EE 284: Convergent Voice and Data Networks

Course Project 1: Session Initiation Protocol

<u>Members:</u>

Arpit Singh(010810514)

Aakash Shah (010819926)

Harsh Raijiwala (010827765)

Overview

The aim of this project is to implement Voice over IP (VoIP) using Computer as interfaces and the SIP Server. The Session Initiation Protocol (SIP) is a signaling protocol used for controlling communication sessions such as voice and video calls over IP networks. Asterisk is an open source framework working as IP PBX system, Communications Server and VoIP gateways. We have used small range, decentralized wireless network i.e. ad-hoc network to connect the VoIP users. After the clients are connected through the ad-hoc network, X-Lite is used to transmit traffic for normal call and conference call. All the traffic capturing is done using Wireshark packet sniffer tool.

Asterisk server is a Linux based software so we have used Ubuntu OS to install and configure asterisk. This Linux machine is our Communication Server. Windows machine are the clients, using X-lite softphone application for connection to ad-hoc network.

PROJECT REQUIREMENTS

Technology and Resource requirements

Hardware Requirements:

- One PC with atleast 4 GB RAM and minimum 20 GB of disk space running on Ubuntu OS to host SIP Proxy.
- Three PC’s or mobile devices capable of running softphone applications like Xlite,
- A network adapter capable of running in adhoc mode to be used as a WiFi hotspot.
- Headsets with microphones for each PC’s.

Software Requirements:

- Ubuntu 14.04.1 LTS
- Asterisk
- Microsoft Windows 7
- X-Lite for Windows

Configuration and Setup

We have assigned following IP addresses for the server and the users.
Server: 192.168.131.152
2000: 192.168.131.156
2010: 192.168.131.157
2020:192.168.131.158

```
sip.conf

[2000] username=2000
type=friend
secret=password
host=dynamic
context=from-sip

[2010] username=2010
type=friend
secret=password
host=dynamic
context=from-sip

[2020] username=2020
type=friend
secret=password
host=dynamic
context=from-sip
```

Fig: Configuration of 2 Users in sip.conf file

```
[from-sip]
exten => 2000, 1, Dial(SIP/2000,20)
exten => 2000, 2, Hangup
exten => 2010, 1, Dial(SIP/2010,20)
exten => 2010, 2, Hangup
exten => 2020, 1, Dial(SIP/2020,20)
exten => 2020, 2, Hangup


exten => 2000,1,Answer()
exten => 2000,2,Busy(10)
exten => 2000,3,Hangup

exten => 2010,1,Answer()
exten => 2010,2,Busy(10)
exten => 2010,3,Hangup

exten => 2020,1,Answer()
exten => 2020,2,Busy(10)
exten => 2020,3,Hangup
```

Fig: Setting time for busy

Phase 1:- Establishing and Analyzing a successful call between 2 SIP clients

Here, two SIP clients (windows OS) registers with the serve and a call is made between then via the server.

