Aalto University
School of Electrical
Engineering

Department of Communications and Networking

ELEC-E7330 Laboratory Course in Internet Technologies

Work Number 29: VoIP
Student Edition
Preliminary Exercises and
Laboratory Assignments

Tommaso Praturlon 06.07.2022
Juha Järvinen 16.11.2020
Dancun Ogenda & Abbas Waqar 20.07.2019

Table of Contents

1. Preliminary Exercises

1.1 Introduction

Voice over Internet Protocol (VoIP) has been a popular technology as it seeks to replace the ageing Public Switched Telephone Networks (PSTN). It enables the transfer of both voice and data over IP. It has the benefits that one, for example, does not have to be restricted to a given geographical location as an internet connection is all that is needed to make phone calls.

There are different scenarios in which VoIP can be implemented. Examples include Skype, WhatsApp and Facebook Messenger. But for this laboratory exercise, FreeSWITCH, which is an open source VoIP solution, will be used.

The laboratory will cover: registering users on the FreeSWITCH server and making calls between users in one domain, making calls between users in different domain, and Session Initiation Protocol (SIP) security and traffic monitoring to see the important details that are involved in the communication.

1.2 VoIP, SIP and FreeSWITCH preliminary questions

PQ1
What is the difference between VoIP and SIP? List a few examples of the popular VoIP solutions that you might have heard of.

PQ2
Draw the protocol stack and show the level at which SIP operates.

PQ3
What is a codec? Provide three different examples of audio codecs that can be used in SIP. Provide the main characteristics such as sampling rate, bit rate and frame size for each of the codecs. Provide two functions of codecs.

PQ4
What are the major differences between VoIP and Public Switched Telephone Network? Provide four features that comes with implementing VoIP that makes it commonly referred to as a killer application of traditional phone systems.

PQ5
Provide an example of codec negotiation in FreeSWITCH in a call between Alice and Bob. Assume Alice has phone with codecs (G722 and PCMU), Bob has codecs (PCMU and PCMA) and the FreeSWITCH server has an inbound codec preference of (G722, G722.1, PCMU, PCMA, G729).

PQ6
Provide the directories that you can use to register a user in FreeSWITCH. In that directory, there are default users. Provide your configuration for a user with username 2000 and password PASS. Show the additional configurations that you have to make in the dialplan file so that calls made to 2000 can be successful.

PQ7
How would you access the FreeSWITCH CLI after it has been installed? How would you save changes to the user files? How would you prove that users have been successfully registered to the FreeSWITCH server? How would you exit the FreeSWITCH CLI? Provide the commands with explanations.

PQ8
By default, in FreeSWITCH everything is transmitted unencrypted. What are the possible disadvantages that exist and how can security be implemented? Provide the directory that can be used to configure security, for example TLS. Provide an exact way in which the media stream can be encrypted.

NOTE: Before coming to the laboratory for the exercises, please read through the FreeSWITCH documentation, or watch videos online on how to set it up and register users. This will make your work easier when you are doing the laboratory work.

2. Laboratory Work

2.1 Introduction

This labwork is done at home and at lab room. We use in this work an ESXi platform installed in an APU4 AMD64 computer. Download images at MyCourses portal.

The laboratory exercise is based on the SIP protocol. The server application used in this exercise is FreeSWITCH, which is installed in Debian 10 servers. There are two laboratory scenarios, one in which calls are made between users which exist in a single domain, the other is between users on different domains. To make calls after registering the users, you will have to configure two physical VoIP phones and at least one softphone that is installed already on the workstation.

To confirm whether users are registered, and any changes made to the configuration files are working, you will need to SSH into the FreeSWITCH servers. The IP addresses are 10.38.0.100 for voip1 and 10.38.10.100 for voip2. The username is *lab* and the password is *lab*.

SSH will not work when trying to configure the VoIP phones. For this, you will have to configure them according to the instructions given in the Appendix section. Once you give them the IP address, you can point to that IP on a web browser and from there configure the users.

All the questions marked with (AT THE LAB) are done at the lab, others at “home”.
Reserve a 1-2 hours of lab session by emailing to juha.tapio.jarvinen@aalto.fi or in the MyCourses portal.

Environment

The topology of the first part of the laboratory exercise is as shown below in Figure 1. As a gateway address use 10.38.0.254.

