



# **San Jose State University**

## **Department of Electrical Engineering**

---

**EE-284 - Section - 2**  
**Voice and Data Networks**

*Course Project 1*

**Voice over Wireless Ad-Hoc Network**  
**A Hands on SIP based VoIP Experiments on : Call Establishment,**  
**Busy Lines, Call on Hold and Conference Calling**

---

*Submitted to -*  
**Prof. Balaji Venkatraman**

*Submitted by -*  
**Shivang Singh**

## **PART - 1**

### **OBJECTIVE -**

To learn the implementation of Voice over IP (VoIP) using Session Initiation Protocol (SIP) over Ad-Hoc Network.

### **HARDWARE SETUP -**

For Phase 1 and 2 , we need two clients and one server machine (a Linux machine). Among a total of three machines, we are using a Linux machine as Server and two Windows machine as our clients.

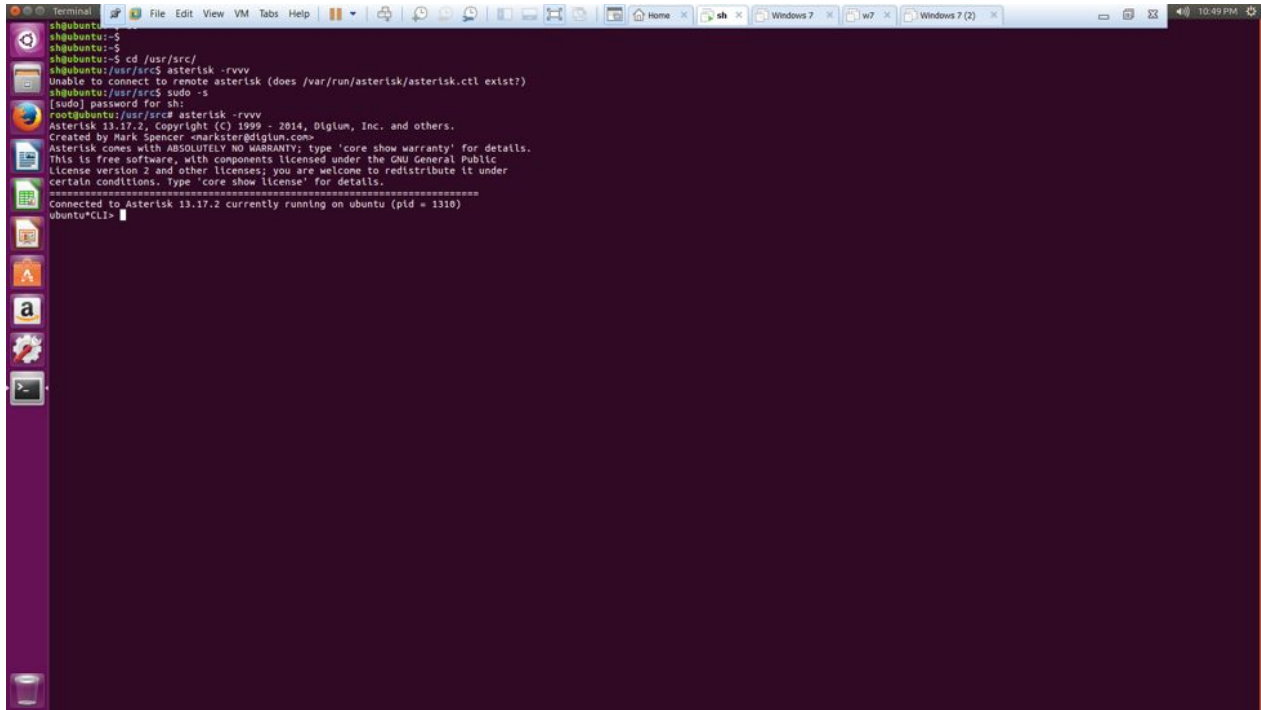
**Server** - The asterisk server is installed on the server machine with the sip.conf and extension.conf files already within them. These files are used for registration of sip clients on the server machine and the call set up.

For creating two clients, we added the following commands in the sip.conf files which we have downloaded along with the asterisk server. They are as below -

```
[general]
port=5060; Port to bind to (SIP is 5060)
bind addr = 10.42.0.1 = Asterisk server IP address allow =
Ulaw ; All all codecs
```

```
[100] username=100
type=friend
secret=password
Host=dynamic
context=from-sip
```

```
[200] username=200
type=friend
secret=password
hots=dynamic
context=from-sip
```

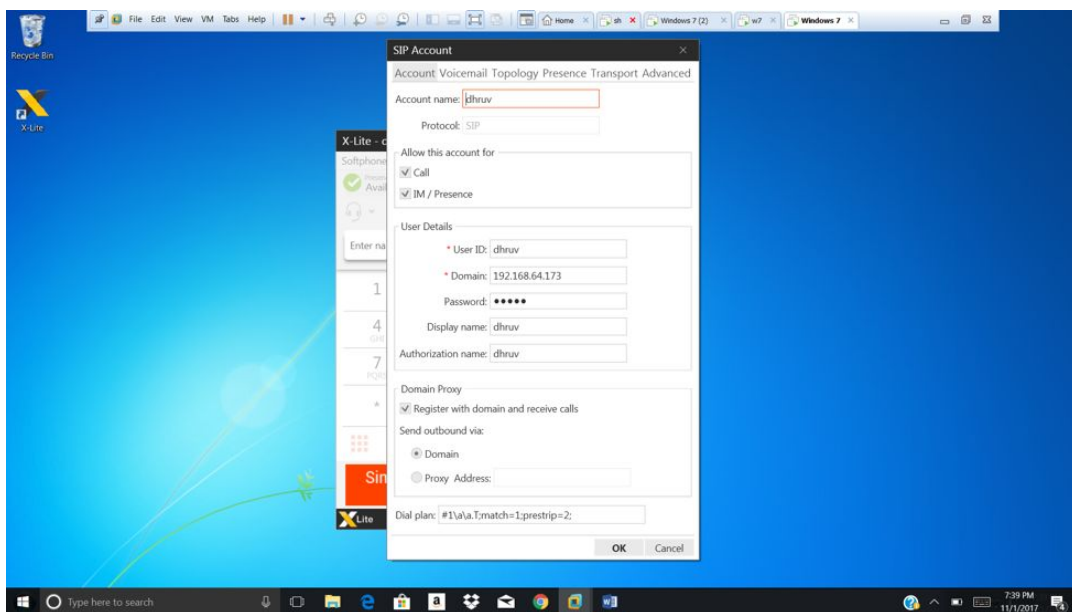
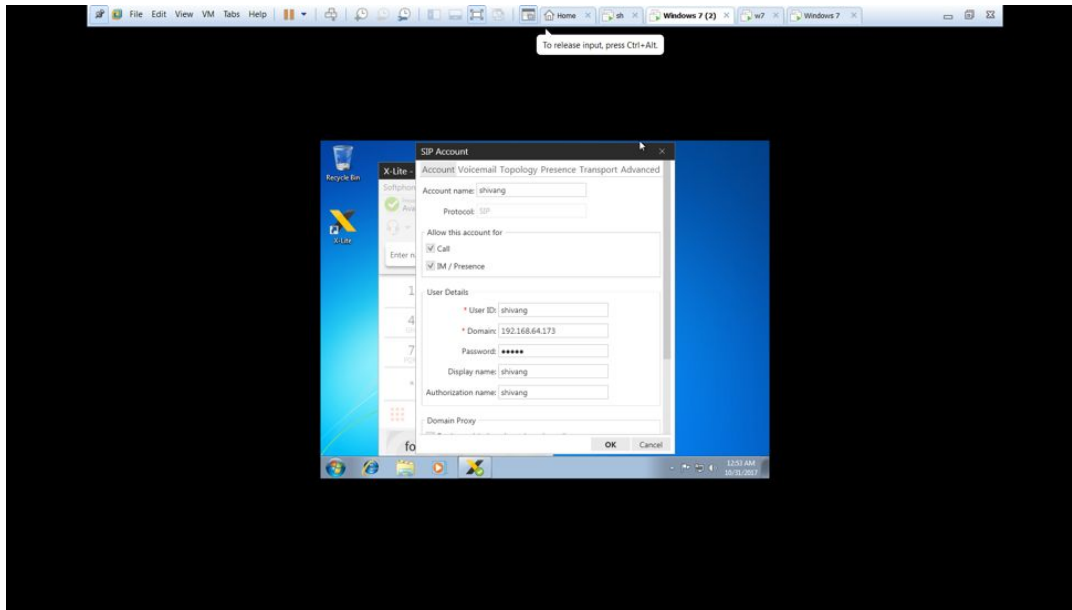


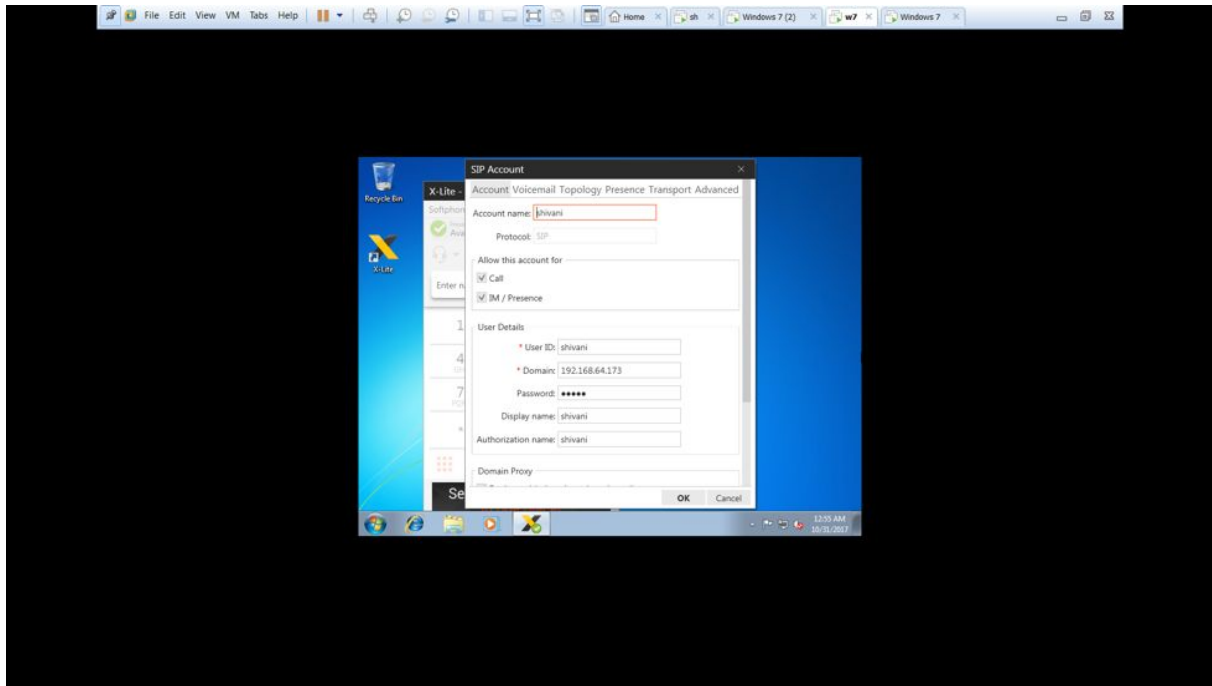
```
sh@ubuntu:~$  
sh@ubuntu:~$  
sh@ubuntu:~$ cd /usr/src/  
sh@ubuntu:/usr/src$ asterisk -rvvv  
Unable to connect to remote asterisk (does /var/run/asterisk/asterisk.ctl exist?)  
sh@ubuntu:/usr/src$ sudo -s  
[sudo] password for sh:  
root@ubuntu:/usr/src# asterisk -rvvv  
Asterisk 13.17.2, Copyright (C) 1999 - 2014, Digium, Inc. and others.  
Created by Mark Spencer <markster@digium.com>  
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.  
This is free software, with components licensed under the GNU General Public  
License version 2 and other licenses; you are welcome to redistribute it under  
certain conditions. Type 'core show license' for details.  
=====  
Connected to Asterisk 13.17.2 currently running on ubuntu (pid = 1310)  
ubuntu*CLI>
```

**Clients** - The X-Lite Softphone software is installed on the to client machines, which is VoIP smartphone and uses SIP (session initiation protocol). One client is assigned with an user ID of 100 and the other is done the same with ID 200.

