A Hands on SIP based VoIP Experiments on:Call Establishment, Busy Lines, Call on Hold and Conference Calling

RYMA NAITAMARA

7 juillet 2018

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.

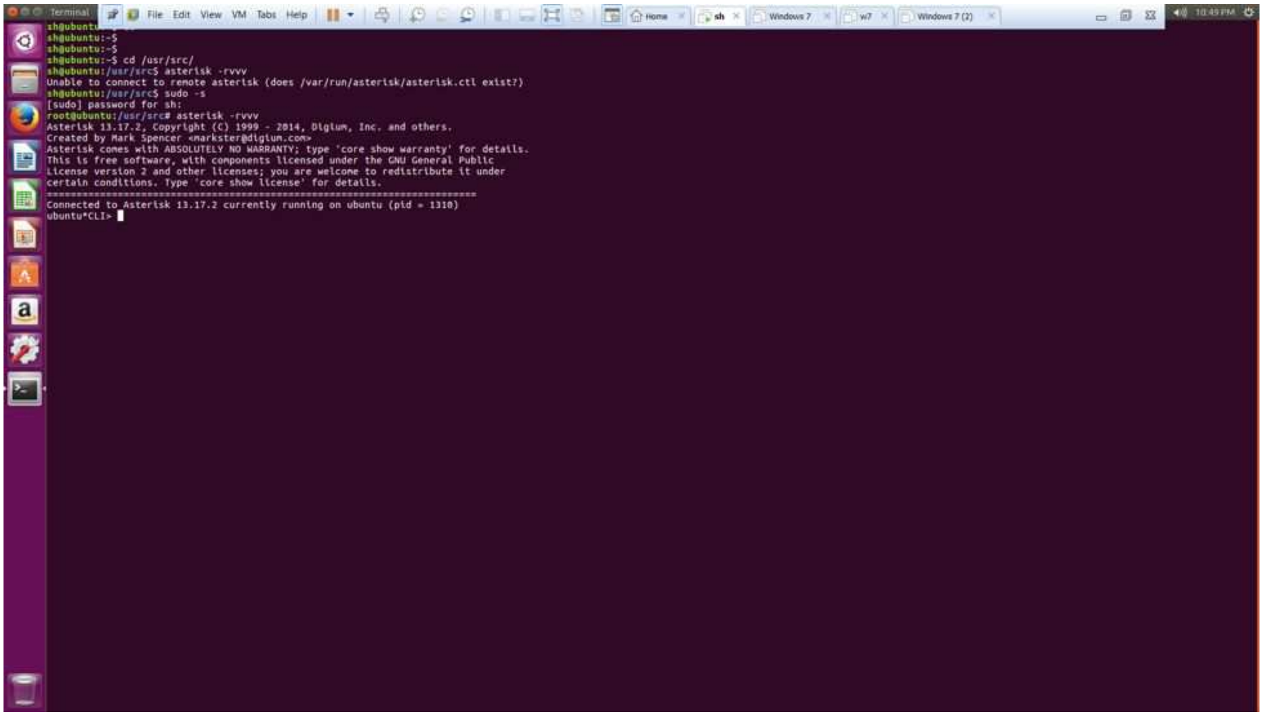
Server - The asterisk server is installed on the server machine with the sip.conf and extension.conf files already within them. These files are used for registration of sip clients on the server machine and the call set up.

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

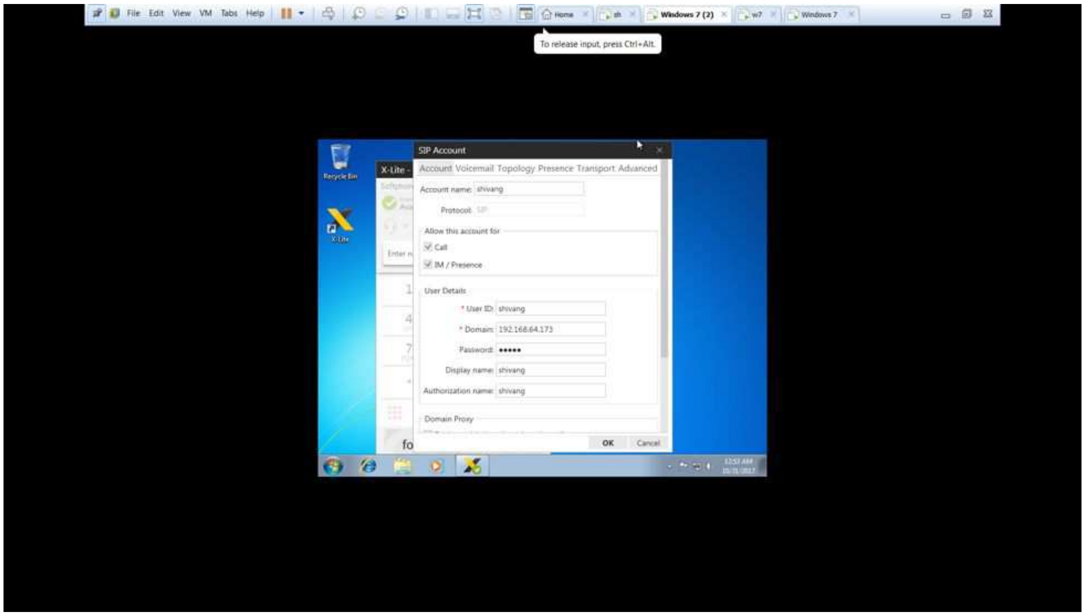
[200] username=200
type=friend
secret=password
hots=dynamic
context=from-sip

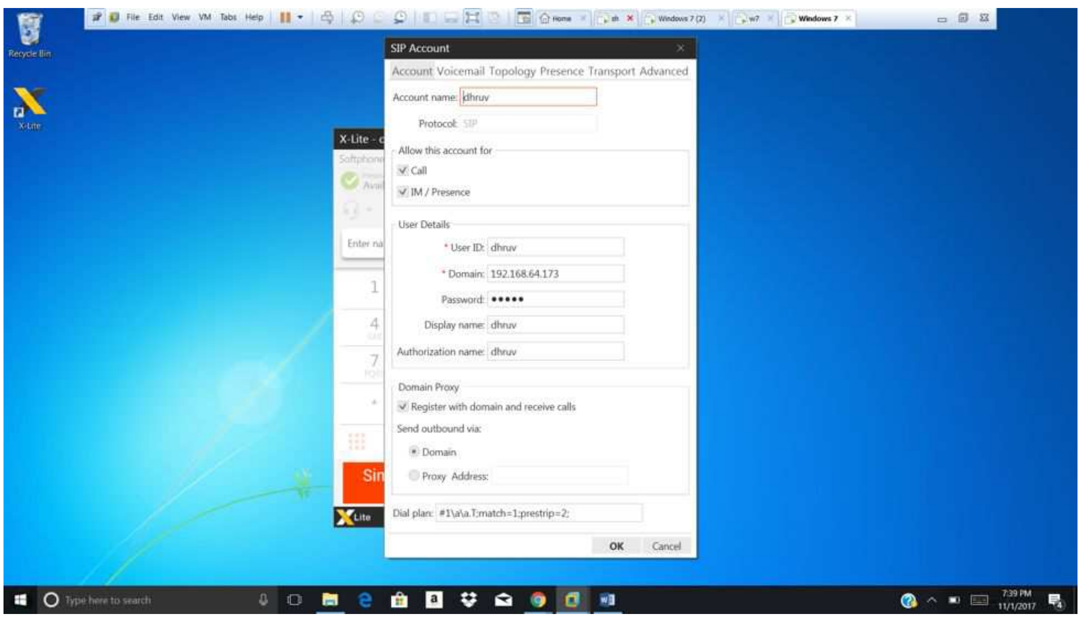


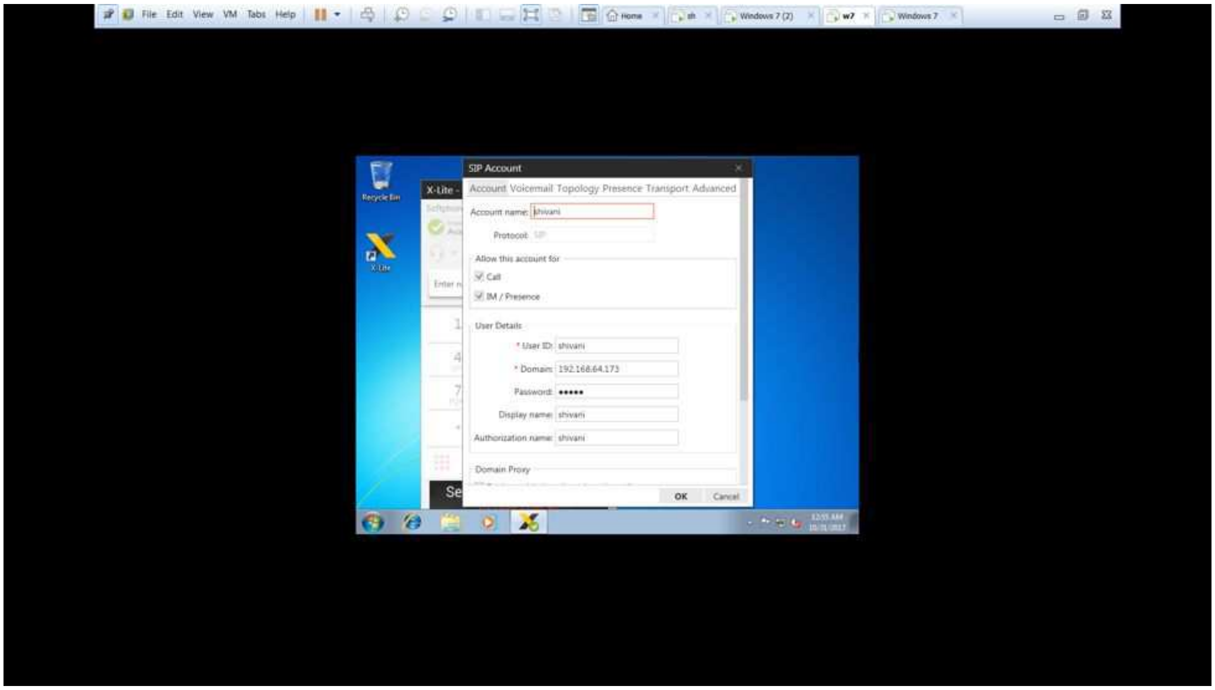
<u>Clients</u> - The X-Lite Softphone software is installed on the to client machines, which is VoIP smartphone and uses SIP (session initiation protocol). One client is assigned with an user ID of 100 and the other is done the same with ID 200.

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 -


To release input, press Ctrl+Alt.
SIP Account
Account Voicemail Topology Presence Transport Advanced
Account name: shivang
Protocol: SIP
Allow this account for
Call
IM / Presence
User Details
* User ID: shivang
* Domain: 192.168.64.173
Password: •••••
Display name: shivang
Authorization name: shivang
Domain Proxy
OK
Cancel


SIP Account
Account Voicemail Topology Presence Transport Advanced
Account name: dhruv
Protocol: SIP
Allow this account for
Call
IM / Presence
User Details
* User ID: dhruv
* Domain: 192.168.64.173
Password: •••••
Display name: dhruv
Authorization name: dhruv
Domain Proxy
Register with domain and receive calls
Send outbound via:
Domain
Proxy Address:
Dial plan: #1\a\a.T;match=1;prestrip=2;
OK
Cancel
Type here to search
7:39 PM
11/1/2017


File Edit View VM Tabs Help
Home
Windows 7 (2)
w7
Windows 7
SIP Account
Account Voicemail Topology Presence Transport Advanced
Account name: shivani
Protocol: SIP
Allow this account for
Call
IM / Presence
User Details
* User ID: shivani
* Domain: 192.168.64.173
Password: •••••
Display name: shivani
Authorization name: shivani
Domain Proxy
OK
Cancel
Recycle Bin
X-Lite

