A PROJECT REPORT

ON

VoIP Communication Using SIP

(SESSION INITIATION PROTOCOL)

Submitted to

BHARAT ELECTRONICS LIMITED, UTTRAKHAND

As a part of Industrial Training in the degree of

Bachelor of Technology

In

Information Technology

SUBMITTED BY:

RAJAT SAXENA

IIT2014503

UNDER SUPERVISION OF:

Mr. LAURANGE KAMAL

Sr. DEPUTY MANAGER

BHARAT ELECTRONICS LIMITED

KOTDWARA - 246149

ACKNOWLEDGEMENT

I take this opportunity to express my profound sense of gratitude and respect to all those who helped me throughout my project.

This report acknowledges to the intense driving and technical competence of the entire individual that have contributed to it. It would have been almost impossible to complete this project without the support of these people. I extend thanks and gratitude to **Mr. Laurange Kamal (Sr. Deputy Manager Development and Engineering)** who has imparted me the guidance in all aspects. He shared his valuable time from the busy schedule to guide me and provide his active and sincere support for my activities.

This report is authentic record of my own work which is accomplished by the sincere and active support of my mentor. I have tried my best to summarize this report.

Rajat Saxena

B. Tech IT, V Semester

Indian Institute Of Information Technology, Allahabad

Allahabad

DATE : Tuesday, 12 July, 2016

BHARAT ELECTRONICS LIMITED KOTDWARA

CERTIFICATE

This is to certify that the project titled "**VoIP Communication using SIP**" is carried out by **Mr. Rajat Saxena** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Information Technology. This is an authentic work carried out by **Mr. Rajat Saxena** under my guidance and supervision.

____

Mr. Laurange Kamal

Sr. Deputy Manager

Development And Engineering (D&E)

Bharat Electronics Limited

Kotdwara - 246149

ABSTRACT

The **VoIP COMMUNICATION USING SIP** is a server based software that creates and manages a session, where a session is considered an exchange of data between an association of participants. It provides user friendly environment for users to communicate between themselves.

It uses SIP (Session Initiation Protocol) which is one of the most common protocol used in VoIP technology. VoIP is a technology that allows us to deliver voice and multimedia content over the internet. It is one of the cheapest way to communicate anytime anywhere with internet availability.

This is a JAVA based server program which handles clients. The software makes excessive use of JAVA Socket API to receive requests and forward respective responses. It maintains detail of registered clients i.e IP address and local port and their numbers. Upon recieveing a call request by any registered user, it generates an appropriate SIP response and forwards it its destination. This software can also act as a proxy server that can communicate as a proxy with another instance of same software running on different machine.

The VoIP COMMUNICATION USING SIP is accessible only by an administrator. The end- user only needs to know the server's IP address to initiate communication upon successful registration by the server. Admin's main function is to maintain the server. Admin will have complete access and control over the complete software.

Thus, major advantage of our software is that it can avoid the toll charge by ordinary telephone service by using fixed charge IP network services such as broadband.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION OF BEL

(BHARAT ELECTRONICS LTD.)

1.1 HISTORY

1954

- Bharat Electronics Limited (BEL) was established at Bangalore, India, by the Government of India under the Ministry of Defence in 1954 to meet the specialised electronic needs of the Indian defence services.
- Over the years, it has grown into a multi-product, multi-technology, multi-unit company servicing the needs of customers in diverse fields in India and abroad.

1964

- The growth and diversification of BEL over the years mirrors the advances in the electronics technology, with which BEL has kept pace. Starting with the manufacture of a few communication equipment in 1956.
- BEL went on to produce Receiving Valves in 1961, Germanium Semiconductors in 1962 and Radio Transmitters for AIR in 1964.

1968

- In 1966, BEL set up a Radar manufacturing facility for the Army and in-house R&D, which has been nurtured over the years.
- Manufacture of Transmitting Tubes, Silicon Devices and Integrated Circuits started in 1967. The PCB manufacturing facility was established in 1968.

1972

- In 1970, manufacture of Black & White TV Picture Tube, X-ray Tube and Microwave Tubes started. The following year, facilities for manufacture of Integrated Circuits and Hybrid Micro Circuits were set up.
- 1972 saw BEL manufacturing TV Transmitters for Doordarshan. The following year, manufacture of Frigate Radars for the Navy began.

1980

- BEL ventured to set up new Units at various places. The second Unit of BEL was set up at Ghaziabad in 1974 to manufacture Radars and Tropo communication equipment for the Indian Air Force.
- The third Unit was established at Pune in 1979 to manufacture Image Converter and Image Intensifier Tubes. In 1980, BEL's first overseas office was set up at New York for procurement of components and materials.

1982

- In 1981, a manufacturing facility for Magnesium Manganese Dioxide batteries was set up at the Pune Unit.
- The Space Electronic Division was set up at Bangalore to support the satellite programme in 1982. The same year saw BEL achieve a turnover of Rs.100 crores.

1985

- In 1983 (ASCO) was taken over by BEL as the fourth manufacturing Unit at Machilipatnam. In 1985, the fifth Unit was set up in Chennai for supply of Tank Electronics.
- The sixth Unit was set up at Panchkula the same year to manufacture Military Communication equipment. 1985 also saw BEL manufacturing on a large scale Low Power TV Transmitters and TVROs for the expansion of Doordarshan's coverage.

1989

- 1986 witnessed the setting up of the seventh Unit at Kotdwara to manufacture Switching Equipment, the eighth Unit to manufacture TV Glass Shell at Taloja (Navi Mumbai) and the ninth Unit at Hyderabad to manufacture Electronic Warfare Equipment.
- 1989 saw the manufacture of Telecom Switching and Transmission Systems.

1992

- The agreement for setting up BEL's first Joint Venture Company, BE DELFT, with M/s Delft of Holland was signed in 1990. Recently this became a subsidiary of BEL with the exit of the foreign partner and has been renamed BEL Optronic Devices Limited.
- The second Central Research Laboratory was established at Ghaziabad in 1992. The first disinvestment (20%) and listing of the Company's shares in Bangalore and Mumbai Stock Exchanges took place the same year.

1998

- TBEL Units obtained ISO 9000 certification in 1993-94. The second disinvestment (4.14%) took place in 1994. In 1996, BEL achieved Rs.1,000 crores turnover.
- In 1997, GE BEL, the Joint Venture Company with M/s GE, USA, was formed. In 1998, BEL set up its second overseas office at Singapore to source components from South East Asia.

2007

- The year 2000 saw the Bangalore Unit, which had grown very large, being reorganized into Strategic Business Units (SBUs). There are now nine SBUs in Bangalore Unit. The same year, BEL shares were listed in the National Stock Exchange.
- In 2002, BEL became the first defence PSU to get operational Mini Ratna Category I status. In June 2007, BEL was conferred the prestigious Navratna status based on its consistent performance.

2016

- During 2014-16, BEL recorded a turnover of Rs.6,695 crores

1.2 VISION

- To be a world-class enterprise in professional electronics.

1.3 MISSION

- To be a customer focused, globally competitive company in defense electronics and in other chosen areas of professional electronics, through quality, technology and innovation.

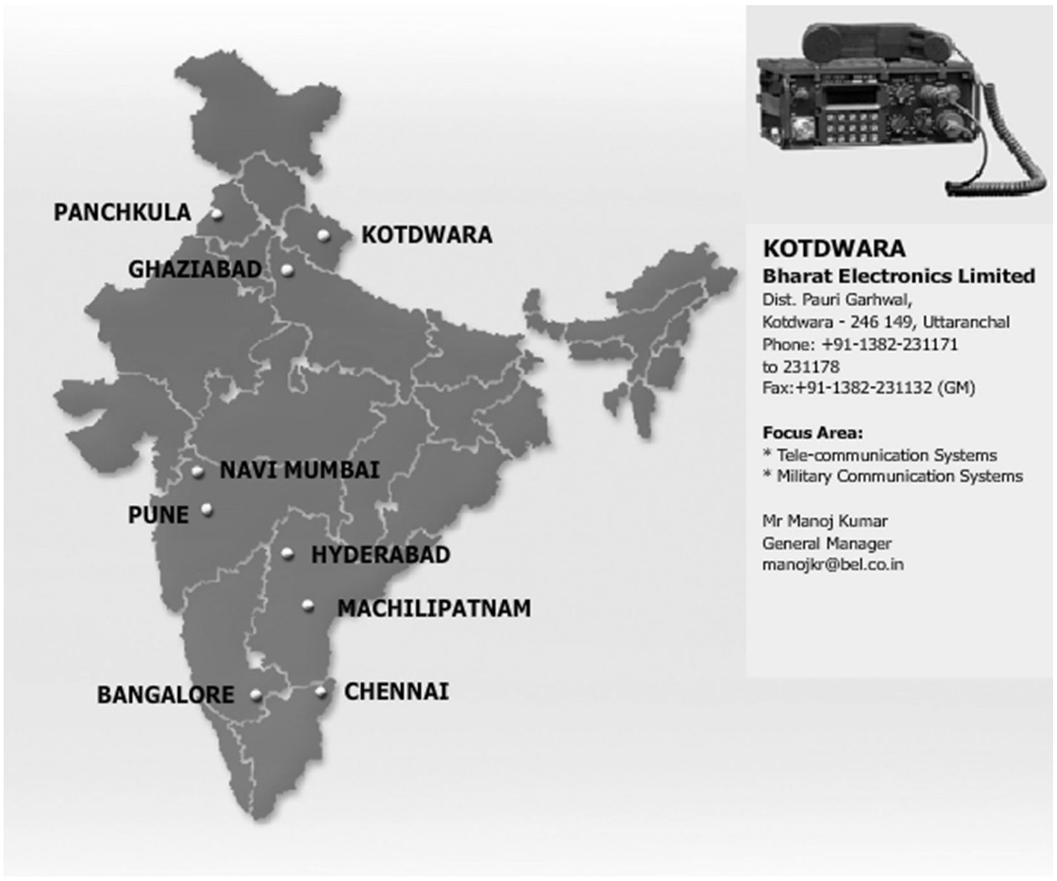
1.4 VALUES

- Putting customers first.
- Working with transparency, honesty & integrity.
- Trusting and respecting individuals.
- Fostering team work.
- Striving to achieve high employee satisfaction.
- Encouraging flexibility & innovation.
- Endeavoring to fulfill social responsibilities.
- Proud of being a part of the organization.

1.5 OBJECTIVES

- To be a customer focused company providing state-of-the-art products & solutions at competitive prices, meeting the demands of quality, delivery & service.
- To generate internal resources for profitable growth.
- To attain technological leadership in defense electronics through in-house R&D, partnership with defense/research laboratories & academic institutions.
- To give thrust to exports.
- To create a facilitating environment for people to realize their full potential through continuous learning & team work.
- To give value for money to customers & create wealth for shareholders.
- To constantly benchmark company's performance with best-in-class internationally.
- To raise marketing abilities to global standards.
- To strive for self-reliance through indigenization.

1.5 MANUFACTURING UNITS



KOTDWARA
Bharat Electronics Limited
Dist. Pauri Garhwal,
Kotdwara - 246 149, Uttaranchal
Phone: +91-1382-231171
to 231178
Fax:+91-1382-231132 (GM)

Focus Area:
* Tele-communication Systems
* Military Communication Systems

Mr Manoj Kumar
General Manager
manojkr@bel.co.in

1.6 PRODUCTS

DIGITAL EXCHANGE (DEX)

- A 128 PORT NON BLOCKING DIGITAL SWITCH WITH MODULAR CONCEPT FOR EASE
- OF MAINTENANCE, SYSTEM CAPACITY ENHANCEMENT AND SCALABILITY.

VERSATILE COMMUNICATION SYSTEM (VCS)

- A 128 PORT NON BLOCKING DIGITAL SWITCH ENABLING, COLLECTING AND
- ANALYSIS OF DATA FROM VARIOUS SHIPS AND WEAPON SYSTEMS.

STAND ALONE COMMUNICATION UNIT (SACU)

- BASED ON INTEL 80386DX MICROPROCESSOR TRIPLE MODEM PROVIDES COMMUNICATION
- OVER THREE INDEPENDENT CHANNELS

UNIT LEVEL SWITCH BOARD (ULSB MK-III)

- Unit level switch board MK-III (ULSB MK-III)equipment is an automatic telephone exchange designed to meet the communication requirements of Army at unit level.

PASSIVE NIGHT VISION GOGGLE

- Passive Night Vision Goggle is a compact and lightweight night vision system used for surveillance, vehicle driving and map reading at night. It is fitted with high performance XD-4 image intensifier tube.

And Many More…….

CHAPTER 2

SOFTWARE AND TECHNOLOGIES USED

2.1 INTRODUCTION

The project aims at developing a communication server entitled as "**VoIP Communication Using SIP**". This application primarily allows the client to easily register his/her device and communicate among various registered clients.

The Software is for communication using UDP.

It maintains two levels of users:

- Server Level
- Client Level

The Software includes:

- Maintaining client registering details.
- Providing important information like transfer of data packets, invite messages, registered device messages etc.
- Generating calls based on certain request number.

2.2 PURPOSE

The goal of our system is to develop and implement the time effective, user friendly software.

2.3 OBJECTIVE

The main objective of the VoIP communication using SIP is to increase communication between client and server which can help in providing a better platform for handling communication among clients. The scope of the project is very wide. This project is very flexible and can be easily expandable.

This application provides creation and management of a session, where a session is considered an exchange of data between an association of participants. The implementation of these applications is complicated by the practices of participants: users may move between endpoints, they may be addressable by multiple names, and they may communicate in several different media – sometimes simultaneously.

UDP (User Datagram Protocol) have been authored that carry various forms of real-time multimedia session data such as voice, video, or text messages. The Session Initiation Protocol (SIP) works in concert with these protocols by enabling Internet endpoints (called user agents) to discover one another and to agree on a characterization of a session they would like to share. For locating prospective session participants, and for other functions, SIP enables the creation of an infrastructure of network hosts (called proxy servers) to which user agents can send registrations, invitations to sessions, and other requests.
SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls.
SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports personal mobility - users can maintain a single externally visible identifier regardless of their network location.

SIP is an agile, general-purpose tool for creating, modifying, and terminating sessions that works independently of underlying transport protocols and without dependency on the type of session that is being established.

2.4 TECHNOLOGY USED

- JAVA
- JAVA Socket API
- UDP
- SIP

2.4.1 SOFTWARE USED

- NetBeans 8.0.2 IDE
- JDK 1.6 or higher
- WireShark
- Phoner (Virtual IP messenger)
- Windows (client)
- Linux (server)

2.4.2 HARDWARE REQUIRED

- A server

2.5 ADVANTAGES

- Low Cost
- Portable
- Reliable
- Cost effective
- No Extra Cables

2.6 SALIENT FEATURES

The major advantage of SIP is in its support for both IP and conventional telephone communication.

- Highly flexible, scalable and customizable.
- It is scalable, easy to implement, and requires less setup time.
- Since SIP can be used to modify any session in progress, a normal telephone call session can be converted into a multi-party videoconference. Users can join in the session no matter what kind of terminal he is using or where he is located. The other person may be logged on to Internet through a PC, or may be traveling with a cell phone.

2.7 OVERVIEW OF SIP

Given below are a few points to note about SIP:

- SIP is a signalling protocol used to create, modify, and terminate a multimedia session over the Internet Protocol. A session is nothing but a simple call between two endpoints. An endpoint can be a Smartphone, a laptop, or any device that can receive and transmit multimedia content over the Internet.
- SIP is an application layer protocol defined by IETF (Internet Engineering Task Force) standard. It is defined in **RFC 3261**.
- SIP is incorporated with two widely used internet protocols: **HTTP** for web browser and **SMTP** used for email. From HTTP, SIP borrowed the client-server architecture and the use of URL and URI. From SMTP, it borrowed a text encoding scheme and a header style.
- SIP takes the help of SDP (Session Description Protocol) which describes a session and RTP (Real Time Transport Protocol) used for delivering voice and video over IP network.
- SIP can be used for two-party (uncast) or multiparty (multicast) sessions.
- Other SIP applications include file transfer, instant messaging, video conferencing, online games, and steaming multimedia distribution.

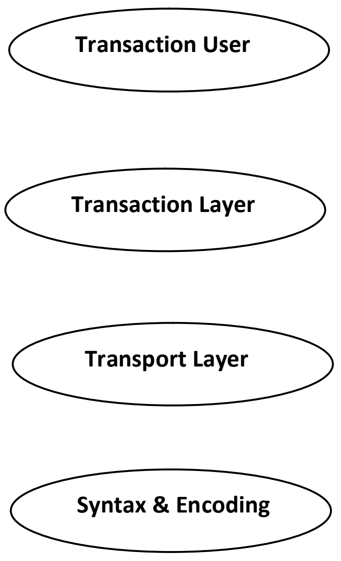
2.8 ADVANTAGES OF VoIP

Some advantages of VOIP include:

- Low cost
- Portability
- No extra cables
- Flexibility
- Video conferencing

2.9 SIP SYSTEM ARCHITECTURE

SIP is structured as a layered protocol, which means its behavior is described in terms of a set of fairly independent processing stages with only a loose coupling between each stage.



