**Communication Networks 2**

**VoIP measurement**

Student report v2.0

Name: Klevis Imeri  
NEPTUN: T4XGKO  
E-mail: klevisimeri11@gmail.com  
Date: 01.05.2024

Please fill in the above part of this report file with your data!  
Change the YOURNEPTUN part of this file’s name to your NEPTUN CODE!  
Please fill in the **GREEN FRAME** parts with YOUR ANSWERS OR SCREENSHOTS!  
Please do NOT take full screen screenshots! Only insert the important parts of it to this document!  
You can use ALT+Print Screen to capture only the current window or better use a tool like **Snipping Tool** which is part of Windows.

If you get stuck, first check the VOIP\_User Guide. If it does not help, you can send questions on Communication Networks VoIP measurement channel in Teams.

Submit this file to Teams channel:

Communication Networks VoIP measurement Assignments

The measurement was developed by: Kulik Ivett

Majdán András

Németh Krisztián

A képen Grafika, Betűtípus, képernyőkép, embléma látható

Automatikusan generált leírás

2021 - 2024

# Content

[Introduction 3](#_30j0zll)

[1. exercise: Start the virtual OpenWrt system 4](#_1fob9te)

[2. exercise: Asterisk on the local machine 7](#_tyjcwt)

[3. exercise: Traffic analysis with protocol analyzer 12](#_1t3h5sf)

[4. exercise: SIP trunk or a connection to a remote Asterisk telephone exchange 18](#_4d34og8)

# Introduction

The VoIP is complicated because it has to solve a complex problem. At the dawn of automatic telephone exchanges, the task was still simple: there was a central exchange, a few subscribers, a telephone number for each member and subscribers were able to call each other. Today, the telephone service is more complicated. Call forwarding, conference calling, voice-mail, call queuing, billing, authorization control, private PBX management, premium rate services and so on.  
To solve this problem, previously complex software was created, which was run on the dedicated computers, named Call Centres (telephone exchanges).

Call Center software running on a general computer has also been released. The best known of this software is Asterisk. Asterisk is pretty complicated. In the laboratory measurements were made with real VoIP telephonies (see 1. picture) connected to a real Call Centre, and a lot of students calling each other by phones. Now the circumstances are different. Everyone at home takes this VoIP measurement. But we hope we have put together a useful, understandable and enjoyable measurement. Let's start the measurement!

# 1. exercise: Start the virtual OpenWrt system

We will use a virtual OpenWrt system, which of course must run on a virtual hardware, and for this we need a ‘hypervisor’ software. A hypervisor, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). OpenWrt was originally targeted as an alternative operating system for cheap home wifi routers. Hence its name: Wrt = wireless router. It is a Linux distribution with very low resource requirements. OpenWrt quickly became popular, as it allows the always-on router to be used not only as a wifi hotspot, but for several other purposes as well, like a file server (NAS) or even a torrent client. In this lab exercise we will use OpenWrt running on a virtual machine as a PBX (Private Branch telephony Exchange).

*1. Installation of the Hypervisor:* If you do not have a virtual machine monitor on your computer, you have to install one. You can install a VirtualBox[[1]](#footnote-0) or another software wich can be used, such as a VMware Player[[2]](#footnote-1).

*2. Dowload of the VM image file:* Download the VM image file named **Asterisk\_OpenWrt17.07.7.ova** from the Teams, VoIP measurement channel -> Files part. Using of the OpenWrt means this is a small file only 1,5MB.

*3. Import a virtual machine in VirtualBox:* In the VirtualBox menu, click File/Machine Import. And you need to import the downloaded file **Asterisk\_OpenWrt17.07.7.ova** in the expert mode. You created your Asterisk OpenWrt virtual machine which will be used during this VoIP measurement.

*4. VM configuration:* If you select this new Asterisk OpenWrt virtual machine (VM) in the VirtualBox you can configure it. You need to set the Network part.

**Network**Interface 1  
Attached to: Bridhed Adapter  
Name: CHOOSE YOUR ACTIVE NETWORK CARD WHERE YOU COMMUNICATE WITH THE INTERNET

If you use wifi now, you need to select the wifi interface here, if you use Ethernet you need to use Ethernet card name here. Be very careful.

Note:  
This will create a network bridge with your network card, so your VM will get an IP address from your router. You cannot use a PC which does the PPPoE dialup or just have a mobile stick.

Advanced  
Adapter type: Paravirtualized Network (virtio-net)

Save the configuration by clicking OK!

Note:  
virtio-net has less overhead than emulating a real network adapter.

*5. Start the VM and IP address finding:* Start the virtual machine!

About 10 seconds after starting (when no more messages arrive), press ENTER and you will get a login screen.

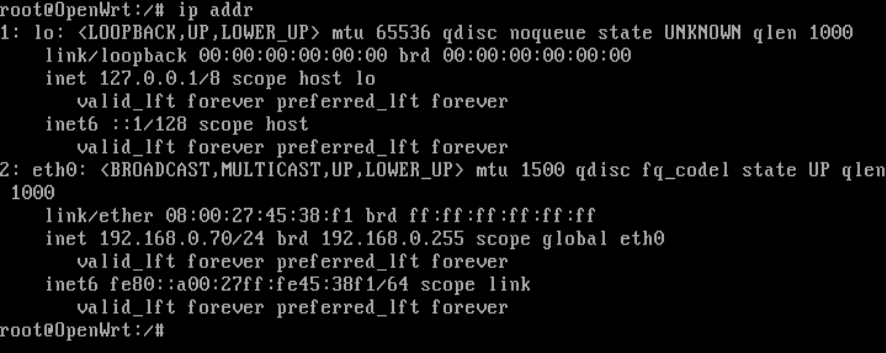
Check the IP address of the system with command:

**ip addr**

**lo interface** is the loopback interface the applications can communicate through this interface with each other on the host in a portable way (without using unix sockets). The **eth0** interface is the one we use for communicating with other computers.

**NOTE:** If you “stuck” into the virtual machine then after the “host key” written at the low right corner (by default the right Ctrl) press the Alt-Tab. And you will get out to your computer again.