Wireshark capture:

```
   5 5… 192.168.131.157 192.168.131.152 SIP/…  964 Request: INVITE sip:2000@192.168.131.152 |
   6 5… 192.168.131.152 192.168.131.157 SIP    614 Status: 401 Unauthorized |
   7 5… 192.168.131.157 192.168.131.152 SIP    380 Request: ACK sip:2000@192.168.131.152 |
   8 5… 192.168.131.157 192.168.131.152 SIP/… 1126 Request: INVITE sip:2000@192.168.131.152 |
   9 5… 192.168.131.152 192.168.131.157 SIP    559 Status: 100 Trying |
  10 5… 192.168.131.152 192.168.131.156 SIP/…  990 Request: INVITE sip:2000@192.168.131.156:50929;rinstance=ed56e6bdee8b35a6 |
  11 5… 192.168.131.156 192.168.131.152 SIP    348 Status: 100 Trying |
  12 5… 192.168.131.156 192.168.131.152 SIP    510 Status: 180 Ringing |
  13 5… 192.168.131.152 192.168.131.157 SIP    575 Status: 180 Ringing |
  19 8… 192.168.131.156 192.168.131.152 SIP/…  820 Status: 200 OK |
  20 8… 192.168.131.152 192.168.131.156 SIP    516 Request: ACK sip:2000@192.168.131.156:50929;rinstance=ed56e6bdee8b35a6 |
  22 8… 192.168.131.152 192.168.131.157 SIP/…  882 Status: 200 OK |
  23 8… 192.168.131.152 192.168.131.156 SIP/…  943 Request: INVITE sip:2000@192.168.131.156:50929;rinstance=ed56e6bdee8b35a6, in-dialog |
  35 8… 192.168.131.157 192.168.131.152 SIP    501 Request: ACK sip:2000@192.168.131.152:5060 |
  36 8… 192.168.131.152 192.168.131.157 SIP/…  912 Request: INVITE sip:2010@192.168.131.157:64103;rinstance=21f8737d9fdf7200, in-dialog |
  43 8… 192.168.131.156 192.168.131.152 SIP    361 Status: 100 Trying |
  78 8… 192.168.131.157 192.168.131.152 SIP    333 Status: 100 Trying |
  79 8… 192.168.131.157 192.168.131.152 SIP/…  799 Status: 200 OK |
  80 8… 192.168.131.152 192.168.131.157 SIP    483 Request: ACK sip:2010@192.168.131.157:64103;rinstance=21f8737d9fdf7200 |
  88 8… 192.168.131.156 192.168.131.152 SIP/…  820 Status: 200 OK |
  89 8… 192.168.131.152 192.168.131.156 SIP    516 Request: ACK sip:2000@192.168.131.156:50929;rinstance=ed56e6bdee8b35a6 |
  90 8… 192.168.131.152 192.168.131.156 SIP/…  943 Request: INVITE sip:2000@192.168.131.156:50929;rinstance=ed56e6bdee8b35a6, in-dialog |
 110 9… 192.168.131.156 192.168.131.152 SIP    361 Status: 100 Trying |
 122 9… 192.168.131.156 192.168.131.152 SIP/…  820 Status: 200 OK |
 123 9… 192.168.131.152 192.168.131.156 SIP    516 Request: ACK sip:2000@192.168.131.156:50929;rinstance=ed56e6bdee8b35a6 |
1255 2… 192.168.131.156 192.168.131.152 SIP    540 Request: BYE sip:2010@192.168.131.152:5060 |
1256 2… 192.168.131.152 192.168.131.156 SIP    564 Status: 200 OK |
```

Connection activities between the SIP clients via SIP Proxy server (Everything from Registering the SIP client to the termination of a call).

MAC address of the server

```
Ethernet II, Src: Vmware_66:d2:f2 (00:0c:29:66:d2:f2), Dst: Vmware_71:83:8e (00:0c:29:71:83:8e)
  > Destination: Vmware_71:83:8e (00:0c:29:71:83:8e)
  > Source: Vmware_66:d2:f2 (00:0c:29:66:d2:f2)
    Type: IPv4 (0x0800)
```

Server MAC: 00:0C:29:66:D2:F2

Phase 2:- Busy User

In this part of the experiment, we make one user busy while the other user tries to call it through the server, and we want to analyze the SIP messages exchanged. We can use this application in case of timeouts or invalid conditions

Wireshark capture:

77 1…	192.168.131.157	192.168.131.152	SIP	590	Request: REGISTER sip:192.168.131.152 (1 binding) |
87 1…	192.168.131.152	192.168.131.157	SIP	622	Status: 401 Unauthorized |
88 1…	192.168.131.157	192.168.131.152	SIP	747	Request: REGISTER sip:192.168.131.152 (1 binding) |
93 1…	192.168.131.152	192.168.131.157	SIP	671	Status: 200 OK (1 binding) |
132 1…	192.168.131.157	192.168.131.152	SIP	658	Request: SUBSCRIBE sip:2010@192.168.131.152 |
133 1…	192.168.131.152	192.168.131.157	SIP	617	Status: 401 Unauthorized |
134 1…	192.168.131.157	192.168.131.152	SIP	820	Request: SUBSCRIBE sip:2010@192.168.131.152 |
135 1…	192.168.131.152	192.168.131.157	SIP	551	Status: 404 Not found (no mailbox) |
180 1…	192.168.131.157	192.168.131.152	SIP	590	Request: REGISTER sip:192.168.131.152 (1 binding) |
181 1…	192.168.131.152	192.168.131.157	SIP	622	Status: 401 Unauthorized |
182 1…	192.168.131.157	192.168.131.152	SIP	747	Request: REGISTER sip:192.168.131.152 (1 binding) |
183 1…	192.168.131.152	192.168.131.157	SIP	671	Status: 200 OK (1 binding) |
289 7…	192.168.131.157	192.168.131.152	SIP/…	964	Request: INVITE sip:2000@192.168.131.152 |
290 7…	192.168.131.152	192.168.131.157	SIP	614	Status: 401 Unauthorized |
291 7…	192.168.131.157	192.168.131.152	SIP	380	Request: ACK sip:2000@192.168.131.152 |
292 7…	192.168.131.157	192.168.131.152	SIP/…	1126	Request: INVITE sip:2000@192.168.131.152 |
293 7…	192.168.131.152	192.168.131.157	SIP	559	Status: 100 Trying |
294 7…	192.168.131.152	192.168.131.156	SIP/…	990	Request: INVITE sip:2000@192.168.131.156:50929;rinstance=ed56e6bdee8b35a6
295 7…	192.168.131.156	192.168.131.152	SIP	348	Status: 100 Trying |
296 7…	192.168.131.156	192.168.131.152	SIP	510	Status: 180 Ringing |
297 7…	192.168.131.152	192.168.131.157	SIP	575	Status: 180 Ringing |
320 9…	192.168.131.152	192.168.131.156	SIP	467	Request: CANCEL sip:2000@192.168.131.156:50929;rinstance=ed56e6bdee8b35a6
321 9…	192.168.131.152	192.168.131.157	SIP	534	Status: 603 Declined |
322 9…	192.168.131.156	192.168.131.152	SIP	473	Status: 200 OK |
323 9…	192.168.131.156	192.168.131.152	SIP	419	Status: 487 Request Terminated |
324 9…	192.168.131.152	192.168.131.156	SIP	516	Request: ACK sip:2000@192.168.131.156:50929;rinstance=ed56e6bdee8b35a6 |
326 9…	192.168.131.152	192.168.131.157	SIP	534	Status: 603 Declined |