We can check the IP addresses on all the machines, of both server and the two clients. Also the IP addresses on all the machines, will be in the same network.

These are the images for the clients -





## **PHASE - 1 : Establishment and Analysis of calls between two clients**

### ***Call Establishment:***

For establishing a call between our two sip clients with user IDs 100 and 200, we modified the extension.conf, which we downloaded along with the asterisk software. Here the user name will be the name as displayed on the X-Lite softphone when we connect it with the server. We are required to add the following commands in the extension. Conf file of the asterisk server.

```
[from-sip]
exten=> 100, 1, Dial (SIP/100,20)
exten=> 100, 2, Hangup
exten=> 100, 1, Dial (SIP/200,20) exten=> 2-1, 2, Hangup
```

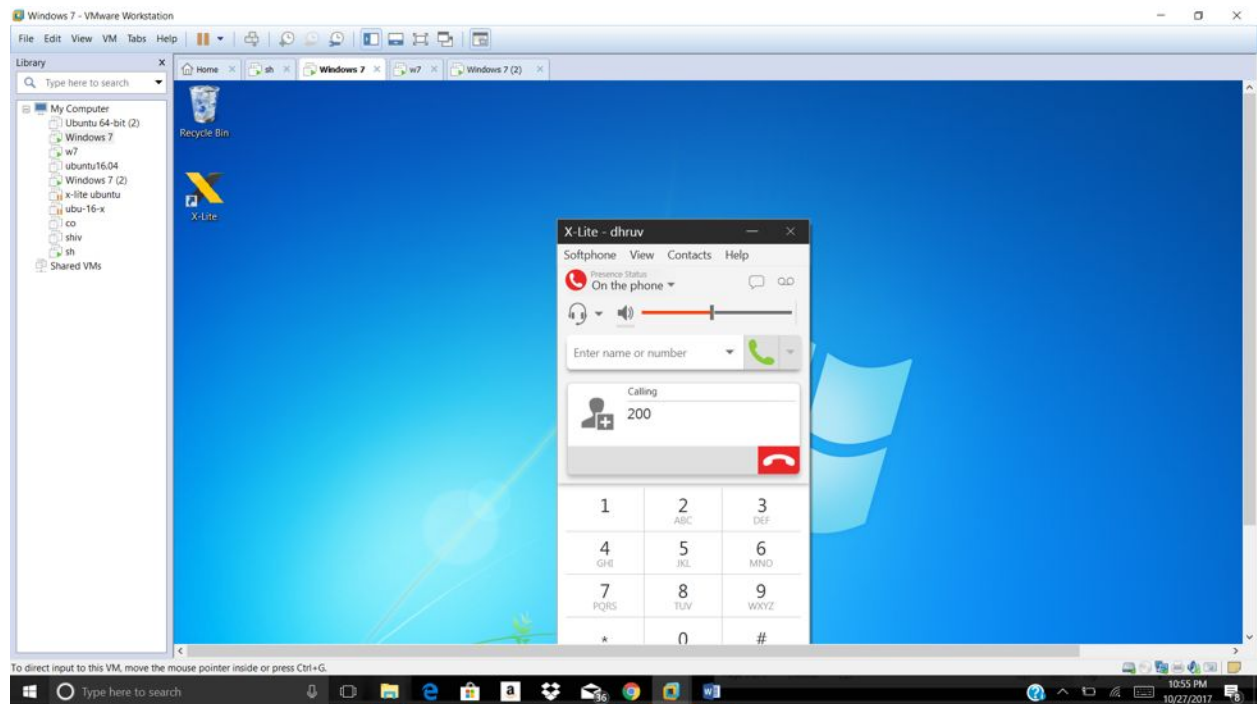
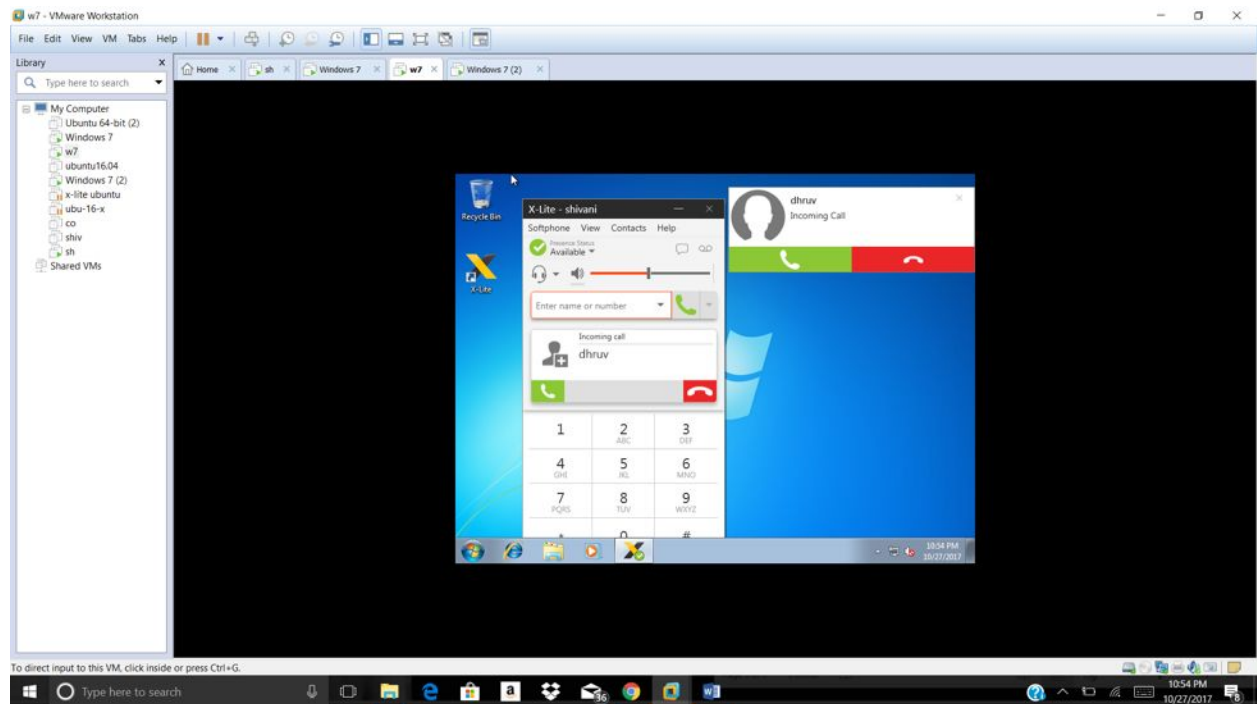
After modified the extension file, we reloaded the asterisk server to save the changes which we made. This is done by typing the reload command. We now call one client to another by dialing the user id and also capture the packets through Wireshark. Here we are calling from client with id 100 to 200.

Here in our project, we have,

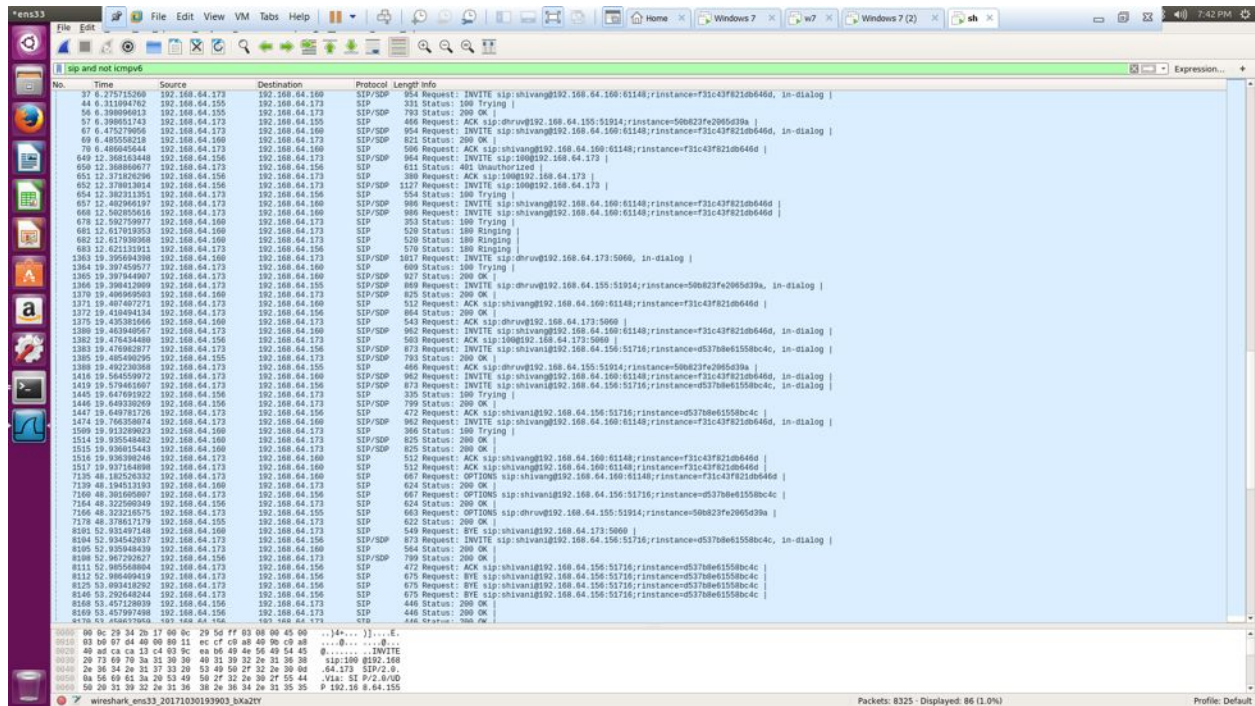
<b><u>Name on X-Lite Phone</u></b>	<b><u>ID (in our project)</u></b>	<b><u>ID (in original file)</u></b>
Shivang	100	2000
Shivani	200	2010
Dhruv	300	2020

***Experiment Result*** : Call was successfully established between two clients.

Calling 200 from 300-



The Wireshark Image will be the following one -





## PHASE-2 : Busy User

In this mode of operation, one user tries to call the other user, but the other user is busy. The time duration for which the user wants to set the busy mode is sent as an argument in the extension.conf file.