**E1.1** Please take a screenshot of the settings printed by the previous command!



**E1.2** Filter out important information! What is the VM IP address and the network mask?

IP address: 192.168.0.70  
Netmask: 255.255.255.0 or /24

If you did not get an IP address to the Eth interface this means you have a problem with your VM network configuration. Please STOP your VM with the command: **halt**

Wait finishing. And check the Network configuration of your virtual machine again, and correct it and Save it. START your Asterisk OpenWrt virtual machine again. Check your IP address again and write down the E1.1 and E1.2 answers into the green frame.

*6. SSH connection:* Although you can use OpenWrt in a text mode through VirtualBox, it is more convenient to connect to it with an SSH client. We would also use an SSH client in case of a real WiFi router. Let's find an SSH client! On Linux systems, the ssh command runs from the command line, and even on Windows 10 sytem runs the ssh command. Still, it seems more convenient to use a dedicated application on Windows.

We like **putty**[[3]](#footnote-2) program, because the putty is small, simple and knows everything what you need, but of course any other ssh client will be fine.

Enter into your virtual machine (VM) by putty or other ssh program:  
*Host name (IP address): OpenWrt IP address (from previous exercise) port: 22  
Connection type: SSH*

If you receive a security warning that the server key is not registered, don’t panic. This is the first time you have joined your VM. So the public key of the VM is not yet known to the SSH client. This warning will not appear next time. Go ahead, and connect to your VM OpenWrt server.

Username: **root** (administrator name in the Unix world)  
Password: **V01Pmeres** (big V zero one big P small meres)

On Linux use command: **ssh root@YOUR\_VM\_IP\_ADDRESS**  
Press Y for security warning (if any)  
password: **V01Pmeres**

# 2. exercise: Asterisk on the local machine

Now, we want to make calls through our newly completed Asterisk call center.

To do this, we have to configure this call center and end-devices for calling are also needed. The latter will now be virtual, although if someone has a real VoIP-capable phone he/she can also try to use it.

Begin with the Asterisk call center setting!

1. *SIP file editing:* The SIP signaling protocol will be used during the measurement. Asterisk also includes two implementations of this. One is known as chan\_sip, the other is res\_pjsip / chan\_pjsip, or just known as PJSIP. Chan\_sip is still part of Asterisk but it is now obsolete. That is why we will use PJSIP.

SIP configuration file is: **/etc/asterisk/pjsip.conf**  
Navigate into this directory with the command: **cd /etc/asterisk**  
and backup this file with the command: **cp pjsip.conf pjsip.conf.bak**

(Note: If a command runs without error in Unix operation system, it typically does not display anything on the screen. This makes it easy to link commands one after the other.)

Edit this SIP configuration file with nano text editor. Use command: **nano pjsip.conf**

The most important commands in nano text editor are:

* you can SAVE your changes with **CTRL+O**  
  you can EXIT with **CTRL+X**
* you can find out which is the current line with **CTRL+C**

We can find around the 285. line in the pjsip.conf file this text: “ENDPOINT CONFIGURED FOR USE WITH A SIP PHONE” Here we need to set information about end devices which are connected to the Asterisk call center. At first two end devices (one software client and your mobile phone or another software client) will be connected.

As shown in the example where lines are commented with semicolon “**;**” a device with the identifier 6001 has got several sections, whose header is enclosed in square brackets **[ ]** . This is followed by the section type, which can be among others **endpoint, auth** and **aor**.

**[*identifier*]  
type = *section\_type***

The **endpoint** describes, among other things, what the device is capable of. The **auth** defines the authentication method, and the **aor** provides information how to access our device.

1. *UDP:* We will use UDP protocol. Therefore we need to allow the **[transport-udp]**section, which we can find around the 128. line. Delete semicolon “**;**” at the begin of the **[transport-udp]** line also delete semicolon “**;**” at the begin of the next 3 lines which are part of this section!
2. *endpoint:* Create a new section directly above the first line containing **[6001]** with the idetifier **[student-phone1]**The section type should be **endpoint**. You won't need semicolons “**;**” at the beginning of lines! Because these lines will be active, the system will use them.

Copy the **transport** from the example. This refers to the previous point.[[4]](#footnote-3) We need the **context** as well. This line tells you which call group the calls coming from here belong to. Do not copy this one by one, let it be the value **from-internal** So, this line in the file will be like this:  
**context=from-internal**

Next two lines (**disallow, allow**) in the **[6001]** example show which codecs will be used.

Only **ulaw** is allowed in the example, which is the PCM μ-law, it is the US PCM version.   
Here in Europe we prefer the A-law (**alaw**) version! Insert these two lines with this change!

After that, all you have to do is to write where the authentication and aor information are located. These sections will also be created soon.  
Now copy/past it from the example **[6001],** rewriting the identifier to **student-phone1**

1. *auth:* New section, the section header is the same, the type is **auth**. Fill in the other three lines based on the example **6001**. Authentication with the username and password, which you need to enter in the next two lines. Username is **student-user1** and the password is **4321** The Asterisk call center will expect this information from the end-device (software client).
2. *aor:* New section, the section header is the same, the type is **aor**. What is this **aor**? *Address of Record,* it’s a bit of a complicated concept, but it’s about knowing where the Asterisk call center is reaching the end-device. This will require an IP address, but as this may change, the end-device will send it when the end-device will be registering to the Asterisk. So now the IP address is not written here. Although it is in the example 6001, do not copy this line! All you need is the type of section (**aor**) and the maximum number of end-devices that can connect to the Asterisk call center as student-user1. This number is one, therefore copy the relevant line from the example! Always delete semicolons “**;**” at begin of lines!

We add another line to the end:  
**remove\_existing=yes**

If there is an already stuck registration (a frozen client has not logged out before), it will be discarded when the new registration arrives.

This completes the configuration for **student-phone1**.