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

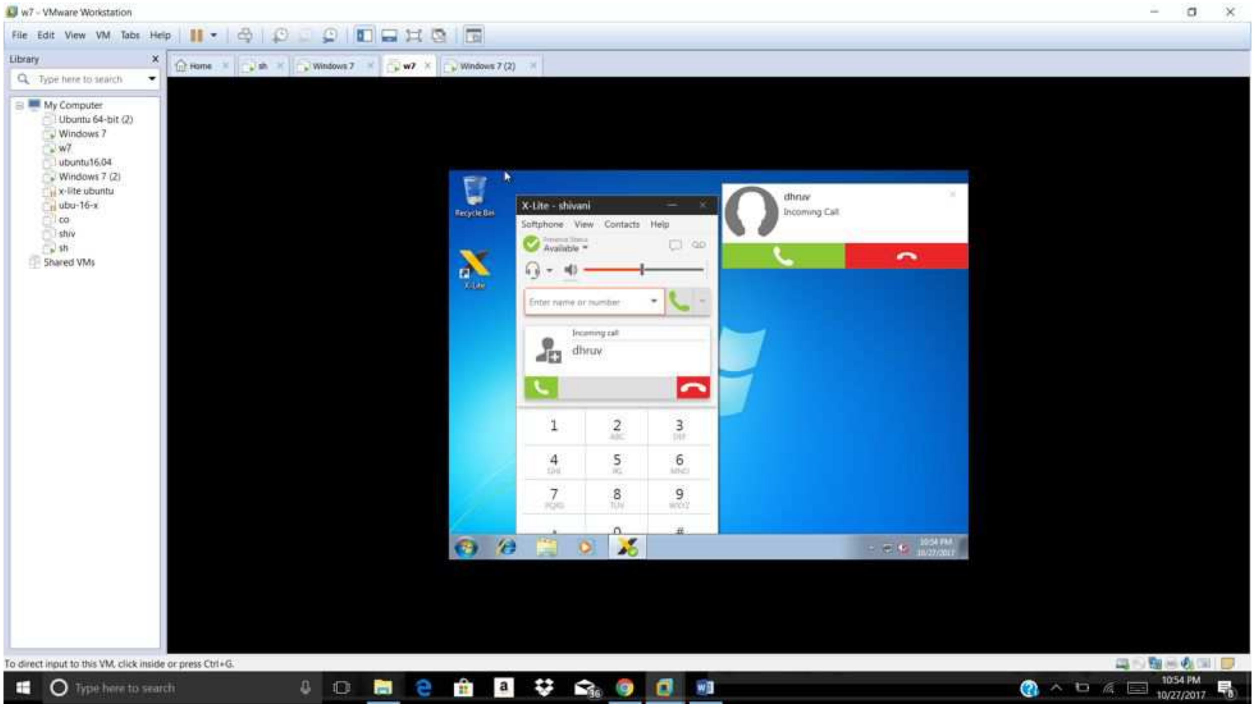
After modified the extension file, we reloaded the asterisk server to save the changes which we made. This is done by typing the reload command. We now call one client to another by dialing the user id and also capture the packets through Wireshark. Here we are calling from client with id 100 to 200.

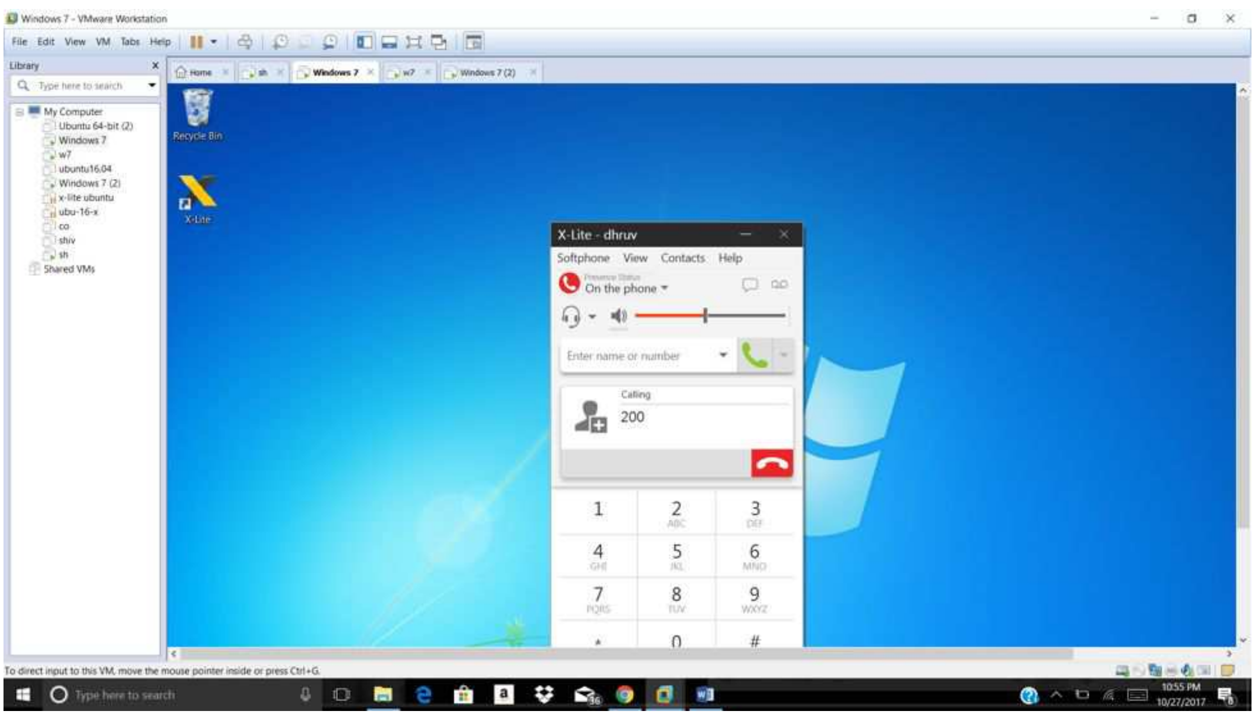
Here in our project, we have,

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

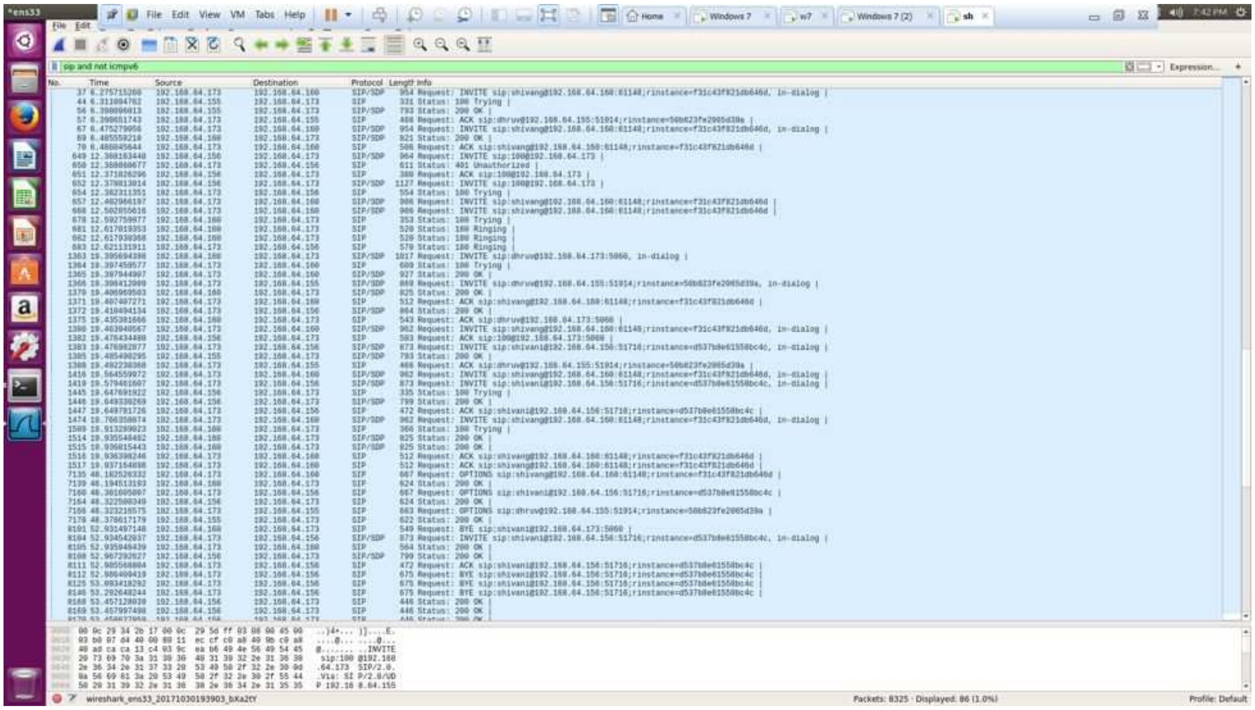
Experiment Result : Call was successfully established between two clients.

Calling 200 from 300-





The Wireshark Image will be the following one -



PHASE-2 : Busy User

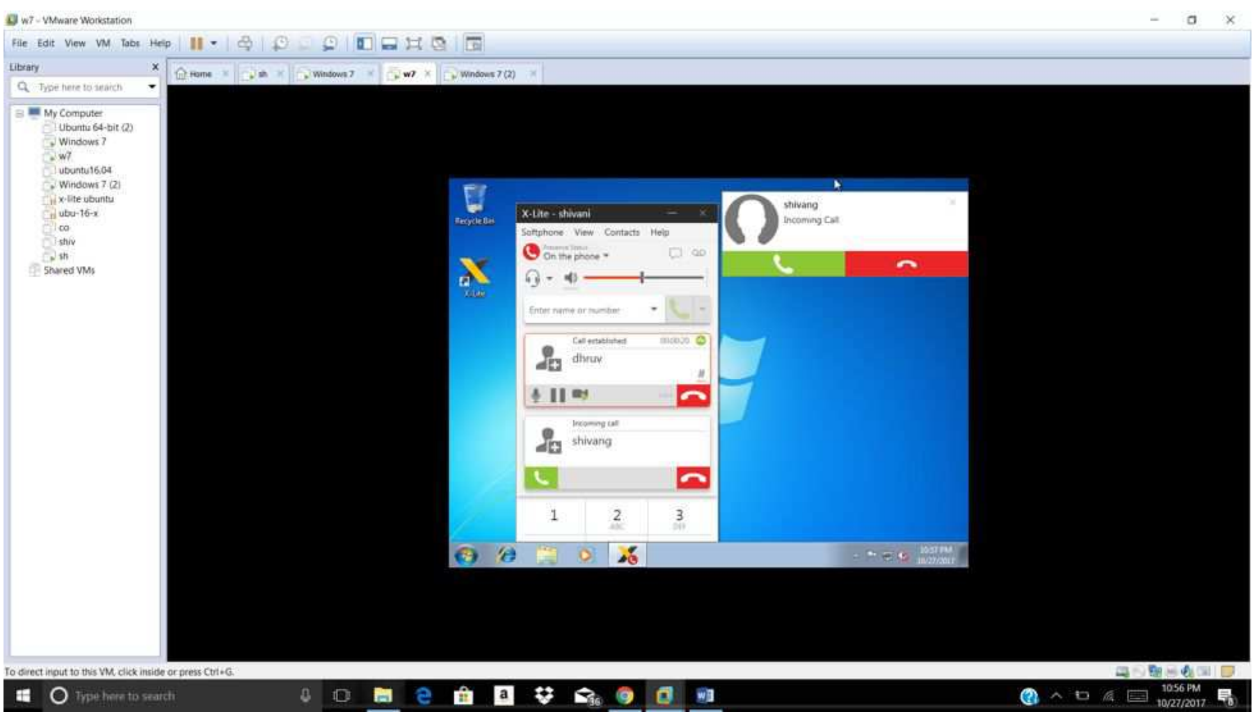
In this mode of operation, one user tries to call the other user, but the other user is busy. The time duration for which the user wants to set the busy mode is sent as an argument in the extension.conf file.