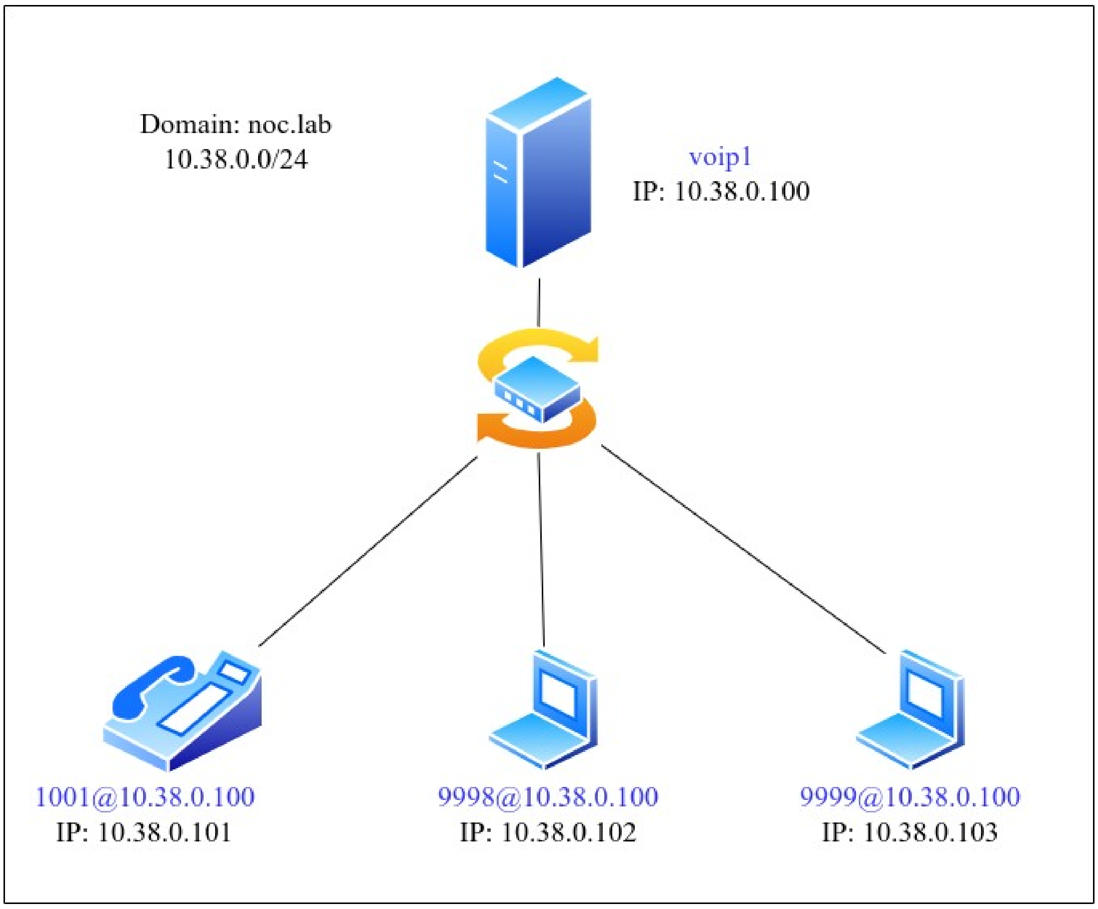

Figure 1. VoIP laboratory topology for the first scenario

Now install one FreeSWITCH server (voip1) in the APU computer through ESXi (connect to management port and login with user *root* and password *VyosVyos!!*). Configure it similarly as in Figure 1.

1. In ESXi connect a physical connections (e.g. vmnic1) to voip1 and use a physical switch to connect the two other laptops and the phone.
2. Install Jitsi softphones to your computers with IPs 10.38.0.102 and 10.38.0.103. Check the Appendix A for instructions on how to do so.
3. At the laboratory, a separate phone (10.38.0.101) will be connected to your installation.

Note! If during the laboratory you run into issues concerning the memory space, which for example impede the functioning of freeswitch, it may be because some files in /var/log/ have reached a big size, you can delete those. If you need even more space, some virtual memory files such as .vswp are sometimes created by ESXi in the datastore. You can delete those, usually they are around 110Mb.

2.3 Configuration of SIP Components

Q1
SSH into voip1. The username is *lab* and the password is *lab*.

A guideline for FreeSWITCH configurations will be found at:
https://freeswitch.org/confluence/display/FREESWITCH/Configuration
First, you have to configure two new users to Jitsi softphones, namely, 9998 and 9999.
Create .xml files for the new users and then register them using the Jitsi softphone. The username has to be that of the created user and the domain the IP address of the server at which they are registered.