MAC address of the server

```
Ethernet II, Src: Vmware_66:d2:f2 (00:0c:29:66:d2:f2), Dst: Vmware_71:83:8e (00:0c:29:71:83:8e)
  > Destination: Vmware_71:83:8e (00:0c:29:71:83:8e)
  > Source: Vmware_66:d2:f2 (00:0c:29:66:d2:f2)
    Type: IPv4 (0x0800)
```

Server MAC: 00:0C:29:66:D2:F2

Phase 3:- Call on Hold

In this scenario, user 2020 tries to call user 2000, when the call between user 2000 and user 2010 is already established. When user 2020 tries to call user 2000, it keeps user 2010 on hold and accepts the call from user 2020. And after completing the call with user 2020, it resumes a call with user 2010.

Wireshark capture:

```
1201 2…  192.168.131.152 192.168.131.158 SIP       614 Status: 401 Unauthorized |
1202 2…  192.168.131.158 192.168.131.152 SIP       380 Request: ACK sip:2000@192.168.131.152 |
1206 2…  192.168.131.158 192.168.131.152 SIP/SDP  1126 Request: INVITE sip:2000@192.168.131.152 |
1207 2…  192.168.131.152 192.168.131.158 SIP       559 Status: 100 Trying |
1251 2…  192.168.131.152 192.168.131.158 SIP       575 Status: 180 Ringing |
3218 4…  192.168.131.152 192.168.131.158 SIP       534 Status: 603 Declined |
3232 4…  192.168.131.158 192.168.131.152 SIP       380 Request: ACK sip:2000@192.168.131.152 |
9516 1…  192.168.131.158 192.168.131.152 SIP/SDP   964 Request: INVITE sip:2000@192.168.131.152 |
9517 1…  192.168.131.152 192.168.131.158 SIP       614 Status: 401 Unauthorized |
9520 1…  192.168.131.158 192.168.131.152 SIP       380 Request: ACK sip:2000@192.168.131.152 |
9521 1…  192.168.131.158 192.168.131.152 SIP/SDP  1126 Request: INVITE sip:2000@192.168.131.152 |
9522 1…  192.168.131.152 192.168.131.158 SIP       559 Status: 100 Trying |
9561 1…  192.168.131.152 192.168.131.158 SIP       575 Status: 180 Ringing |
104… 1…  192.168.131.152 192.168.131.158 SIP/SDP   882 Status: 200 OK |
105… 1…  192.168.131.158 192.168.131.152 SIP       501 Request: ACK sip:2000@192.168.131.152:5060 |
105… 1…  192.168.131.152 192.168.131.158 SIP/SDP   912 Request: INVITE sip:2020@192.168.131.158:65273;rinstance=b32701279717a0…
105… 1…  192.168.131.158 192.168.131.152 SIP       333 Status: 100 Trying |
105… 1…  192.168.131.158 192.168.131.152 SIP/SDP   799 Status: 200 OK |
105… 1…  192.168.131.152 192.168.131.158 SIP       483 Request: ACK sip:2020@192.168.131.158:65273;rinstance=b32701279717a085 …
113… 1…  192.168.131.152 192.168.131.158 SIP/SDP   912 Request: INVITE sip:2020@192.168.131.158:65273;rinstance=b32701279717a0…
113… 1…  192.168.131.158 192.168.131.152 SIP       333 Status: 100 Trying |
113… 1…  192.168.131.158 192.168.131.152 SIP/SDP   799 Status: 200 OK |
113… 1…  192.168.131.152 192.168.131.158 SIP       483 Request: ACK sip:2020@192.168.131.158:65273;rinstance=b32701279717a085 …
113… 1…  192.168.131.152 192.168.131.158 SIP       682 Request: BYE sip:2020@192.168.131.158:65273;rinstance=b32701279717a085 …
114… 1…  192.168.131.152 192.168.131.158 SIP       682 Request: BYE sip:2020@192.168.131.158:65273;rinstance=b32701279717a085 …
114… 1…  192.168.131.158 192.168.131.152 SIP       442 Status: 200 OK |
114… 1…  192.168.131.158 192.168.131.152 SIP       442 Status: 200 OK |
```