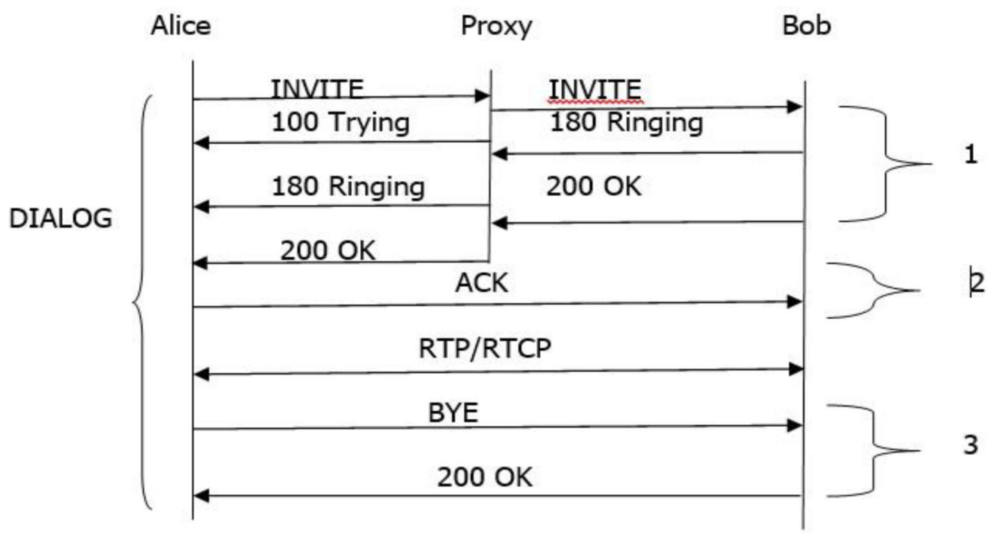
- The lowest layer of SIP is its **syntax and encoding**. Its encoding is specified using an augmented **Backus-Naur Form grammar** (BNF).

- At the second level is the **transport layer**. It defines how a Client sends requests and receives responses and how a Server receives requests and sends responses over the network. All SIP elements contain a transport layer.

- Next comes the **transaction layer**. A transaction is a request sent by a Client transaction (using the transport layer) to a Server transaction, along with all responses to that request sent from the server transaction back to the client. Any task that a user agent client (UAC) accomplishes takes place using a series of transactions. **Stateless proxies** do not contain a transaction layer.

- The layer above the transaction layer is called the **transaction user**. Each of the SIP entities, except the **stateless proxy**, is a transaction user.

2.10 BASIC CALL FLOW IN SIP



Given below is a step-by-step explanation of the above call flow:

- An INVITE request that is sent to a proxy server is responsible for initiating a session.

- The proxy server sends a **100 Trying** response immediately to the caller (Alice) to stop the re-transmissions of the INVITE request.

- The proxy server searches the address of Bob in the location server. After getting the address, it forwards the INVITE request further.

- Thereafter, **180 Ringing** (Provisional responses) generated by Bob is returned back to Alice.
- A **200 OK** response is generated soon after Bob picks the phone up.
- Bob receives an **ACK** from the Alice, once it gets **200 OK**.
- At the same time, the session gets established and RTP packets (conversations) start flowing from both ends.
- After the conversation, any participant (Alice or Bob) can send a **BYE** request to terminate the session.
- **BYE** reaches directly from Alice to Bob bypassing the proxy server.
- Finally Bob sends a **200 OK** response to confirm the BYE and the session is terminated.
- In the above basic call flow, three **transactions** are (marked as 1, 2, 3) available.
- The complete call (from INVITE to 200 OK) is known as a **Dialog**.

2.11 INTRODUCTION TO SIP MESSAGES

SIP messages are of two types: **REQUESTS** and **RESPONSES**.

- The opening line of a request contains a method that defines the request, and a Request-URI that defines where the request is to be sent.
- Similarly the opening line of a response contains a response code.

REQUEST METHODS

SIP requests are the codes used to establish a communication. To complement them, there are **SIP responses** that generally indicate whether a request succeeded or failed.

There are commands known as METHODS that make a SIP message workable.

- METHODS can be regarded as SIP requests, since they request a specific action to be taken by another user agent or server.
- METHODS are distinguished into two types:
  - Core Methods
  - Extension Methods

CORE METHODS

There are six core methods as discussed below.

INVITE

- INVITE is used to initiate a session with a user agent. In other words, an INVITE method is used to establish a media session between the user agents.
- INVITE can contain the media information of the caller in the message body.
- A session is considered established if an INVITE has received a success response (2xx) or an ACK has been sent.
- A successful INVITE request establishes a **dialog** between the two user agents which continues until a BYE is sent to terminate the session.
- An INVITE sent within an established dialog is known as a **re-INVITE**.
- Re-INVITE is used to change the session characteristics or refresh the state of a dialog.

INVITE Example

The following code shows how INVITE is used.

```
INVITE sips:Bob@TMC.com SIP/2.0
Via: SIP/2.0/TLS client.ANC.com:5061;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice<sips:Alice@atlanta.com>;tag=1234567
To: Bob<sips:Bob@TMC.com>
Call-ID: 12345601@ANC.com
CSeq: 1 INVITE
Contact: <sips:Alice@client.ANC.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces
Content-Type: application/sdp
Content-Length: ...
v=0
o=Alice 2890844526 2890844526 IN IP4 client.ANC.com
s=Session SDP
c=IN IP4 client.ANC.com
t=3034423619 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

BYE

BYE is the method used to terminate an established session. This is a SIP request that can be sent by either the caller or the callee to end a session.

- It cannot be sent by a proxy server.
- BYE request normally routes end to end, bypassing the proxy server.
- BYE cannot be sent to a pending an INVITE or an unestablished session.

REGISTER

REGISTER request performs the registration of a user agent. This request is sent by a user agent to a registrar server.

- The REGISTER request may be forwarded or proxied until it reaches an authoritative registrar of the specified domain.
- It carries the AOR (Address of Record) in the **To**header of the user that is being registered.
- REGISTER request contains the time period (3600 sec).

- One user agent can send a REGISTER request on behalf of another user agent. This is known as **third-party registration**. Here, the **From** tag contains the URI of the party submitting the registration on behalf of the party identified in the **To** header.

CANCEL

CANCEL is used to terminate an unestablished session. User agents use this request to cancel a pending call attempt initiated earlier.

- It can be sent either by a user agent or a proxy server.
- CANCEL is a **hop by hop** request, i.e., it goes through the elements between the user agent and receives the response generated by the next stateful element.

ACK

- ACK is used to acknowledge the final responses to an INVITE method. An ACK always goes in the direction of INVITE. ACK may contain SDP body (media characteristics), if it is not available in INVITE.
- ACK may not be used to modify the media description that has already been sent in the initial INVITE.
- A stateful proxy receiving an ACK must determine whether or not the ACK should be forwarded downstream to another proxy or user agent.
- For 2xx responses, ACK is end to end, but for all other final responses, it works on hop by hop basis when stateful proxies are involved.

OPTIONS

OPTIONS method is used to query a user agent or a proxy server about its capabilities and discover its current availability. The response to a request lists the capabilities of the user agent or server. A proxy never generates an OPTIONS request.

EXTENSION METHODS

SUBSCRIBE

- SUBSCRIBE is used by user agents to establish a subscription for the purpose of getting notification about a particular event.
- It has a time period in the **Expires** header field that indicates the desired duration of existence of a subscription.
- After the specified time period passes, the subscription is automatically terminated.
- A successful subscription establishes a dialog between the user agents.
- A subscription can be refreshed by sending another SUBSCRIBE within the dialog before the expiration time.
- The server accepting a subscription returns a 200 OK.
- Users can unsubscribe by sending another SUBSCRIBE method with Expires value 0 (zero).

NOTIFY

- NOTIFY is used by user agents to convey the occurrence of a particular event. A NOTIFY is always sent within a dialog when a subscription exists between the subscriber and the notifier.
- A 200 OK response is received for every NOTIFY to indicate that it has been received.
- NOTIFY requests contain an **Event** header field indicating the package and a **subscription-state** header field indicating the current state of the subscription.
- A NOTIFY is always sent at the start of a subscription and at the termination of a subscription.

PUBLISH

- PUBLISH is used by a user agent to send event state information to a server known as an event state compositor.
- PUBLISH is mostly useful when there are multiple sources of event information.
- A PUBLISH request is similar to a NOTIFY, except that it is not sent in a dialog.

- A PUBLISH request must contain an Expires header field and a Min-Expires header field.

INFO

INFO is used by a user agent to send call signalling information to another user agent with which it has established a media session. This is an end-to-end request and never generate by proxies. A proxy will always forward an INFO request.

UPDATE

UPDATE is used to modify the state of a session without changing the state of the dialog. UPDATE is used if a session is not established and the user wants to change the codec.

IF a session is established, a re-Invite is used to change/update the session.

MESSAGE

It is used to send an instant message or **IM** using SIP. An IM usually consists of short messages exchanged in real time by participants engaged in text conversation.

- MESSAGE can be sent within a dialog or outside a dialog.
- The contents of a MESSAGE are carried in the message body as a **MIME** attachment.

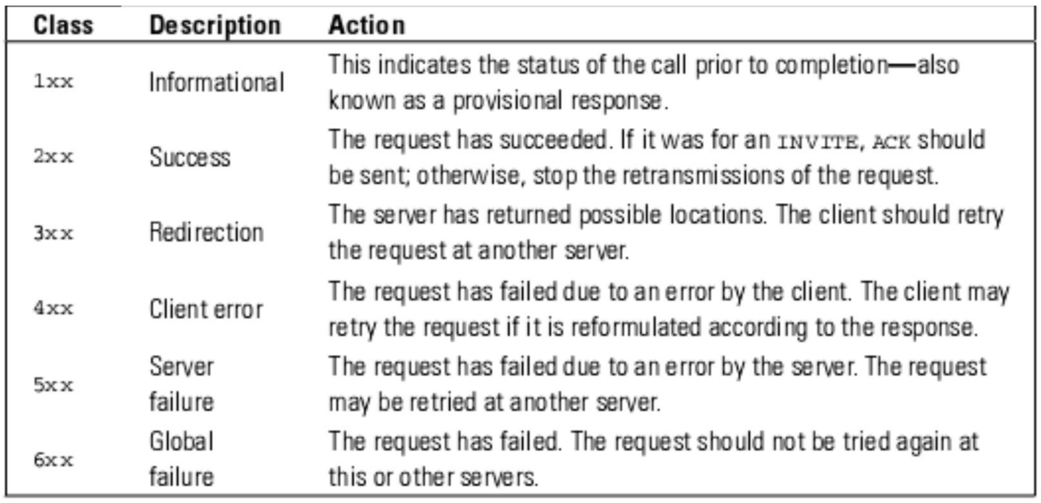
A **200 OK** response is normally received to indicate that the message has been delivered at its destination.

2.12 RESPONSE CODE IN SIP

A SIP response is a message generated by a user agent server (UAS) or SIP server to reply a request generated by a client. It could be a formal acknowledgement to prevent retransmission of requests by a UAC.

- A response may contain some additional header fields of info needed by a UAC.
- SIP has six responses.
- 1xx to 5xx has been borrowed from HTTP and 6xx is introduced in SIP.
- 1xx is considered as a **provisional** response and the rest are **final** responses.

Given below is the description of each response code :



Class	Description	Action
1xx	Informational	This indicates the status of the call prior to completion—also known as a provisional response.
2xx	Success	The request has succeeded. If it was for an INVITE, ACK should be sent; otherwise, stop the retransmissions of the request.
3xx	Redirection	The server has returned possible locations. The client should retry the request at another server.
4xx	Client error	The request has failed due to an error by the client. The client may retry the request if it is reformulated according to the response.
5xx	Server failure	The request has failed due to an error by the server. The request may be retried at another server.
6xx	Global failure	The request has failed. The request should not be tried again at this or other servers.

2.13 ABOUT SDP

SDP stands for Session Description Protocol. It is used to describe multimedia sessions in a format understood by the participants over a network. Depending on this description, a party decides whether to join a conference or when or how to join a conference.

- The owner of a conference advertises it over the network by sending multicast messages which contain description of the session e.g. the name of the owner, the name of the session, the coding, the timing etc. Depending on these information the recipients of the advertisement take a decision about participation in the session.
- SDP is generally contained in the body part of Session Initiation Protocol popularly called SIP.
- SDP is defined in RFC 2327. An SDP message is composed of a series of lines, called fields, whose names are abbreviated by a single lower-case letter, and are in a required order to simplify parsing.

2.13.1 PURPOSE OF SDP

The purpose of SDP is to convey information about media streams in multimedia sessions to help participants join or gather info of a particular session.

- SDP is a short structured textual description.

- It conveys the name and purpose of the session, the media, protocols, codec formats, timing and transport information.
- A tentative participant checks these information and decides whether to join a session and how and when to join a session if it decides to do so.
- The format has entries in the form of <type>= <value>, where the <type>defines a unique session parameter and the <value>provides a specific value for that parameter.
- The general form of a SDP message is:

  x=parameter1 parameter2 ... parameterN

- The line begins with a single lower-case letter, for example, x. There are never any spaces between the letter and the =, and there is exactly one space between each parameter. Each field has a defined number of parameters.

2.13.2 SESSION DESCRIPTION PARAMETERS

Session description (* denotes optional)

- ▪ v= (protocol version)
- ▪ o= (owner/creator and session identifier)
- ▪ s= (session name)
- ▪ i=* (session information)
- ▪ u=* (URI of description)
- ▪ e=* (email address)
- ▪ p=* (phone number)
- ▪ c=* (connection information -not required if included in all media)
- ▪ b=* (bandwidth information)
- ▪ z=* (time zone adjustments)
- ▪ k=* (encryption key)
- ▪ a=* (zero or more session attribute lines)

An SDP Example

Given below is an example session description, taken from RFC 2327 :