Take a look at how the default user .xml files have been configured to know what to include to make the new users that you are going to create work.

After creating the .xml file, you have to give file ownership to *freeswitch* otherwise it won't load the .xml files. You can do that by following commands.

```
chmod 000 <filename> // remove all rights from the file
chmod gu+wr <filename> // give the group and user read and write permissions to
the file
chown -R freeswitch:freeswitch <filename> // change the ownership of the file to
freeswitch
```

After creating the .xml files for both users, you can access the FreeSWITCH CLI, by using command `fs_cli`, reload all the xml configurations by using `reloadxml` command.
To check the registered users, use `show registrations` command.
Hopefully, you can see the users that you have created registered. If no registrations can be found for users, then something might be wrong with your user configuration. Try it by yourself and if you do not succeed you can ask for help from the assistant(s).

Then go back to the FreeSWITCH CLI and confirm using the command `show registrations`. If from the console, you can see the total number of registrations to be two, then everything is working as it should be.

Proceed by making a call between user 9998 and 9999. However, before making a call, start `tcpdump` on the Ethernet interface of voip1. To be sure that tcpdump is installed you can type the command `sudo which tcpdump`. Capture the packets for a small period between the call.

At the lab: make a call between users 9998 and 1001. Before making a call, start tcpdump on the Ethernet interface of voip1. Capture the packets for a small period between the call.

Transfer the .pcap file to your lab computer and open it using Wireshark.

Q2
Analyze the file and describe everything that is happening. What sort of packets and protocols can be seen from the capture? Can you see the origin and the destination of the call? What codecs are being used, if any?

Provide short and concise descriptions with Wireshark VoIP calls and SIP functions to back up your answers.

2.4 SIP Trunking

Environment

Add voip2 server and NetEm to the installation as presented in Figure 2. Configure voip2 on a similar way as voip1 and configure the rest of network as in the figure; 10.38.200.0/30 is the internal link in ESXi. Move the 9999 laptop to the other domain.
Remember to IP route networks between VoIP hosts and **make sure that voip1 can exchange packets with voip2 via netem**.

NetEm is "*an enhancement of the Linux traffic control facilities that allow to add delay, packet loss, duplication and more other characteristics to packets outgoing from a selected network interface. NetEm is built using the existing Quality Of Service (QOS) and Differentiated Services (diffserv) facilities in the Linux kernel.*" (https://man7.org/linux/man-pages/man8/tc-netem.8.html). In the image there is a network bridge between two interfaces. "Cut" a link between voip1 and voip2 servers and add Netem between them. Remember to name links with different names. The just start the NetEm image in ESXi and leave it running so long as told to something with that computer. You can access the virtual machine with the credentials *lab/lab*.