1. *Once more:* We need two end-devices. So we need repeat the whole configuration for the second end-device as well. So, repeat the previous three points (3,4,5) inserting new sections, just now write **student-phone2** everywhere. Both the username for authentication (**student-user2**) and the password (let it be **9876**) must be rewritten. Be very careful!
2. *Done:* SAVE the file (Ctrl+O) with the original filename and EXIT (**CTRL+X**)
3. *Reload:* The Asterisk software, running as a service in the background, must be instructed to read its settings again. To do this, start the Asterisk command line interface with command:

**asterisk -r**

Then **inside the Asterisk terminal** the configuration files re-read is needed with command:

**core reload**

You can then see any **error messages** if something has been written wrong in the configuration. However, it only shows the errors once, if we don't save the config file again, Asterisk won't reload it. In addition, not all mistakes are our fault, for example, **some errors are related to other part of system, they do not bother us**. For example:

ERROR[20981]: config\_options.c:710 aco\_process\_config: Unable to load config file 'cdr.conf'  
NOTICE[20987]: sorcery.c:1334 sorcery\_object\_load: Type 'system' is not reloadable, maintaining previous values  
ERROR[20987]: res\_pjsip/config\_system.c:261 system\_create\_resolver\_and\_set\_nameservers: There are no local system nameservers configured, resorting to system resolution  
WARNING[20981]: pbx.c:8717 ast\_context\_verify\_includes: Context 'local' tries to include nonexistent context 'parkedcalls'

Note: You can also get more information from Asterisk. As we have seen, starting the program with the asterisk –r command we will get some messages about the operation, which we can need for debugging. **So you will get even more messages if you start Asterisk with this command:**

**asterisk –rvvvv**

The Asterisk document about this command only says “the more the letter v, the more information in output”.

And in the running interactive Asterisk, the following command specifically allows you to display more details about the pjsip module:

**pjsip set logger on**

Exit from the Asterisk terminal with command exit or quit

Note: You can re-read the configuration without going into the Asterisk with command:  
**service asterisk reload**

or with the command: **/etc/init.d/asterisk reload**

You can even **restart Asterisk** with the command: **service asterisk restart**

However, we chose interactive login because that is the only way we can see any error messages!

**E2.1** Copy the newly written sections of **pjsip.conf** file here into the frame!

[student-phone1]

type=endpoint

transport=transport-udp

context=from-internal

disallow=all

allow=alaw

allow=gsm

auth=student-phone1

aors=student-phone1

[student-phone1]

type=auth

auth\_type=userpass

password=4321

username=student-user1

[student-phone1]

type=aor

max\_contacts=1

remove\_existing=yes

[student-phone2]

type=endpoint

transport=transport-udp

context=from-internal

disallow=all

allow=alaw

allow=gsm

auth=student-phone2

aors=student-phone2

[student-phone2]

type=auth

auth\_type=userpass

password=9876

username=student-user2

[student-phone2]

type=aor

max\_contacts=1

1. *Dialplan:* The end-devices could already be connected to the Asterisk, but something is still missing. We need a telephone number for calling, and this has not even been mentioned yet. This is because it does not belong to the SIP module it must be set somewhere else.

We can set the phone-number in the file **extensions.conf** in the same directory. At first backup this file with the command: **cp extensions.conf extensions.conf.bak** Now, we can edit the **extensions.conf** file with the command: **nano extensions.conf**

It is a long, complicated file. **We simply add to the beginning what we need.** The first line should be the identifier of this section in square brackets like in the pjsip.conf file. The section name should now be **from-internal** (calls from **inside**). So, the first line will be: **[from-internal]**

The dialing rules come down. These are in the following format:  
**exten => 300,1,Dial(PJSIP/student-phone1)  
exten => 300,n,Hangup()**

The first line means that if the called extension is 300 then as a first step (1) dial and ring the student-phone-1 end-device, defined in the PJSIP module, by sending the corresponding SIP message. Then it rings, and when it is picked up, the connection is established.

When Asterisk is done with this, according to the second line: we are still talking about the number 300, the next priority task (n) is to end the conversation (**Hangup()** = hang up).

Following this pattern (still in the same section), type these two exten lines also for extension 301 to rign the student-phone2 end-device! Save the file and exit!

**E2.2** Copy the newly written first five lines of **extensions.conf** here!

[from-internal]

exten => 300,1,Dial(PJSIP/student-phone1)

exten => 300,n,Hangup()

exten => 301,1,Dial(PJSIP/student-phone2)

exten => 301,n,Hangup()

1. *Dialplan Reload:* Reload the config files with one of methods written in the 8. *Reload:* point.
2. *Installation of SIP clients:* We need two end-devices for calling. Two devices which are connected to your home network, it has a SIP client running on it and it has a microphone and speaker.

*Installation of SIP clients:* We need two end-devices for calling. Two devices which are connected to your home network, it has a SIP client running on it and it has a microphone and speaker. They will be your computer and mobile-phone (if you use your home WiFi on mobile-phone and not mobilnet). But it could be also another computer or tablet. For example, you can use MicroSIP application under the Windows system. (You can dowload it for the site: [**https://www.microsip.org/downloads**](https://www.microsip.org/downloads) )

It's a pretty small program, and there's a "portable" version, so you don't even need to install anything on your computer. It’s worth noting that when MicroSIP is closed, it only hides in the taskbar! **You need to permanently exit with Exit**! Under Linux system, the Linphone application will definitely be good, but you can use also something else.

The second device, your mobile-phone need a SIP client too. From the same website [**https://www.microsip.org/downloads**](https://www.microsip.org/downloads) you can also download a SIP client named **ZoiPer** onto your mobile-phone, if you have Android. Or it can be anything else like that. After ZoiPer installing, ZoiPer aggressively try to force the paid version, feel free to reject these attempts (by SKIP button)!

1. *Configuration of SIP client:* Configure your SIP client on your computer! Account Name will be **student-phone1** The SIP server and the Domain is you IP address of your Asterisk call center. It is the VM IP address from exercise E2.1. The Username is the identifier **student-phone1** The SIP configuration was created for this. Login and Password are required for authentication (in pjsip.conf). Login user for authentication **student-user1** and password **4321** You can set also Display name it is the name which will be displayed on the called side during the ringing. Everything else should remain the default! (In principle, we also need to set the PCM A-law codec to be the only one active codec, but in the software I was looking at, that was the default.)