### Steps -

1. Configure the two SIP users in “sip.conf” file as done in Phase 1 of the experiment.
2. Configure the extension.conf file for the 2 SIP users as given below -

[from-sip]

exten=> 100,1, Dial(SIP/100,20)

exten=> 100,2, Hangup exten => 200,1, Answer() exten => 200,2, Busy (10)

exten=> 200,3, Hangup

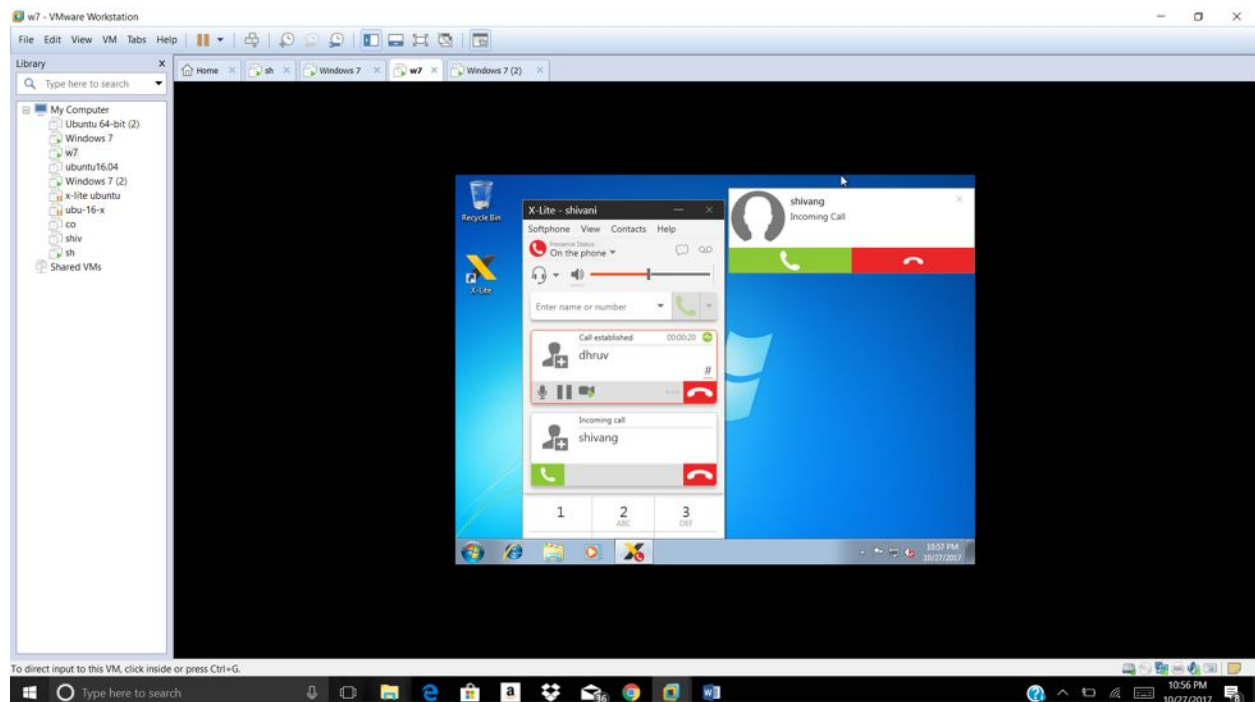
Now, here in this one client with ID 200 is made busy with the busy period of the time mentioned in the brackets; here in our case we have considered the time to be busy as of 10 seconds.

exten=> 200,1, Answer

exten=> 200,2, Busy(10) exten=>

200,3, Hangup

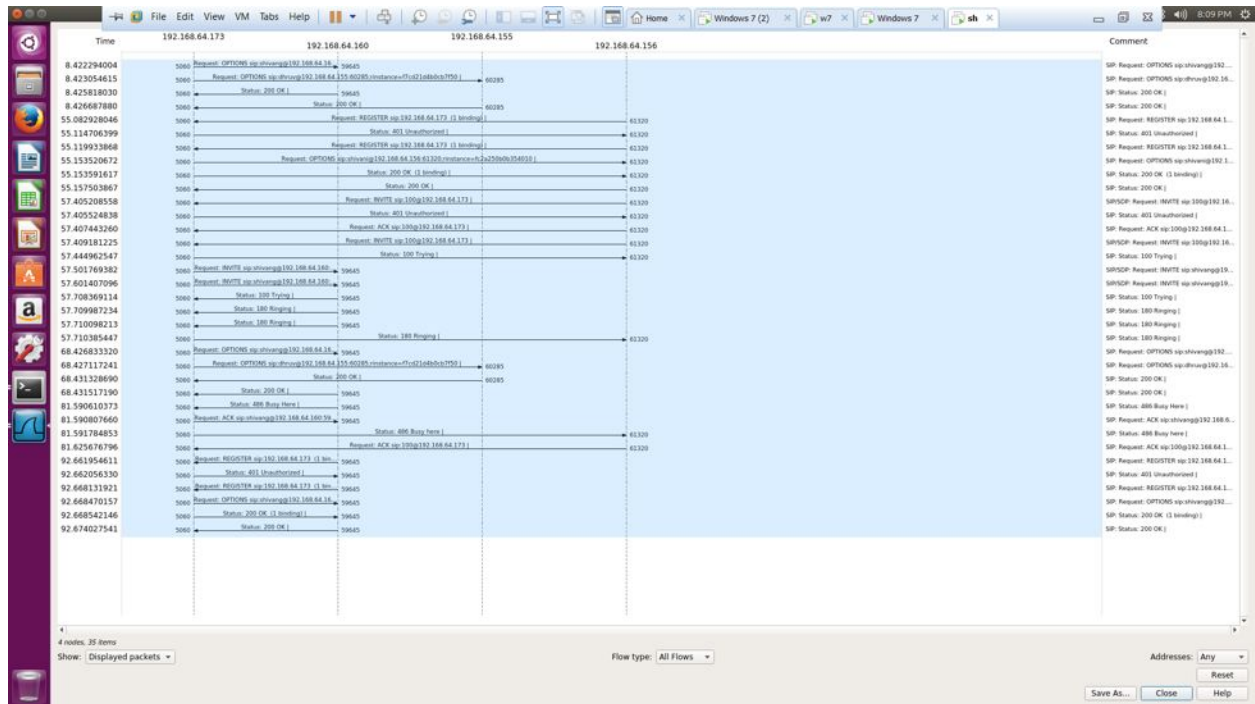
200=> Busy



```
sh@ubuntu:~$ cd /usr/src/
sh@ubuntu:~/src$ asterisk -rvvv
Unable to connect to remote asterisk (does /var/run/asterisk/asterisk.ctl exist?)
sh@ubuntu:~/src$ sudo -s
[sudo] password for sh:
root@ubuntu:~/src# asterisk -rvvv
Asterisk 13.17.2, Copyright (C) 1999 - 2014, Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=====
Connected to Asterisk 13.17.2 currently running on ubuntu (pid = 1310)
-- Registered SIP 'dhruv' at 192.168.64.155:50348
[Oct 27 22:51:49] NOTICE[1404]: chan_sip.c:24586 handle_response_peerpoke: Peer 'dhruv' is now Reachable. (44ms / 2000ms)
-- Unregistered SIP 'dhruv'
-- Registered SIP 'dhruv' at 192.168.64.155:50348
[Oct 27 22:51:51] NOTICE[1404]: chan_sip.c:28404 handle_request_subscribe: Received SIP subscribe for peer without mailbox: dhruv
-- Using SIP RTP cos mark 5
-- Executing [100@phones:1] NoOp("SIP/dhruv-00000000", "Call for shivang") in new stack
-- Executing [100@phones:2] Dial("SIP/dhruv-00000000", "SIP/shivang") in new stack
[Oct 27 22:51:58] WARNING[2285][c-00000000]: app_dial.c:2525 dial_exec_full: Unable to create channel of type 'SIP' (cause 20 - Subscriber absent)
-- Everyone is busy/congested at this time (1:0/0/1)
-- Executing [100@phones:3] Hangup("SIP/dhruv-00000000", "") in new stack
-- Span extension (phones, 100, 2) exited non-zero on 'SIP/dhruv-00000000'
-- Registered SIP 'shivang' at 192.168.64.160:61044
[Oct 27 22:53:58] NOTICE[1404]: chan_sip.c:24586 handle_response_peerpoke: Peer 'shivang' is now Reachable. (8ms / 2000ms)
-- Unregistered SIP 'shivang'
-- Registered SIP 'shivang' at 192.168.64.160:61044
[Oct 27 22:54:00] NOTICE[1404]: chan_sip.c:28404 handle_request_subscribe: Received SIP subscribe for peer without mailbox: shivang
-- Registered SIP 'shivani' at 192.168.64.156:61475
[Oct 27 22:54:02] NOTICE[1404]: chan_sip.c:24586 handle_response_peerpoke: Peer 'shivani' is now Reachable. (8ms / 2000ms)
-- Unregistered SIP 'shivani'
-- Registered SIP 'shivani' at 192.168.64.156:61475
[Oct 27 22:54:05] NOTICE[1404]: chan_sip.c:28404 handle_request_subscribe: Received SIP subscribe for peer without mailbox: shivani
-- Using SIP RTP cos mark 5
-- Executing [100@phones:1] NoOp("SIP/dhruv-00000001", "Call for shivang") in new stack
-- Executing [100@phones:2] Dial("SIP/dhruv-00000001", "SIP/shivang") in new stack
-- Using SIP RTP cos mark 5
-- Called SIP/shivang
-- SIP/shivang-00000002 is ringing
-- Span extension (phones, 100, 2) exited non-zero on 'SIP/dhruv-00000001'
-- Using SIP RTP cos mark 5
-- Executing [200@phones:1] NoOp("SIP/dhruv-00000003", "Call for shivani") in new stack
-- Executing [200@phones:2] Dial("SIP/dhruv-00000003", "SIP/shivani") in new stack
-- Using SIP RTP cos mark 5
-- Called SIP/shivani
-- SIP/shivani-00000004 is ringing
-- SIP/shivani-00000004 is ringing
-- Got SIP response 480 'Busy Here' back from 192.168.64.156:61475
-- SIP/shivani-00000004 is busy
-- Everyone is busy/congested at this time (1:1/0/0)
-- Executing [200@phones:3] Hangup("SIP/dhruv-00000003", "") in new stack
-- Span extension (phones, 200, 2) exited non-zero on 'SIP/dhruv-00000003'
-- Using SIP RTP cos mark 5
-- Executing [200@phones:1] NoOp("SIP/shivang-00000005", "Call for shivani") in new stack
-- Executing [200@phones:2] Dial("SIP/shivang-00000005", "SIP/shivani") in new stack
-- Using SIP RTP cos mark 5
```

We can see in the following Wireshark capture, the of user being BUSY.  
From 100-

No.	Time	Source	Destination	Protocol	Length	Info
1	0.42224804	192.168.64.173	192.168.64.155	SIP	667	Request: OPTIONS sip/...
2	0.42904815	192.168.64.173	192.168.64.155	SIP	667	Request: OPTIONS sip/...
3	4.45047020	192.168.64.155	192.168.64.173	SIP	624	Status: 200 OK
4	55.06252804	192.168.64.155	192.168.64.173	SIP	759	Request: REGISTER sip/...
5	55.14170839	192.168.64.173	192.168.64.155	SIP	624	Status: 401 Unauthorized
6	55.13993868	192.168.64.155	192.168.64.173	SIP	759	Request: REGISTER sip/...
7	55.13502972	192.168.64.173	192.168.64.155	SIP	667	Request: OPTIONS sip/...
8	55.13509427	192.168.64.155	192.168.64.173	SIP	677	Status: 200 OK (SIP/2.0)
9	55.15758367	192.168.64.155	192.168.64.173	SIP	624	Status: 200 OK
10	57.48028058	192.168.64.155	192.168.64.173	SIP/SDP	964	Request: INVITE sip/...
11	57.48952408	192.168.64.173	192.168.64.155	SIP	611	Status: 401 Unauthorized
12	57.48742436	192.168.64.155	192.168.64.173	SIP/SDP	380	Request: ACK sip/...
13	57.48958125	192.168.64.155	192.168.64.173	SIP/SDP	1227	Request: INVITE sip/...
14	57.44489247	192.168.64.173	192.168.64.155	SIP	554	Status: 200 OK
15	57.50170382	192.168.64.173	192.168.64.155	SIP/SDP	984	Request: INVITE sip/...
16	57.50148709	192.168.64.173	192.168.64.155	SIP/SDP	984	Request: INVITE sip/...
17	57.78058914	192.168.64.155	192.168.64.173	SIP	353	Status: 300 Trying
18	57.78998723	192.168.64.155	192.168.64.173	SIP	520	Status: 180 Ringing
19	57.78989213	192.168.64.155	192.168.64.173	SIP	520	Status: 180 Ringing
20	57.78988547	192.168.64.173	192.168.64.155	SIP	570	Status: 180 Ringing
21	68.42033320	192.168.64.173	192.168.64.155	SIP	667	Request: OPTIONS sip/...
22	68.42711721	192.168.64.173	192.168.64.155	SIP	667	Request: OPTIONS sip/...
23	68.43128890	192.168.64.155	192.168.64.173	SIP	622	Status: 200 OK
24	68.43151799	192.168.64.155	192.168.64.173	SIP	624	Status: 200 OK
25	81.590819373	192.168.64.155	192.168.64.173	SIP	415	Status: 480 Busy Here
26	81.590807660	192.168.64.173	192.168.64.155	SIP	512	Request: ACK sip/...
27	81.591748053	192.168.64.155	192.168.64.173	SIP	532	Status: 480 Busy Here
28	81.555676796	192.168.64.155	192.168.64.173	SIP	380	Request: ACK sip/...
29	82.861954611	192.168.64.155	192.168.64.173	SIP	759	Request: REGISTER sip/...
30	82.862956330	192.168.64.173	192.168.64.155	SIP	626	Status: 401 Unauthorized
31	82.868131921	192.168.64.155	192.168.64.173	SIP	759	Request: REGISTER sip/...
32	82.868479157	192.168.64.173	192.168.64.155	SIP	667	Request: OPTIONS sip/...
33	82.86852446	192.168.64.173	192.168.64.155	SIP	677	Status: 200 OK (SIP/2.0)
34	82.874027541	192.168.64.155	192.168.64.173	SIP	624	Status: 200 OK



### PHASE-3: Call on Hold

Here the third client, with the client ID 300, calls to the user with ID 100, which is already in a connection with user 200. User 100 then keeps the 200 user on HOLD and establishes a call with ID 300. After completing the call with 300, he (ID#100) continues the call with user 200.