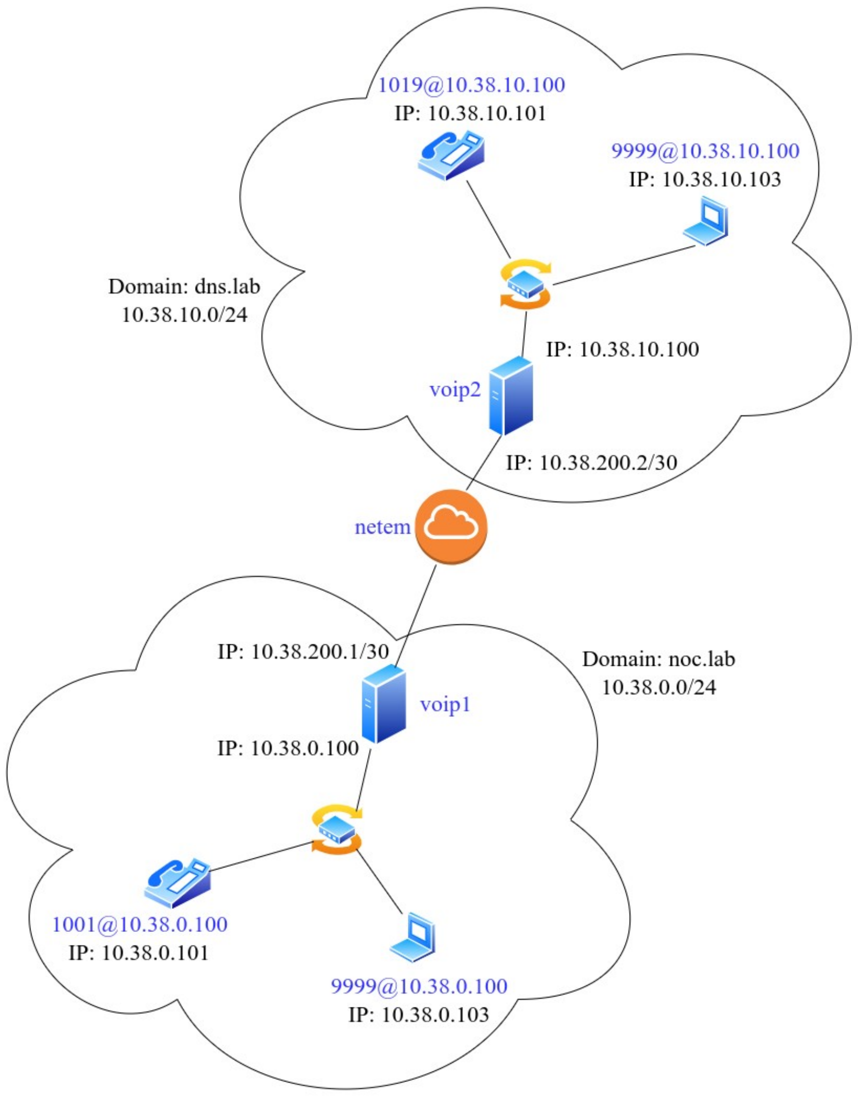

Figure 2. VoIP laboratory topology for the second scenario

Q3
Next remove user 9999 from voip1 and configure it on voip2 server (10.38.10.100). You can SSH into voip2 by using username *lab* and password *lab*. Show that your registration is performed on voip2, make a call from 9998 to 9999. Is the call successful? If not, why?

Towards SIP trunking

SIP Trunking refers to the interconnection between two domains to enable end point's Phone Exchange Systems (PBX) to send and receive calls via the Internet. The SIP protocol is able to control voice, video and even messaging applications.

As you might have noticed from the previous question, a call from user 9998 (voip1) to 9999 (voip2) was not successful because SIP Trunking was not implemented and the first SIP server did not know how to route calls to the other.

To enable outbound calls, we need a gateway that will allow our SIP servers to route outbound calls, that is, a SIP trunking that enables users 1001 and 9998 in voip1 server to call users 1019 and 9999 on voip2 server.

The task for you is to configure the FreeSWITCH servers, and in order to do so you will need to edit the configuration files in both voip1 and voip2.
We want to have a configuration such that

i. when user 1001 in noc.lab dials 51019, user 1019 in dns.lab is called.
ii. when user 1019 in dns.lab dials 61001, user 1001 in noc.lab is called.
iii. when user 9998 in the domain noc.lab dials 59999, user 9999 in dns.lab is called.
iv. when user 9999 in the domain dns.lab dials 69998, user 9998 in noc.lab is called.

Starting from voip2, edit the `/etc/freeswitch/sip_profiles/external.xml`, and locate the section that has `ext-rtp-ip` and `ext-sip-ip`. Change those setting from the default value to the IP address of that domain.

Secondly, enable the gateway configuration in `/etc/freeswitch/sip_profiles/external/`, create a new .xml file, taking example.xml as reference and resources online on the topic. Here, you need to specify which requires the gateway name, gateway IP, and authentication details. These latter are the username and password of the registered user in the other domain, which will be able to answer the call if configured on the gateway for incoming calls, as we will do in the following. The codecs that will be used for outbound and inbound calls can also be configured in this file.

Thirdly, we need to configure the Direct Inward Dial (DID) service, which will indicate that when a dialed number is not recognized as internal, the server will try to search for another domain, placed outbound, and will also tell our server to route this call to a specified gateway, the one create above. This is configured in `/etc/freeswitch/dialplan/default/outbound.xml`, where for voip2 the destination number is 61001.

Finally, we need to set up our SIP server to be able to receive incoming calls from other domains. This is configured in `/etc/freeswitch/diaplan/public/00_inbound_did.xml`. Here, we specify which dialplan to refer to if we receive something from another domain, then look for the specified dialplan against incoming DID plan.
This shows that if the destination number matches the number configured in this file, then the call is inbound and should be handled as such.