Note: In case of Linphone-application only one usename can be set, Use the username **student-phone1**. Later during the registration you will be asked about the authentication user and password, In this point you need to set **student-user1** and **4321**

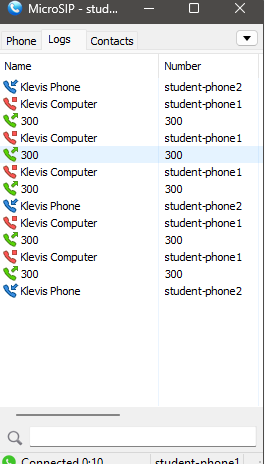
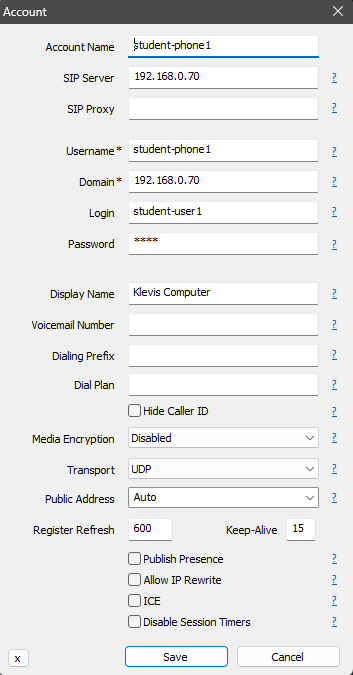
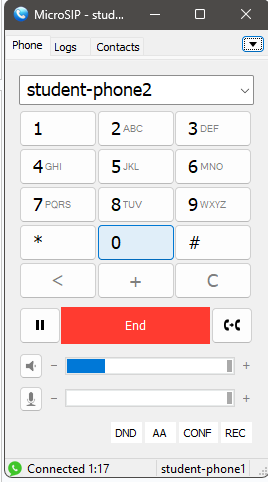
What might be still interesting is the “expiration” or “register refresh”. This tells you how often the client will re-register itself. For **ZoiPer**, this default time is 60 seconds, which is too often and it could take a few seconds. So, change this value to 5-10 minutes and even 60 minutes. You need to write this value in seconds!

Then set up the mobile-phone client (or another computer client) in the same way, except that the user name and the name-password pair used for authentication will be different!

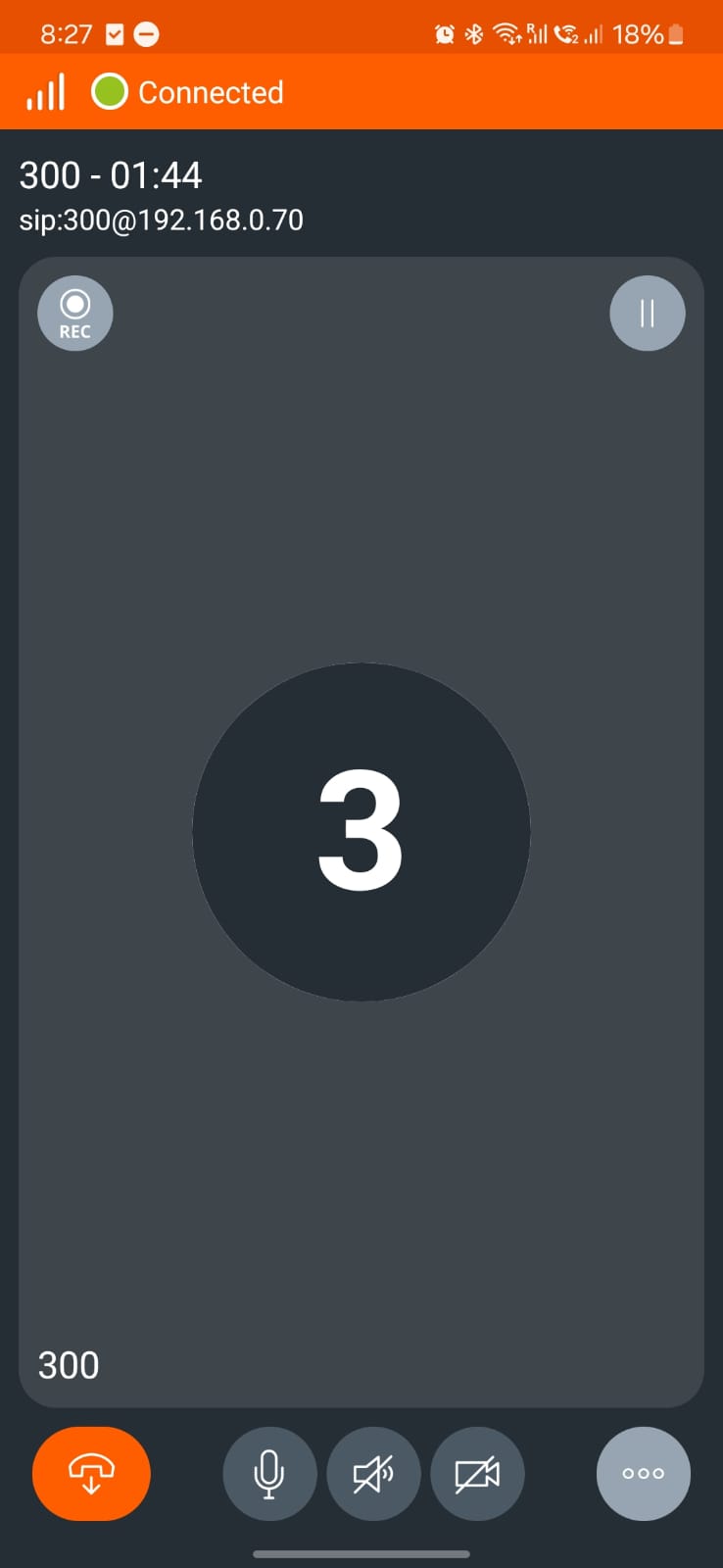
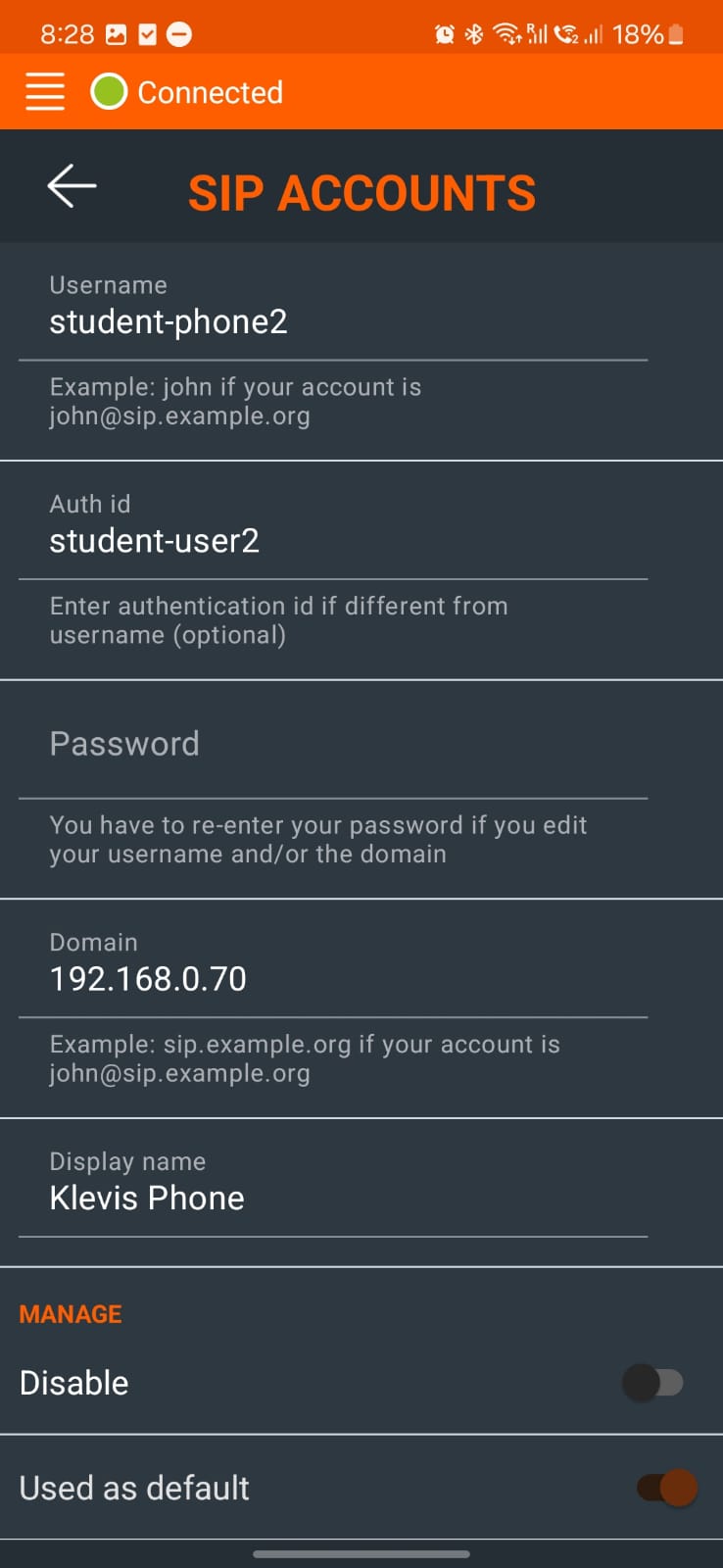
1. *Call:* Call your mobile from your computer (**301**) and vice versa **(300**). Pick it up, talk to yourself a little. If someone is there, you can talk together. If it works, let's be happy! If not, fix it!