For this part, we register the third client (ID#300) in sip.conf file as described below -

[300]

username=300

type=friend

secret=password

host=dynamic

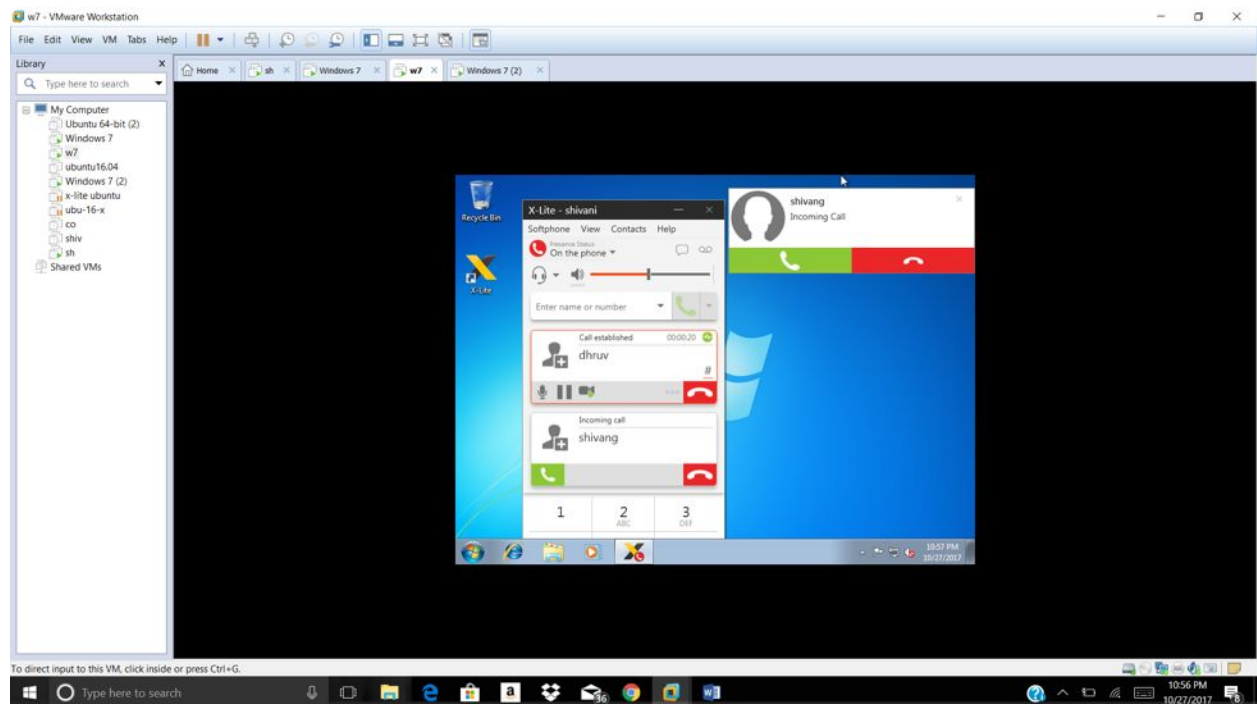
context =from-sip

Also, since we want to establish the call with user (ID#300) we have to add the following commands in the extension.conf file -

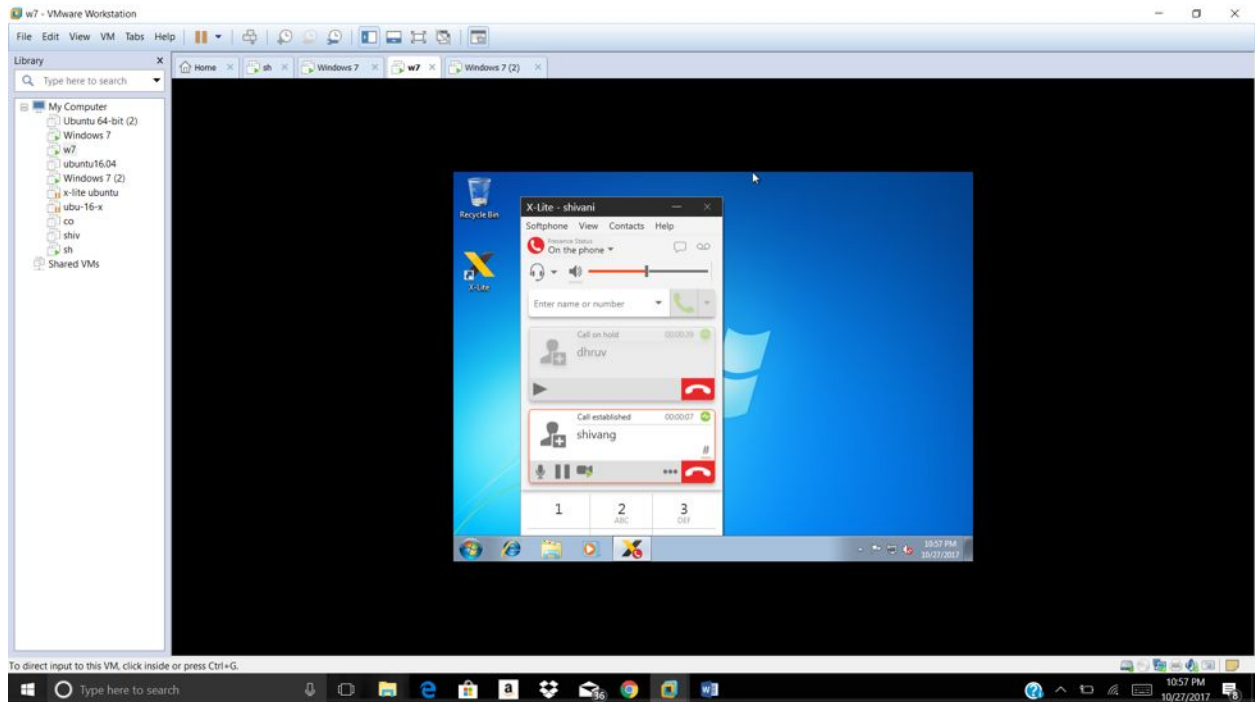
Exten => 300,1,Dial (SIP/300,20)

Exten => 300,2,Hangup

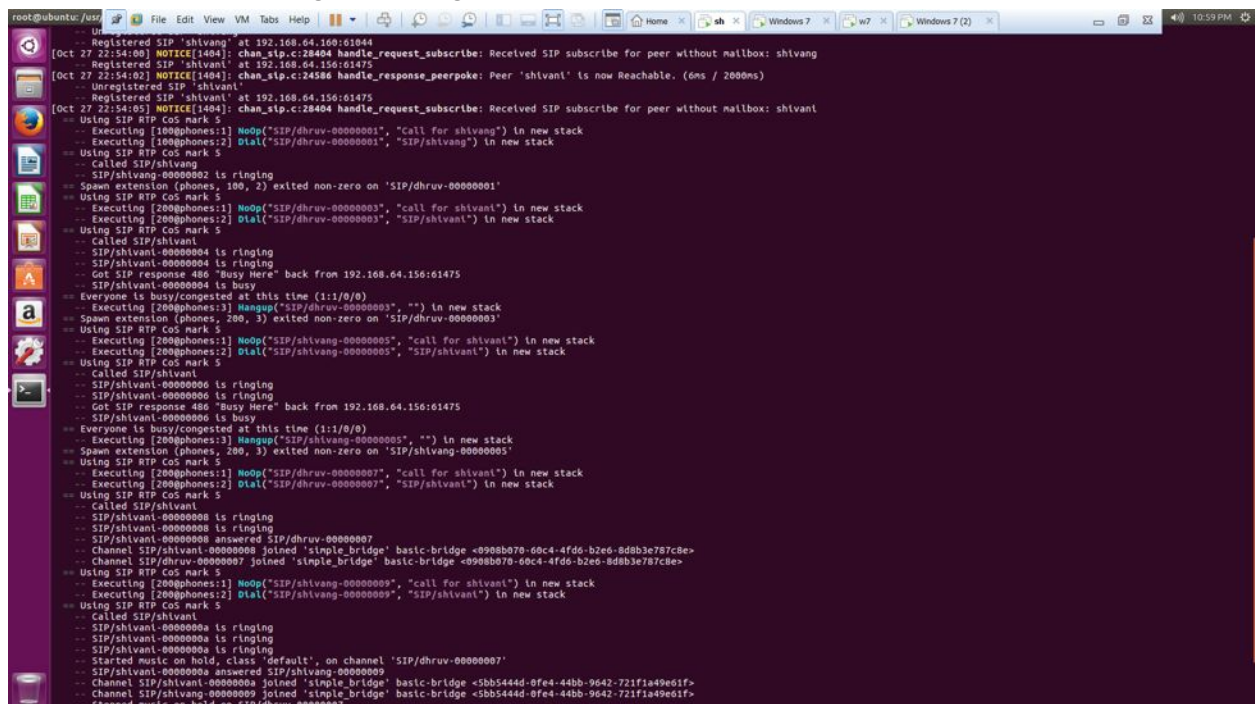
In this image we can see that user 300 is calling user 100 and then the user 100 holds call of user 200 and continues call with 100 once it gets completed.



Call on Hold for Dhruv (User 300), while user 200, established call with user 100-



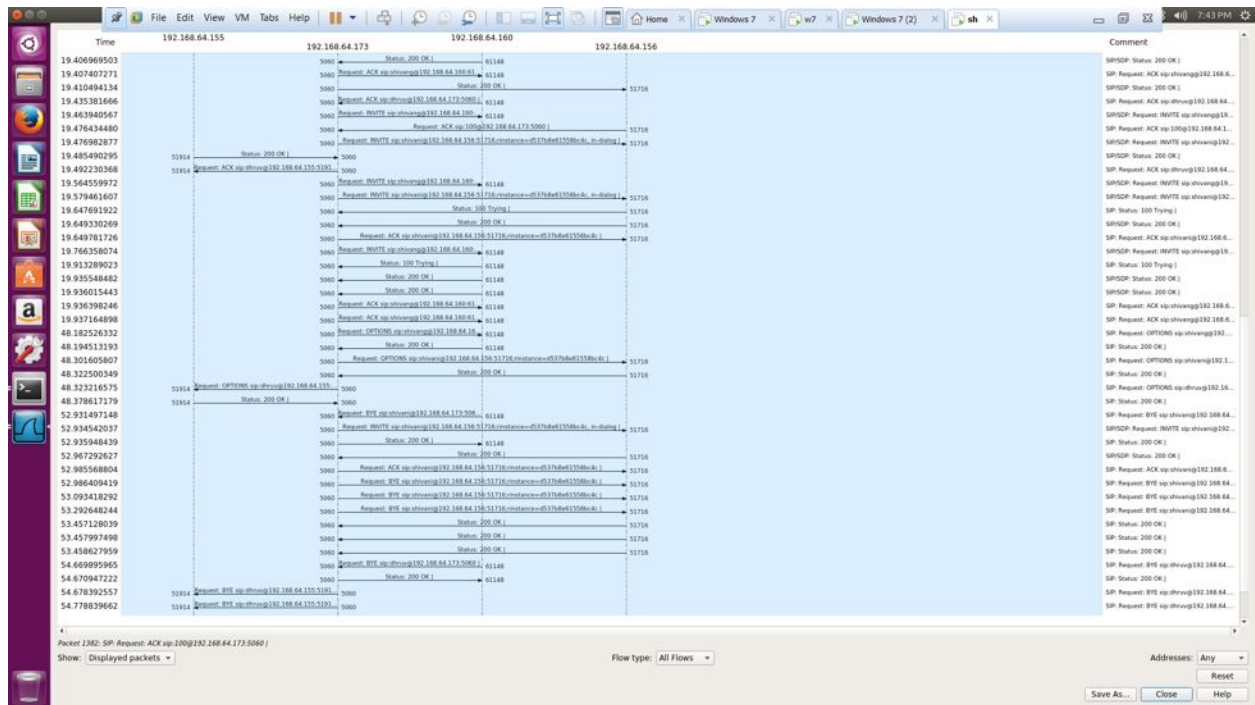
We can also see the working in this image below -