Perform the same operations for the voip1 server, with gateway 10.38.10.100 and outbound number 51019. For these changes to take effect, you can reload the external profiles with the command `sofia profile external restart reloadxml`, even though sometimes you might also need to restart the freeswitch service from the server with `sudo systemctl restart freeswitch.service`.
After a successful configuration, you can check whether both servers are able to communicate with each other. This can be tested by opening the FreeSWITCH CLI and entering the command `sofia status`. You should be able to see the external registration that will handle outbound and inbound calls between the two FreeSWITCH servers in different domain.

Q4 (AT THE LAB)
Make a successful call from user 1001 in voip1 domain to user 1019 in voip2 domain and vice versa.

Q5
Start `tcpdump` to capture packets, on any of the FreeSWITCH servers (voip1 or voip2) and make an outbound call from any domain. Accept the call at the other end and talk with your group member for few seconds. Transfer the captured file to the lab computer and open it using Wireshark.

Analyze the file and describe everything that is happening. What sort of packets and protocols can be seen from the capture? Can you see the origin and the destination of the call? What CODECS are being used, if any?

Provide short and concise descriptions with Wireshark VoIP calls and SIP functions to back up your answers.

Q6 (AT THE LAB)
Repeat the test with 9998 and 9999. Is there any difference in dumps?

2.5 QoS and codecs

Simulating a WAN

In most VoIP networks, there are challenges that always affect the quality of calls between different endpoints. An example of the challenges are latency, jitter and delays, among others. Various audio codecs have been designed to support VoIP calls under different network conditions.

Real network consists of limited bandwidth along with latency, jitter, delays. These can be introduced in our network using `tc`, `netem` and `wondershaper`.

There is a virtual machine between voip1 and voip2 to emulate a Wide Area Network which is called netem (Network Emulation). It allows to test protocols by emulating variable delays, loss, duplication and reordering. Netem is used with the command line tool `tc` (traffic control) and you can learn more about it from the link:
http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

Some useful commands for the lab are as below:

To add a fixed amount of delay to all the packets that go out of the local interface, the command used is: -

```
#tc qdisc add dev <interface> root netem delay 100ms
```

After the command has been executed, the results of the ping before and after should be different in how long they take. A new ping should show an increase in the delay with a figure of about 100 milliseconds.

It is also possible to introduce random variation to the network. The command used to introduce this variation is as shown: -

```
#tc qdisc change dev <interface> root netem delay 100ms 10ms
```

For packet loss, in the `tc` command the value has to be specified using the percentage sign. An example command that will drop 1 out of 100 packets is as shown: -

```
#tc qdisc change dev <interface> root netem loss 1%
```

In addition, it is also possible to duplicate the number of packets. The command used is as follows: -

```
#tc qdisc change dev <interface> root netem duplicate 1%
```

To corrupt packets through the introduction of random noise, the command used is as follows: -

```
#tc qdisc change dev <interface> root netem corrupt 1%
```

NOTE: Read the reference about the use of netem in addition to the commands that have been given above.

For bandwidth limitation, you can use another traffic shaping script called `wondershaper`. Wondershaper allows you to dictate what amount of bandwidth you want in a link, both in the uplink and the downlink directions. An example is shown as follows: -

```
#wondershaper <interface> 1024 512
```

This will configure on the link a downlink speed of 1024 Kbits/sec and an uplink speed of 512 Kbits/sec

Q7

Which codec would you consider useful in low and high bandwidth conditions? Provide the reasoning behind the choice. What is the best solution and the sacrifice that has to be taken in order to make calls in situation where the bandwidth is limited?

Q8

You can configure the outgoing codec in any of the FreeSWITCH servers by adding a variable tag in the gateway configuration file. The file is located at

```
/etc/freeswitch/sip_profiles/external/<gateway>.xml

<variables>
<variable name="absolute_codec_string" value="insert_codec_here"
direction="outbound"/>
</variables>
```

Force FreeSWITCH to use one of the codecs supported by the IP phones (G722 or G711) for outgoing calls. Restart FreeSWITCH after the changes made to the gateway configuration files. Introduce a delay, and then constantly increase it until the point at which the VoIP calls will no longer be audible. At least use values 0ms, 200ms, 500ms and 1000ms to simulate calls from local call to a satellite call.

Change the delay to zero and then alter the packet loss values. Analyze the results in few sentences and also write the highest loss value for acceptable VoIP call. How does packet duplication and corruption affect in the call?

NOTE: Remove any configured network limitations that are introduced earlier, before going to the next task. Check man pages for the commands (e.g. `man wondershaper`*).*

2.6 VoIP Security and threats

VoIP hasn't survived without security risks in physical and virtual environments and you could read many different articles exploring the topic. This part of the laboratory scratches only the surface of VoIP security, specifically, the encryption of SIP and voice. The idea of this part is to show how unsecured VoIP traffic may be.