MAC address of the server

```
Ethernet II, Src: Vmware_66:d2:f2 (00:0c:29:66:d2:f2), Dst: Vmware_71:83:8e (00:0c:29:71:83:8e)
  Destination: Vmware_71:83:8e (00:0c:29:71:83:8e)
  Source: Vmware_66:d2:f2 (00:0c:29:66:d2:f2)
  Type: IPv4 (0x0800)
```

Server MAC: 00:0C:29:66:D2:F2

Phase 4:- Call Conferencing: In this scenario, user 2020 tries to call user 2000, when the call between user 2000 and user 2010 is already established. When user 2020 tries to call user 2000, user 2000 keeps user 2010 on hold and accepts the call from user 2020. Then, user 2000 invites user 2010 again to join the pre-established call. As a result, a call conference is established.

Wireshark capture:

```
 903 1… 192.168.131.152 192.168.131.157 SIP/…  943 Request: INVITE sip:2010@192.168.131.157:56841;rinstance=d24cdd8c62238612, in-dialog |
 930 1… 192.168.131.157 192.168.131.152 SIP    361 Status: 100 Trying |
 931 1… 192.168.131.157 192.168.131.152 SIP/…  820 Status: 200 OK |
 933 1… 192.168.131.152 192.168.131.157 SIP    516 Request: ACK sip:2010@192.168.131.157:56841;rinstance=d24cdd8c62238612 |
 963 1… 192.168.131.156 192.168.131.152 SIP    501 Request: ACK sip:2010@192.168.131.152:5060 |
 987 1… 192.168.131.158 192.168.131.152 SIP    501 Request: ACK sip:2000@192.168.131.152:5060 |
 988 1… 192.168.131.152 192.168.131.158 SIP/…  910 Request: INVITE sip:2020@192.168.131.158:65273;rinstance=b32701279717a085, in-dialog |
1004 1… 192.168.131.156 192.168.131.152 SIP    361 Status: 100 Trying |
1005 1… 192.168.131.156 192.168.131.152 SIP/…  820 Status: 200 OK |
1006 1… 192.168.131.152 192.168.131.156 SIP    516 Request: ACK sip:2000@192.168.131.156:61795;rinstance=9e654bd39254cdc0 |
1007 1… 192.168.131.158 192.168.131.152 SIP/…  799 Status: 200 OK |
1009 1… 192.168.131.152 192.168.131.158 SIP    483 Request: ACK sip:2020@192.168.131.158:65273;rinstance=b32701279717a085 |
1010 1… 192.168.131.152 192.168.131.156 SIP/…  943 Request: INVITE sip:2000@192.168.131.156:61795;rinstance=9e654bd39254cdc0, in-dialog |
1065 1… 192.168.131.156 192.168.131.152 SIP    361 Status: 100 Trying |
1074 1… 192.168.131.156 192.168.131.152 SIP/…  820 Status: 200 OK |
1075 1… 192.168.131.152 192.168.131.156 SIP    516 Request: ACK sip:2000@192.168.131.156:61795;rinstance=9e654bd39254cdc0 |
1804 1… 192.168.131.156 192.168.131.152 SIP/… 1151 Request: INVITE sip:2010@192.168.131.152:5060, in-dialog |
1805 1… 192.168.131.152 192.168.131.156 SIP    574 Status: 100 Trying |
1806 1… 192.168.131.152 192.168.131.156 SIP/…  882 Status: 200 OK |
1807 1… 192.168.131.152 192.168.131.157 SIP/…  943 Request: INVITE sip:2010@192.168.131.157:56841;rinstance=d24cdd8c62238612, in-dialog |
1818 1… 192.168.131.156 192.168.131.152 SIP    501 Request: ACK sip:2010@192.168.131.152:5060 |
1825 1… 192.168.131.157 192.168.131.152 SIP/…  820 Status: 200 OK |
1826 1… 192.168.131.152 192.168.131.157 SIP    516 Request: ACK sip:2010@192.168.131.157:56841;rinstance=d24cdd8c62238612 |
2632 2… 192.168.131.156 192.168.131.152 SIP    540 Request: BYE sip:2020@192.168.131.152:5060 |
2633 2… 192.168.131.156 192.168.131.152 SIP    668 Request: BYE sip:2010@192.168.131.152:5060 |
2634 2… 192.168.131.152 192.168.131.156 SIP    564 Status: 200 OK |
2635 2… 192.168.131.152 192.168.131.156 SIP    525 Status: 200 OK |
```