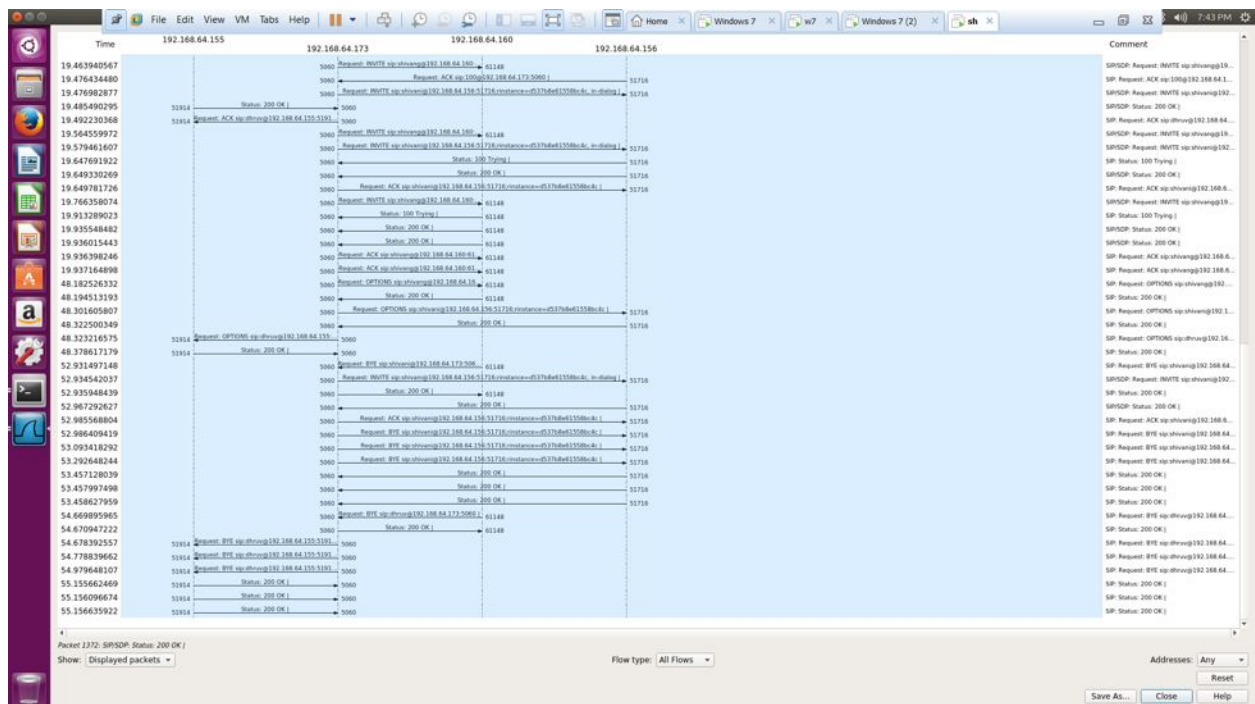




100-



Combined-



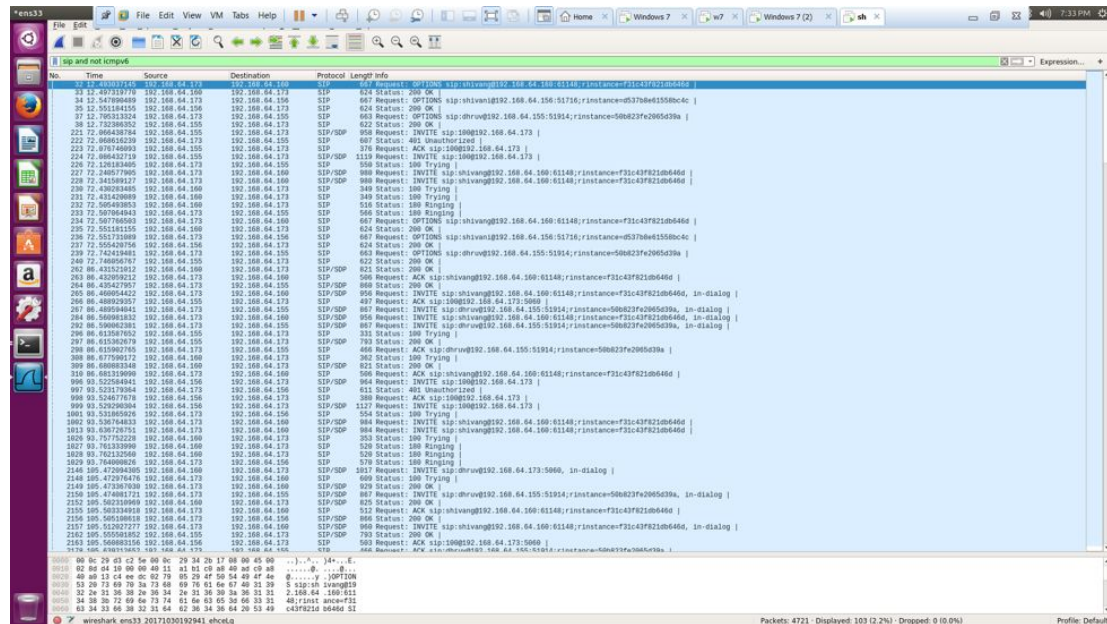
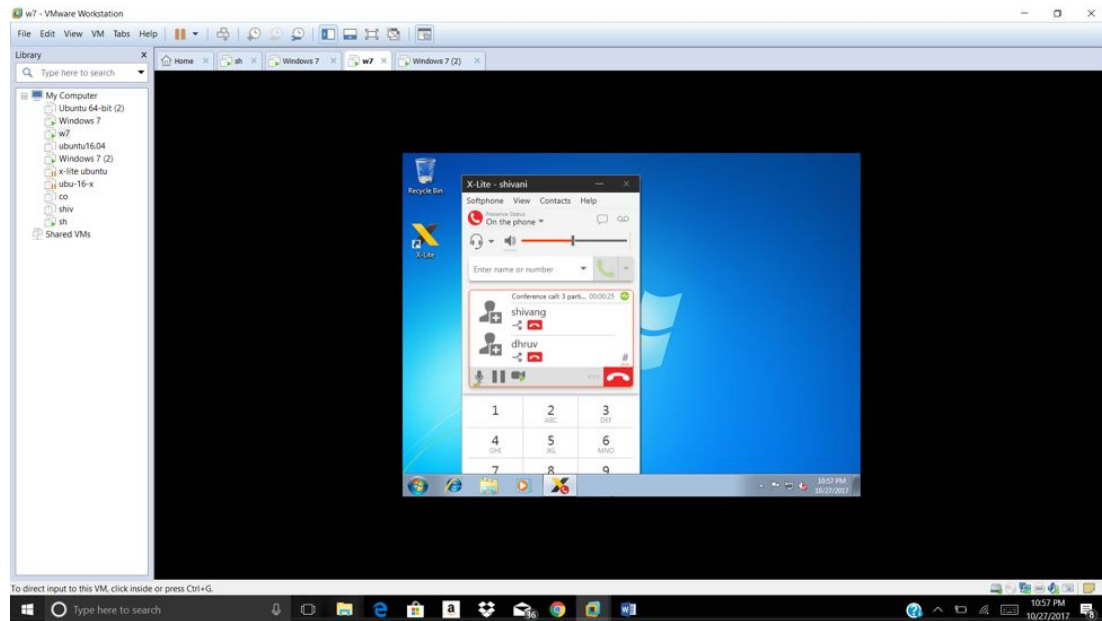
## PHASE - 4: Call Conference

In this Call Conferencing scenario, the user with ID 300 calls user (ID#100), whereas a call is already in progress with two users (IDs#100 and 200). The user 2000, puts the call by 200 on Hold and establishes the call with user 300. Once the call has been set up between them, user again invites the other user (ID#200) to join the call. So, a conferencing call starts in between the users with IDs # 100, 200,300.

```
root@ubuntu:/usr#
[Oct 27]
-- Using SIP RTP Ccs mark 5
-- Executing [100gphones:1] NoOp("SIP/dhruv-00000001", "Call for shivang") in new stack
-- Executing [100gphones:2] Dial("SIP/dhruv-00000001", "SIP/shivang") in new stack
-- Using SIP RTP Ccs mark 5
-- Called SIP/shivang
-- SIP/shivang-00000002 is ringing
-- SIP/shivang-00000002 exited non-zero on 'SIP/dhruv-00000001'
-- Spawn extension (phones, 100, 2) exited non-zero on 'SIP/dhruv-00000001'
-- Using SIP RTP Ccs mark 5
-- Executing [200gphones:1] NoOp("SIP/dhruv-00000003", "call for shivani") in new stack
-- Executing [200gphones:2] Dial("SIP/dhruv-00000003", "SIP/shivani") in new stack
-- Using SIP RTP Ccs mark 5
-- Called SIP/shivani
-- SIP/shivani-00000004 is ringing
-- SIP/shivani-00000004 is ringing
-- Got SIP response 486 "Busy Here" back from 192.168.64.156:61475
-- SIP/shivani-00000004 is busy
-- Everyone is busy/congested at this time (1:1/0/0)
-- Executing [200gphones:3] Hangup("SIP/dhruv-00000003", "") in new stack
-- Spawn extension (phones, 200, 3) exited non-zero on 'SIP/dhruv-00000003'
-- Using SIP RTP Ccs mark 5
-- Executing [200gphones:1] NoOp("SIP/shivang-00000005", "call for shivani") in new stack
-- Executing [200gphones:2] Dial("SIP/shivang-00000005", "SIP/shivani") in new stack
-- Using SIP RTP Ccs mark 5
-- Called SIP/shivani
-- SIP/shivani-00000006 is ringing
-- SIP/shivani-00000006 is ringing
-- Got SIP response 486 "Busy Here" back from 192.168.64.156:61475
-- SIP/shivani-00000006 is busy
-- Everyone is busy/congested at this time (1:1/0/0)
-- Executing [200gphones:3] Hangup("SIP/shivang-00000005", "") in new stack
-- Spawn extension (phones, 200, 3) exited non-zero on 'SIP/shivang-00000005'
-- Using SIP RTP Ccs mark 5
-- Executing [200gphones:1] NoOp("SIP/dhruv-00000007", "call for shivani") in new stack
-- Executing [200gphones:2] Dial("SIP/dhruv-00000007", "SIP/shivani") in new stack
-- Using SIP RTP Ccs mark 5
-- Called SIP/shivani
-- SIP/shivani-00000008 is ringing
-- SIP/shivani-00000008 is ringing
-- SIP/shivani-00000008 answered SIP/dhruv-00000007
-- Channel SIP/shivani-00000008 joined 'simple_bridge' basic-bridge <0908b070-60c4-4fd6-b2e6-8dbb3e787c8e>
-- Channel SIP/dhruv-00000007 joined 'simple_bridge' basic-bridge <0908b070-60c4-4fd6-b2e6-8dbb3e787c8e>
-- Using SIP RTP Ccs mark 5
-- Executing [200gphones:1] NoOp("SIP/shivang-00000009", "call for shivani") in new stack
-- Executing [200gphones:2] Dial("SIP/shivang-00000009", "SIP/shivani") in new stack
-- Using SIP RTP Ccs mark 5
-- Called SIP/shivani
-- SIP/shivani-00000009 is ringing
-- SIP/shivani-00000009 is ringing
-- SIP/shivani-00000009 is ringing
-- Started music on hold, class 'default', on channel 'SIP/dhruv-00000007'
-- SIP/shivani-00000009 answered SIP/shivang-00000009
-- Channel SIP/shivani-00000009 joined 'simple_bridge' basic-bridge <5bb5444d-0fe4-44bb-9642-721fa49e61f>
-- Channel SIP/shivang-00000009 joined 'simple_bridge' basic-bridge <5bb5444d-0fe4-44bb-9642-721fa49e61f>
-- Stopped music on hold on SIP/dhruv-00000007
-- Channel SIP/shivani-00000009 left 'native_rtp' basic-bridge <5bb5444d-0fe4-44bb-9642-721fa49e61f>
-- Channel SIP/shivang-00000009 left 'native_rtp' basic-bridge <5bb5444d-0fe4-44bb-9642-721fa49e61f>
-- Spawn extension (phones, 200, 2) exited non-zero on 'SIP/shivang-00000009'
-- Channel SIP/shivani-00000008 left 'native_rtp' basic-bridge <0908b070-60c4-4fd6-b2e6-8dbb3e787c8e>
-- Channel SIP/dhruv-00000007 left 'native_rtp' basic-bridge <0908b070-60c4-4fd6-b2e6-8dbb3e787c8e>
-- Spawn extension (phones, 200, 2) exited non-zero on 'SIP/dhruv-00000007'
ubuntu@cli:~$
```