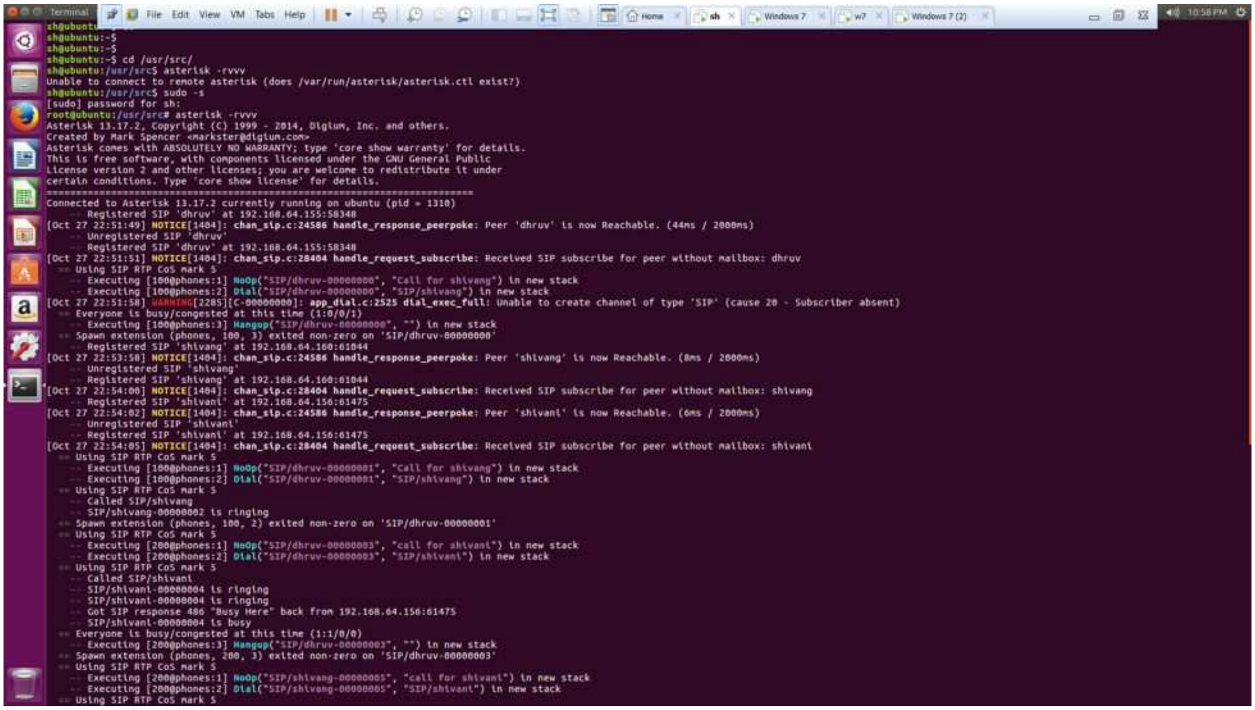
Steps -

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

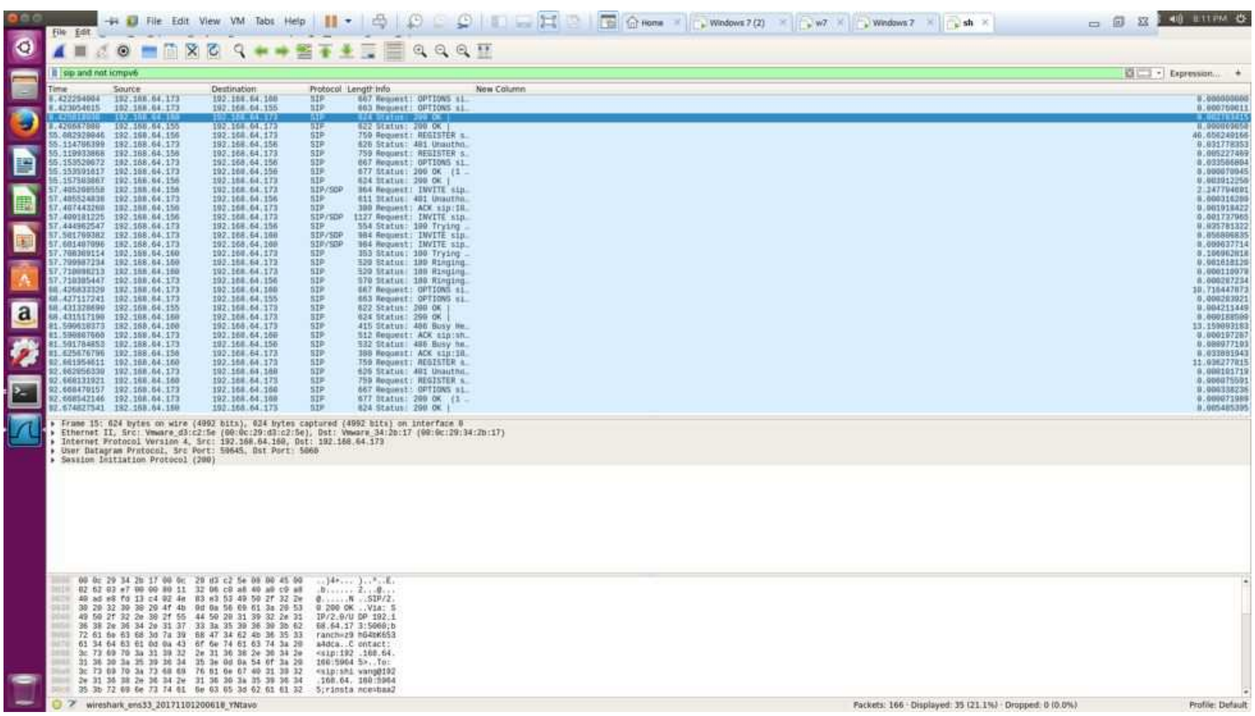
Now, here in this one client with ID 200 is made busy with the busy period of the time mentioned in the brackets; here in our case we have considered the time to be busy as of 10 seconds.
exten=> 200,1,Answer
exten=> 200,2,Busy(10) exten=>
200,3,Hangup
200=>Busy

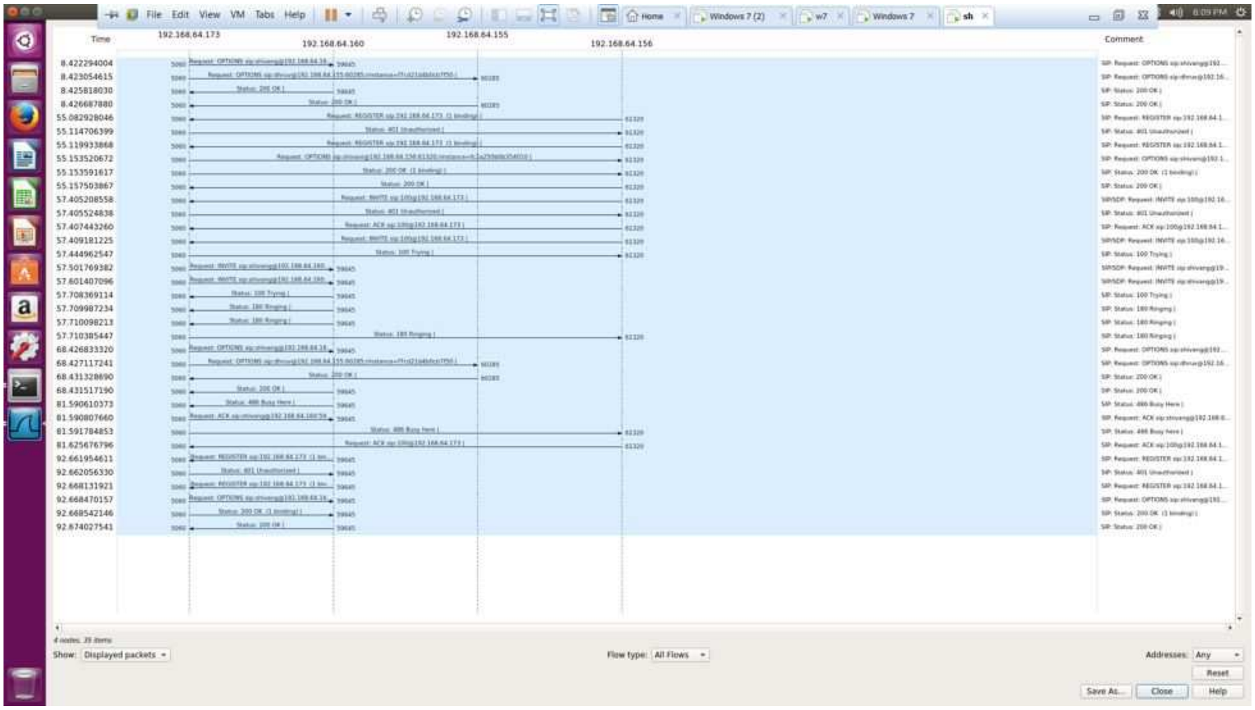




We can see in the following Wireshark capture, the of user being BUSY.

From 100-




Time
192.168.64.173
192.168.64.160
192.168.64.155
192.168.64.156
Comment
8.422294004
8.423054615
8.425818030
8.426687880
55.082928046
55.114706399
55.119933868
55.153520672
55.153591617
55.157503867
57.405208558
57.405524838
57.407443260
57.409181225
57.444962547
57.501769382
57.601407096
57.708369114
57.709987234
57.710098213
57.710385447
68.426833320
68.427117241
68.431328690
68.431517190
81.590610373
81.590807660
81.591784853
81.625676796
92.661954611
92.662056330
92.668131921
92.668470157
92.668542146
92.674027541
Show: Displayed packets
Flow type: All Flows
Addresses: Any
Reset
Save As...
Close
Help

PHASE-3: Call on Hold

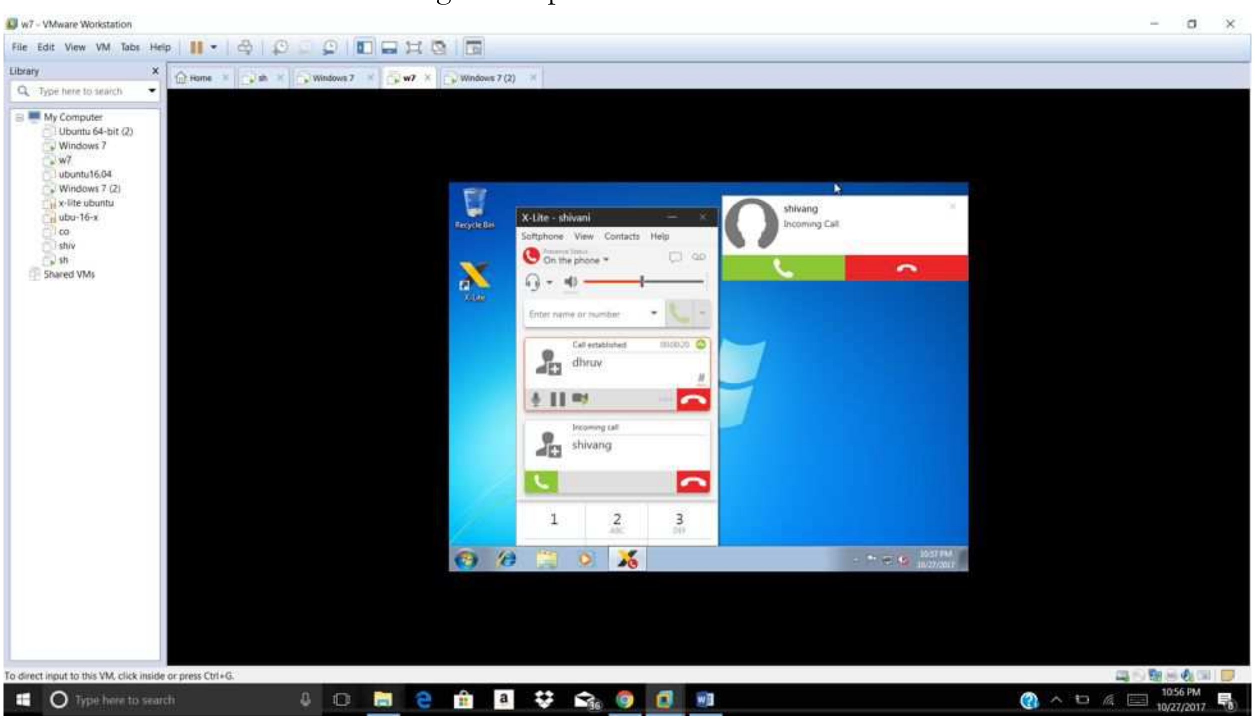
Here the third client, with the client ID 300, calls to the user with ID 100, which is already in a connection with user 200. User 100 then keeps the 200 user on HOLD and establishes a call with ID 300. After completing the call with 300, he (ID#100) continues the call with user 200.
For this part, we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

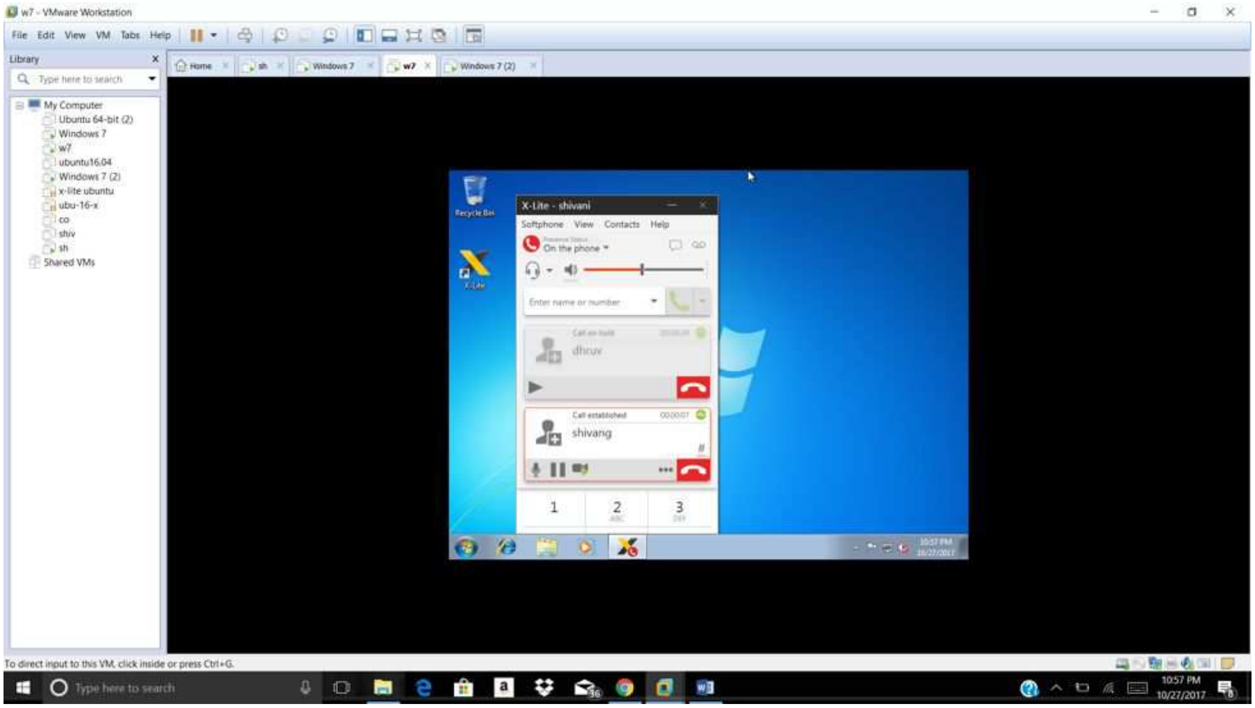
Also, since we want to establish the call with user (ID#300) we have to add the following commands in the extension.conf file -

