network Layer) and Message Queue Depth (Measure at Transport Layer), which we achieved because of preemptive behavior of Linux Operating System.

Currently, we developed the SIP UDP Transport Process under Linux Environment, in Future we are planning to Develop the other two common transport Layer processes at connection manager level, such as TCP (transmission congestion protocol) and TLS (transport layer security) .The Aim of these two processes are developing under Linux Environment also to reduce the Socket Depth at NL and Message Queue Depth at TL. In order to implement these two processes under Linux Environment, there are challenging issues associated with Timers and Security aspects. We Measured the Socket Depth and Message Queue Depth by Using the SIPP tool for performance testing of the SIP UDP Transport Process.

VI. REFERENCES

[1]. C. Holmberg, E. Burger, and H. Kaplan, “Session Initiation Protocol (SIP) INFO Method and Package Framework,”RFC 6086 (Proposed Standard), Internet Engineering Task Force, January 2011.

[2]. C. Jennings, R. Mahy, and F. Audet, “Managing Client-Initiated Connections in the Session Initiation Protocol (SIP),” RFC 5626 (Proposed Standard), Internet Engineering Task Force, Oct. 2009.

[3]. R. Mahy, B. Biggs, and R. Dean, “The Session Initiation Protocol (SIP) “Replaces” Header,” RFC 3891 (Proposed Standard), Internet Engineering Task Force, Sept. 2004.

[4]. SIP: Understanding the Session Initiation Protocol Alan B. Johnston, Artech House, second edition, 2004. ISBN 1-58053-655-7

[5]. R. Mahy and D. Petrie, “The Session Initiation Protocol (SIP) “Join” Header,” RFC 3911 (Proposed Standard), Internet Engineering Task Force, Oct. 2004

[6]. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “SIP: Session Initiation Protocol,” RFC 3261 (Proposed Standard), Internet Engineering Task Force, June 2002.

[7]. J. Rosenberg and H. Schulzrinne, “Reliability of Provisional Responses in Session Initiation Protocol (SIP),” RFC 3262 (Proposed Standard), Internet Engineering Task Force, and June 2002.

[8]. B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, “Session Initiation Protocol (SIP) Extension for Instant Messaging,” RFC 3428 (Proposed Standard), Internet Engineering Task Force, Dec. 2002.

[9]. Cisco IOS Programmer’s Guide/ Architecture Reference, Software Release 12.0, Fifth Edition, February 1999, Text Part Number: 78-2051-05.

[10]. M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, “SIP: Session Initiation Protocol,” RFC 2543 (Proposed Standard), Internet Engineering Task Force, Mar. 1999, obsolete by RFCs 3261, 3262, 3263, 3264, 3265.

[11]. H. Schulzrinne and J. Rosenberg, "The Session Initiation Protocol: Providing Advanced Telephony Access across the Internet," *Bell Labs Technical Journal*, October-December 1998.

[12]. Postal, J., “User Datagram Protocol, “RFC 768, 1980.

[13]. H. Sinnreich and A. Johnston, Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol, John Wiley & Sons, New York, NY, USA, 2nd edition.

[14]. http://sipp.sourceforge.net/doc3.0/reference.html

[15]. http://linux.die.net/man/7/mq_overview.

[16]. http://manpages.courier/mta.org/htmlman2/select.html