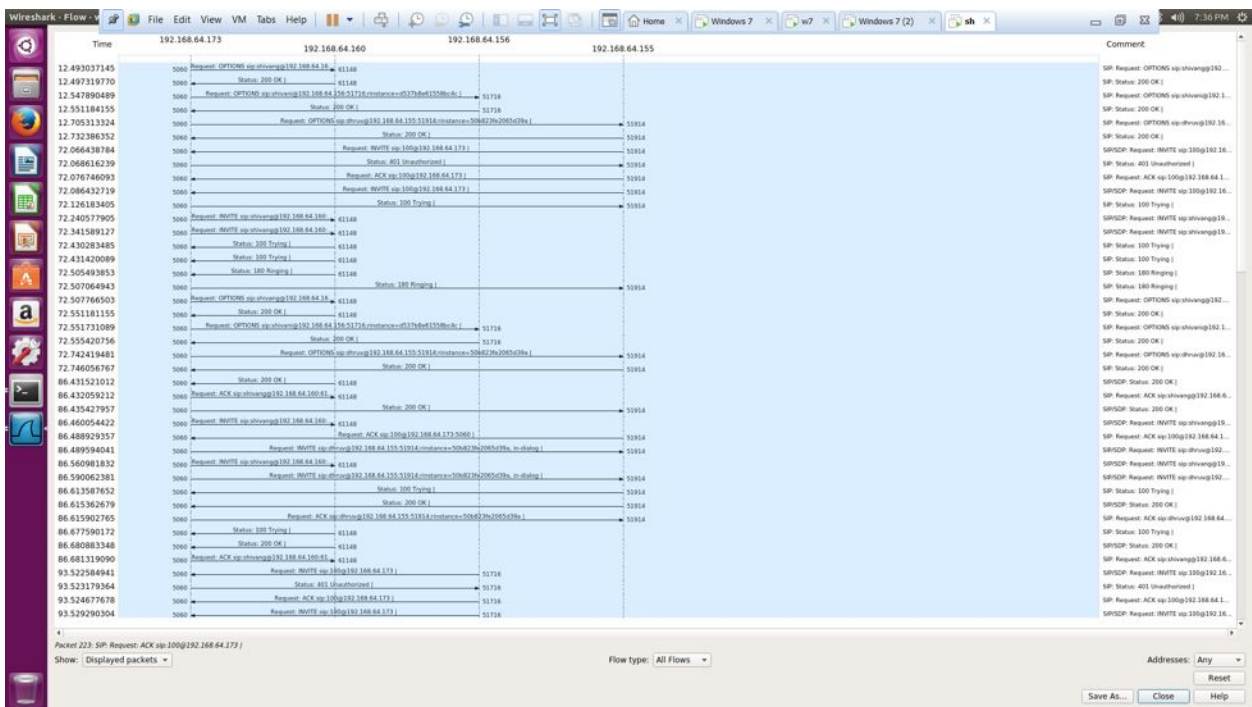
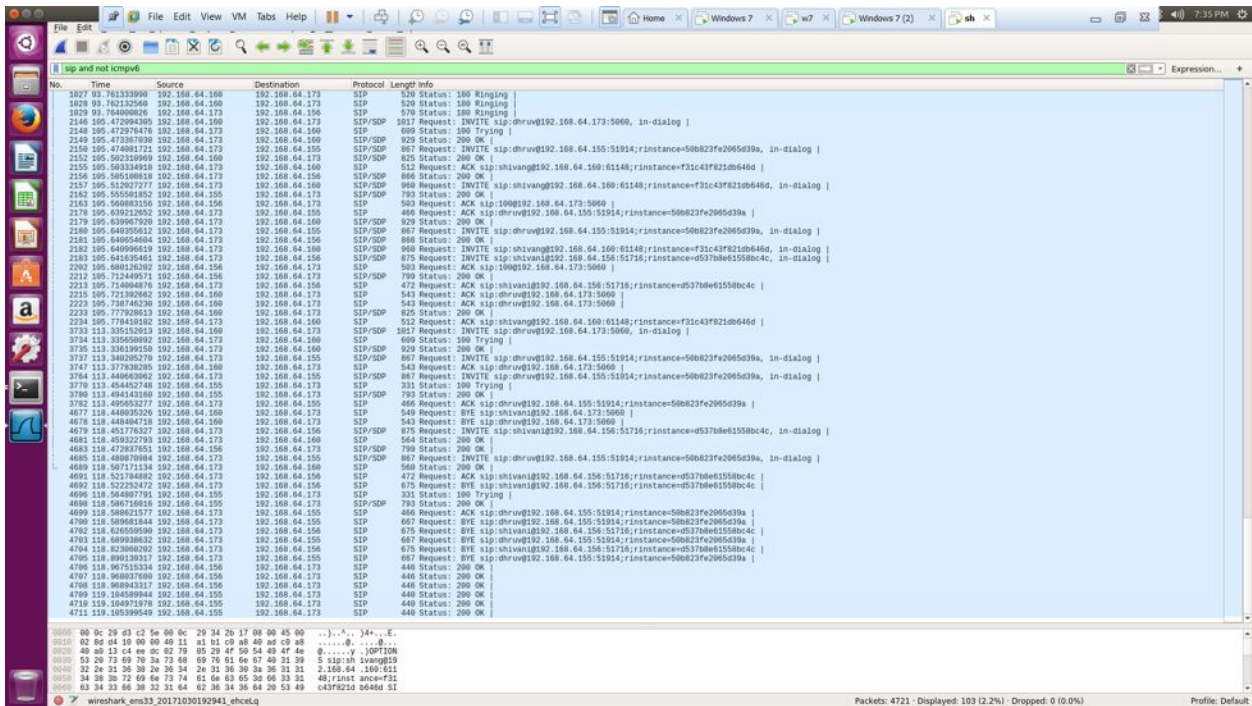
## Call conferencing using all three users -