```
v=0
o=mhandley2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu(Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=application 32416udp wb
a=orient:portrait
```

2.14 NETBEANS IDE SOFTWARE

NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform, including the NetBeans integrated development environment (IDE), can be extended by third party developers.

The NetBeans IDE is primarily intended for development in Java, but also supports other languages , in particular PHP,C/C++ and HTML.

NetBeans is a cross-platform and runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM.

2.15 METHODOLOGY ADOPTED

I followed a 5-Step Development Methodology to develop the "**VoIP Communication using SIP**". The 5-Step Development Methodology that I adopted is listed below:

- **Study and Learning** : Learning java and studying all software to be used like sever and communication software.
- **Layout** : Scoping out the details and determining what type of design, programming, function, etc. is best suited for theparticular solution for Communication between two clients that a development needs. Then refining and documenting of the design was carried out.
- **Development** : Once the specs were detailed, I began the process to build a "VoIP using SIP". During this phase the faculty/guide monitored the development through work in-progress.
- **Implementation** : This is when the site went through tweaking and testing. I refined the codes and finalized the test of communication between two clients as per the project.
- **Demo** : The final step of the methodology that I plan to follow will be to give a demo of the fully developed and tested "VoIP software using SIP".

CHAPTER 3

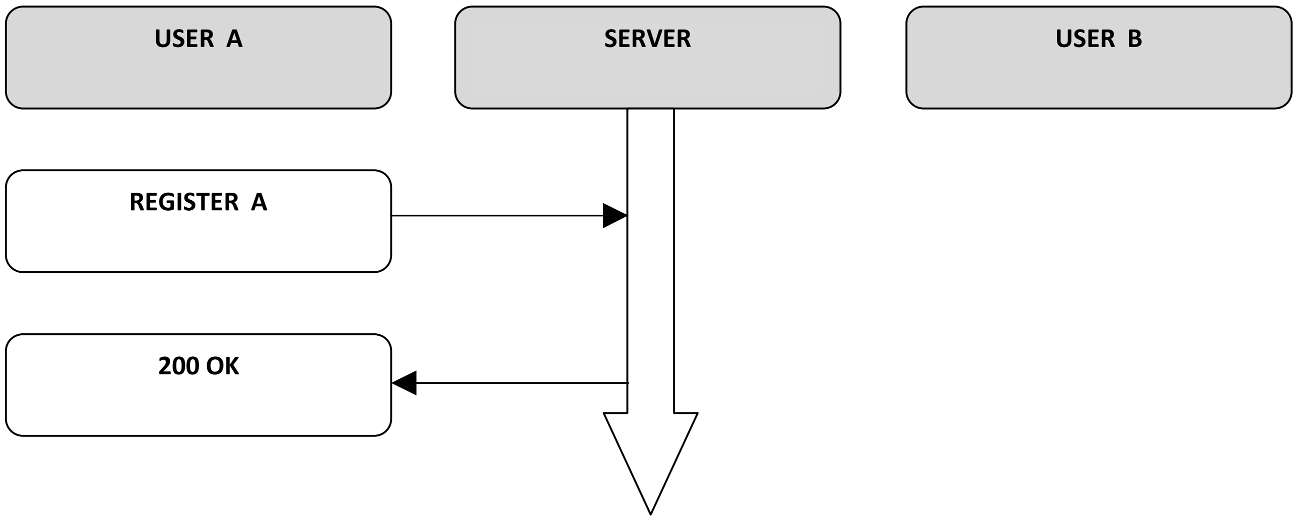
DESIGN AND IMPLEMENTATION

This chapter deals with the basic design and its implementation in creation of this software. This JAVA based server is designed using best OOM practices and takes full advantage of JAVA's Object Oriented Methodology.

3.1 BASIC FLOW CHARTS

3.1.1 REGISTRATION OF USERS

A user sends a REGISTER request to the server. The request includes user's contact list. The SIP Server validates the user's credentials and if it succeeds then user is successfully registered and server replies with 200 OK response.



Actual Message flow of Registration Process is as follows :

REGISTER USER A -> SERVER

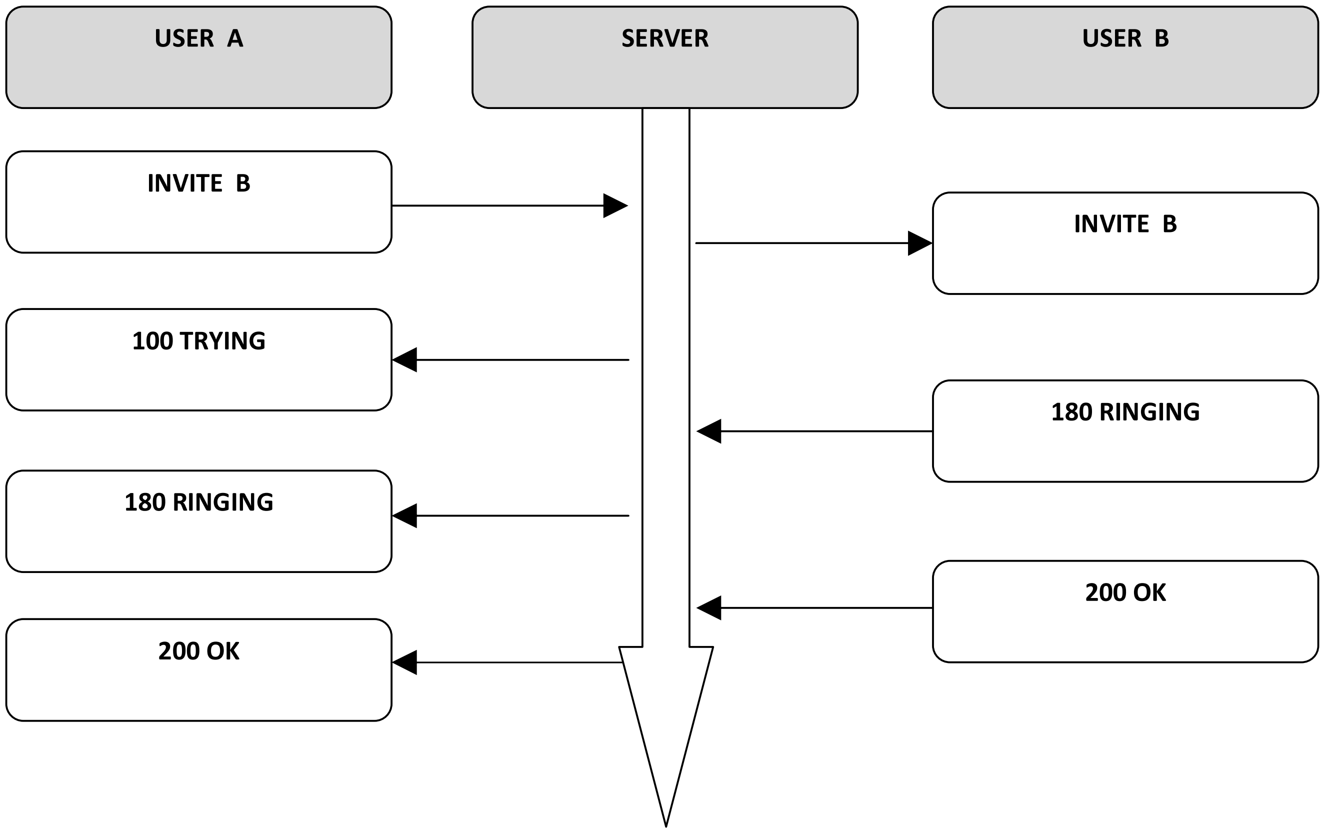
```
REGISTER sip:192.168.43.23 SIP/2.0
Call-ID: e94e44bed7d447e6707ff5c8d5582410@0:0:0:0:0:0:0:0
CSeq: 1 REGISTER
From: "4060" <sip:4060@192.168.43.23>;tag=5296f111
To: "4060" <sip:4060@192.168.43.23>
Via: SIP/2.0/UDP 192.168.43.81:61284;branch=z9hG4bK-363638-4a734d557563fd1c90978c058a68bea3
Max-Forwards: 70
User-Agent: Jitsi2.8.5426Windows 8
Expires: 600
Contact: "4060" <sip:4060@192.168.43.81:61284;transport=udp;registering_acc=192_168_43_23>;expires=600
Content-Length: 0
```

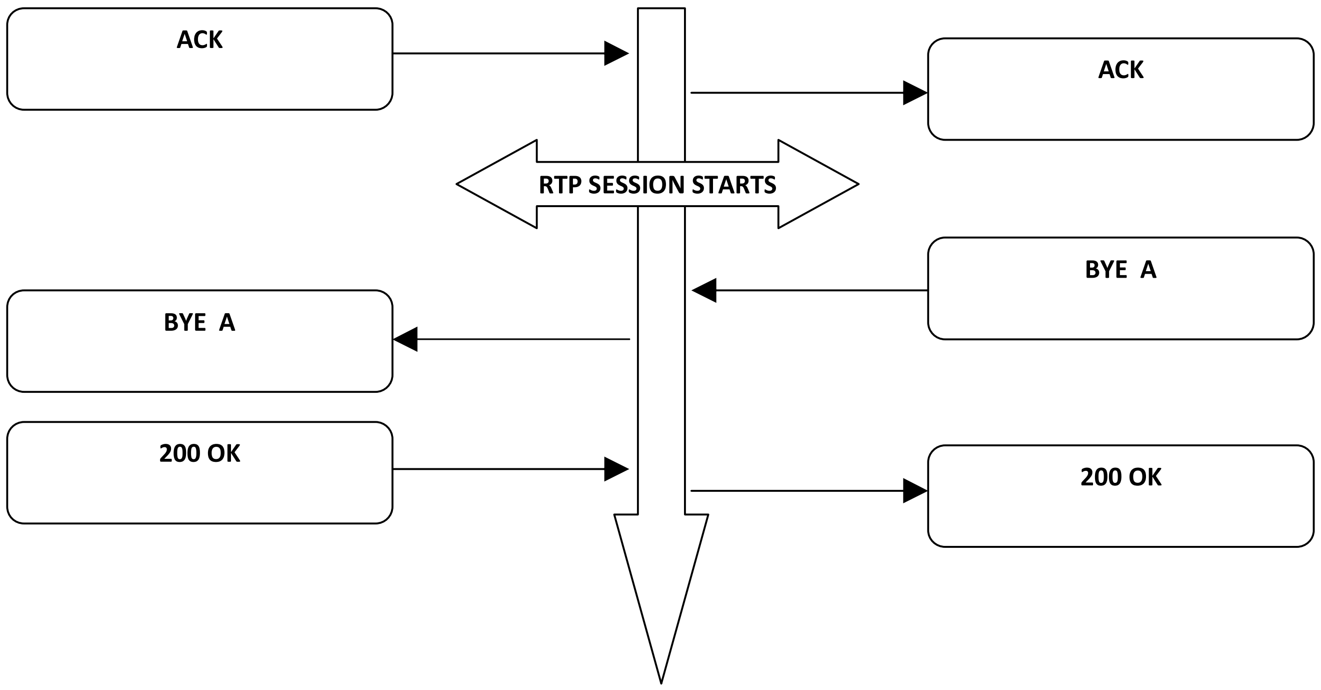
OK SERVER -> USER A

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.43.81:61284;branch=z9hG4bK-363638-4a734d557563fd1c90978c058a68bea3
From: "4060" <sip:4060@192.168.43.23>;tag=5296f111
To: "4060" <sip:4060@192.168.43.23>
Call-ID: e94e44bed7d447e6707ff5c8d5582410@0:0:0:0:0:0:0:0
CSeq: 1 REGISTER
Contact: "4060" <sip:4060@192.168.43.81:61284;transport=udp;registering_acc=192_168_43_23>;expires=600
Allow:
Max-Forwards: 70
Allow-Events:
User-Agent: Jitsi2.8.5426Windows 8
Supported:
Expires: 600
Content-Length: 0
```

3.1.2 NORMAL FLOW

In this flow of control, the caller makes a call through an INVITE request which is handled by the server. The server figures out the correct recipient of the call and forwards it. It also sends back a TRYING 100 response back to the caller. The receiver recognizes the INVITE message and there is a ringing bell on the phone of the receiver. He picks up the call and a successful RTP connection is established between both the users. Any user can drop the call by sending a BYE request to the other user. The other user sends an appropriate ACK response and the session terminates.





The Actual Message flow recorded is as follows :

INVITE USER A -> SERVER

```
INVITE sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Contact: <sip:8060@192.168.43.81:5000>
Content-Type: application/sdp
Allow: INVITE, ACK, BYE, CANCEL, INFO, MESSAGE, NOTIFY, OPTIONS, REFER, UPDATE
Max-Forwards: 70
Supported: 100rel, replaces, from-change
P-Early-Media: supported
User-Agent: SIPPER for phoner
P-Preferred-Identity: <sip:8060@192.168.43.23>
Content-Length:   476

v=0
o=- 3361950259 1 IN IP4 192.168.43.81
s=SIPPER for phoner
c=IN IP4 192.168.43.81
t=0 0
m=audio 5002 RTP/AVP 8
a=rtpmap:8 PCMA/8000
a=sendrecv
```

INVITE SERVER -> USER B

```
INVITE sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Contact: <sip:8060@192.168.43.23>
Content-Type: application/sdp
Max-Forwards: 69
User-Agent: SIPPER for phoner
Content-Length:   476
```

```
v=0
o=- 3361950259 1 IN IP4 192.168.43.81
s=SIPPER for phoner
c=IN IP4 192.168.43.81
t=0 0
m=audio 5002 RTP/AVP 8
a=rtpmap:8 PCMA/8000
a=sendrecv
```

TRYING SERVER -> USER A

```
SIP/2.0 100 TRYING
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Allow: INVITE, ACK, BYE, CANCEL, INFO, MESSAGE, NOTIFY, OPTIONS, REFER, UPDATE
User-Agent: SIPPER for phoner
Supported: 100rel, replaces, from-change
Content-Length: 0
```

RINGING USER B -> SERVER

```
SIP/2.0 180 Ringing
CSeq: 3 INVITE
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790,SIP/2.0/UDP
192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
Contact: "4060" <sip:4060@192.168.43.81:61284;transport=udp;registering_acc=192_168_43_23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

RINGING SERVER -> USER A

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Contact: "4060" <sip:4060@192.168.43.23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

OK USER B -> SERVER

```
SIP/2.0 200 OK
CSeq: 3 INVITE
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790,SIP/2.0/UDP
192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
Contact: "4060" <sip:4060@192.168.43.81:61284;transport=udp;registering_acc=192_168_43_23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Type: application/sdp
Content-Length: 167

v=0
o=4060-jitsi.org 0 0 IN IP4 192.168.43.81
s=-
c=IN IP4 192.168.43.81
t=0 0
m=audio 5001 RTP/AVP 8 0
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtcp:5004
```

OK SERVER -> USER A

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Content-Length: 167

v=0
o=4060-jitsi.org 0 0 IN IP4 192.168.43.81
s=-
c=IN IP4 192.168.43.81
t=0 0
m=audio 5001 RTP/AVP 8 0
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtcp:5004
```

ACK USER A -> SERVER

```
ACK sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK00e851e8c345e611906d832a577aa99e;rport
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 ACK
Contact: <sip:8060@192.168.43.81:5000>
Max-Forwards: 70
Content-Length: 0
```

ACK SERVER -> USER B

```
ACK sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK00e851e8c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 ACK
Contact: <sip:8060@192.168.43.81:5000>
Content-Length: 0
```

BYE USER B -> SERVER

```
BYE sip:8060@192.168.43.23 SIP/2.0
CSeq: 1 BYE
From: <sip:4060@192.168.43.23>;tag=ff84cb15
To: <sip:8060@192.168.43.23>;tag=1443862458
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
Max-Forwards: 70
Via: SIP/2.0/UDP 192.168.43.81:61284;branch=z9hG4bK-363638-e9845c7bc20c1c01b1db6c8d9e7786ba
Contact: "4060" <sip:4060@192.168.43.81:61284;transport=udp;registering_acc=192_168_43_23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

BYE SERVER -> USER A

```
BYE sip:8060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790
Via: SIP/2.0/UDP 192.168.43.81:61284;branch=z9hG4bK-363638-e9845c7bc20c1c01b1db6c8d9e7786ba;recieved=192.168.43.81
From: <sip:4060@192.168.43.23>;tag=ff84cb15
To: <sip:8060@192.168.43.23>;tag=1443862458
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 1 BYE
Contact: "4060" <sip:4060@192.168.43.23>
Max-Forwards: 69
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

OK USER A -> SERVER

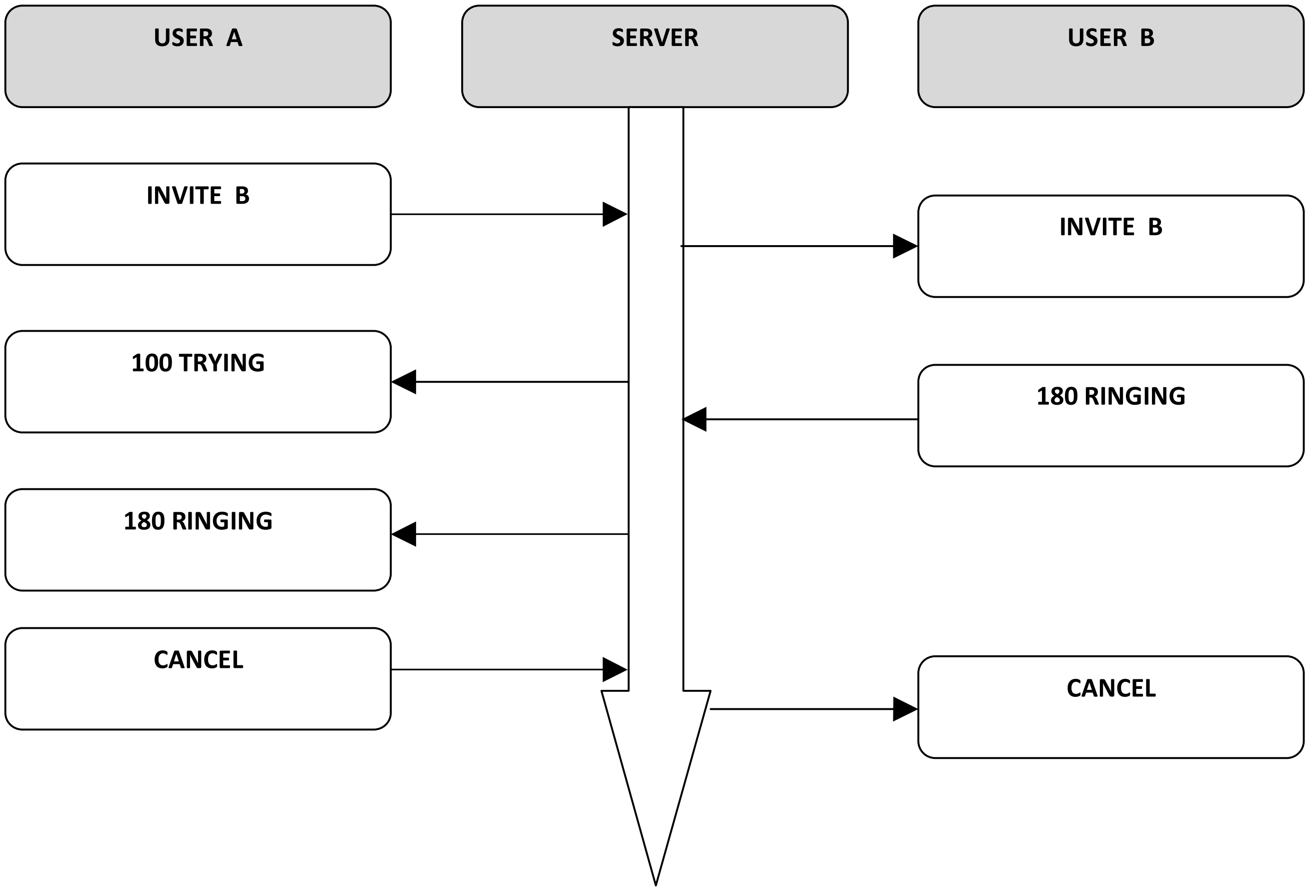
```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790
Via: SIP/2.0/UDP 192.168.43.81:61284;branch=z9hG4bK-363638-e9845c7bc20c1c01b1db6c8d9e7786ba;recieved=192.168.43.81
From: <sip:4060@192.168.43.23>;tag=ff84cb15
To: <sip:8060@192.168.43.23>;tag=1443862458
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 1 BYE
Contact: <sip:8060@192.168.43.81:5000>
Server: SIPPER for phoner
Content-Length: 0
```