**E2.3**  Put here the screenshot about your talking (either mobile or computer screenshot)

From computer:



From phone:



1. *Let’s play a little bit!* What if you call yourself (I mean student-phone1 for example 300)? What does Auto Answer and DND (do not disturb) do in MicroSIP? Try it if you would like to.

*Exact time:*  There is a phone number that, when we call it, it will tell you the exact time. We can do it if the service is implemented by the service provider. This is the number **180**. We will implement this service into our Asterisk call center.

To do this, we need to make three lines into the **extensions.conf** file:

**exten => 180,1,Answer()  
exten => 180,n,SayUnixTime()  
exten => 180,n,Hangup()**

The first line answers the call, the connection is established. The rest should be clear.

1. Let's try it: write these line into the **extensions.conf** Reload the config files with one of methods written in the 8. *Reload:* point.

**E2.4** What is the first information (first word) that the machine voice says when we call this number? Is the time zone same like your local time? Fill the frame with your answer

What is the first information? The day of the week (in my case wednesday)

Timezone is the same like in your country?   
No it uses UTC (GTM) but I am in GMT+2.



(Check the command **date** on the OpenWrt. You can compare the output with the timezone.)

# 3. exercise: Traffic analysis with protocol analyzer

1. *Wireshark installation:* Download Wireshark on your computer from the site <https://www.wireshark.org/download.html> and install it! If you need to restart your computer, at first shut down your VM OpenWrt with the **poweroff** command!
2. *Tcpdump:* Warning! VirtualBox's bridged network solution bypasses the Windows network stack, so we can't start catching packets directly on the network interface. The solution is to receive packets within the VM OpenWrt and then forward the received traffic via SSH to the input of the Wireshark program on your computer. Sounds complicated, but we help with it.

At first install **tcpdump** on the virtual machine OpenWrt to catch packets:  
**opkg update  
opkg install tcpdump**

1. *Start of the packages’ catching :* Two things you need to do. You need to start the following Windows command with the goal of the traffic capturing on the guest OpenWrt operating system and you need to redirect this traffic to the Wireshark program on your computer through an SSH connection on you computer. Open a command prompt window or terminal window (in Windows with **cmd** command) and write the command (in row):

**ssh root@*YOUR\_VM\_IP\_ADDRESS*** "**tcpdump -ni eth0 -s 0 -U -w - not port 22**" **| wireshark -k -i –**

**IMPORTANT:** In case of Windows 10 you need to writhe the whole path of the Wireshark program! "C:\Program Files\Wireshark\Wireshark.exe" into the command or at first you need to change the directory with the command: **cd "C:\Program Files\Wireshark"**And rewrite the ***YOUR\_VM\_IP\_ADDRESS*** to your own VM OpenWrt IP address before running it!

When it is our first connection by SSH you need to answer by YES before the password – you will not see what you will write.