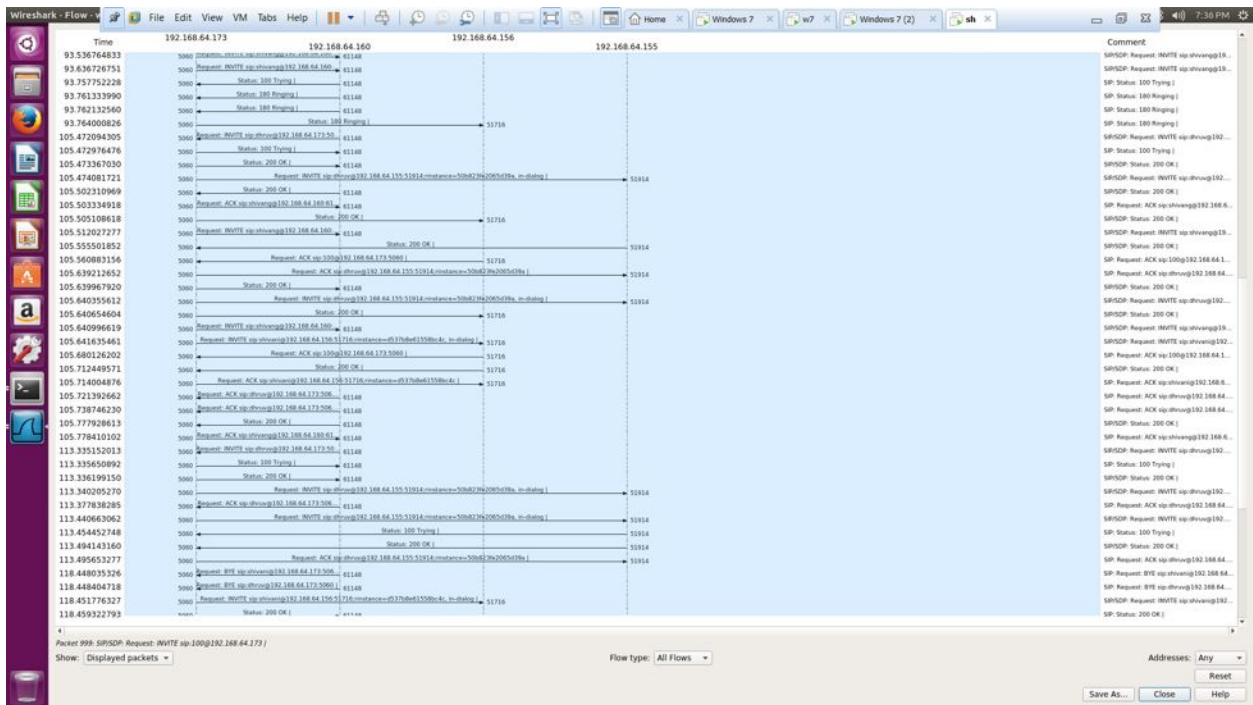


## Only SIP files-

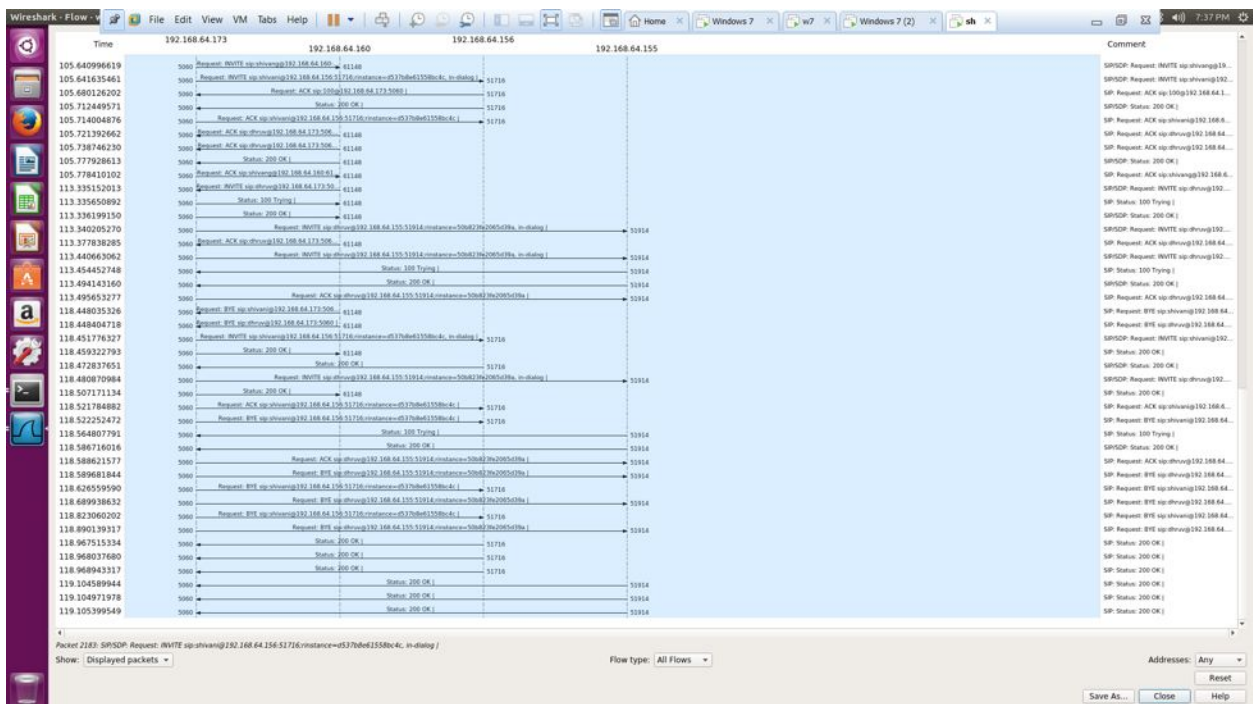




192.168.64.173 multicasts to all



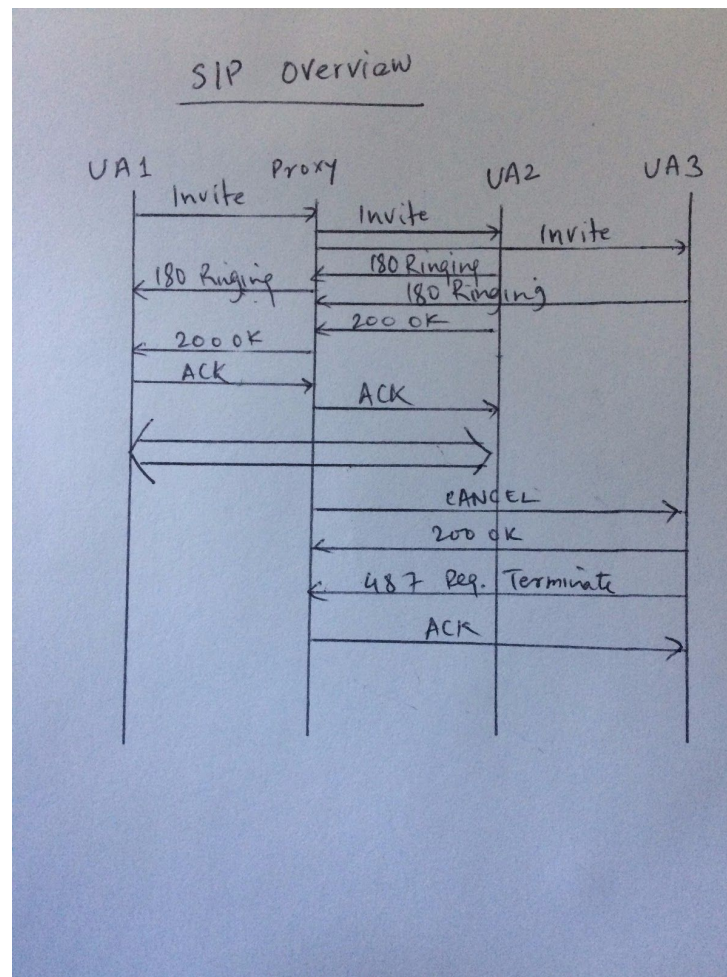
192.168.64.173 to 192.168.64.156 via 192.168.64.160



## PART - 2

### SIP Overview-

Session Initiation Protocol (SIP) is used to establish the session between two parties. It is a communication and media exchange protocol. It is an application as well as session layer protocol which varies with the functionality it performs at an instance. It works in corporate with other application layer protocols like H323. Three way handshaking can be seen here in the following figure -



## **PJSIP -**

- It is a free and open source multimedia communication library which is written in C language implementing standard based protocol such as SIP, SDP, RTP, ICE and others.
- It is both compact and feature-full. It supports all video, audio and instant messaging facilities.

The parts of our script, requiring special attention as described as below-

### **1. Lib Class-**

- This is the most most library class. It needs to be initialized only one single time in the entire program from the point we start the library.
- It is used to create other important objects like transport and accounts.
- We have created the callback class as “log\_cb”.

```
# Creating the library instance of Lib Class
lib = pj.Lib()
# Instantiate library with default configuration
lib.init(log_cfg = pj.LogConfig(level=3, callback=log_cb))
```

### **2. Call Class-**

- A call class is created to do two things here -
  - a. To Answer a call
  - b. For hanging up the call

After successful registration, the PJSIP client attempts to send the INVITE message to the server to call the client in the argument. We can notice that if “call” instance is created successfully then the INVITE is sent to the server with which it is registered. “SRDialCallBack()” is called here to get the notifications on the change of state of events of call.

Here ahead of that the program will wait for the user to exit on the input line and if user gives the ENTER command, program will be exited and main class instance will be destroyed and hence successful call will be completed.

```
# Start calling process
b=raw_input("Enter destination URI: ")
call = acc.make_call(b, SRDialCallback())
```

### **3. Callback Classes -**

We use Callback classes to get the notification. There are various callback classes such as Account Callback class, call callback class and many more to handle the relevant methods and install the necessary callback instance to the object.

```

# To retrieve events from the call
class SRDialCallback(pj.DialCallback):
    def __init__(self, call=None):
        pj.DialCallback.__init__(self, call)

    def on_state(self):
        print("Call is ON :", self.call.info().state_text),
        print ("Last code is :", self.call.info().last_code),
        print ("(" + self.call.info().last_reason + ")")

# Notification whenever there is a change in media state
def on_media_state(self):
    global lib
    if self.call.info().media_state == pj.MediaState.ACTIVE:

        #Connecting the call to a different sound device
        call_slot = self.call.info().conf_slot
        lib.conf_connect(call_slot, 0)
        lib.conf_connect(0, call_slot)
        print ("Hi. How are you doing today???)
        print (lib)

# Main Part

```

## Python-Client Code :

```

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help
For 284 Project - untitled3 - [~/PycharmProjects/untitled3]
No Python interpreter configured for the project
1 import sys
2 import pjsua as pj
3
4 LOG_LEVEL=3
5 current_call = None
6
7 # For printing the log
8 def log_callback(level, string, len):
9     print string
10
11 # To get the notifications for Account Registration
12 class MyAccCallback(pj.AccountCallback):
13     def __init__(self, acc=None):
14         pj.AccountCallback.__init__(self, acc)
15     def on_reg_state(self):
16         if self.ses:
17             if self.call.info().reg_status >= 200:
18                 self.ses.release()
19
20 # Incoming call
21 def on_incoming_call(self, call):
22     global current_call
23     if current_call:
24         call.answer(486, "Busy")
25     return
26
27 print "Incoming call from: ", call.info().remote_uri
28 print "Type to answer"
29
30 current_call = call
31
32 call_cb = MyAccCallback(current_call)
33 current_call.set_callback(call_cb)
34
35 #For Ringing
36 current_call.answer(180)
37
38
39
40 # To retrieve events from the call
41 class SRDialCallback(pj.DialCallback):
42     def __init__(self, call=None):
43         pj.DialCallback.__init__(self, call)
44
45     def on_state(self):
46         print("Call is ON :", self.call.info().state_text),
47         print ("Last code is :", self.call.info().last_code),
48         print ("(" + self.call.info().last_reason + ")")
49
50 # Notification whenever there is a change in media state
51 def on_media_state(self):
52     global lib
53     if self.call.info().media_state == pj.MediaState.ACTIVE:
54
55         #Connecting the call to a different sound device
56         call_slot = self.call.info().conf_slot
57         lib.conf_connect(call_slot, 0)
58         lib.conf_connect(0, call_slot)
59         print ("Hi. How are you doing today???)
60         print (lib)
61
62 # Main Part
63 on_incoming_cal...

```

g: TODO Python Console Terminal  
 Error running 'scratch\_11': @HostNull method com.intellij.executor.configurations.GeneralCommandLine.getExePath must not return null (a minute ago)  
 22:33 LF: UTF-8



```
untitled3 For 284 Project scratch_11
No Python interpreter configured for the project
Configure Python Interpreter

# Main Part
def make_call(uri):
    try:
        print "making call to", uri
        return acc.make_call(uri, cb=MyAccCallback())
    except pj.Error, e:
        print "exception: " + str(e)
        return None

# Creating the library instance of Lib Class
lib = pj.Lib()

# Instantiate library with default configuration
lib.init(log_cfg = pj.LogConfig(level=LOG_LEVEL, callback=log_cb))

# Configure one Transport Object and fixing it for listening to 5060 port and UDP protocol
trans_conf = pj.TransportConfig()

print "-----REGISTRATION BELOW-----"
print "\n\n"
trans_conf.port = 5060
a=raw_input("Enter the IP Address of your client : ")
print "We have our Default port number 5060 for SIP"
trans_conf.bound_addr = a
transport = lib.create_transport(pj.TransportType.UDP, trans_conf)
# Initiate the instance for Library Class
lib.start()

#When no sound card is found
lib.set_null_snd_dev()

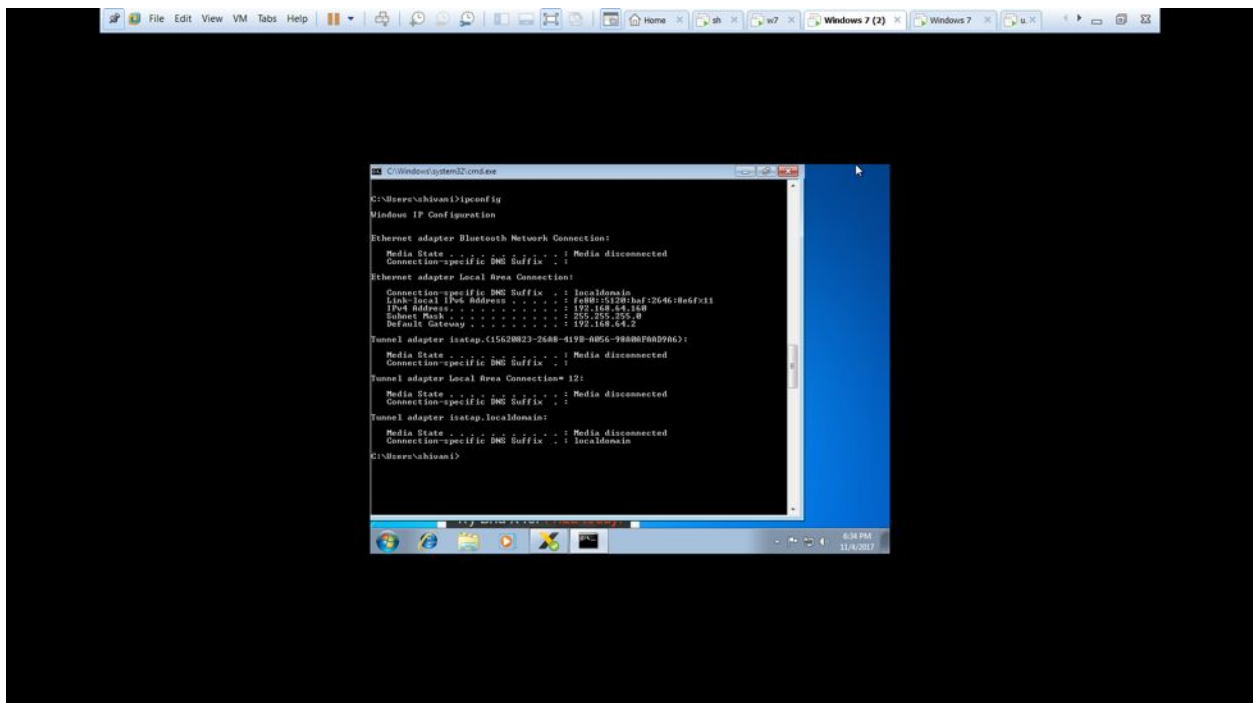
#Configuring account class to register with registrar server
#Giving info to create header of REGISTER SIP message
ab4=raw_input("Enter the IP address of your server: ")
ab=raw_input("Enter username: ")
ab1=raw_input("Enter password: ")
ab2=raw_input("Do you want to use same Display name and username ? Enter valid entry Y/N ?")
if ab2=="Y" or ab2=="y":
    ab3=ab
else:
    ab3=raw_input("Enter display name: ")
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 + ':5060'
acc_callback = MyAccCallback(acc_conf)
acc = lib.create_account(acc_conf, cb=acc_callback)
#Creating instance of AccCallback Class
acc.set_callback(acc_callback)
print "\n\n"
print "Registration is completed"
print("Status: ", acc.info().reg_status, "\n")
print("Reason: ", acc.info().reg_reason, "\n")
on_incoming_call...
```