MAC address of the server

```
Ethernet II, Src: Vmware_66:d2:f2 (00:0c:29:66:d2:f2), Dst: Vmware_71:83:8e (00:0c:29:71:83:8e)
  > Destination: Vmware_71:83:8e (00:0c:29:71:83:8e)
  > Source: Vmware_66:d2:f2 (00:0c:29:66:d2:f2)
    Type: IPv4 (0x0800)
```

Server MAC: 00:0C:29:66:D2:F2

SIP FLOW in Wireshark

SIP protocol is used to initiate a session between two endpoints. Its job is not to carry any voice or video data itself, it only allows two endpoints to set up connection using SDP encapsulated in SIP messages, for transferring that traffic (voice or video) between each other via other protocol, the Real-time Transport Protocol (RTP).

This is the SIP Request header that tells us what kind of SIP message this is. This particular packet is a SIP INVITE request for below extension.

```
Session Initiation Protocol (INVITE)
  Request-Line: INVITE sip:2010@192.168.131.152 SIP/2.0
    Method: INVITE
    Request-URI: sip:2010@192.168.131.152
      Request-URI User Part: 2010
      Request-URI Host Part: 192.168.131.152
```

The “Via” header contains a list of all SIP proxy servers that this packet has passed through, including the initiating client.

```
Message Header
  Via: SIP/2.0/UDP 192.168.131.156:61795;branch=z9hG4bK-524287-1---45dd62721e6f5519;rport
    Transport: UDP
    Sent-by Address: 192.168.131.156
    Sent-by port: 61795
    Branch: z9hG4bK-524287-1---45dd62721e6f5519
    RPort: rport
  Max-Forwards: 70
```

The “To” header specifies the SIP packet’s destination

```
To: <sip:2010@192.168.131.152>
  SIP to address: sip:2010@192.168.131.152
    SIP to address User Part: 2010
    SIP to address Host Part: 192.168.131.152
```

The “From” header specified who sent the SIP packet

```
From: "2000"<sip:2000@192.168.131.152>;tag=e275434c
  SIP Display info: "2000"
  SIP from address: sip:2000@192.168.131.152
    SIP from address User Part: 2000
    SIP from address Host Part: 192.168.131.152
  SIP from tag: e275434c
Call-ID: 82158ODUyZWUzOTNlNDA3MDhmZWZkMjU5YjVkZTIxZjBhNGY
```

The CSeq header field serves as a way to identify and order transactions. It consists of a sequence number and a method. The method MUST match that of the request.

```
˅ CSeq: 1 INVITE
    Sequence Number: 1
    Method: INVITE
  Allow: SUBSCRIBE, NOTIFY, INVITE, ACK, CANCEL, BYE, REFER, INFO, OPTIONS, MESSAGE
  Content-Type: application/sdp
  Supported: replaces
  User-Agent: X-Lite release 4.9.6 stamp 82158
  Content-Length: 340
```

This particular packet is a SDP packet, it contains a SDP message with information the remote client needs to open an RTP session for this call.

```
˅ Session Description Protocol
    Session Description Protocol Version (v): 0
```

The IP address of the SIP client that created this packet

```
˅ Owner/Creator, Session Id (o): - 13122259317790375 1 IN IP4 192.168.131.156
    Owner Username: -
    Session ID: 13122259317790375
    Session Version: 1
    Owner Network Type: IN
    Owner Address Type: IP4
    Owner Address: 192.168.131.156
  Session Name (s): X-Lite release 4.9.6 stamp 82158
```

The IP address the destination SIP client should contact to open an RTP session.

```
˅ Connection Information (c): IN IP4 192.168.131.156
    Connection Network Type: IN
    Connection Address Type: IP4
    Connection Address: 192.168.131.156
```