Note the meanings:  
-s 0 capture whole packets  
-n we don’t want to convert addresses to names  
-i eth0 capture on the eth0 network interface  
-U prevent buffering  
-w - write not to a file, but to STDOUT (so to the console if not redirected)  
not port 22 we don’t want to capture packets of SSH  
| feed the output of the previous command to the following command’s input  
-k start capturing immediately  
-i - read packets from STDIN (so from the console if not redirected)

And then you need to write the password. In this point Wireshark starts and collects packages. If you do not have Windows 7 and everzthing works you can go to point 5 *Capture!*

1. *plink:* This point is needed if your computer has got Windows 7 system! But if the *3. point Start of the packages’ catching* didn’t work correctly this plink point could be used as well. Windows 7 syetem doesn’t include directly SSH command. So, you need to download the **plink.exe** for SSH redirection under Windows system from the Putty program website: [**https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html**](https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html) Copy this **plink.exe** file into your Wireshark installation directory (this directory should be C: \ Program Files \ Wireshark) On Linux the built-in SSH command will be fine, no need to get plink! See end of the next point. WARNING: plink.exe will work ONLY if you have run putty.exe with authentication key before plink.exe! Because plink.exe is part of the putty package!

The following Windows command starts the traffic capture on the guest OpenWrt operating system and the Wireshark program on your computer. And this creates the connection between these two systems. Open a command prompt window in Windows with **cmd** and go to the Wireshark directory:

**cd "C:\Program Files\Wireshark"**

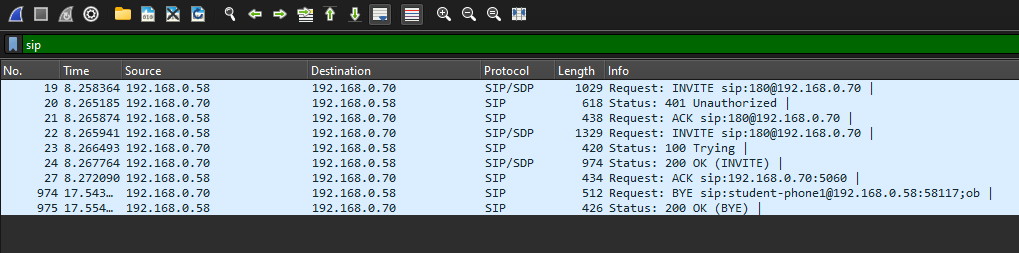
Then write the following command (in a row):

**plink -batch -ssh -pw *PASSWORD* root@*YOUR\_VM\_IP\_ADDRESS* "tcpdump -ni eth0 -s 0 -U -w - not port 22" | Wireshark.exe -k -i -**

**WARNING:** In the above command, of course, rewrite ***PASSWORD*** and ***YOUR\_VM\_IP\_ADDRESS*** to your own password and VM OpenWrt IP address before running it!

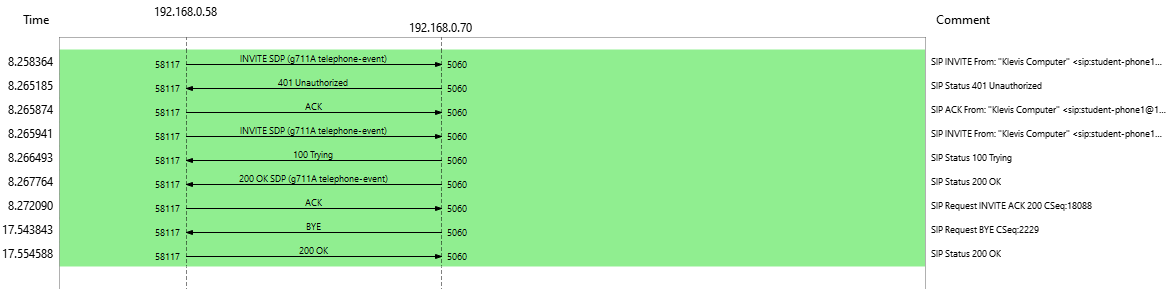
1. *Capture!* Now make a call to **180**-from your MicroIP client and listen to the exact time. Then stop the packets capturing in Wireshark. The packets for this call should appear in the Wireshark.
2. *Interpret the packets!* Set the display filter to **sip** in Wireshark on the top of the Wireshark window. (Apply a display filter…)

**E3.1** Take a screeshot of the captured packets in the Wireshark!



1. *SIP Flow:* Now in the Telephony menu, choose SIP Flows (in Wireshark)!  
   Choose the flow and then click "Flow Sequence"! A chronological representation of the SIP packets in the stream is displayed

**E3.2** Take a screenshot of SIP Flow sequence!



1. *Authentication:* If everzthing is going well, in this Sip Flow sequence you should see an INVITE, 401 Unauthorized, INVITE, OK message sequence (this is a call setup)

Note:  
If you see more INVITE messages then you have forgot to enable only the G.711 A-law codec. In this case the first two INVITE messages contain all the supported codecs and the third (in-dialog) INVITE (and the second OK message) contains the chosen codec.

But why are there two INVITE messages?  
And what is that “401 Unauthorized” message?

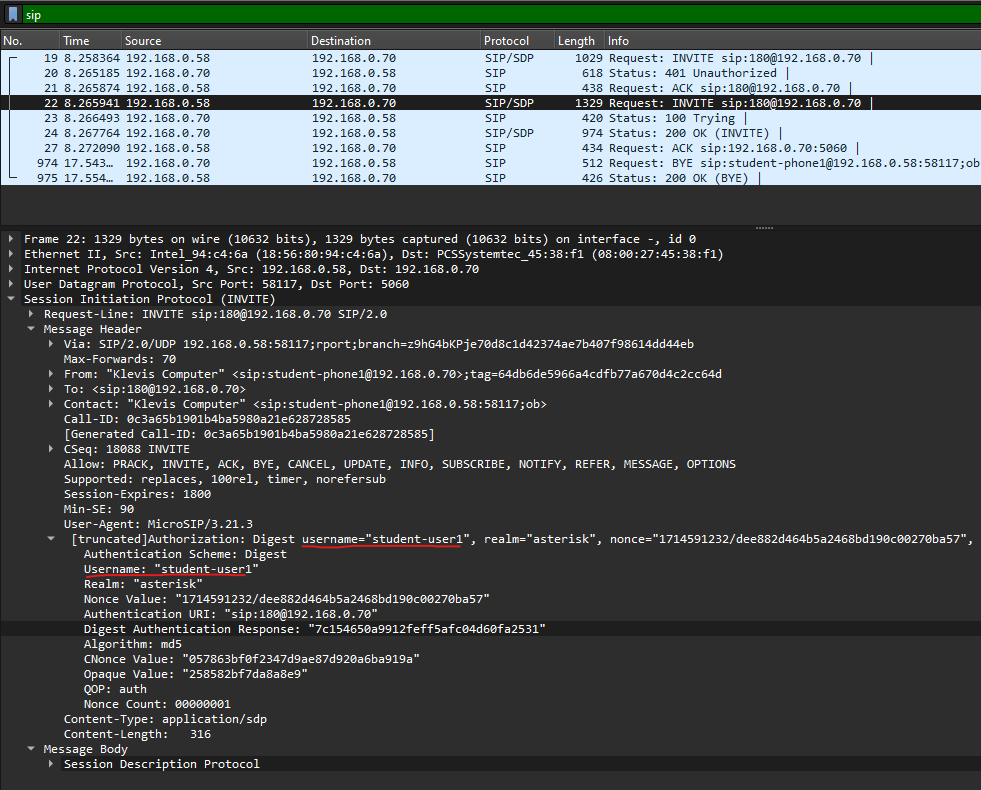
If you look at the SIP message header of this first INVITE, you will see that there are no authentication at all. However you cannot call without authentication (you had setup a password in **/etc/asterisk/pjsip.conf**. So Asterisk send back a “401 Unauthorized” message. Moreover, it also tells you how to wait for authentication.