Q9 (AT THE LAB)
Capture a VoIP call from any of the FreeSWITCH servers and decode the RTP traffic in Wireshark. Are you able to successfully tap into the VoIP call?

Q10 (AT THE LAB)
Enable a secure call between the FreeSWITCH servers. By default, FreeSWITCH supports also encrypted calls but for the purpose of this lab it is sufficient that the signaling part of the call is secured. Remember, since the call is outbound you will have to look at the dialplan and at the SIP profiles.

Hint for the dialplan:
```
<action application="bridge"
data="sofia/gateway/<gateway_name>/<user>;transport=tls"/>
```

3. Final Report

Answer all the questions marked Q from the previous sections. Additionally, answer the questions below in the final report.

FQ1
There are many more codecs involved with the VoIP technology. Find and list here at least four more along with their basic properties: sampling rate, bit rate, frame size (if frame-based).

FQ2
What security threats can be found for VoIP traffic or system? Mention at least four of them.

FQ3
Give few tips to improve VoIP security.

Appendix A: Installation of Jitsi softphones

This tutorial will require the use of the free virtualization software Oracle Virtual Box (https://www.virtualbox.org/wiki/Downloads), in which we you will create an Ubuntu virtual machine. In this machine you will install Jitsi and Wireshark which will be used in the laboratory. Jitsi Desktop is a free open-source client application using VoIP and SIP, while Wireshark is a powerful packet analyzer.

Note! If you are working on a Windows laptop, the installation of Jitsi Desktop from the website is straightforward.

Installation of the Ubuntu 20.04LTE machine in VirtualBox

Download the Ubuntu 20.04LTE Desktop image from the official website. (https://releases.ubuntu.com/20.04.4/?_ga=2.71112592.1276185110.1655983032-1464998365.1655983032)
In VirtualBox, create a new virtual machine with the following specifications:

1. type is Linux and version is Ubuntu 64bit
2. give it 2Gb of memory and create a new virtual disk, a file size of 20Gb dynamically allocated will be enough
3. in Storage: choose as disk file the ISO image you downloaded before
4. in Network: attach to NAT to be able to access the Internet if your host machine is connected to it, this will be helpful when downloading the required software. Later on you can switch to Bridged adapter to perform the laboratory work
5. power one the virtual machine and follow the instructions for the installation of the OS, select Minimal installation
6. It is suggested to install the VboxGuestAdditions. To do so, in the guest machine go to to Devices → Insert Guest Additions and follow the instructions.

Installation of Jitsi softphones

Download the Ubuntu package from the official Jitsi website and the package `jitsi-archive-keyring_1.0.1_all.deb` from the following links. The latter package is necessary for the installation of Jitsi since it does not come preinstalled in Linux.

Jitsi Desktop: https://desktop.jitsi.org/Main/Download

Installation package: http://download.jitsi.org/stable/

In the terminal of the virtual machine
```
sudo apt update
sudo apt upgrade
sudo apt install libappindicator1
sudo apt install openjdk-8-jre
```
change directory to the folder where the .deb files have been downloaded
```
sudo dpkg -i jitsi-archive-keyring_1.0.1_all.deb
sudo dpkg -i jitsi_2.10.5550-1_amd64.deb
```

You can now start the application by running `jitsi` or clicking on the application icon.

Installation of Wireshark

You can install Wireshark with

```
sudo apt install wireshark
```

Note that Wireshark can read and play OPUS codec only from version 3.4.0 onward. You can play calls with codecs PCMU and G711 with the Linux version, for example.

Appendix B: configuring Yealink VoIP phones

VOIP Phones checking or configuring via keypad:

	Check	**Configure**
IP Address	**90#	**80# then <IP address>#
Subnet Mask	**91#	**81# then <subnet mask>#
Gateway	**92#	**82# then <gateway>#
DNS1	**93#	**83# then <dns1>#
DNS2	**94#	**84# then <dns2>#
DHCP		**88# then 0# to disable DHCP **88# then 1# to enable DHCP

VOIP Phones checking or configuring via web:
Dial **90# to get the IP address of the phone. Then enter the address in address the bar of the browser. Login: *admin*, password: *admin*.
In the account settings, activate the account and change the required parameters.

Note: PC used for login should be on the same network as the phone.