The key pieces of information in this header are audio, 65350 and RTP/AVP. The audio component obviously signifies that this is an audio call, 65350 specifies the port where want to receive the RTP stream.

```
⌄ Media Description, name and address (m): audio 65350 RTP/AVP 9 8 120 0 84 101
    Media Type: audio
    Media Port: 65350
    Media Protocol: RTP/AVP
    Media Format: ITU-T G.722
    Media Format: ITU-T G.711 PCMA
    Media Format: DynamicRTP-Type-120
    Media Format: ITU-T G.711 PCMU
    Media Format: Unassigned
    Media Format: DynamicRTP-Type-101
```

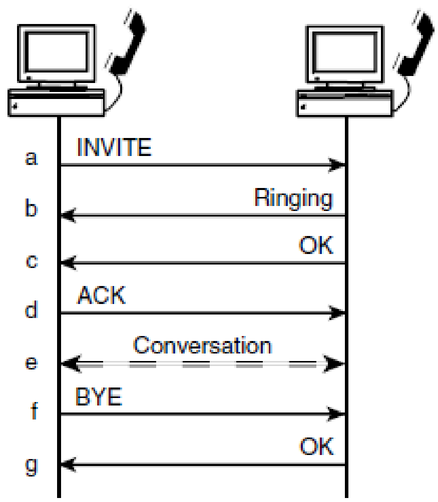
SIP 3-way Handshake

SIP implements a three-way handshake as follows:

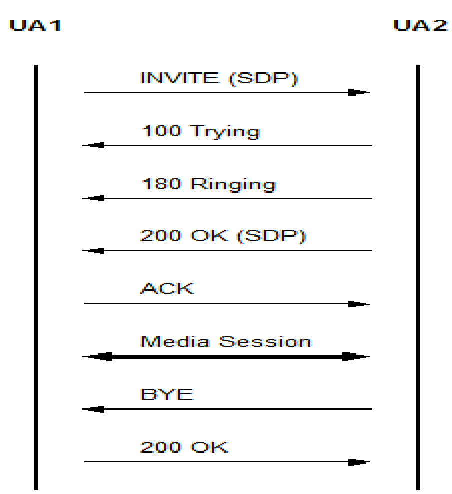
1. The caller (SIP client) sends an INVITE to the receiver.
2. The receiver (SIP client) sends an 200 OK to accept the call
3. The caller sends an ACK to indicate that the handshake is done and a call is going to be setup.

Point to remember is:

If the first INVITE message for the sender to receiver includes a SDP call description, the 200 OK response message includes the receiver (callee's) SDP.



The steps until ACK, are the part of 3-way handshake. After that, the media session starts between the SIP participants. Another figure is shown below:



Short note on Ad-Hoc Networks

Unlike the infrastructure networks which includes some Local Area Networks, the ad-hoc networks are comprised of a group of workstations or other wireless devices which communicate directly with each other to exchange information

An ad hoc is a network where there are no access points between participants or the devices connected to that network. Generally, it is a network without any base stations or central information hub in between. It can be called as Infrastructure less or multi-hop functionality.

It is a collection where two or more devices with wireless communications capability are connected onto the same network. It is a peer to peer network.

Ad- hoc is a closed network where none of the computers are connected to the internet, but are created between the participants. But, if one of the participants has a connection to a public or private network, this connection can be shared among other members of the network. This will allow other users on the spontaneous ad hoc network to connect to the Internet as well.

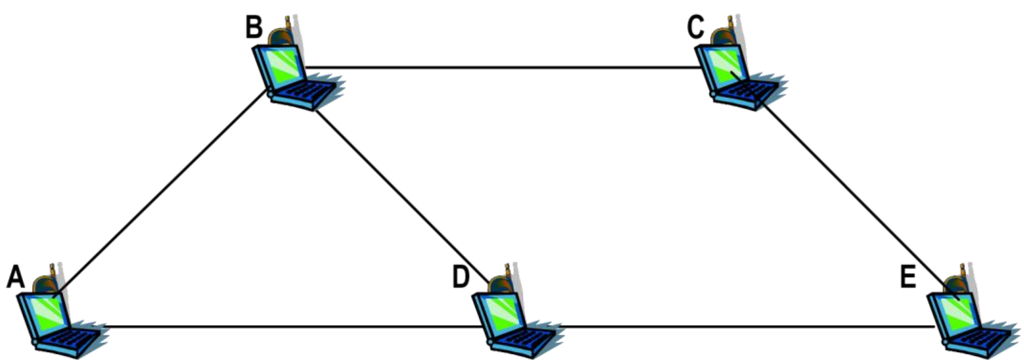
For example, early computers could connect to each other to exchange information, usually in a smaller office environment without the need for domains and the additional management.

It allows spontaneous formation and deformation of mobile networks.

Each mobile or node acts as a router itself.

Administration cost reduces.