Find this part of the “401 Unauthorized” message’s header:  
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="5930c42c"

This part tells that you have to authenticate yourself by using MD5 hashing.  
Now find your username and password in the second INVITE message’s SIP header! You will find your username in clear text, however the password in the “response=” field is hashed.

**F3.3** Take a screeshot about the authorization part! Mark the user name with a red underline! Put here the authentication part!

The Username is underlined with red.   
The authentication response (i.e Digest Authentication Response is:  
**7c154650a9912feff5afc04d60fa2531**



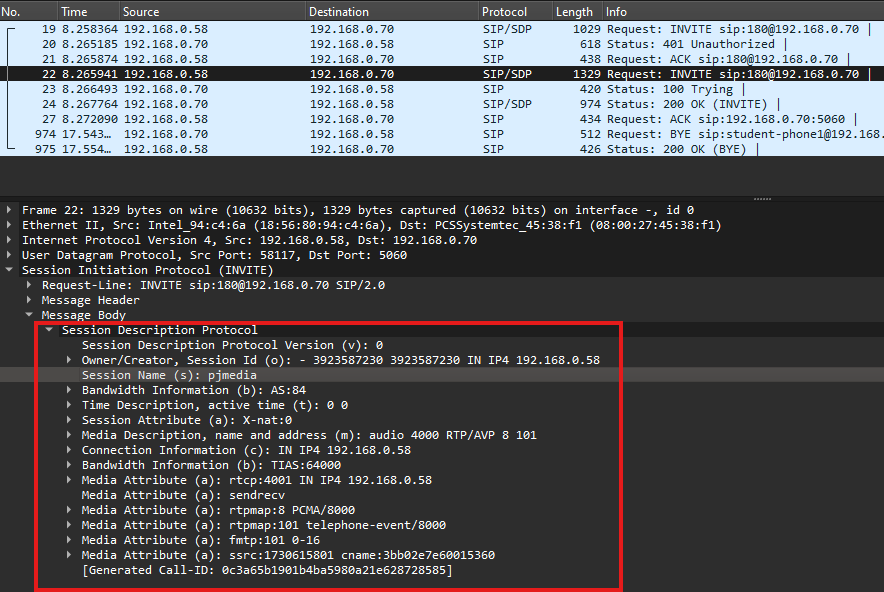
Note:  
So the password was not sent in cleartext. But you can repeat the same message (repeat attack), can’t you? No, you cannot because the password is hashed with the previously given nonce, which is changing. This is a kind of “challenge–response authentication”.

1. *SDP - Session Description Protocol:* As you know SIP is only a signaling protocol and it uses a separate protocol – RTP - for carrying the voice. But how the parties known how to connect to each other? What IP, ports and codec should they use for RTP?

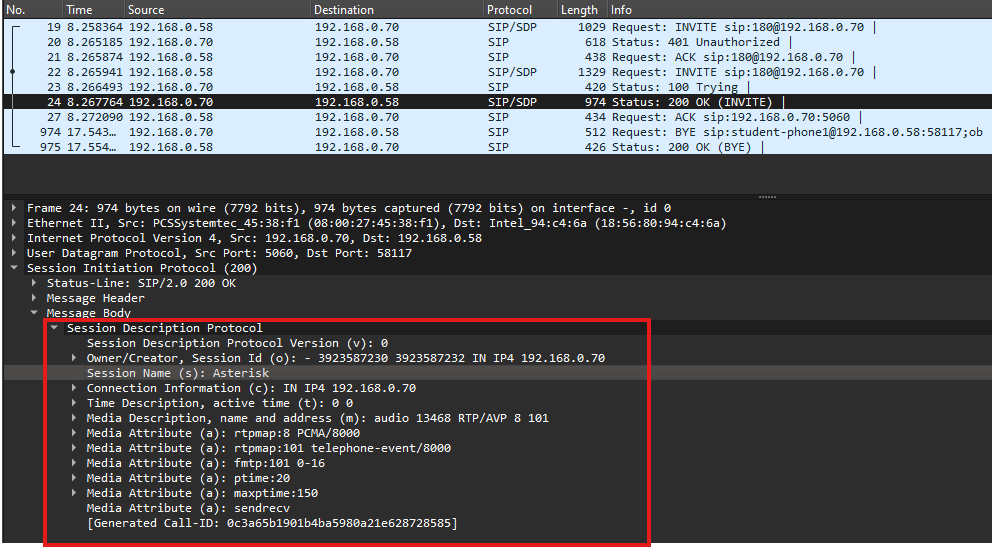
They agree on these parameters in the SDP header of INVITE/OK messages.

Check the INVITE’s and the first OK’s “Message Body” for the “Session Description Protocol” part!

**E3.4** Take a screenshot of Session Description Protocol part of the second INVITE SIP message!



**F3.5** Take a screenshot of Session Description Protocol part of the first 200 OK message!



Under the “Media Description, name and address (m):” part of these messages you will find the “Media Port:” field which contains the RTP port number of each party. Also the “Media Format:” field contains the voice codec. The IP address of each party is under the “Connection Information (c):” in the “Connection Address:” field. (In the Wireshark click on the small triangle in front of the relevant row to get a detailed explanation of the data.)