Exten => 300,1,Dial (SIP/300,20)
Exten => 300,2,Hangup

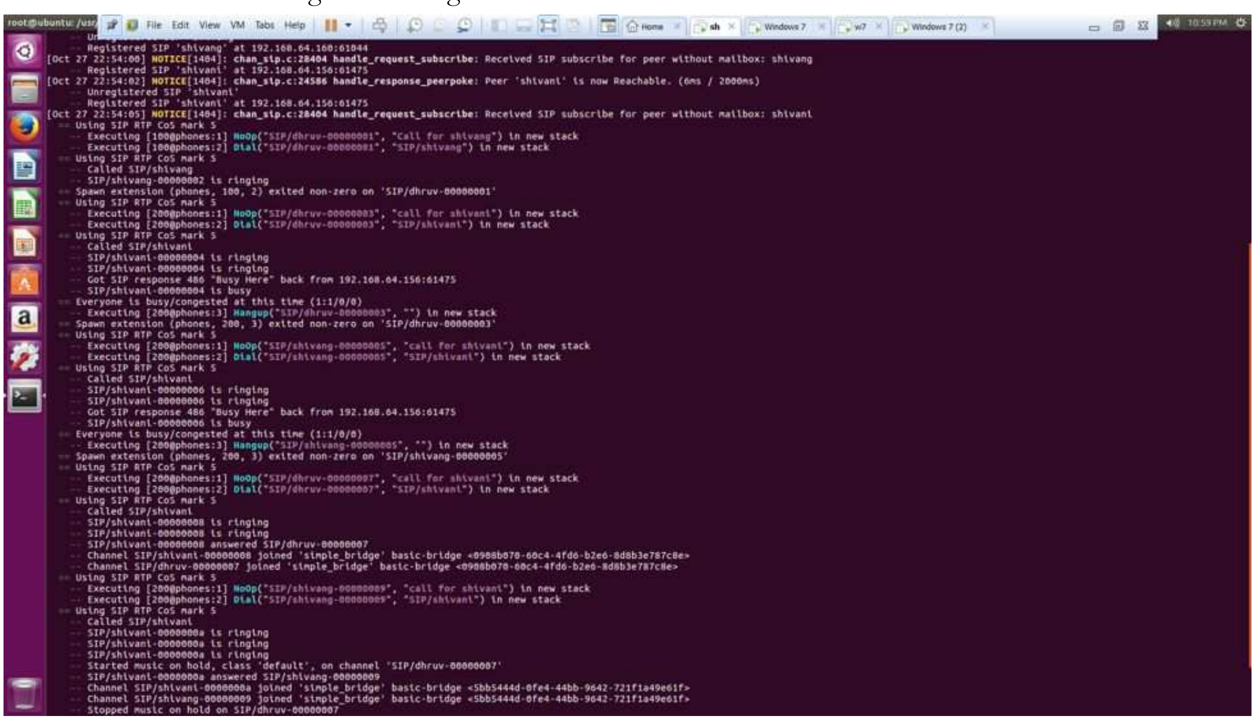
In this image we can see that user 300 is calling user 100 and then the user 100 holds call of user 200 and continues call with 100 once it gets completed.



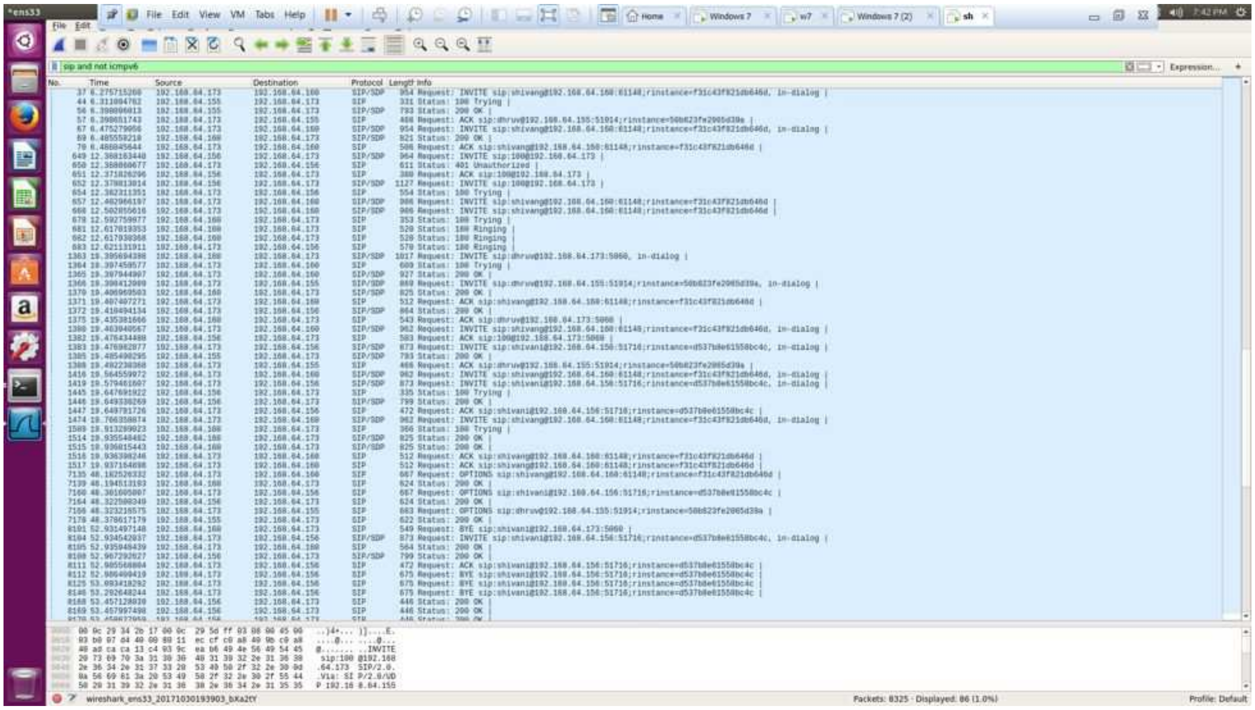
Call on Hold for Dhruv (User 300), while user 200, established call with user 100-



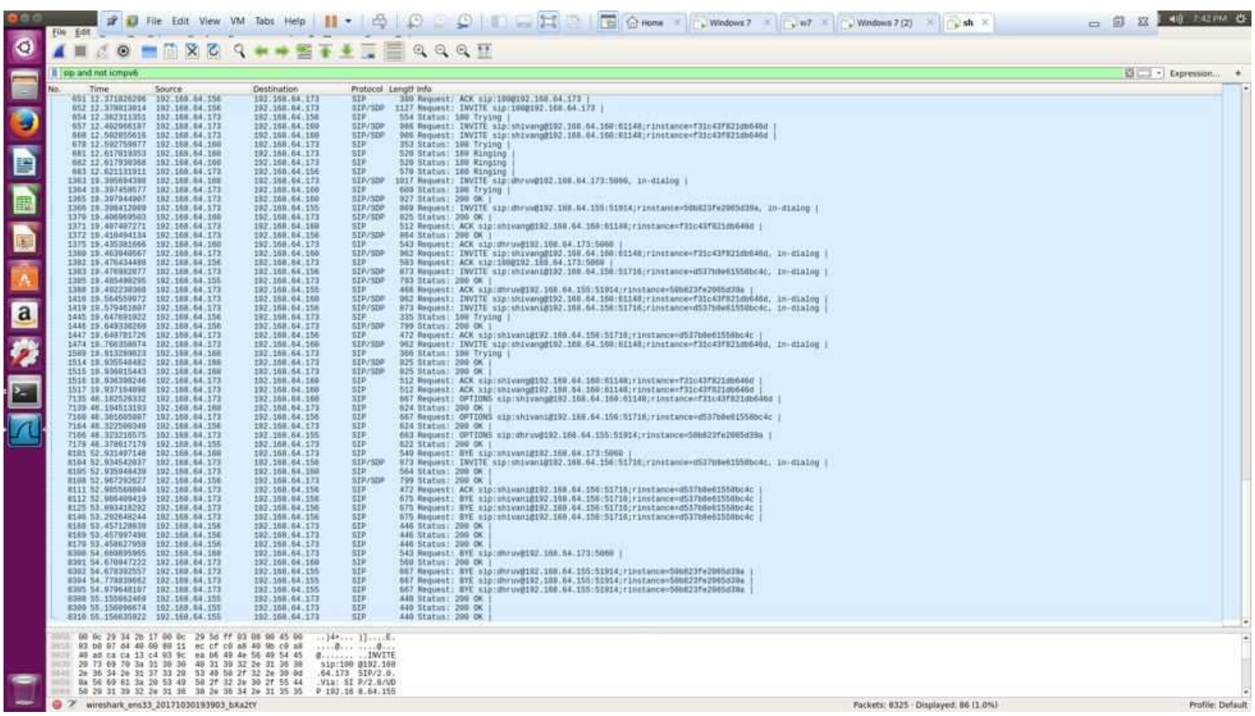
We can also see the working in this image below -