The basic operating principle is shown below:



Here, mobile node A communicates directly with B when a channel is available, because of a single hop.
But if channel is not available, then multi-hop communication is necessary i.e. it will take path A-D-B.
For implementation of these multi-hop communications, the intermediate nodes should act as a router and pass the packets.
For communication between A and E via mobile D, the mobile D should act as a router.

Part2

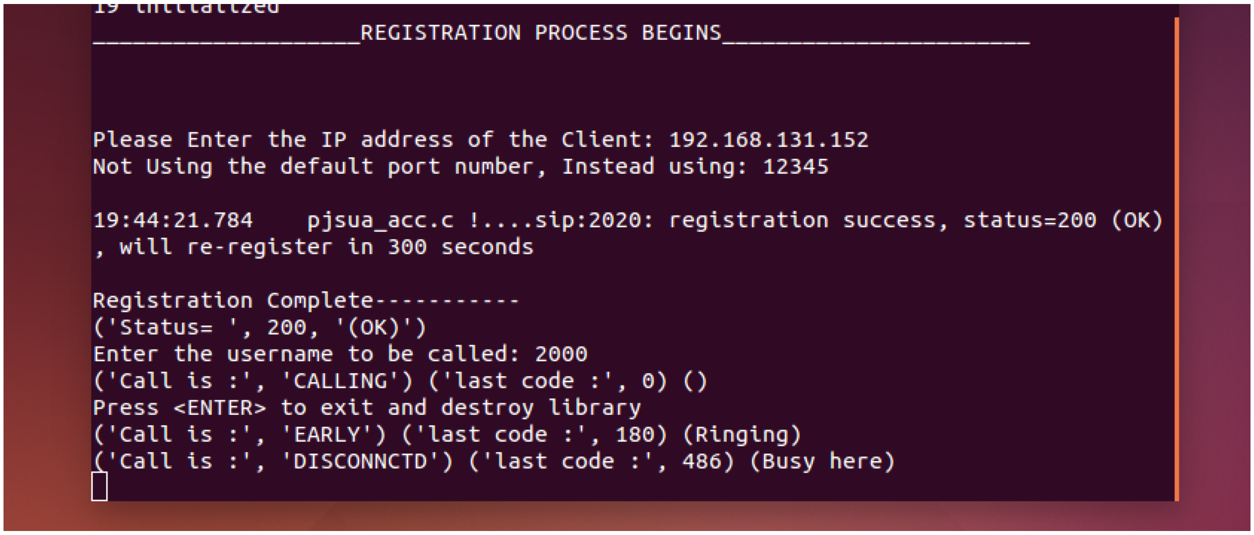


Fig: Screenshot of the terminal window

The Script on running will ask the user for the client address, but in this case it is same as the server's address.

In our source code, we have hard coded the user IP address in the form of username @ server:port