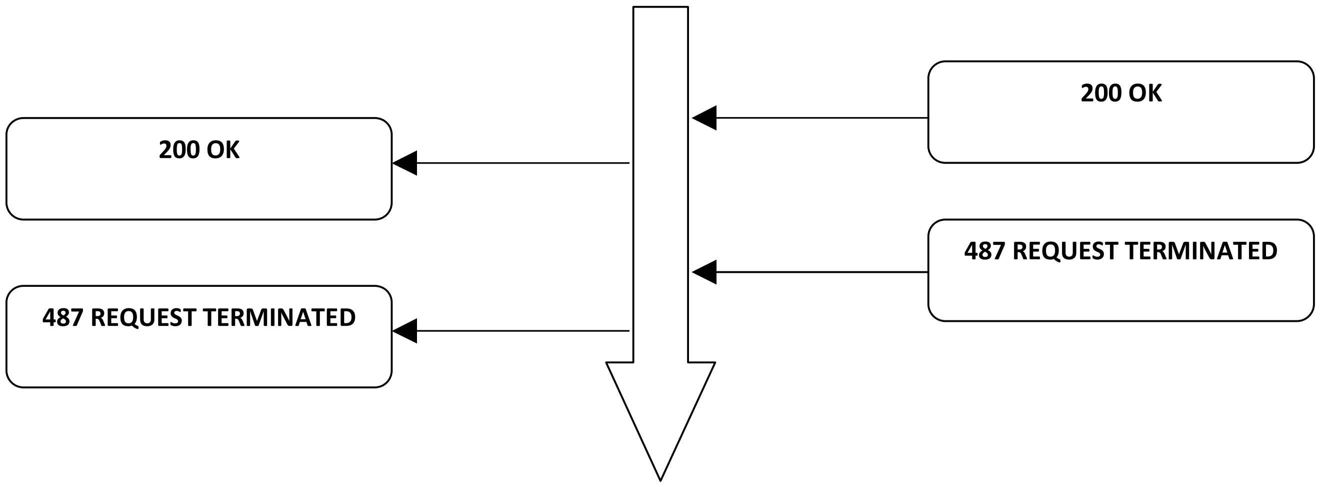
OK SERVER -> USER B

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.43.81:61284;branch=z9hG4bK-363638-e9845c7bc20c1c01b1db6c8d9e7786ba;recieved=192.168.43.81
From: <sip:4060@192.168.43.23>;tag=ff84cb15
To: <sip:8060@192.168.43.23>;tag=1443862458
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 1 BYE
Content-Length: 0
```

3.1.3 CANCELLATION PROCESS

When the user who is calling, ends the call then a CANCEL request is sent to the server. The server forwards the CANCEL request to the receiver. Receiver sends an 200 OK and a 487 REQUEST TERMINATED response. The session is then cancelled successfully.





The Actual Message flow recorded is as follows :

INVITE USER A -> SERVER

```
INVITE sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Contact: <sip:8060@192.168.43.81:5000>
Content-Type: application/sdp
Allow: INVITE, ACK, BYE, CANCEL, INFO, MESSAGE, NOTIFY, OPTIONS, REFER, UPDATE
Max-Forwards: 70
Supported: 100rel, replaces, from-change
P-Early-Media: supported
User-Agent: SIPPER for phoner
P-Preferred-Identity: <sip:8060@192.168.43.23>
Content-Length:   476

v=0
o=- 3361950259 1 IN IP4 192.168.43.81
s=SIPPER for phoner
c=IN IP4 192.168.43.81
t=0 0
m=audio 5002 RTP/AVP 8
a=rtpmap:8 PCMA/8000
a=sendrecv
```

INVITE SERVER -> USER B

```
INVITE sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Contact: <sip:8060@192.168.43.23>
Content-Type: application/sdp
Max-Forwards: 69
User-Agent: SIPPER for phoner
Content-Length:   476

v=0
o=- 3361950259 1 IN IP4 192.168.43.81
s=SIPPER for phoner
c=IN IP4 192.168.43.81
t=0 0
```

```
m=audio 5002 RTP/AVP 8
a=rtpmap:8 PCMA/8000
a=sendrecv
```

TRYING SERVER -> USER A

```
SIP/2.0 100 TRYING
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Allow: INVITE, ACK, BYE, CANCEL, INFO, MESSAGE, NOTIFY, OPTIONS, REFER, UPDATE
User-Agent: SIPPER for phoner
Supported: 100rel, replaces, from-change
Content-Length: 0
```

RINGING USER B -> SERVER

```
SIP/2.0 180 Ringing
CSeq: 3 INVITE
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790,SIP/2.0/UDP
192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
Contact: "4060" <sip:4060@192.168.43.81:61284;transport=udp;registering_acc=192_168_43_23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

RINGING SERVER -> USER A

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Contact: "4060" <sip:4060@192.168.43.23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

CANCEL USER A -> SERVER

```
CANCEL sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK80f677efc145e6119d2ef15dce95fe22;rport
From: <sip:8060@192.168.43.23>;tag=696308355
To: <sip:4060@192.168.43.23>
Call-ID: 80F677EF-C145-E611-9D2D-F15DCE95FE22@192.168.43.81
CSeq: 3 CANCEL
Allow: INVITE, ACK, BYE, CANCEL, INFO, MESSAGE, NOTIFY, OPTIONS, REFER, UPDATE
Max-Forwards: 70
User-Agent: SIPPER for phoner
Content-Length: 0
```

CANCEL SERVER -> USER B

```
CANCEL sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK80f677efc145e6119d2ef15dce95fe22;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=696308355
To: <sip:4060@192.168.43.23>
Call-ID: 80F677EF-C145-E611-9D2D-F15DCE95FE22@192.168.43.81
CSeq: 3 CANCEL
Max-Forwards: 69
User-Agent: SIPPER for phoner
Content-Length: 0
```

OK USER B -> SERVER

```
SIP/2.0 200 OK
CSeq: 3 CANCEL
Call-ID: 80F677EF-C145-E611-9D2D-F15DCE95FE22@192.168.43.81
From: <sip:8060@192.168.43.23>;tag=696308355
To: <sip:4060@192.168.43.23>;tag=fb48d1b7
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790,SIP/2.0/UDP
192.168.43.81:5000;branch=z9hG4bK80f677efc145e6119d2ef15dce95fe22;rport;recieved=192.168.43.81
Contact: "4060" <sip:4060@192.168.43.81:6819;transport=udp;registering_acc=192_168_43_23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

OK SERVER -> USER A

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK80f677efc145e6119d2ef15dce95fe22;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=696308355
To: <sip:4060@192.168.43.23>;tag=fb48d1b7
Call-ID: 80F677EF-C145-E611-9D2D-F15DCE95FE22@192.168.43.81
CSeq: 3 CANCEL
Contact: "4060" <sip:4060@192.168.43.23>
Content-Length: 0
```

REQUEST TERMINATED USER B -> SERVER

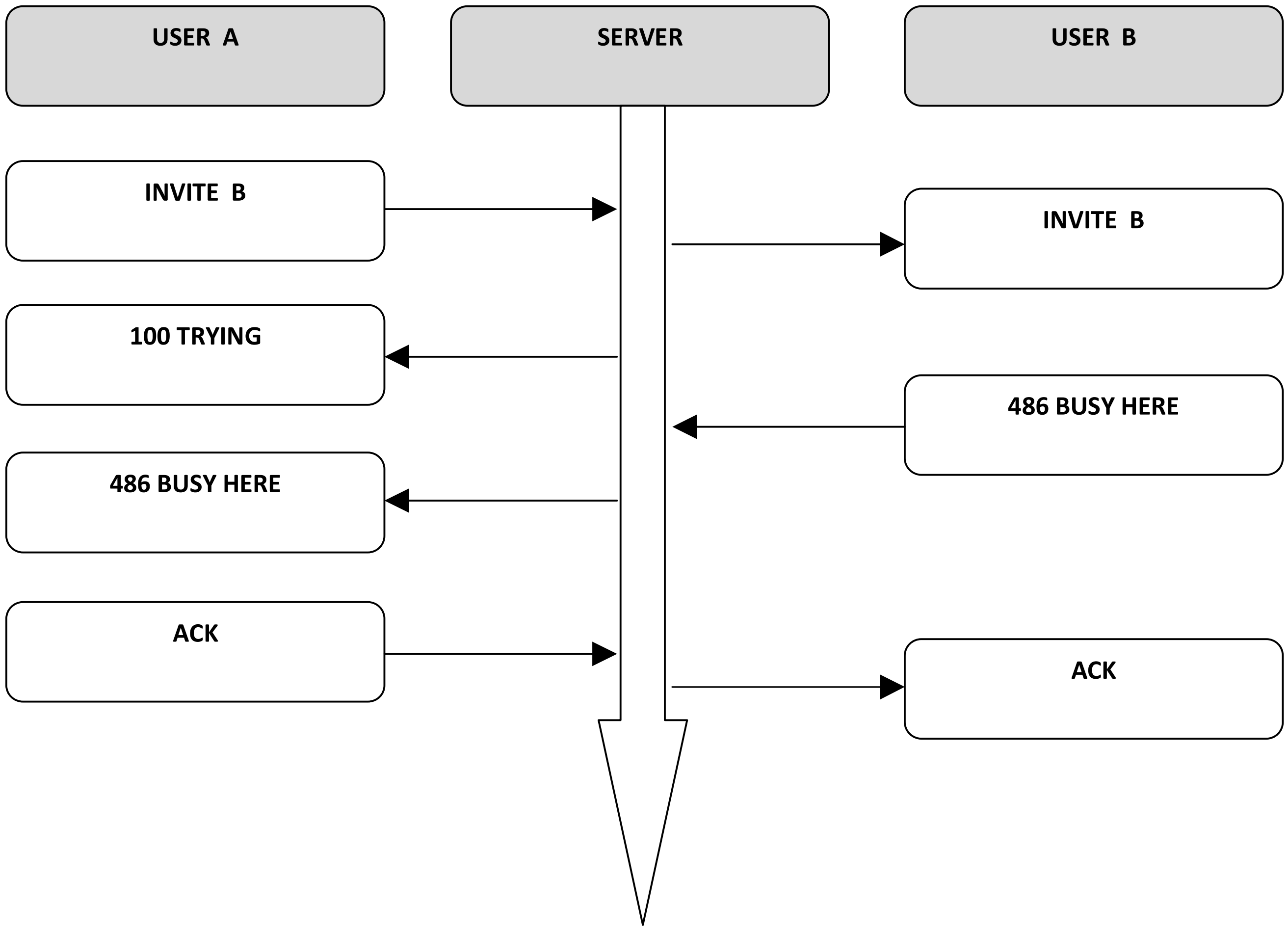
```
SIP/2.0 487 Request Terminated
CSeq: 3 INVITE
Call-ID: 80F677EF-C145-E611-9D2D-F15DCE95FE22@192.168.43.81
From: <sip:8060@192.168.43.23>;tag=696308355
To: <sip:4060@192.168.43.23>;tag=fb48d1b7
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790,SIP/2.0/UDP
192.168.43.81:5000;branch=z9hG4bK80f677efc145e6119d2ef15dce95fe22;rport;recieved=192.168.43.81
Contact: "4060" <sip:4060@192.168.43.81:6819;transport=udp;registering_acc=192_168_43_23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

REQUEST TERMINATED SERVER -> USER A

```
SIP/2.0 487 Request Terminated
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK80f677efc145e6119d2ef15dce95fe22;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=696308355
To: <sip:4060@192.168.43.23>;tag=fb48d1b7
Call-ID: 80F677EF-C145-E611-9D2D-F15DCE95FE22@192.168.43.81
CSeq: 3 INVITE
Contact: "4060" <sip:4060@192.168.43.23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

3.1.4 WHEN BUSY

When the receiver terminates the call during the ringing phase, then the caller receives 486 BUSY HERE response. In this scenario, reciever is busy and sends a 486 BUSY HERE response to caller's INVITE. Note that the non 2xx response is acknowledged on a hop-by-hop basis instead of end-to-end. Also note that many SIP UAs will not return a 486 response, as they have multiple lines and other features.



The Actual Message flow recorded is as follows :

INVITE USER A -> SERVER

```
INVITE sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Contact: <sip:8060@192.168.43.81:5000>
Content-Type: application/sdp
Allow: INVITE, ACK, BYE, CANCEL, INFO, MESSAGE, NOTIFY, OPTIONS, REFER, UPDATE
Max-Forwards: 70
Supported: 100rel, replaces, from-change
P-Early-Media: supported
User-Agent: SIPPER for phoner
P-Preferred-Identity: <sip:8060@192.168.43.23>
Content-Length:   476

v=0
o=- 3361950259 1 IN IP4 192.168.43.81
s=SIPPER for phoner
c=IN IP4 192.168.43.81
t=0 0
m=audio 5002 RTP/AVP 8
a=rtpmap:8 PCMA/8000
a=sendrecv
```

INVITE SERVER -> USER B

```
INVITE sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Contact: <sip:8060@192.168.43.23>
Content-Type: application/sdp
Max-Forwards: 69
User-Agent: SIPPER for phoner
Content-Length:   476

v=0
o=- 3361950259 1 IN IP4 192.168.43.81
s=SIPPER for phoner
c=IN IP4 192.168.43.81
t=0 0
m=audio 5002 RTP/AVP 8
a=rtpmap:8 PCMA/8000
a=sendrecv
```

TRYING SERVER -> USER A

```
SIP/2.0 100 TRYING
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Allow: INVITE, ACK, BYE, CANCEL, INFO, MESSAGE, NOTIFY, OPTIONS, REFER, UPDATE
User-Agent: SIPPER for phoner
Supported: 100rel, replaces, from-change
Content-Length: 0
```

RINGING USER B -> SERVER

```
SIP/2.0 180 Ringing
CSeq: 3 INVITE
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790,SIP/2.0/UDP
192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
Contact: "4060" <sip:4060@192.168.43.81:61284;transport=udp;registering_acc=192_168_43_23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

RINGING SERVER -> USER A

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK802488e6c345e611906d832a577aa99e;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=1443862458
To: <sip:4060@192.168.43.23>;tag=ff84cb15
Call-ID: 802488E6-C345-E611-906C-832A577AA99E@192.168.43.81
CSeq: 3 INVITE
Contact: "4060" <sip:4060@192.168.43.23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

BUSY HERE USER B -> SERVER

```
SIP/2.0 486 Busy here
CSeq: 3 INVITE
Call-ID: 80D32C76-C245-E611-AF8B-005EA72B6F73@192.168.43.81
From: <sip:8060@192.168.43.23>;tag=3059277927
To: <sip:4060@192.168.43.23>;tag=725a0d84
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790,SIP/2.0/UDP
192.168.43.81:5000;branch=z9hG4bK80d32c76c245e611af8c005ea72b6f73;rport;recieved=192.168.43.81
Contact: "4060" <sip:4060@192.168.43.81:29034;transport=udp;registering_acc=192_168_43_23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