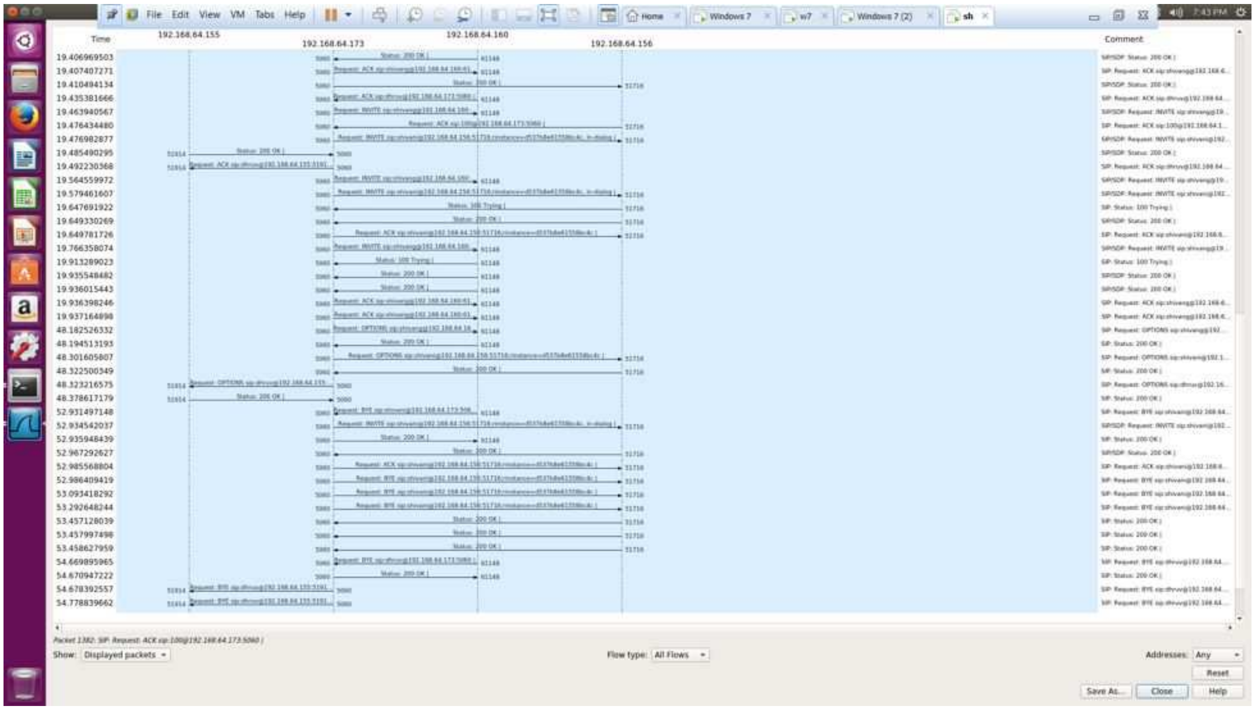
From 100-



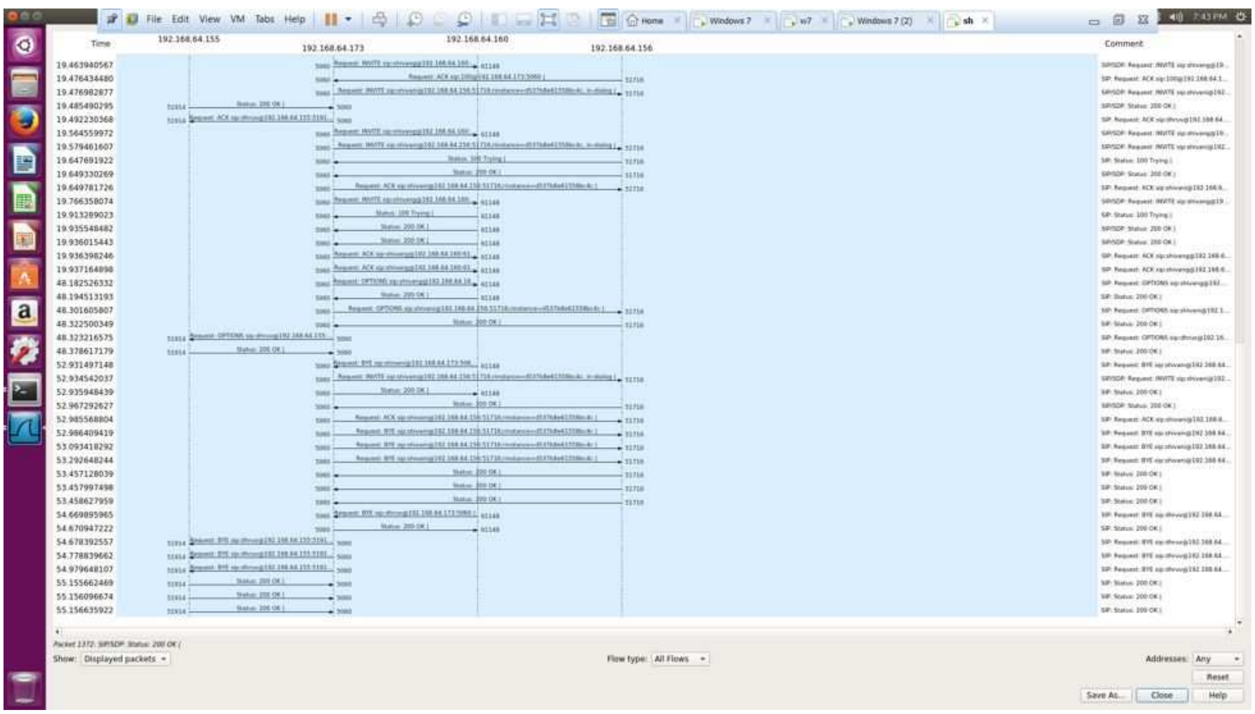
200-



100-

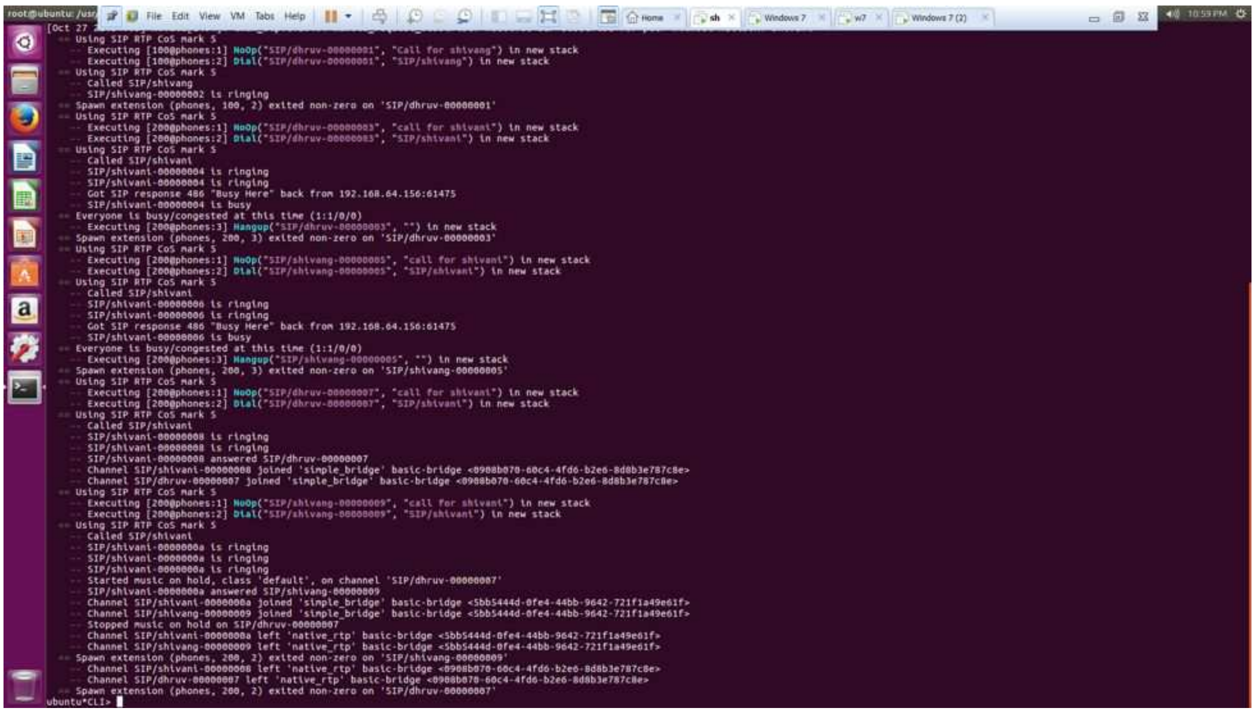


Combined-

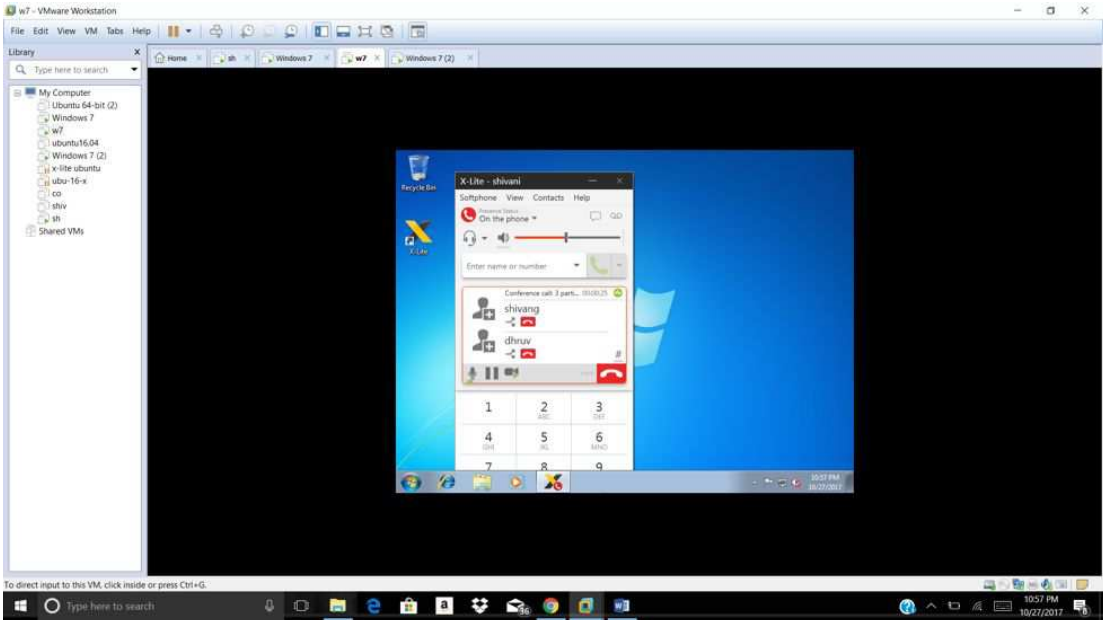


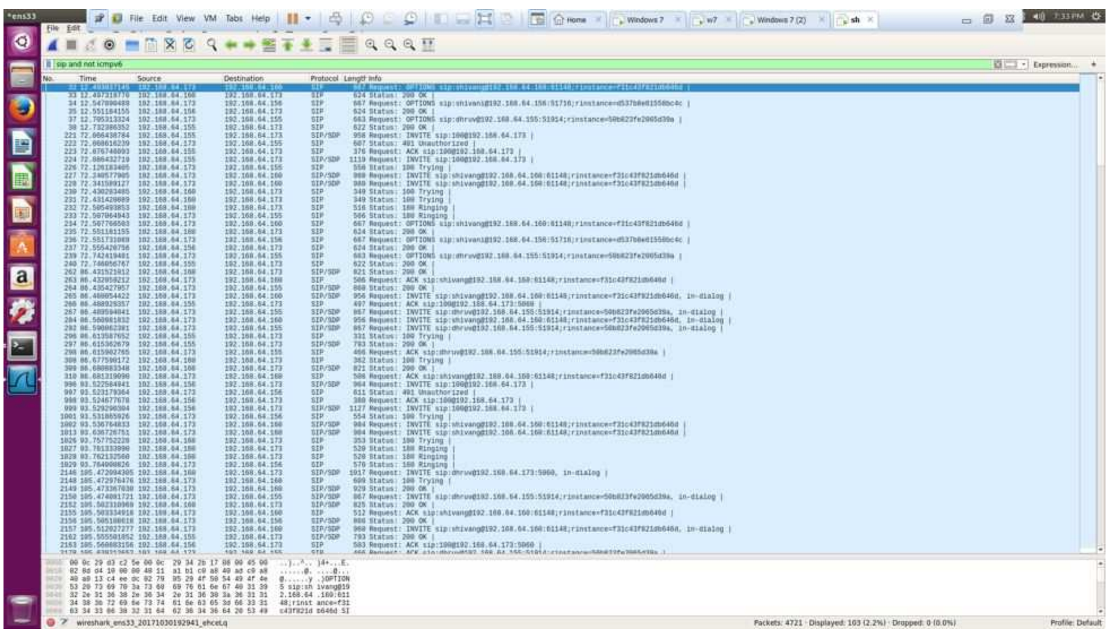
PHASE - 4: Call Conference

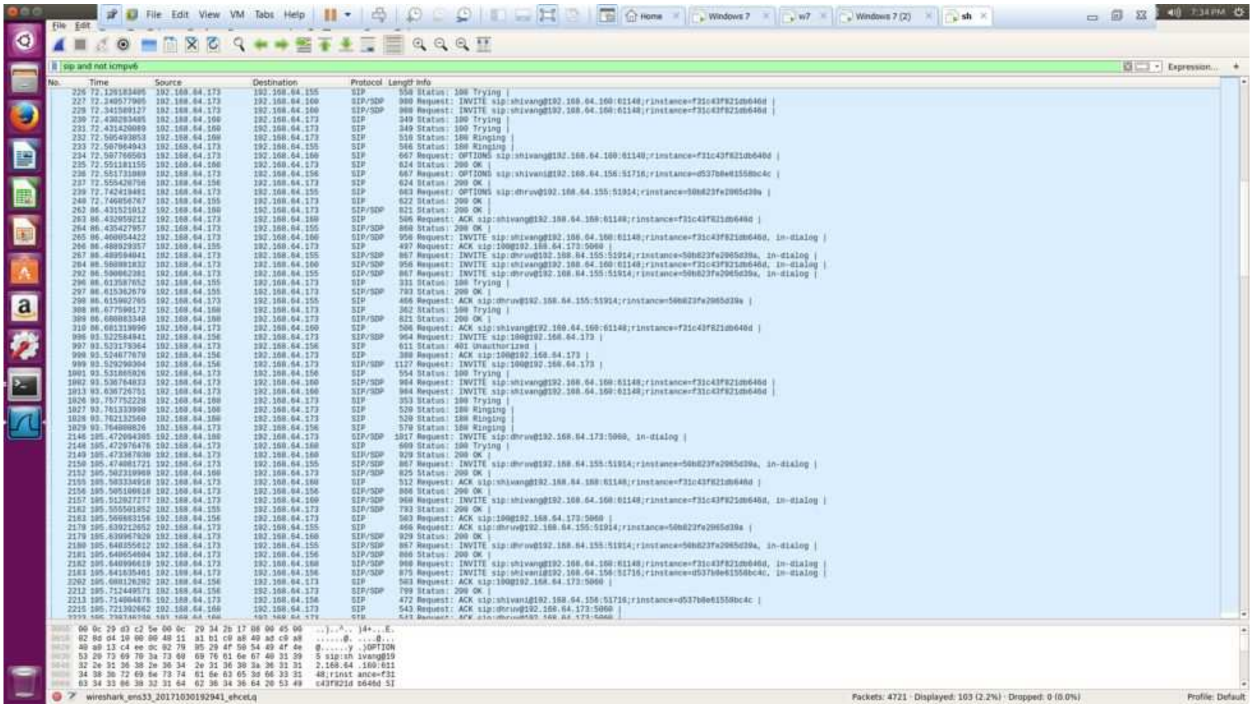
In this Call Conferencing scenario, the user with ID 300 calls user (ID#100), whereas a call is already in progress with two users (IDs#100 and 200). The user 2000, puts the call by 200 on Hold and establishes the call with user 300. Once the call has been set up between them, user again invites the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