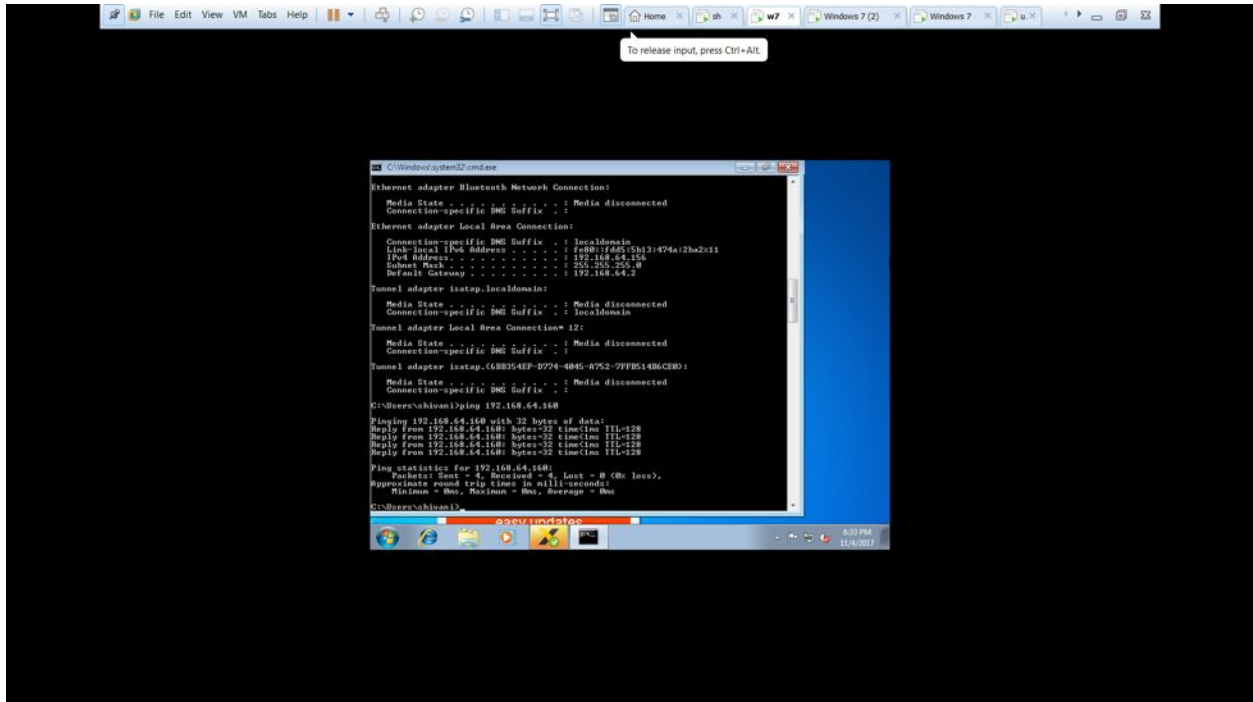
```
untitled3 For 284 Project scratch_11
No Python interpreter configured for the project
Configure Python Interpreter

ab4=raw_input("Enter the IP address of your server: ")
ab=raw_input("Enter username: ")
ab1=raw_input("Enter password: ")
ab2=raw_input("Do you want to use same Display name and username ? Enter valid entry Y/N ?")
if ab2=="Y" or ab2=="y":
    ab3=ab
else:
    ab3=raw_input("Enter display name: ")
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 + ':5060'
acc_callback = MyAccCallback(acc_conf)
acc = lib.create_account(acc_conf, cb=acc_callback)
#Creating instance of AccCallback Class
acc.set_callback(acc_callback)
print "\n\n"
print "Registration is completed"
print("Status: ", acc.info().reg_status, "\n")
print("Reason: ", acc.info().reg_reason, "\n")
ab5=raw_input("Do you want to make a call now ?? Y/N/n")
print "\n\n"
if ab5=="Y" or ab5=="y":
    # Start calling process
    b=raw_input("Enter destination URI: ")
    call = acc.make_call(b, SRDialCallback())
    #Client Side Waiting for ENTER Command to exit
    print ("Press <ENTER> to exit and destroy library")
    input = sys.stdin.readline().rstrip("\n")
    # To shut down the library
    lib.destroy()
    lib = None
else:
    print "Unregistering"
    time.sleep(2)
    print "Destroying the libraries"
    time.sleep(2)
    lib.destroy()
    lib = None
    sys.exit(1)
except pj.Error, e:
    print ("Exception: " + str(e))
    lib.destroy()
    lib = None
    sys.exit(1)
on_incoming_call...
```

# MoS VALUE CALCULATION

- Mean Opinion Score (MoS) is a measure which is used representing the overall quality of a system.
  - MoS value is the result of underlying network attributes and is extremely useful in accessing the call quality.
  - It is a commonly used measure for the quality of audio, video and audiovisual entities.
- In our experiment we have pinged two users and then calculated the MoS value.
- Also since we are pinging through a LAN connection, so the packet loss is Zero or near to zero, which means a perfect transmission over the connection to the two entities.





- Here we have pinged 192.168.64.168 with 32 bytes of data and we see that-

Total number of packets sent = 4

The number of packet received = 4

LOSS = 0 (0% loss)

So the MoS becomes= **5** (max.) and the label is **EXCELLENT**

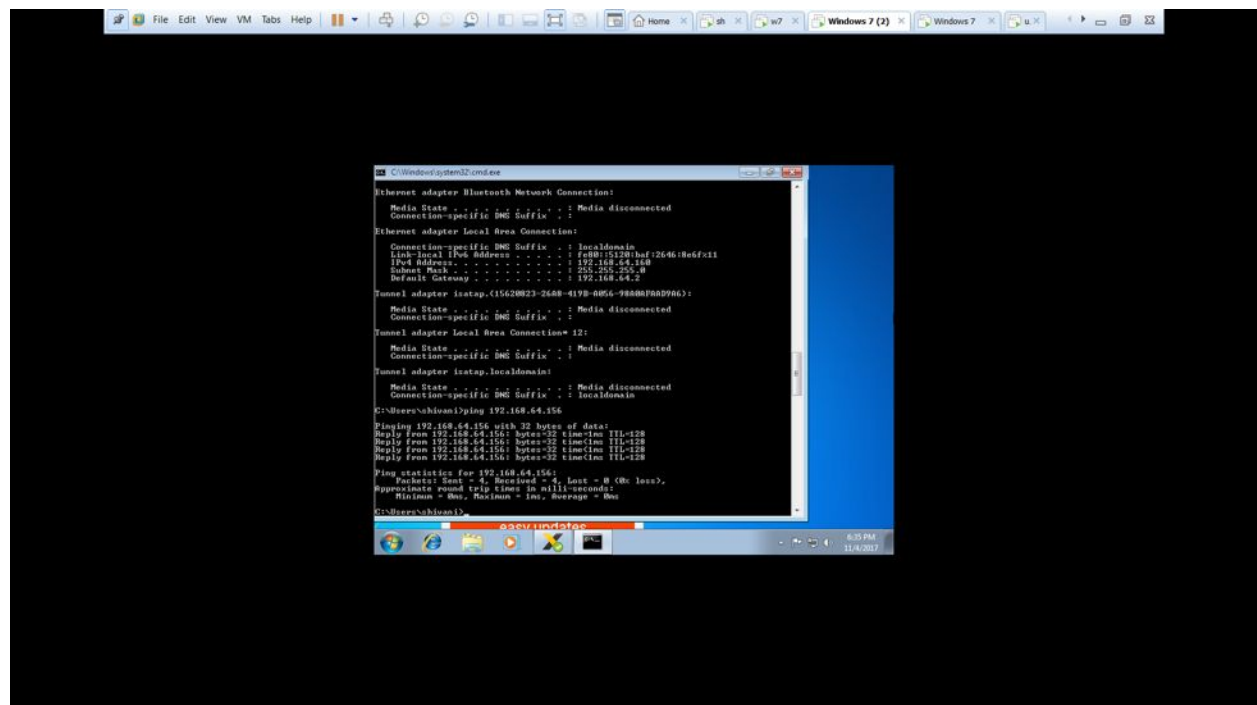
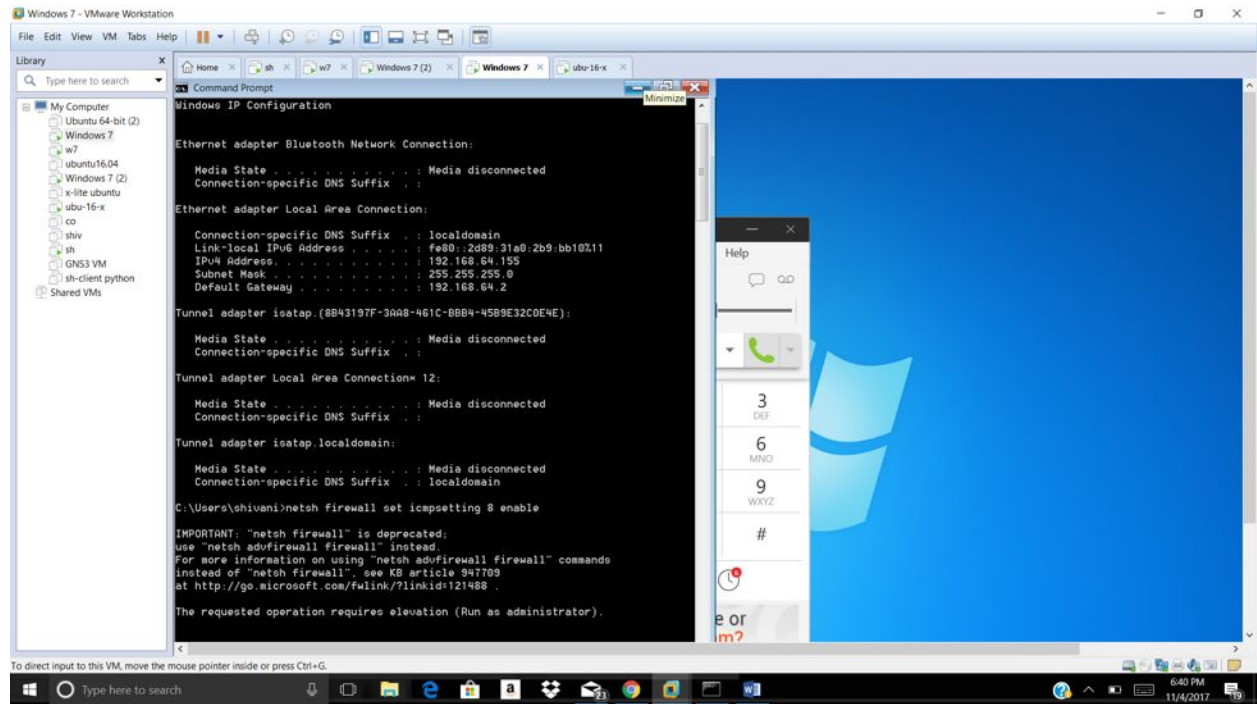
- In this one, I have pinged 192.168.64.156 and see that -

Total number of packets sent = 4

The number of packet received = 4

LOSS = 0 (0% loss)

So the MoS becomes= **5** (max.) and the label is **EXCELLENT**



This is how we pinged and calculated the MoS value and found our communication to be with the maximum rating of **5** as “**EXCELLENT**”, as defined by the parameters given by ITU- Telecommunication Standardization Sector (ITU-T).