BUSY HERE SERVER -> USER A

```
SIP/2.0 486 Busy here
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK80d32c76c245e611af8c005ea72b6f73;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=3059277927
To: <sip:4060@192.168.43.23>;tag=725a0d84
Call-ID: 80D32C76-C245-E611-AF8B-005EA72B6F73@192.168.43.81
CSeq: 3 INVITE
Contact: "4060" <sip:4060@192.168.43.23>
User-Agent: Jitsi2.8.5426Windows 8
Content-Length: 0
```

ACK USER A -> SERVER

```
ACK sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK80d32c76c245e611af8c005ea72b6f73;rport
From: <sip:8060@192.168.43.23>;tag=3059277927
To: <sip:4060@192.168.43.23>;tag=725a0d84
Call-ID: 80D32C76-C245-E611-AF8B-005EA72B6F73@192.168.43.81
CSeq: 3 ACK
Content-Length: 0
```

ACK SERVER -> USER B

```
ACK sip:4060@192.168.43.23 SIP/2.0
Via: SIP/2.0/UDP 192.168.43.23:5060;branch=z9hG4bK2d4790
Via: SIP/2.0/UDP 192.168.43.81:5000;branch=z9hG4bK80d32c76c245e611af8c005ea72b6f73;rport;recieved=192.168.43.81
From: <sip:8060@192.168.43.23>;tag=3059277927
To: <sip:4060@192.168.43.23>;tag=725a0d84
Call-ID: 80D32C76-C245-E611-AF8B-005EA72B6F73@192.168.43.81
CSeq: 3 ACK
Contact:
Content-Length: 0
```

CHAPTER 4

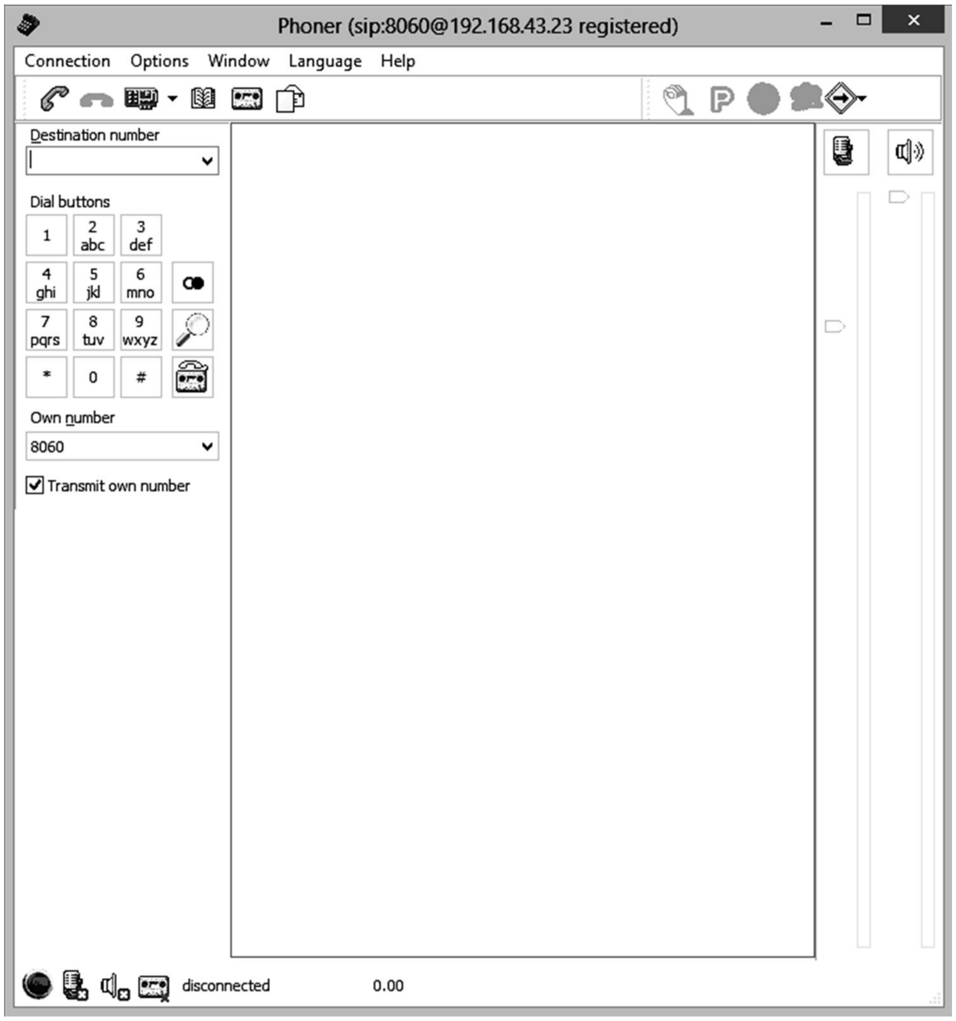
USAGE AND DISCUSSION

This chapter contains screenshots of the working of the software and basic guidelines of its usage. This is a step by step method that answers the question of how to actually use the software. Please note that this software is thoroughly tested under various conditions and found to be reliable for the user.

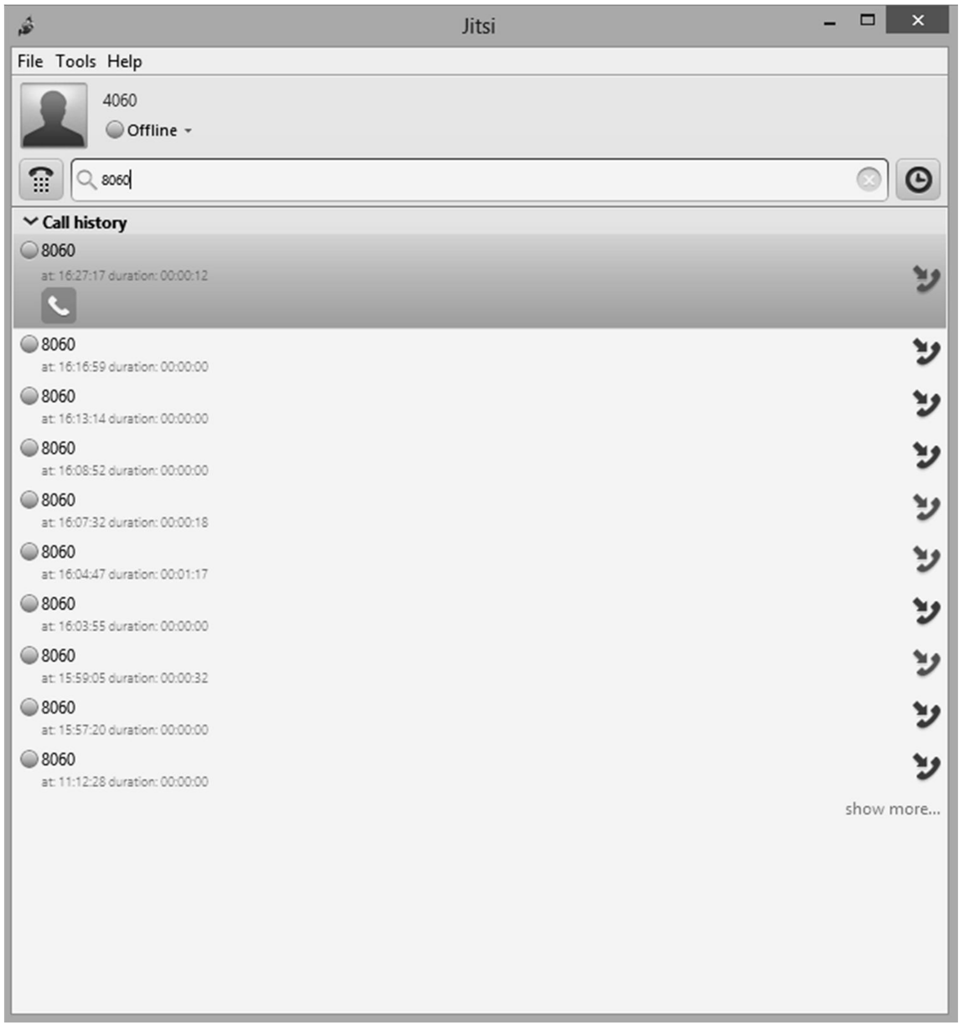
4.1 STEPS WITH SCREENSHOTS

- **DOWNLOAD PHONER (SIP COMMUNICATOR) ON CLIENT**

  Phoner is a freeware application for Microsoft Windows, from Win2k up to Win7. This application enables voice connections to landline, cellular network and VoIP. Phoner can be used as softphone on stationary and mobile computer devices attached to analogue or digital (ISDN) subscriber lines and as SIP client.

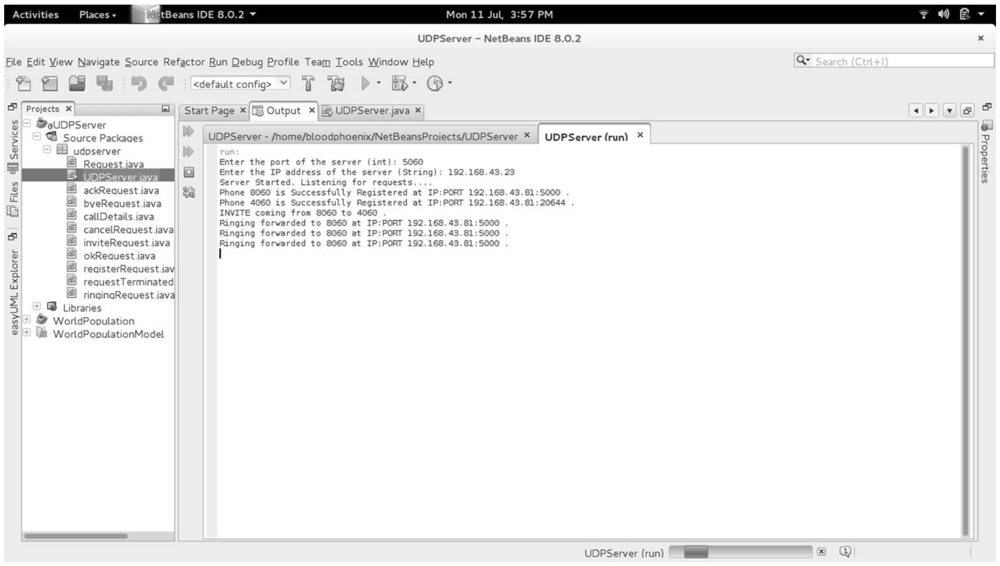


Jitsi is also a popular SIP Communicator.



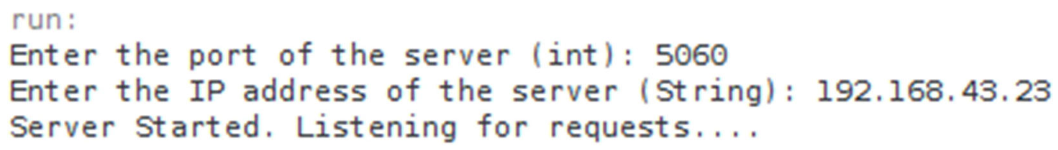
8060
at: 16:27:17 duration: 00:00:12
8060
at: 16:16:59 duration: 00:00:00
8060
at: 16:13:14 duration: 00:00:00
8060
at: 16:08:52 duration: 00:00:00
8060
at: 16:07:32 duration: 00:00:18
8060
at: 16:04:47 duration: 00:01:17
8060
at: 16:03:55 duration: 00:00:00
8060
at: 15:59:05 duration: 00:00:32
8060
at: 15:57:20 duration: 00:00:00
8060
at: 11:12:28 duration: 00:00:00

- **SET UP NETBEANS 8.0.2 or later with JDK 1.6 or later**



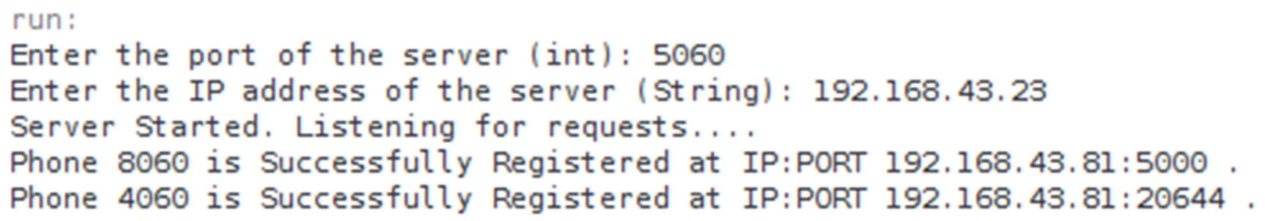
```
run:
Enter the port of the server (int): 5060
Enter the IP address of the server (String): 192.168.43.23
Server Started. Listening for requests....
Phone 8060 is Successfully Registered at IP:PORT 192.168.43.81:5000 .
Phone 4060 is Successfully Registered at IP:PORT 192.168.43.81:20644 .
INVITE coming from 8060 to 4060 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
```

- **<u>RUN THE CODE ON A SERVER SPECIFYING THE PORT AND IP</u>**



```
run:
Enter the port of the server (int): 5060
Enter the IP address of the server (String): 192.168.43.23
Server Started. Listening for requests....
```

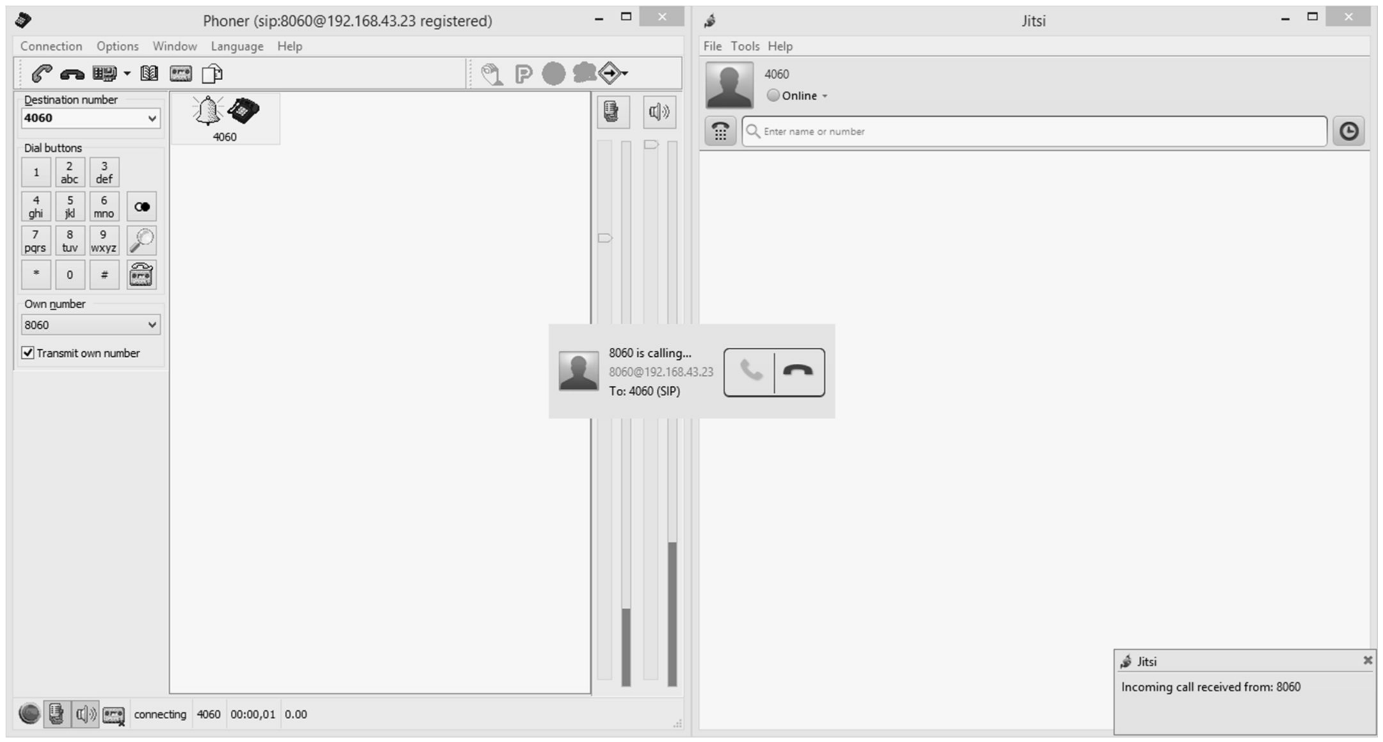
Clients will register….



```
run:
Enter the port of the server (int): 5060
Enter the IP address of the server (String): 192.168.43.23
Server Started. Listening for requests....
Phone 8060 is Successfully Registered at IP:PORT 192.168.43.81:5000 .
Phone 4060 is Successfully Registered at IP:PORT 192.168.43.81:20644 .
```