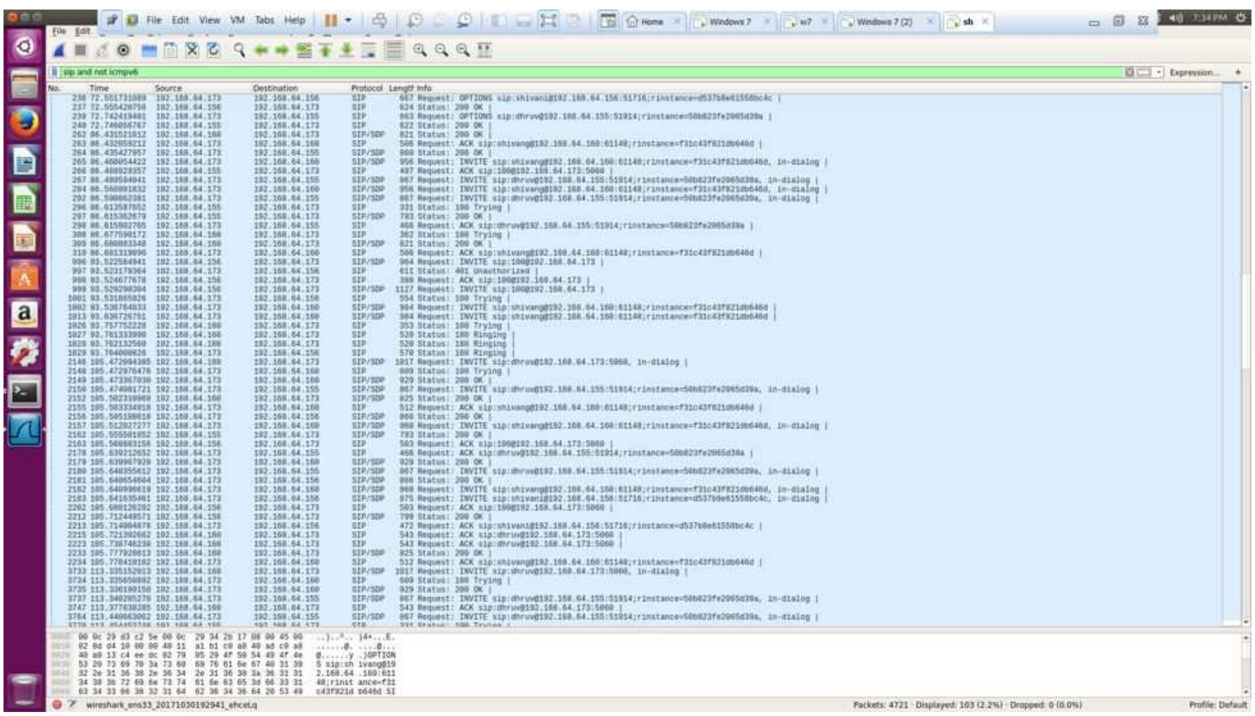


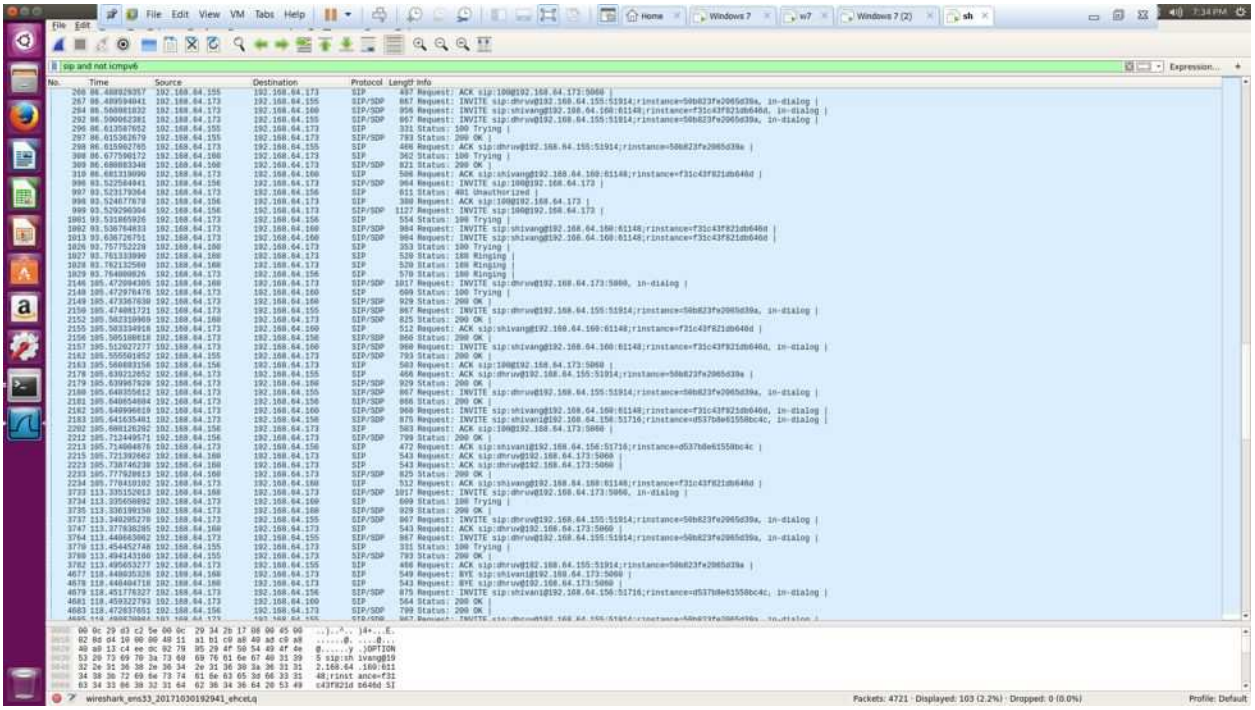
Call conferencing using all three users -



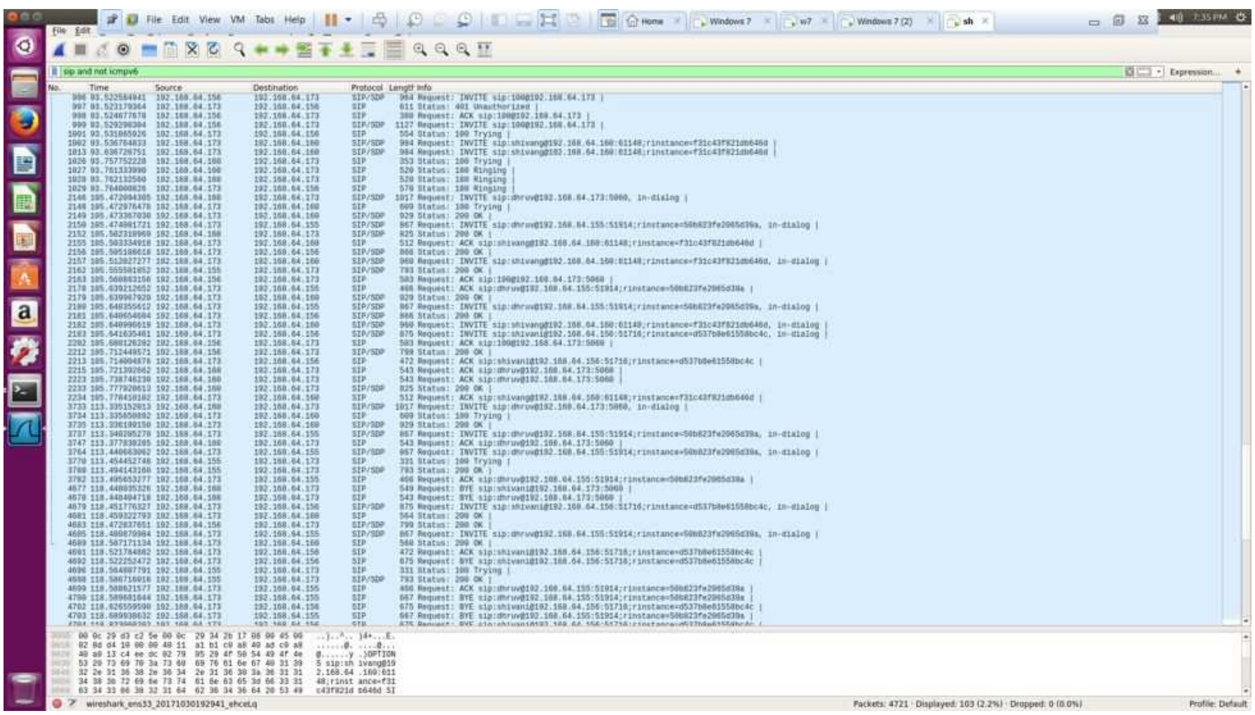


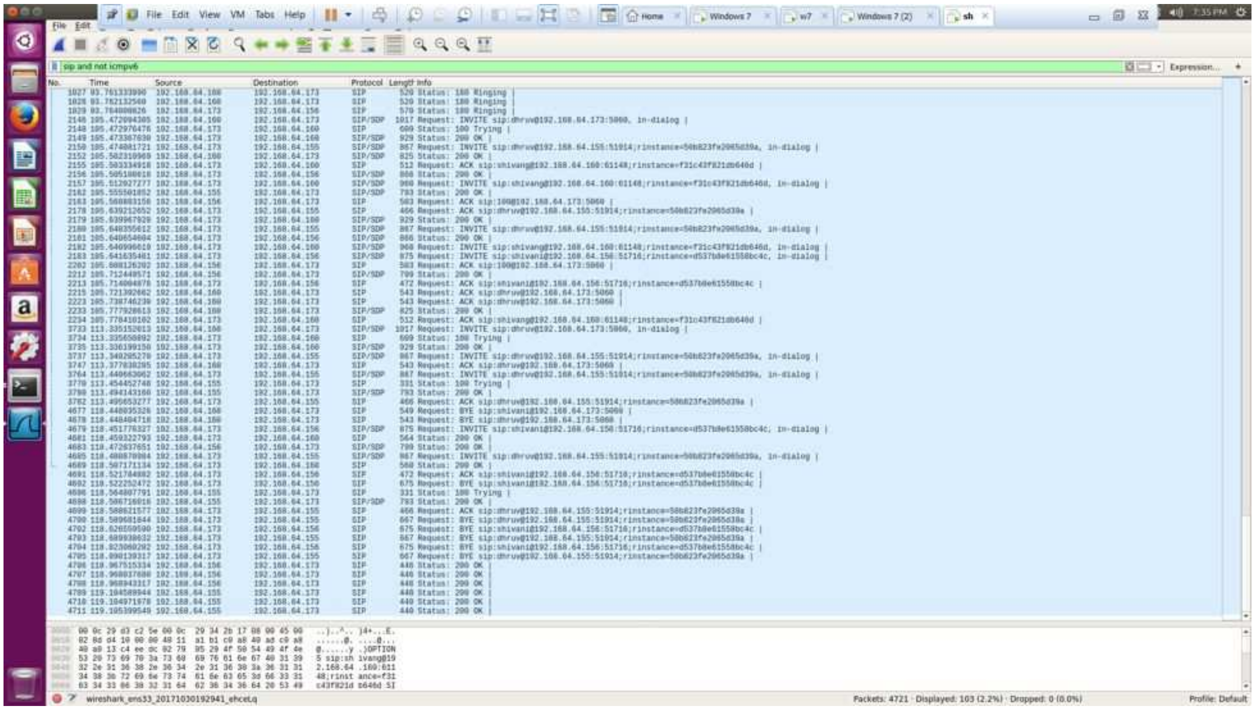

sip and not icmpv6
wireshark_ens33_20171030192941_ehceLq
Packets: 4721 · Displayed: 103 (2.2%) · Dropped: 0 (0.0%)
Profile: Default