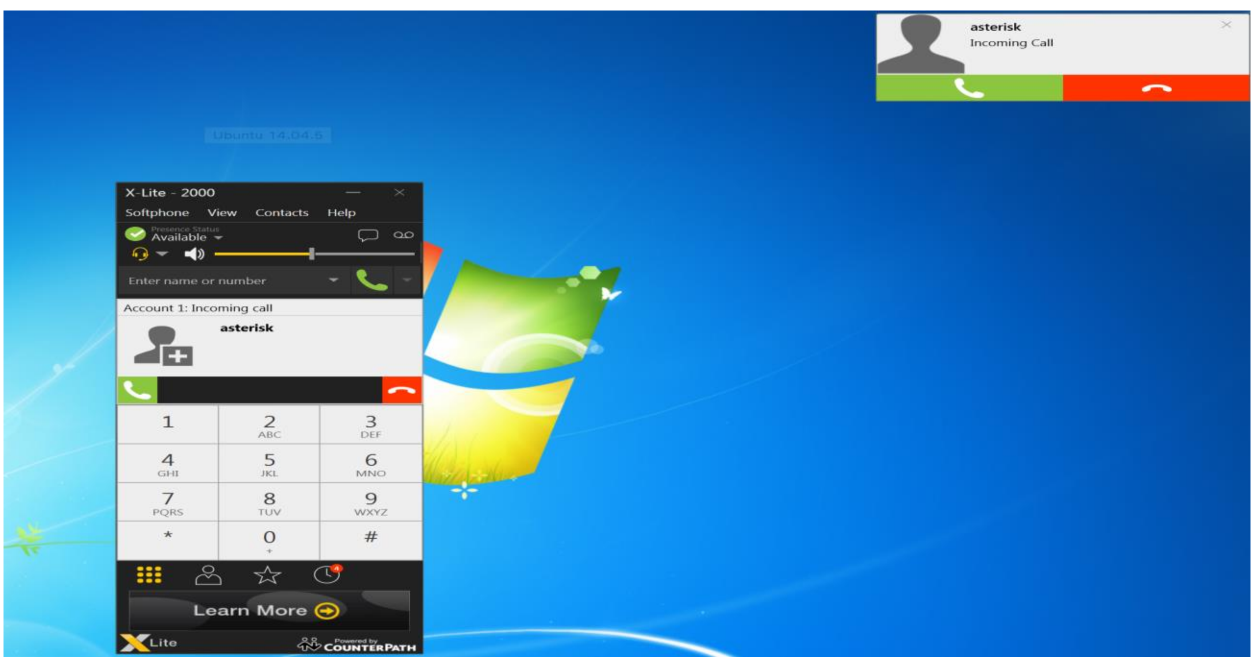


Fig: Established call between the script and XLite phone.

Source Code:

```
# SIP Client (Soft phone prototype)
# Created by Arpit Singh, Akash Shah, Harsh Raijiwala
# Run the server before running this file "$sudo asterisk -r" in other terminal
#In user URI enter only the Called user name
import sys
import pjsua as pj
import threading
import time
# Log of callback class
def log_cb(level, str, len):
    print(str),
# Account Callback class to get notifications of Account registration
class MyAccountCallback(pj.AccountCallback):
    def __init__(self, acc):
        pj.AccountCallback.__init__(self, acc)
# Call Callback to receive events from Call
class SRCallCallback(pj.CallCallback):
    def __init__(self, call=None):
        pj.CallCallback.__init__(self, call)
    def on_state(self):
        print("Call is :", self.call.info().state_text),
        print("last code :", self.call.info().last_code),
        print("(" + self.call.info().last_reason + ")")
# Notification when call's media state is changed
    def on_media_state(self):
        global lib
        if self.call.info().media_state == pj.MediaState.ACTIVE:
            # Connect the call to sound device
            call_slot = self.call.info().conf_slot
            lib.conf_connect(call_slot, 0)
            lib.conf_connect(0, call_slot)
            print("Hey !!!!! Can you hear me !!!!!!!!!!")
            print (lib)
# Main loop
try:
    # Start of the Main Class
    # Create library instance of Lib class
    lib = pj.Lib()
    # Instantiate library with default config
    lib.init(log_cfg = pj.LogConfig(level=3, callback=log_cb))
    # Configuring one Transport Object and setting it to listen at 5060 port and UDP protocol
    trans_conf = pj.TransportConfig()
```

```
    print "___________ REGISTRATION PROCESS BEGINS __________"
    print "\n\n"
    # 12345 is default port for SIP
    trans_conf.port = 12345
    # Here the client address is same as the Servers Address
    a=raw_input("Please Enter the IP address of the Client: ")
    print "Not Using the default port number, Instead using: 12345"
    trans_conf.bound_addr = a
    transport = lib.create_transport(pj.TransportType.UDP,trans_conf)
   # Starting the instance of Lib class
    lib.start()
    lib.set_null_snd_dev()
    # Configuring Account class to register with Registrar server
    # Giving information to create header of REGISTER SIP message
    # Hardcoded these values
    ab4="192.168.131.152" # Server's address
    ab='2020' # This clients User name
    ab1="password" # Password same as "password"
    ab2='y'
    ab3=ab
    acc_conf = pj.AccountConfig(domain = ab4, username = ab, password =ab1, display = ab3)
    # registrar = 'sip:'+ab4+':5060', proxy = 'sip:'+ab4+':5060')
    acc_conf.id ="sip:"+ab
    acc_conf.reg_uri ='sip:'+ab4+':12345'
    acc_callback = MyAccountCallback(acc_conf)
    acc = lib.create_account(acc_conf,cb=acc_callback)
    # creating instance of AccountCallback class
    acc.set_callback(acc_callback)
    print('\n')
    print "Registration Complete-----------"
    print('Status= ',acc.info().reg_status, \
         '(' + acc.info().reg_reason + ')')
            # Starting Calling process.
    b=raw_input("Enter the username to be called: ")
            #sip and address are hard coded here
    b1="sip:"+ str(b)+"@192.168.0.1:5066"
    call = acc.make_call(b1, SRCallCallback())
    print('Press <ENTER> to exit and destroy library')
    input = sys.stdin.readline().rstrip('\r\n')
# Shutting down the library
    lib.destroy()
    lib = None
except pj.Error, e:
```

```
print("Exception, error occured at : " + str(e))
lib.destroy()
lib = None
sys.exit(1)
```