- **<u>RUN PHONER ON CLIENT COMPUTER AND SET SERVER IP AS REGISTRAR</u>**

- **<u>NOW DIAL THE NUMBER OF ANY REGISTERED USER AND CALL</u>**



Phoner (sip:8060@192.168.43.23 registered)

Destination number: 4060

Own number: 8060

☑ Transmit own number

connecting 4060 00:00,01 0.00

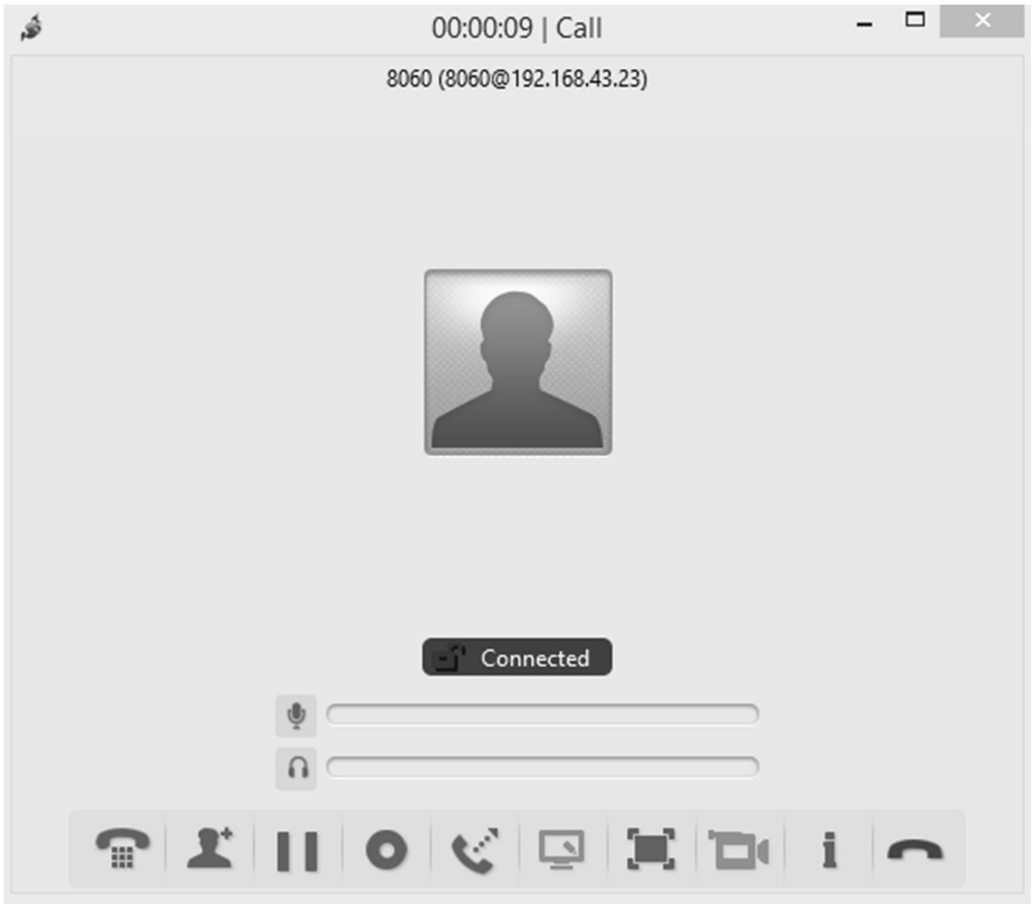
Jitsi

4060 Online

8060 is calling...
8060@192.168.43.23
To: 4060 (SIP)

Incoming call received from: 8060

- **<u>RTP SESSION IS ETABLISHED WHEN 'CONNECTED' IS DISPLAYED</u>**

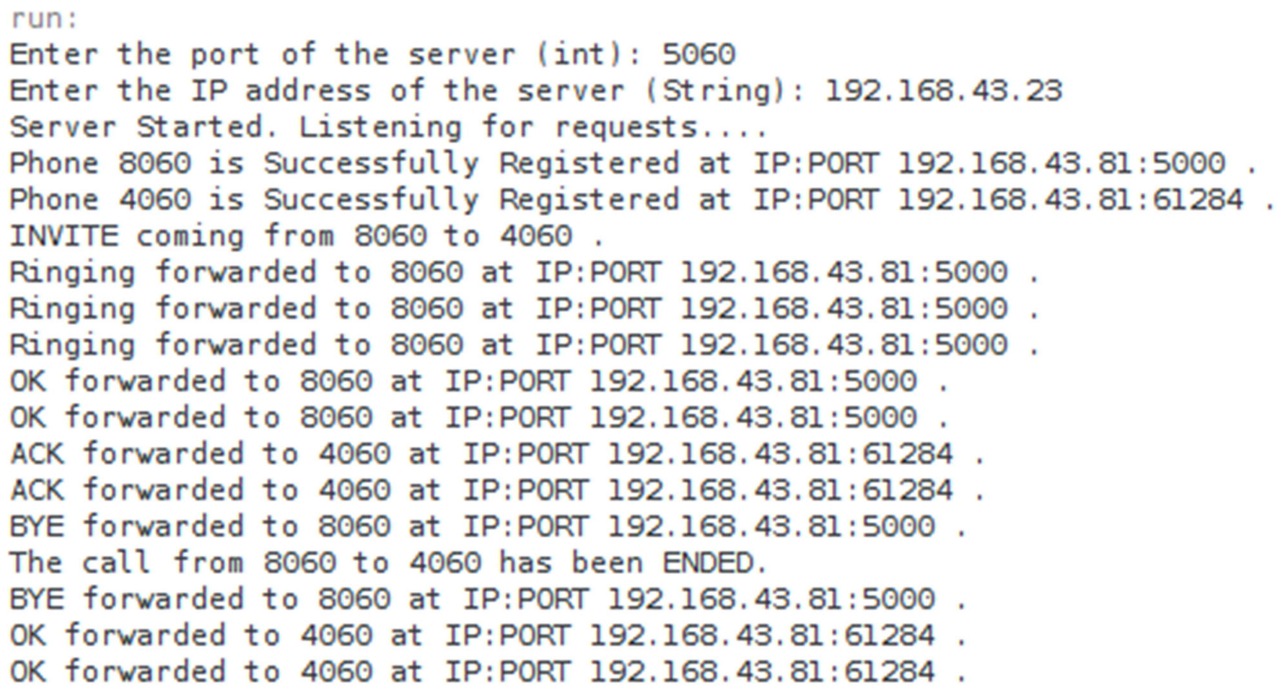


- **<u>CLICK THE RED 'END CALL' WHEN FINISHED TALKING</u>**

- **<u>LOG SCREEN DURING 'BUSY HERE'</u>**

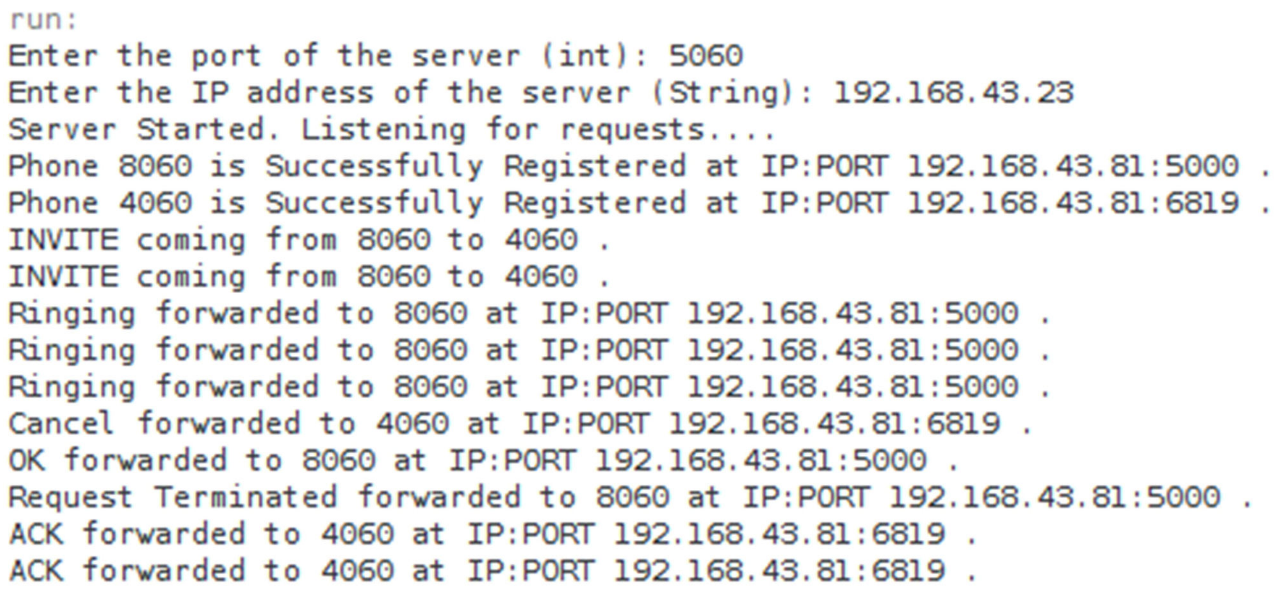
```
run:
Enter the port of the server (int): 5060
Enter the IP address of the server (String): 192.168.43.23
Server Started. Listening for requests....
Phone 8060 is Successfully Registered at IP:PORT 192.168.43.81:5000 .
Phone 4060 is Successfully Registered at IP:PORT 192.168.43.81:29034 .
INVITE coming from 8060 to 4060 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Busy Here forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Busy Here forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
ACK forwarded to 4060 at IP:PORT 192.168.43.81:29034 .
ACK forwarded to 4060 at IP:PORT 192.168.43.81:29034 .
ACK forwarded to 4060 at IP:PORT 192.168.43.81:29034 .
```

- **LOG SCREEN DURING NORMAL FLOW**



```
run:
Enter the port of the server (int): 5060
Enter the IP address of the server (String): 192.168.43.23
Server Started. Listening for requests....
Phone 8060 is Successfully Registered at IP:PORT 192.168.43.81:5000 .
Phone 4060 is Successfully Registered at IP:PORT 192.168.43.81:61284 .
INVITE coming from 8060 to 4060 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
OK forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
OK forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
ACK forwarded to 4060 at IP:PORT 192.168.43.81:61284 .
ACK forwarded to 4060 at IP:PORT 192.168.43.81:61284 .
BYE forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
The call from 8060 to 4060 has been ENDED.
BYE forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
OK forwarded to 4060 at IP:PORT 192.168.43.81:61284 .
OK forwarded to 4060 at IP:PORT 192.168.43.81:61284 .
```

- **LOG SCREEN DURING CANCELATION**



```
run:
Enter the port of the server (int): 5060
Enter the IP address of the server (String): 192.168.43.23
Server Started. Listening for requests....
Phone 8060 is Successfully Registered at IP:PORT 192.168.43.81:5000 .
Phone 4060 is Successfully Registered at IP:PORT 192.168.43.81:6819 .
INVITE coming from 8060 to 4060 .
INVITE coming from 8060 to 4060 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Ringing forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Cancel forwarded to 4060 at IP:PORT 192.168.43.81:6819 .
OK forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
Request Terminated forwarded to 8060 at IP:PORT 192.168.43.81:5000 .
ACK forwarded to 4060 at IP:PORT 192.168.43.81:6819 .
ACK forwarded to 4060 at IP:PORT 192.168.43.81:6819 .
```

CONCLUSION

The **VoIP Communication using SIP** provides easy communication among users using only an internet connection or a local area network if available. The technologies used in developing this project are easily implemented and have vast support and user base worldwide.

Whole project is maintained on **Github** under the name **UDP-SIP-Server** which can be forked and modified as per the need of the user.

Project Link : https://github.com/srajat/UDP-SIP-Server

Please visit **www.github.com/srajat** for more projects.