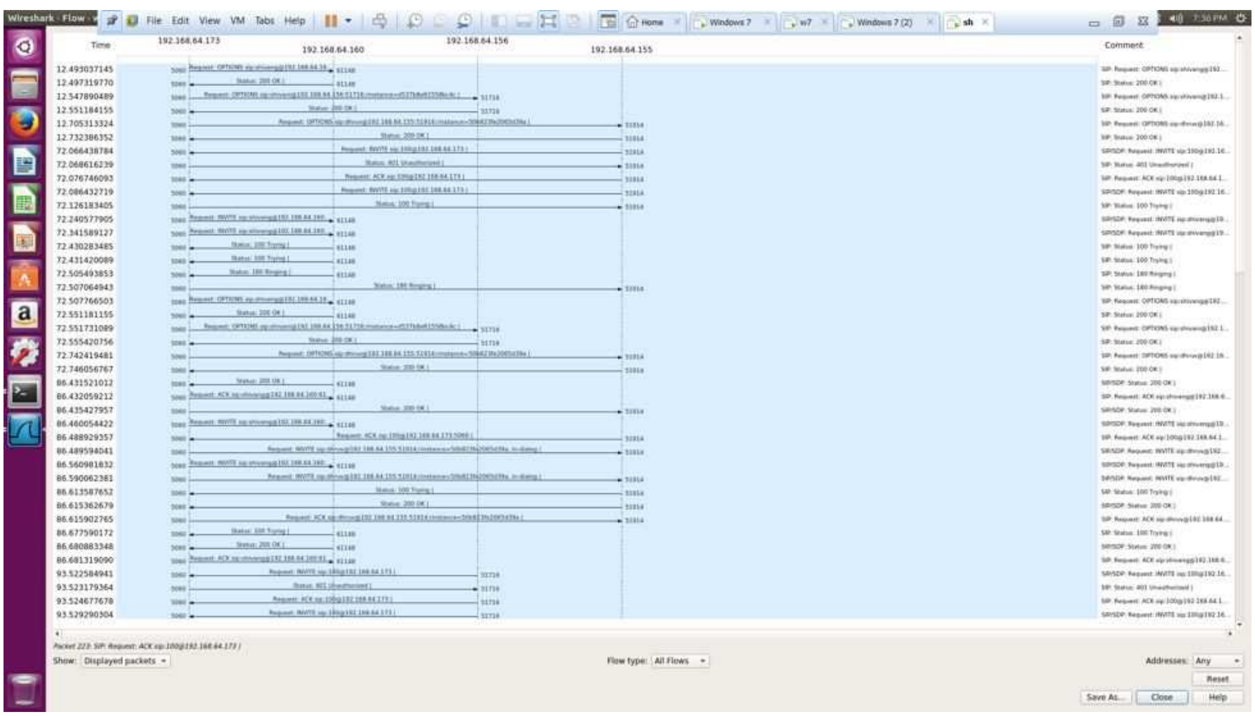

sip and not icmpv6
wireshark_ens33_20171030192941_ehceLq
Packets: 4721 · Displayed: 103 (2.2%) · Dropped: 0 (0.0%)
Profile: Default



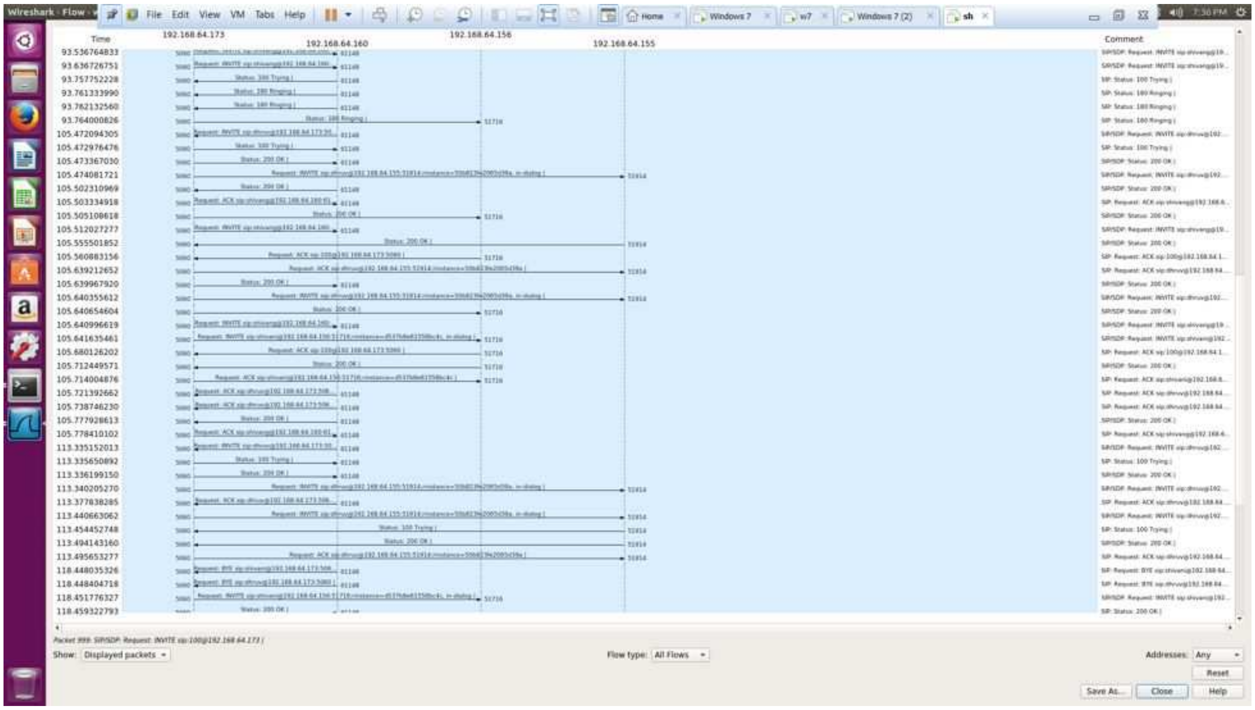
Only SIP files-



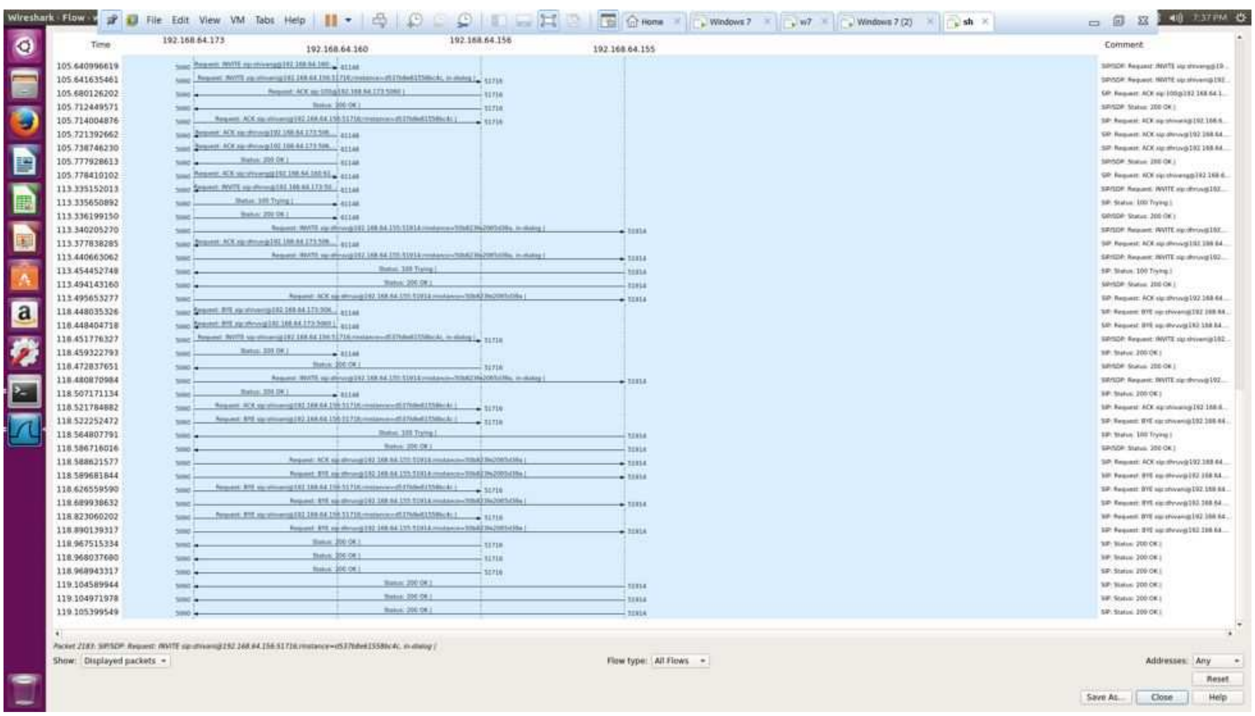




192.168.64.173 multicasts to all



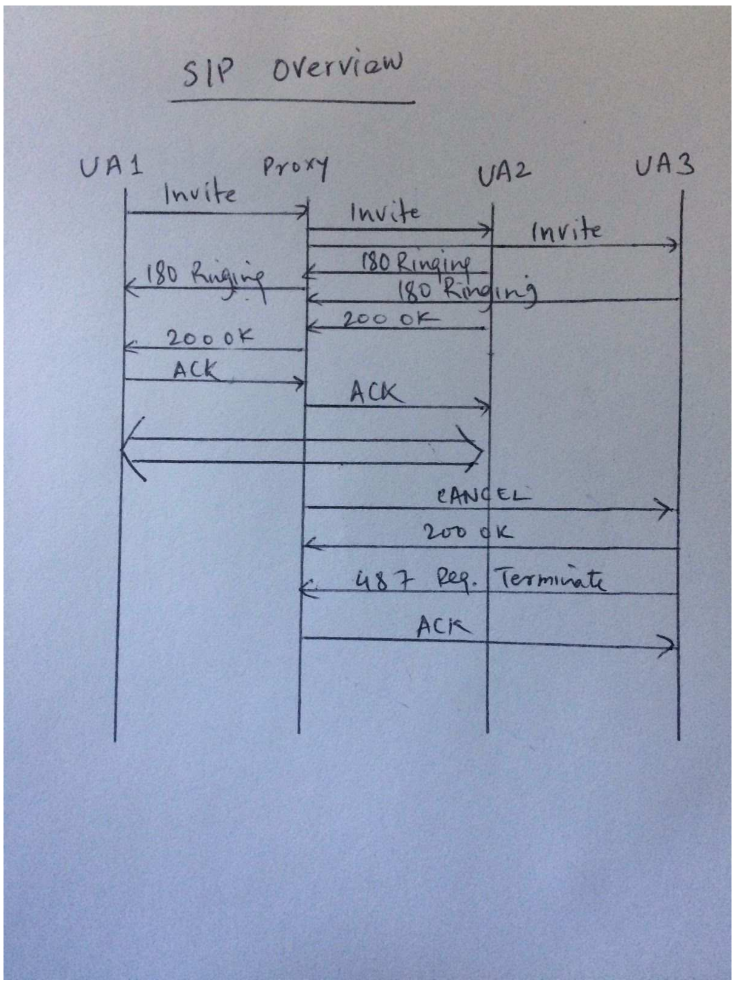
192.168.64.173 to 192.168.64.156 via 192.168.64.160



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 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.

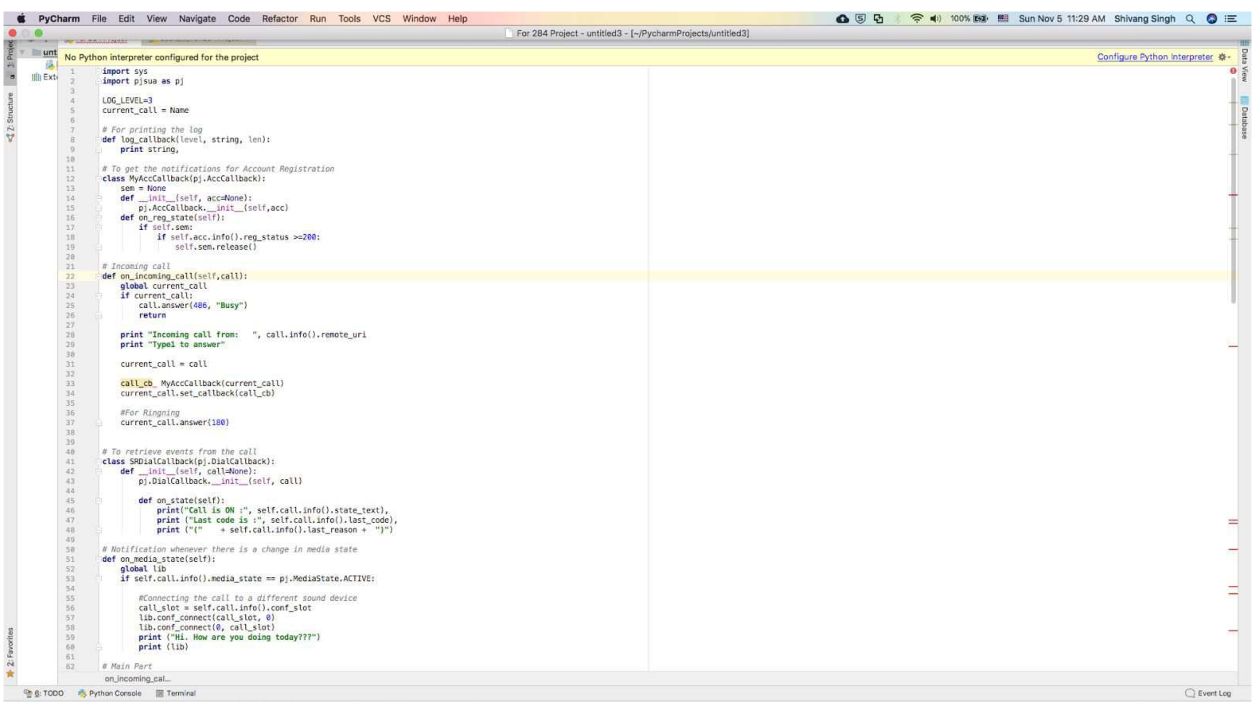
```
    # Start calling process
b=raw_input("Enter destination URI: ")
call = acc.make_call(b, SRDialCallback())
```

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.

```python
# 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
```

<u>Python-Client Code :</u>



untitled3 〉 For 284 Project 〉

For 284 Project × | Users/.../For 284 Project ×

No Python interpreter configured for the project — Configure Python Interpreter

```python

# 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 listning 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 Adderess 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()

    #Configurig 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')
    print "Registration is completed"
    print('Status=  ',acc.info().reg_status, \
        '(' + acc.info().reg_reason + ')')
```

on_incoming_cal...