APPENDIX

CODE

MAIN CLASS

```
package udpserver;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File UDPServer.java
 */
import java.net.*;
import java.io.*;
import java.util.HashMap;
import java.util.StringTokenizer;

public class UDPServer
{

      private static final int ECHOMAX = 2048;    //Stores max length of recieved
      message                           in bytes
      private static HashMap<String,String> REGISTERED = new HashMap<String,String>();
      //HashMap of REGISTERED users
      private static HashMap<String,callDetails> CURRENTCALLS = new
      HashMap<String,callDetails>();      //HashMap of Current Calls going on


      public static void main(String[] args) throws IOException
      {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("Enter the port of the server (int): ");
        int servPort = Integer.parseInt(br.readLine());

        System.out.print("Enter the IP address of the server (String): ");
        String servIp = br.readLine();

        System.out.println("Server Started. Listening for requests....");

        //Create new packet to recieve into
        DatagramSocket socket = new DatagramSocket(servPort) ;
        DatagramPacket packet = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);


        for (;;)    //Loops infinitely
        {
            socket.receive(packet);     //Blocks till packet recieved

            InetAddress clientAddress = packet.getAddress();    //client's IP
            int clientPort = packet.getPort();      //client's Port

            byte[] arr = new byte[packet.getLength()];
        System.arraycopy(packet.getData(), packet.getOffset(), arr, 0, arr.length);
        String requestMsg = new String(arr);    //recieved Message


        StringTokenizer st = new StringTokenizer(requestMsg,"\r\n");
        String line1 = st.nextToken();  //stores first line of recieved message
        String typeOfMsg = line1.substring(0, line1.indexOf(" "));

        if("REGISTER".equals(typeOfMsg))    //If message is of type Register
        {
            registerRequest r = new registerRequest();  //new register object
            String feildName = "";
            String nextLine = "";
            while(st.hasMoreTokens())
            {
                nextLine = st.nextToken();
                feildName = nextLine.substring(0,nextLine.indexOf(":"));
                if("Via".equals(feildName))
             r.via = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("From".equals(feildName))
            r.from = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("To".equals(feildName))
              r.to = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Call-ID".equals(feildName))
          r.callId = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("CSeq".equals(feildName))
            r.cSeq = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Contact".equals(feildName))
         r.contact = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Allow".equals(feildName))
           r.allow = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Max-Forwards".equals(feildName))
     r.maxForwards = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Allow-Events".equals(feildName))
     r.allowEvents = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("User-Agent".equals(feildName))
       r.userAgent = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Supported".equals(feildName))
       r.supported = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Expires".equals(feildName))
         r.expires = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Content-Length".equals(feildName))
   r.contentLength = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            }

            //add to registered the user toRegister
            String sipUri = r.contact.substring(r.contact.indexOf(":")+1,
            r.contact.indexOf(";"));
            String number,ipPort;

            if(!sipUri.contains(">"))
            {
                //this means there is no > (for Jitsi)
                number = sipUri.substring(0, sipUri.indexOf("@"));
                ipPort = sipUri.substring(sipUri.indexOf("@")+1, sipUri.length());
            }
            else
            {
               //this means there is a > (for Phoner)
               number = sipUri.substring(0, sipUri.indexOf("@"));
               ipPort = sipUri.substring(sipUri.indexOf("@")+1, sipUri.length()-1);
            }
        //check if already registered
        boolean isRegistered = REGISTERED.containsKey(number);
        int expires = Integer.parseInt(r.expires.trim());

        if(!isRegistered && expires > 0)    //register
        {
          System.out.println("Phone "+number+" is Successfully Registered at
          IP:PORT "+ipPort+" .");
          REGISTERED.put(number, ipPort);
        }

        else if(isRegistered && expires == 0)   //unregister
        {
          System.out.println("Phone "+number+" is Successfully UNREGISTERED.");
          REGISTERED.remove(number);
        }

        //send OK response to caller
        byte[] send = r.OK_200().getBytes();
        DatagramPacket p = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
        p.setAddress(clientAddress);
        p.setPort(clientPort);
        p.setData(send);
        socket.send(p);

    }

    if("INVITE".equals(typeOfMsg))  //If message is of type Invite
    {
        inviteRequest r = new inviteRequest();  //new Invite object
        String feildName = "";
        String nextLine = "";
        while(st.hasMoreTokens())
        {
            nextLine = st.nextToken();
            feildName = nextLine.substring(0,nextLine.indexOf(":"));

            if("Via".equals(feildName))
      r.via.add(nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length()));
            else if("From".equals(feildName))
        r.from = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("To".equals(feildName))
          r.to = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("Call-ID".equals(feildName))
      r.callId = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("CSeq".equals(feildName))
        r.cSeq = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("Contact".equals(feildName))
     r.contact = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("Content-Type".equals(feildName))
r.contentType = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("Allow".equals(feildName)
    r.allow = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("Max-Forwards".equals(feildName))
r.maxForwards = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("Supported".equals(feildName))
  r.supported = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("P-Early-Media".equals(feildName))
r.pEarlyMedia = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            else if("User-Agent".equals(feildName))
  r.userAgent = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
```

```
                    else if("P-Preferred-Identity".equals(feildName))
 r.prefferedIdentity = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                    else if("Content-Length".equals(feildName))
                    {
                        r.contentLength = nextLine.substring(nextLine.indexOf("
                        ")+1,nextLine.length());
                        break;
                    }
                }
                while(st.hasMoreTokens())   //Fill data into SDP part of msg
                {
                    nextLine = st.nextToken();
                    feildName = nextLine.substring(0,nextLine.indexOf("="));
                    if("v".equals(feildName))
                   r.v = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                    else if("o".equals(feildName))
                   r.o = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                    else if("s".equals(feildName))
                   r.s = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                    else if("c".equals(feildName))
                   r.c = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                    else if("t".equals(feildName))
                   r.t = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                    else if("m".equals(feildName))
                   r.m = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());

                    else if("a".equals(feildName))
                r.a.add(nextLine.substring(nextLine.indexOf("=")+1,nextLine.length()));

                }

                //find who is calling
                String callerNumber = r.from.substring(r.from.indexOf(":")+1,
                  r.from.indexOf("@"));

                //find who to send this invite
                String calleeNumber =
                line1.substring(line1.indexOf(":")+1,line1.indexOf("@"));
                String calleeIp = extractIpOrPort(REGISTERED.get(calleeNumber),0);
                String calleePort = extractIpOrPort(REGISTERED.get(calleeNumber),1);

                System.out.println("INVITE coming from "+callerNumber+" to " +
                calleeNumber + " .");

                //send trying back to caller
                byte[] send = r.TRYING_100().getBytes();
                DatagramPacket p = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
                p.setAddress(clientAddress);
                p.setPort(clientPort);
                p.setData(send);
                socket.send(p);

                //Now forward this packet to callee
                byte[] send1 = r.forwardInvite(line1, servIp, servPort).getBytes();
                DatagramPacket p1 = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
                p1.setAddress(InetAddress.getByName(calleeIp));
                p1.setPort(Integer.parseInt(calleePort.trim()));
                p1.setData(send1);
                socket.send(p1);

                //add details to CURRENTCALLS
                callDetails cd = new callDetails();
```

```
            cd.caller = callerNumber;
            cd.called = calleeNumber;

            String callId = r.callId.substring(0, r.callId.indexOf("@"));
            CURRENTCALLS.put(callId, cd);

        }

        if("SIP/2.0 180 Ringing".equals(line1))     //If message is of type Ringing
        {
            ringingRequest r = new ringingRequest();    //new Ringing Object
            String feildName = "";
            String nextLine = "";
            while(st.hasMoreTokens())
            {
                nextLine = st.nextToken();
                feildName = nextLine.substring(0,nextLine.indexOf(":"));
                if("Via".equals(feildName))
          r.via.add(nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length()));
                else if("From".equals(feildName))
            r.from = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("To".equals(feildName))
             r.to = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Call-ID".equals(feildName))
          r.callId = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("CSeq".equals(feildName))
           r.cSeq = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Contact".equals(feildName))
         r.contact = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Allow".equals(feildName))
           r.allow = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Max-Forwards".equals(feildName))
    r.maxForwards = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());

                else if("User-Agent".equals(feildName))
      r.userAgent = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Supported".equals(feildName))
      r.supported = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());

                else if("Content-Length".equals(feildName))
  r.contentLength = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            }

            //extract details of who to forward
            String fwdNumber = r.from.substring(r.from.indexOf(":")+1,
            r.from.indexOf("@"));
            String fwdIp = extractIpOrPort(REGISTERED.get(fwdNumber),0);
            String fwdPort = extractIpOrPort(REGISTERED.get(fwdNumber),1);

            System.out.println("Ringing forwarded to "+fwdNumber+" at IP:PORT
            "+fwdIp+":"+fwdPort+" .");

            //forward ringing
            byte[] send = r.forwardRinging(servIp).getBytes();
            DatagramPacket p = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
            p.setAddress(InetAddress.getByName(fwdIp));
            p.setPort(Integer.parseInt(fwdPort.trim()));
            p.setData(send);
            socket.send(p);
        }
        if("SIP/2.0 200 OK".equals(line1))      //If message is of type OK
        {
            //recieved OK from callee
            okRequest r = new okRequest();
            String feildName = "";
            String nextLine = "";
            while(st.hasMoreTokens())
            {
                nextLine = st.nextToken();
                feildName = nextLine.substring(0,nextLine.indexOf(":"));

                if("Via".equals(feildName))
         r.via.add(nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length()));
                else if("From".equals(feildName))
            r.from = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("To".equals(feildName))
          r.to = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Call-ID".equals(feildName))
          r.callId = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("CSeq".equals(feildName))
            r.cSeq = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Contact".equals(feildName))
         r.contact = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Content-Type".equals(feildName))
     r.contentType = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Allow".equals(feildName))
      r.allow = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Max-Forwards".equals(feildName))
     r.maxForwards = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Supported".equals(feildName))
       r.supported = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("P-Early-Media".equals(feildName))
     r.pEarlyMedia = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("User-Agent".equals(feildName))
       r.userAgent = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("P-Preferred-Identity".equals(feildName))
r.prefferedIdentity = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Content-Length".equals(feildName))
                {
   r.contentLength = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                    break;
                }
            }

              while(st.hasMoreTokens())
            {
                nextLine = st.nextToken();
                feildName = nextLine.substring(0,nextLine.indexOf("="));
                if("v".equals(feildName))
               r.v = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                else if("o".equals(feildName))
               r.o = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                else if("s".equals(feildName))
               r.s = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                else if("c".equals(feildName))
               r.c = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                else if("t".equals(feildName))
               r.t = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                else if("m".equals(feildName))
               r.m = nextLine.substring(nextLine.indexOf("=")+1,nextLine.length());
                else if("a".equals(feildName))
        r.a.add(nextLine.substring(nextLine.indexOf("=")+1,nextLine.length()));
            }

            //Find out fwd Ok to whom
            String fwdNumber = r.from.substring(r.from.indexOf(":")+1,
            r.from.indexOf("@"));
            String fwdIp = extractIpOrPort(REGISTERED.get(fwdNumber),0);
            String fwdPort = extractIpOrPort(REGISTERED.get(fwdNumber),1);

            System.out.println("OK forwarded to "+fwdNumber+" at IP:PORT
            "+fwdIp+":"+fwdPort+" .");

            //forward OK
            byte[] send = r.forwardOk(servIp).getBytes();
            DatagramPacket p = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
            p.setAddress(InetAddress.getByName(fwdIp));
            p.setPort(Integer.parseInt(fwdPort.trim()));
            p.setData(send);
            socket.send(p);

        }

        if("ACK".equals(typeOfMsg))     //If message is of type ACK
        {

            ackRequest r = new ackRequest();    //New ACK object
            String feildName = "";
            String nextLine = "";
            while(st.hasMoreTokens())
            {
                nextLine = st.nextToken();
                feildName = nextLine.substring(0,nextLine.indexOf(":"));
                if("Via".equals(feildName))
         r.via.add(nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length()));
                else if("From".equals(feildName))
            r.from = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("To".equals(feildName))
              r.to = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Call-ID".equals(feildName))
          r.callId = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("CSeq".equals(feildName))
            r.cSeq = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Contact".equals(feildName))
         r.contact = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Allow".equals(feildName))
          r.allow = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Max-Forwards".equals(feildName))
     r.maxForwards = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("User-Agent".equals(feildName))
       r.userAgent = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Supported".equals(feildName))
       r.supported = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Content-Length".equals(feildName))
   r.contentLength = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            }

            //whom to send this ack
            String fwdNumber =
            r.to.substring(r.to.indexOf(":")+1,r.to.indexOf("@"));
            String fwdIp = extractIpOrPort(REGISTERED.get(fwdNumber),0);
            String fwdPort = extractIpOrPort(REGISTERED.get(fwdNumber),1);
            System.out.println("ACK forwarded to "+fwdNumber+" at IP:PORT
            "+fwdIp+":"+fwdPort+" .");

            //forward ack
            byte[] send = r.forwardAck(line1, servIp, servPort).getBytes();
            DatagramPacket p = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
            p.setAddress(InetAddress.getByName(fwdIp));
            p.setPort(Integer.parseInt(fwdPort.trim()));
            p.setData(send);
            socket.send(p);
        }

        if("BYE".equals(typeOfMsg))     //If message is of type Bye
        {
            byeRequest r = new byeRequest();    //new Bye object
            String feildName = "";
            String nextLine = "";
            while(st.hasMoreTokens())
            {
                nextLine = st.nextToken();
                feildName = nextLine.substring(0,nextLine.indexOf(":"));
                if("Via".equals(feildName))
         r.via.add(nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length()));
                else if("From".equals(feildName))
            r.from = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("To".equals(feildName))
            r.to = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Call-ID".equals(feildName))
          r.callId = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("CSeq".equals(feildName))
     r.cSeq = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Contact".equals(feildName))
         r.contact = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Allow".equals(feildName))
           r.allow = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Max-Forwards".equals(feildName))
   r.maxForwards = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("User-Agent".equals(feildName))
     r.userAgent = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Supported".equals(feildName))
     r.supported = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Content-Length".equals(feildName))
 r.contentLength = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            }

            //whom to send this bye
            String fwdNumber =
            r.to.substring(r.to.indexOf(":")+1,r.to.indexOf("@"));
            String fwdIp = extractIpOrPort(REGISTERED.get(fwdNumber),0);
            String fwdPort = extractIpOrPort(REGISTERED.get(fwdNumber),1);

            System.out.println("BYE forwarded to "+fwdNumber+" at IP:PORT
            "+fwdIp+":"+fwdPort+" .");

            //Forward BYE
            byte[] send = r.forwardBye(line1, servIp, servPort).getBytes();
            DatagramPacket p = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
            p.setAddress(InetAddress.getByName(fwdIp));
            p.setPort(Integer.parseInt(fwdPort.trim()));
            p.setData(send);
            socket.send(p);
            //remove this call from CURRENTCALLS
            String callId = r.callId.substring(0, r.callId.indexOf("@"));

            if(CURRENTCALLS.containsKey(callId))
            {
                System.out.println("The call from 
              "+CURRENTCALLS.get(callId).caller+" to 
              "+CURRENTCALLS.get(callId).called+" has been ENDED.");

                CURRENTCALLS.remove(callId);
            }
        }

        if("CANCEL".equals(typeOfMsg))      //If message is of type Cancel
        {
            cancelRequest r = new cancelRequest();  //new Cancel object
            String feildName = "";
            String nextLine = "";
            while(st.hasMoreTokens())
            {
                nextLine = st.nextToken();
                feildName = nextLine.substring(0,nextLine.indexOf(":"));
                if("Via".equals(feildName))
          r.via.add(nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length()));
                else if("From".equals(feildName))
           r.from = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("To".equals(feildName))
           r.to = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Call-ID".equals(feildName))
         r.callId = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("CSeq".equals(feildName))
           r.cSeq = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Contact".equals(feildName))
         r.contact = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Allow".equals(feildName))
           r.allow = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Max-Forwards".equals(feildName))
     r.maxForwards = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());

                else if("User-Agent".equals(feildName))
      r.userAgent = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Supported".equals(feildName))
       r.supported = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());

                else if("Content-Length".equals(feildName))
   r.contentLength = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            }

            //Find forward cancel to whom
            String fwdNumber = r.to.substring(r.from.indexOf(":")+1,
            r.from.indexOf("@"));
            String fwdIp = extractIpOrPort(REGISTERED.get(fwdNumber),0);
            String fwdPort = extractIpOrPort(REGISTERED.get(fwdNumber),1);

            System.out.println("Cancel forwarded to "+fwdNumber+" at IP:PORT 
            "+fwdIp+":"+fwdPort+" .");

            //forward cancel
            byte[] send = r.forwardCancel(line1,servIp,servPort).getBytes();
            DatagramPacket p = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
            p.setAddress(InetAddress.getByName(fwdIp));
            p.setPort(Integer.parseInt(fwdPort.trim()));
            p.setData(send);
            socket.send(p);

            //remove callDetails
            String callId = r.callId.substring(0,r.callId.indexOf("@"));
            if(CURRENTCALLS.containsKey(callId))
                CURRENTCALLS.remove(callId);
        }

        if(line1.contains("487 Request"))   //If message is of type 487
        {
            requestTerminatedRequest r = new requestTerminatedRequest();
            String feildName = "";
            String nextLine = "";
            while(st.hasMoreTokens())
            {
                nextLine = st.nextToken();
                feildName = nextLine.substring(0,nextLine.indexOf(":"));
                if("Via".equals(feildName))
         r.via.add(nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length()));
                else if("From".equals(feildName))
          r.from = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("To".equals(feildName))
             r.to = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Call-ID".equals(feildName))
          r.callId = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("CSeq".equals(feildName))
            r.cSeq = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Contact".equals(feildName))
         r.contact = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Allow".equals(feildName))
          r.allow = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Max-Forwards".equals(feildName))
     r.maxForwards = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("User-Agent".equals(feildName))
       r.userAgent = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Supported".equals(feildName))
       r.supported = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Content-Length".equals(feildName))
   r.contentLength = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            }

            //forward to whom
            String fwdNumber = r.from.substring(r.from.indexOf(":")+1,
            r.from.indexOf("@"));
            String fwdIp = extractIpOrPort(REGISTERED.get(fwdNumber),0);
            String fwdPort = extractIpOrPort(REGISTERED.get(fwdNumber),1);

            System.out.println("Request Terminated forwarded to "+fwdNumber+" at
            IP:PORT "+fwdIp+":"+fwdPort+" .");

            //forward 487
            byte[] send =
            r.forwardrequestTerminated(line1,servIp,servPort).getBytes();
            DatagramPacket p = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
            p.setAddress(InetAddress.getByName(fwdIp));
            p.setPort(Integer.parseInt(fwdPort.trim()));
            p.setData(send);
            socket.send(p);
        }
        if(line1.contains("486 Busy"))  //If message is of type Busy
        {
            requestTerminatedRequest r = new requestTerminatedRequest();
            String feildName = "";
            String nextLine = "";
            while(st.hasMoreTokens())
            {
                nextLine = st.nextToken();
                feildName = nextLine.substring(0,nextLine.indexOf(":"));
                if("Via".equals(feildName))
          r.via.add(nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length()));
                else if("From".equals(feildName))
            r.from = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("To".equals(feildName))
           r.to = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Call-ID".equals(feildName))
         r.callId = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("CSeq".equals(feildName))
        r.cSeq = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Contact".equals(feildName))
        r.contact = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Allow".equals(feildName))
          r.allow = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Max-Forwards".equals(feildName))
    r.maxForwards = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("User-Agent".equals(feildName))
       r.userAgent = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Supported".equals(feildName))
       r.supported = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
                else if("Content-Length".equals(feildName))
   r.contentLength = nextLine.substring(nextLine.indexOf(" ")+1,nextLine.length());
            }

            //forward to whom
            String fwdNumber = r.from.substring(r.from.indexOf(":")+1,
            r.from.indexOf("@"));
            String fwdIp = extractIpOrPort(REGISTERED.get(fwdNumber),0);
            String fwdPort = extractIpOrPort(REGISTERED.get(fwdNumber),1);

            System.out.println("Busy Here forwarded to "+fwdNumber+" at IP:PORT
            "+fwdIp+":"+fwdPort+" .");

            //forward 486
            byte[] send =
            r.forwardrequestTerminated(line1,servIp,servPort).getBytes();
            DatagramPacket p = new DatagramPacket(new byte[ECHOMAX], ECHOMAX);
            p.setAddress(InetAddress.getByName(fwdIp));
            p.setPort(Integer.parseInt(fwdPort.trim()));
            p.setData(send);
            socket.send(p);

            //remove from CURRENTCALLS
            String callId = r.callId.substring(0,r.callId.indexOf("@"));
            if(CURRENTCALLS.containsKey(callId))
                CURRENTCALLS.remove(callId);
        }


        packet.setLength(ECHOMAX);

    }
}
```

```
    private static String extractIpOrPort(String s,int choice) //Returns IP or PORT
                                                              from SIP URI
    {
        if(choice == 0) //returns IP if choice == 0
        {
            return s.substring(0, s.indexOf(":"));
        }
        else //else returns PORT
            return s.substring(s.indexOf(":")+1);
    }
}
```

ACK REQUEST CLASS

```
package udpserver;
/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File ackRequest.java
 */
import java.util.ArrayList;


public class ackRequest extends Request
{
    ArrayList<String> via;  //List of via headers

    ackRequest()    //constructor
    {
        super();
        via = new ArrayList<>();
    }

    public String forwardAck(String line1,String servIp,int servPort)
    {
        String fwd_res = line1 + "\r\n";

        //add recieved to topmost via feild
        String upperViaFeild = via.get(0);
        String recieved = upperViaFeild.substring(upperViaFeild.indexOf(" ")+1,
        upperViaFeild.indexOf(":"));
        upperViaFeild = upperViaFeild + ";recieved=" + recieved;
        via.set(0, upperViaFeild);

        //add this server's Via tag
        via.add(0,"SIP/2.0/UDP "+servIp+":"+servPort+";branch=z9hG4bK2d4790");

        //add via feilds
        for(int in=0;in<via.size();in++)
            fwd_res = fwd_res + "Via: " + via.get(in) + "\r\n";

        fwd_res = fwd_res + "From: " + from + "\r\n";
        fwd_res = fwd_res + "To: " + to + "\r\n";
        fwd_res = fwd_res + "Call-ID: " + callId + "\r\n";
        fwd_res = fwd_res + "CSeq: " + cSeq + "\r\n";
        fwd_res = fwd_res + "Contact: " + contact + "\r\n";
        fwd_res = fwd_res + "Content-Length: 0\r\n\r\n";

        return fwd_res;
    }
}
```

BYE REQUEST CLASS

```
package udpserver;

import java.util.ArrayList;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File byeRequest.java
 */
public class byeRequest extends Request
{
    ArrayList<String> via;
    byeRequest()
    {
        super();
        via = new ArrayList<>();
    }

    public String forwardBye(String line1,String servIp,int servPort)
    {
        String fwd_res = line1 + "\r\n";

        //add recieved to topmost via feild
        String upperViaFeild = via.get(0);
        String recieved = upperViaFeild.substring(upperViaFeild.indexOf(" ")+1,
        upperViaFeild.indexOf(":"));
        upperViaFeild = upperViaFeild + ";recieved=" + recieved;
        via.set(0, upperViaFeild);

        //add this server's Via tag
        via.add(0,"SIP/2.0/UDP "+servIp+":"+servPort+";branch=z9hG4bK2d4790");

        for(int in=0;in<via.size();in++)    //add all via headers
            fwd_res = fwd_res + "Via: " + via.get(in) + "\r\n";


        fwd_res = fwd_res + "From: " + from + "\r\n";
        fwd_res = fwd_res + "To: " + to + "\r\n";
        fwd_res = fwd_res+ "Call-ID: " + callId + "\r\n";
        fwd_res = fwd_res + "CSeq: " + cSeq + "\r\n";

        String modifiedContact = contact.substring(0,contact.indexOf("@")+1)+
        servIp+">";
        fwd_res = fwd_res + "Contact: " + modifiedContact + "\r\n";

        fwd_res = fwd_res + "Max-Forwards: " + (Integer.parseInt(maxForwards.trim())-1)
        + "\r\n";
        fwd_res = fwd_res + "User-Agent: " + userAgent + "\r\n";
        fwd_res = fwd_res + "Content-Length: " + contentLength + "\r\n\r\n";
        return fwd_res;
    }
}
```

CALL DETAILS CLASS

```
package udpserver;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File callDetails.java
 */
public class callDetails
{
    String caller;  //Stores caller Phone number
    String called;  //Stores the Called Phone number


    callDetails()   //constructor
    {
        caller = "";
        called = "";

    }
}
```

CANCEL REQUEST CLASS

```
package udpserver;

import java.util.ArrayList;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File cancelRequest.java
 */
public class cancelRequest extends Request
{
    ArrayList<String> via;
    cancelRequest()
    {
        super();
        via = new ArrayList<>();
    }
    public String forwardCancel(String line1,String servIp,int servPort)
    {
        String fwd_res = line1 + "\r\n";

        //add recieved to topmost via feild
        String upperViaFeild = via.get(0);
        String recieved = upperViaFeild.substring(upperViaFeild.indexOf(" ")+1,
        upperViaFeild.indexOf(":"));
        upperViaFeild = upperViaFeild + ";recieved=" + recieved;
        via.set(0, upperViaFeild);

        //add this server's Via tag
        via.add(0,"SIP/2.0/UDP "+servIp+":"+servPort+";branch=z9hG4bK2d4790");

        //add via feilds
        for(int in=0;in<via.size();in++)
            fwd_res = fwd_res + "Via: " + via.get(in) + "\r\n";

        fwd_res = fwd_res + "From: " + from + "\r\n";
        fwd_res = fwd_res + "To: " + to + "\r\n";
        fwd_res = fwd_res+ "Call-ID: " + callId + "\r\n";
        fwd_res = fwd_res + "CSeq: " + cSeq + "\r\n";

        fwd_res = fwd_res + "Max-Forwards: " + (Integer.parseInt(maxForwards.trim())-1)
        + "\r\n";
        fwd_res = fwd_res + "User-Agent: " + userAgent + "\r\n";

        fwd_res = fwd_res + "Content-Length: 0" + "\r\n\r\n";

        return fwd_res;
    }
}
```

INVITE REQUEST CLASS

```
package udpserver;

import java.util.ArrayList;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File inviteRequest.java
 */
public class inviteRequest extends Request
{
    String contentType,pEarlyMedia,prefferedIdentity;
    String v,o,s,c,t,m;
    ArrayList<String> a,via;

    public inviteRequest()
    {
        super();
        this.contentType = "";
        this.pEarlyMedia = "";
        this.prefferedIdentity = "";

        this.v = "";
        this.c = "";
        this.m = "";
        this.o = "";
        this.s = "";
        this.t = "";

        a = new ArrayList<>();
        via = new ArrayList<>();
    }
    public String TRYING_100()      //method to generate trying response
    {
        String trying_res = "SIP/2.0 100 TRYING\r\n";

        //add recieved to topmost via feild
        String upperViaFeild = via.get(0);
        String recieved = upperViaFeild.substring(upperViaFeild.indexOf(" ")+1,
        upperViaFeild.indexOf(":"));
        upperViaFeild = upperViaFeild + ";recieved=" + recieved;
        via.set(0, upperViaFeild);

        for(int in=0;in<via.size();in++)
            trying_res = trying_res + "Via: " + via.get(in) + "\r\n";

        trying_res = trying_res + "From: " + from + "\r\n";
        trying_res = trying_res + "To: " + to + "\r\n";
        trying_res = trying_res + "Call-ID: " + callId + "\r\n";
        trying_res = trying_res + "CSeq: " + cSeq + "\r\n";
        trying_res = trying_res + "Allow: " + allow + "\r\n";
        trying_res = trying_res + "User-Agent: " + userAgent + "\r\n";
        trying_res = trying_res + "Supported: " + supported + "\r\n";
        trying_res = trying_res + "Content-Length: 0" + "\r\n";

        trying_res = trying_res + "\r\n";
        return trying_res;
    }

    public String forwardInvite(String line1,String servIp,int servPort)    //to
                                                              generate fwd response
    {
        String fwd_res = line1 + "\r\n";

        //add this server's Via tag
        via.add(0,"SIP/2.0/UDP "+servIp+":"+servPort+";branch=z9hG4bK2d4790");

        for(int in=0;in<via.size();in++)
            fwd_res = fwd_res + "Via: " + via.get(in) + "\r\n";

        fwd_res = fwd_res + "From: " + from + "\r\n";
        fwd_res = fwd_res + "To: " + to + "\r\n";
        fwd_res = fwd_res+ "Call-ID: " + callId + "\r\n";
        fwd_res = fwd_res + "CSeq: " + cSeq + "\r\n";

        String modifiedContact = contact.substring(0,contact.indexOf("@")+1)+
        servIp+">";
        fwd_res = fwd_res + "Contact: " + modifiedContact + "\r\n";

        fwd_res = fwd_res + "Content-Type: " + contentType + "\r\n";
        fwd_res = fwd_res + "Max-Forwards: " + (Integer.parseInt(maxForwards.trim())-1)
        + "\r\n";
        fwd_res = fwd_res + "User-Agent: " + userAgent + "\r\n";
        fwd_res = fwd_res + "Content-Length: " + contentLength + "\r\n\r\n";

        fwd_res = fwd_res + "v=" + v + "\r\n";
        fwd_res = fwd_res + "o=" + o + "\r\n";
        fwd_res = fwd_res + "s=" + s + "\r\n";
        fwd_res = fwd_res + "c=" + c + "\r\n";
        fwd_res = fwd_res + "t=" + t + "\r\n";
        fwd_res = fwd_res + "m=" + m + "\r\n";

        for(int in=0;in<a.size();in++)
            fwd_res = fwd_res + "a=" + a.get(in) + "\r\n";

        return fwd_res;
    }
}
```

OK REQUEST CLASS

```
package udpserver;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File okRequest.java
 */
public class okRequest extends inviteRequest
{
    okRequest()
    {
        super();
    }

    public String forwardOk(String servIp)
    {
        String fwd_res = "SIP/2.0 200 OK\r\n";

        if(via.get(0).contains(","))
        {
            String modifiedVia =
            via.get(0).substring(via.get(0).indexOf(",")+1,via.get(0).length());
            via.set(0,modifiedVia);
        }
        else
            via.remove(0);


        //add via feilds
        for(int in=0;in<via.size();in++)
            fwd_res = fwd_res + "Via: " + via.get(in) + "\r\n";

        fwd_res = fwd_res + "From: " + from + "\r\n";
        fwd_res = fwd_res + "To: " + to + "\r\n";
        fwd_res = fwd_res+ "Call-ID: " + callId + "\r\n";
        fwd_res = fwd_res + "CSeq: " + cSeq + "\r\n";

        if(cSeq.contains("CANCEL"))     //If this OK is in response to a CANCEL then
        {
            String modifiedContact = contact.substring(0,contact.indexOf("@")+1)+
            servIp+">";
            fwd_res = fwd_res + "Contact: " + modifiedContact + "\r\n";
        }

        fwd_res = fwd_res + "Content-Length: " + contentLength + "\r\n\r\n";

        if(!cSeq.contains("BYE") && !cSeq.contains("CANCEL"))   /*If this OK is not in
                                                                  response to a
                                                                  BYE or a CANCEL then
                                                                 */
        {
            fwd_res = fwd_res + "v=" + v + "\r\n";
            fwd_res = fwd_res + "o=" + o + "\r\n";
            fwd_res = fwd_res + "s=" + s + "\r\n";
            fwd_res = fwd_res + "c=" + c + "\r\n";
            fwd_res = fwd_res + "t=" + t + "\r\n";
            fwd_res = fwd_res + "m=" + m + "\r\n";
            for(int in=0;in<a.size();in++)
                fwd_res = fwd_res + "a=" + a.get(in) + "\r\n";
        }

        return fwd_res;
    }
}
```

REGISTER REQUEST CLASS

```
package udpserver;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File registerRequest.java
 */
public class registerRequest extends Request
{
    String allowEvents,expires;

    public registerRequest(String via,String from,String to,String callId,String
      cSeq,String contact,String allow
    ,String maxForwards,String allowEvents,String userAgent,String supported,String
      expires,
    String contentLength)
    {
        this.via = via;
        this.from = from;
        this.to = to;
        this.callId = callId;
        this.cSeq = cSeq;
        this.contact = contact;
        this.allow = allow;
        this.maxForwards = maxForwards;
        this.allowEvents = allowEvents;
        this.userAgent = userAgent;
        this.supported = supported;
        this.expires = expires;
        this.contentLength = contentLength;
    }
    public registerRequest()
    {
        super();
        this.allowEvents = "";
        this.expires = "";
    }
    public String OK_200()
    {
        String ok_res = "SIP/2.0 200 OK\r\n";

        ok_res = ok_res + "Via: " + via + "\r\n";
        ok_res = ok_res + "From: " + from + "\r\n";
        ok_res = ok_res + "To: " + to + "\r\n";
        ok_res = ok_res + "Call-ID: " + callId + "\r\n";
        ok_res = ok_res + "CSeq: " + cSeq + "\r\n";
        ok_res = ok_res + "Contact: " + contact + "\r\n";
        ok_res = ok_res + "Allow: " + allow + "\r\n";
        ok_res = ok_res + "Max-Forwards: " + maxForwards + "\r\n";
        ok_res = ok_res + "Allow-Events: " + allowEvents + "\r\n";
        ok_res = ok_res + "User-Agent: " + userAgent + "\r\n";
        ok_res = ok_res + "Supported: " + supported + "\r\n";
        ok_res = ok_res + "Expires: " + expires + "\r\n";
        ok_res = ok_res + "Content-Length: " + contentLength + "\r\n";

        ok_res = ok_res + "\r\n";
        return ok_res;
    }
}
```

REQUEST CLASS

```
package udpserver;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File Request.java
 */

public class Request
{
    String via,from,to,callId,cSeq,contact,allow,maxForwards,
            userAgent,supported,contentLength;

    public Request()    //constructor
    {
        this.allow = "";
        this.cSeq = "";
        this.callId = "";
        this.contact = "";
        this.contentLength = "";
        this.from = "";
        this.maxForwards = "";
        this.supported = "";
        this.to = "";
        this.userAgent = "";
        this.via = "";
    }
}
```

REQUEST TERMINATED CLASS

```
package udpserver;

import java.util.ArrayList;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File requestTerminatedRequest.java
 */
public class requestTerminatedRequest extends Request
{
    ArrayList<String> via;
    requestTerminatedRequest()
    {
        super();
        via = new ArrayList<>();
    }
    public String forwardrequestTerminated(String line1,String servIp,int servPort)
    {
        String fwd_res = line1 + "\r\n";

        if(via.get(0).contains(","))
        {
            String modifiedVia =
            via.get(0).substring(via.get(0).indexOf(",")+1,via.get(0).length());
            via.set(0,modifiedVia);
        }
        else
            via.remove(0);


        //add via feilds
        for(int in=0;in<via.size();in++)
            fwd_res = fwd_res + "Via: " + via.get(in) + "\r\n";

        fwd_res = fwd_res + "From: " + from + "\r\n";
        fwd_res = fwd_res + "To: " + to + "\r\n";
        fwd_res = fwd_res+ "Call-ID: " + callId + "\r\n";
        fwd_res = fwd_res + "CSeq: " + cSeq + "\r\n";

        String modifiedContact = contact.substring(0,contact.indexOf("@")+1)+
        servIp+">";
        fwd_res = fwd_res + "Contact: " + modifiedContact + "\r\n";
        fwd_res = fwd_res + "User-Agent: " + userAgent + "\r\n";
        fwd_res = fwd_res + "Content-Length: 0" + "\r\n\r\n";

        return fwd_res;
    }
}
```

RINGING REQUEST CLASS

```
package udpserver;

import java.util.ArrayList;

/**
 *
 * @author Rajat Saxena & Shivam Dabral & Biwas Bisht
 * @date 13/Jun/2016
 * @project UDP_Server
 * @File ringingRequest.java
 */
public class ringingRequest extends Request
{
    ArrayList<String> via;
    ringingRequest()
    {
        super();
        via = new ArrayList<>();
    }
    public String forwardRinging(String servIp)
    {
        String fwd_res = "SIP/2.0 180 Ringing\r\n";

        String modifiedVia =
        via.get(0).substring(via.get(0).indexOf(",")+1,via.get(0).length());
        via.set(0,modifiedVia);

        //add via feilds
        for(int in=0;in<via.size();in++)
            fwd_res = fwd_res + "Via: " + via.get(in) + "\r\n";

        fwd_res = fwd_res + "From: " + from + "\r\n";
        fwd_res = fwd_res + "To: " + to + "\r\n";
        fwd_res = fwd_res+ "Call-ID: " + callId + "\r\n";
        fwd_res = fwd_res + "CSeq: " + cSeq + "\r\n";

        String modifiedContact = contact.substring(0,contact.indexOf("@")+1)+
        servIp+">";
        fwd_res = fwd_res + "Contact: " + modifiedContact + "\r\n";

        fwd_res = fwd_res + "User-Agent: " + userAgent + "\r\n";

        fwd_res = fwd_res + "Content-Length: 0" + "\r\n\r\n";

        return fwd_res;
    }
}
```

REFRENCES

- **https://www.ietf.org/rfc/rfc3261.txt** - RFC3261 Standard for SIP
- **https://tools.ietf.org/html/rfc3665** - RFC3665 Standard for SIP Call flow Examples
- **TCP/IP Sockets in JAVA, Second Edition** : Practical Guide for programmers (The Practical Guides) 2nd Edition by Kenneth L. Calvert and Michael J. Donahoo
- **www.tutorialspoint.com/session_initiation_protocol** - Understanding SIP
- **www.phoner.de** - Phoner download
- **Voice over IP Fundamentals (2nd Edition)** : Written by Jonathan Davidson (Author), James F. Peters (Author), Manoj Bhatia (Author), Satish Kalidindi (Author), Sudipto Mukherjee (Author)