6: TODO | Python Console | Terminal | Event Log

Error running 'scratch_11': @NotNull method com/intellij/execution/configurations/GeneralCommandLine.getExePath must not return null (a minute ago)

22:33 LF UTF-8

untitled3 〉 For 284 Project 〉

For 284 Project × | Users/.../For 284 Project ×

No Python interpreter configured for the project — Configure Python Interpreter

```python
    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')
    print "Registration is completed"
    print('Status=  ',acc.info().reg_status, \
        '(' + acc.info().reg_reason + ')')

    ab5=raw_input("Do you want to make a call now ?? Y/N\n")
    print "\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('\r\n')

    # To shut down the library
    lib.destroy()
    lib = None

else:

print "Unregistering"
time.sleep(2)
print "Destroying the librarires"
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_cal...

6: TODO | Python Console | Terminal | Event Log

Error running 'scratch_11': @NotNull method com/intellij/execution/configurations/GeneralCommandLine.getExePath must not return null (a minute ago)

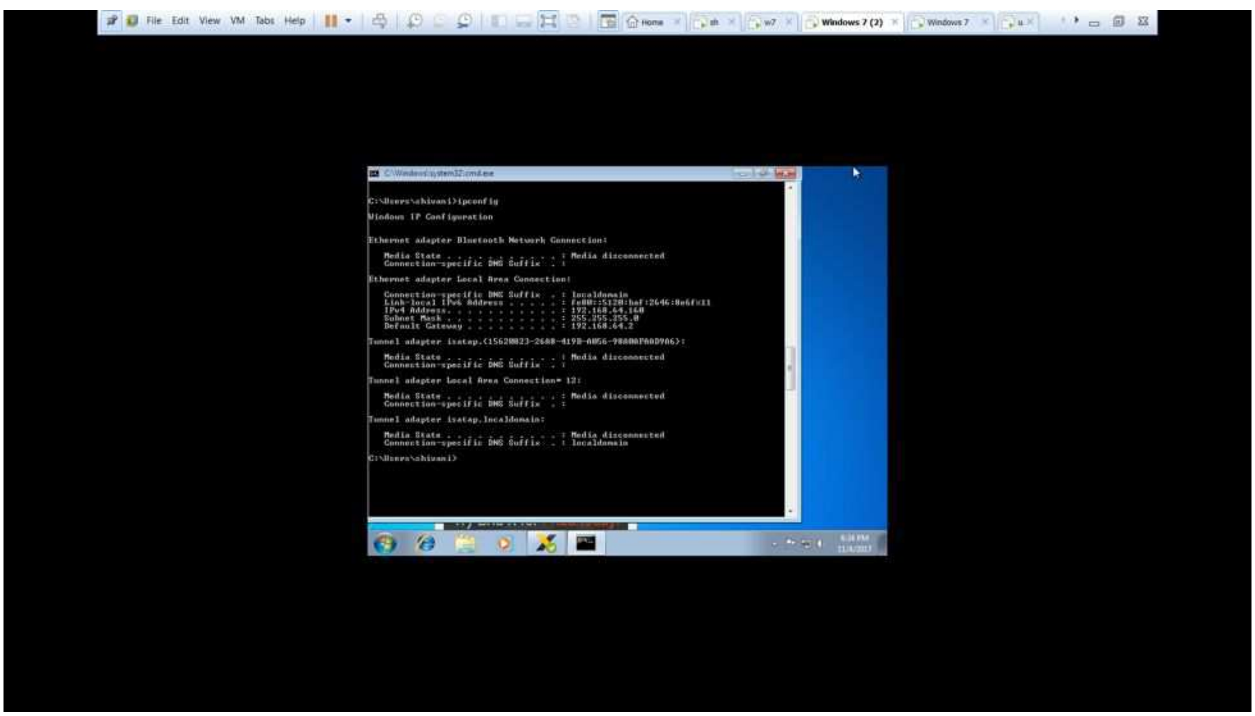
22:33 LF UTF-8

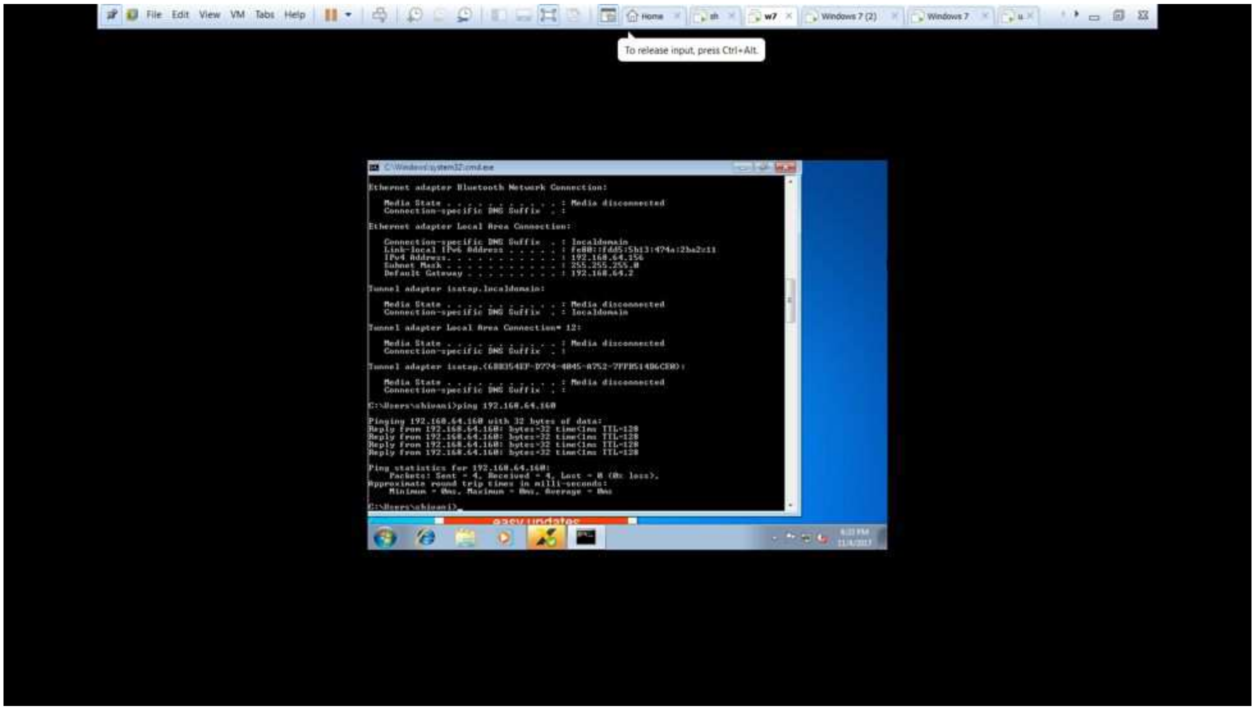
MoS VALUE CALCULATION

- 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.



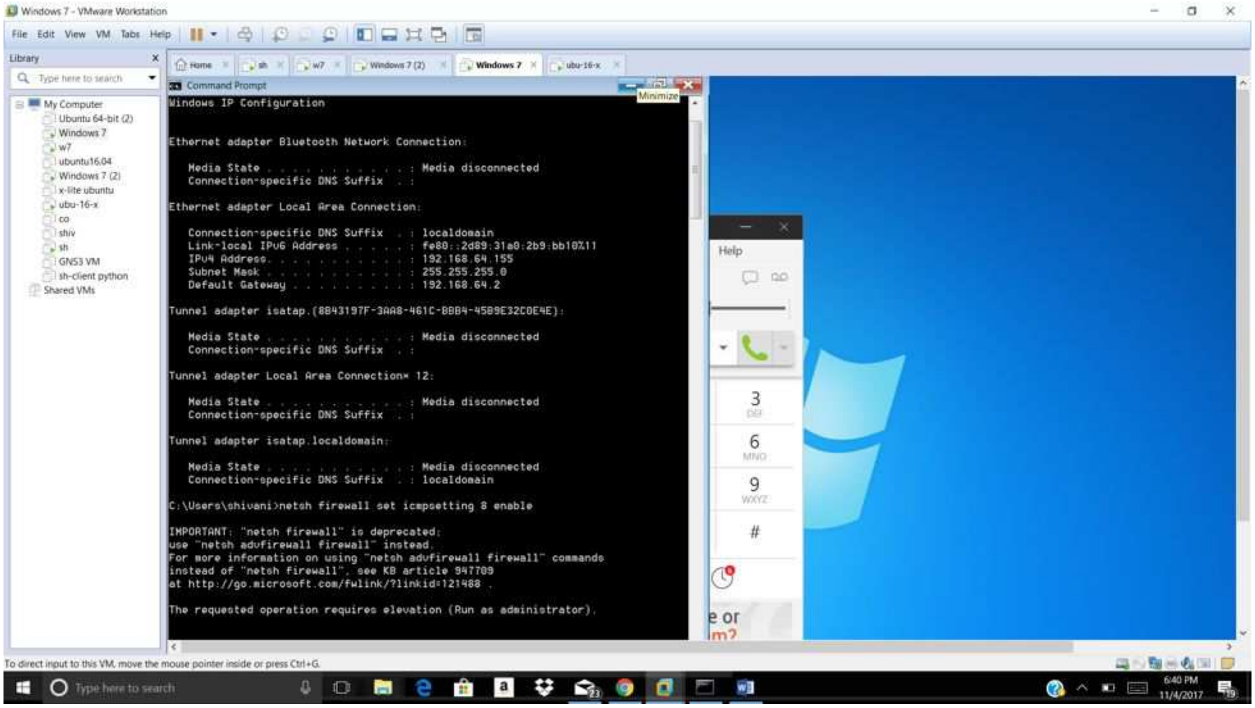


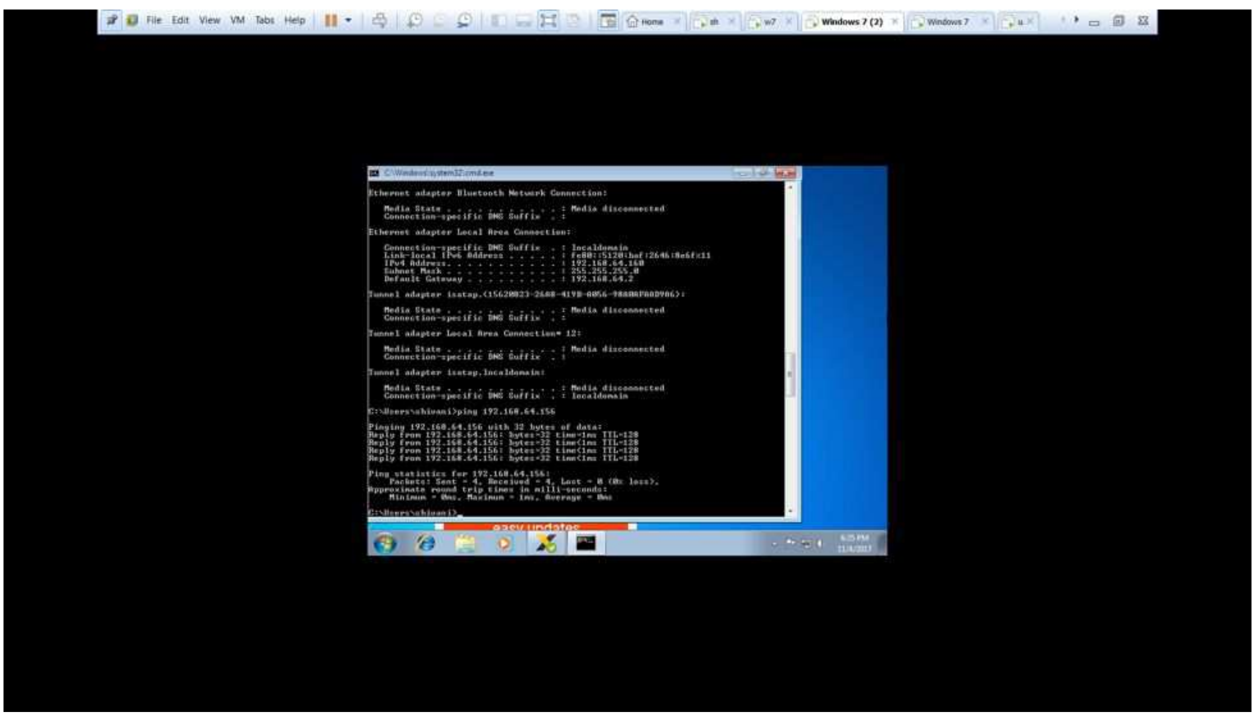
- Here we have pinged 192.168.64.168 with 32 bytes of data and we see that-

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