**E3.6** Write the read data below instead of XXXXX!

End-device (client) IP address: 192.168.0.58 RTP Port: 4000  
Asterisk call center IP address: 192.168.0.70 RTP Port: 13468

Type of codec: ITU G.711 PCMA 8(a-law)

Note: i found the type of PCMA at Media Attribute -> Media Format: 8  
8 stands for a-law and 0 for μ-Law.

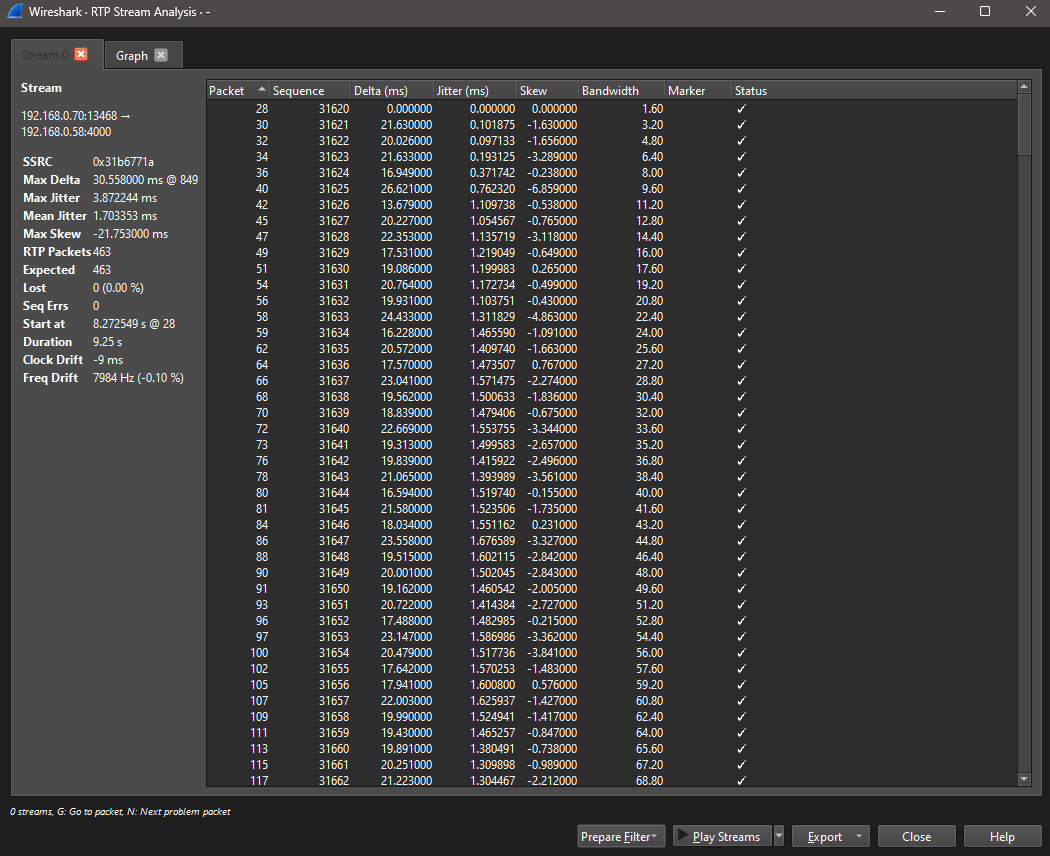
**Note:**  
5060 is certainly a WRONG answer for the RTP port because it is the port number of SIP protocol!

1. *SIP theory:* But why do we need the IP address of each party? Why can’t we just use the SIP messages’ source IP address?

Because it is the main benefit of SIP, the signaling path requires only few resources (eg. bandwidth) while the voice path consumes a lot. You can use a low-end router as your PBX without passing the voice through it (so the signaling goes through the router, but the voice path is between the parties). In this version of Asterisk this configuration option is called **direct\_media** in **pjsip.conf** file with default value **yes**. In larger PBX systems with thousands of SIP extensions it is not uncommon that it contains multiple DSP cards. Each DSP card has its own IP address, so when you make a call, in the INVITE’s SDP field you will find the IP address of one of these cards. So the main CPU of the PBX gets less load because it has to only process the signaling (SIP).

# *RTP investigation:* Clear the SIP display filter in Wireshark! (x at the right) In the Telephony menu, choose RTP -> RTP Streams! You will see two streams. One stream flows from your computer (source address=IP address of your computer) to the Openwrt (destination address=IP address of your OpenWrt). and the other stream flows from your Openwrt (source address=IP address of your OpenWrt) to your computer (destination address=IP address of your computer). This is due to the two-way connection in case of exact time query. Choose the RTP stream of “exect time announcement” This is the flow from the Asterisk center (OpenWrt) to the end-device (your computer). And click “Analyze”!

**F3.7** Take a screeshot of the analyze screen of Wireshark!



Wireshark analyzes the entire RTP stream and provides various statistics about it.

1. *Interpret it!* Wireshark analyzes the entire RTP stream and provides various statistics about it. Examples include the mean jitter, the number of RTP packets, the number and proportion of RTP packets lost, the time length of the stream, and the average bandwidth used by the stream (in kbit/s). With the exception of the latter, we can read the data from the summary table on the left, while the estimated bandwidth is given by Wireshark per packet, so you have to scroll to the bottom of the packet list to read the stabilized average!

**E3.8** Write the following data from the forward flow (Asterisk → end-device)!

mean jitter (with unit of measure!): 1.703353ms

the number of RTP packets: 463

the proportion of RTP packets lost (%): 0(0.00%)

Duration ((with unit of measure!): 9.25s

the average bandwidth (kb/s): 80.00kib/s~**81.92kb/s**

**Note:**  
The “kilo” is a decimal-based prefix in the SI system and refers to multiplication by a thousand. Thus, 1 kB = 1000 bytes, 1 kb = 1000 bits, 1 kb/s = 1000 bits/second, and so on. “Kibi” is a double-based prefix that refers to multiplication by 210 = 1024. Thus 1 KiB = 1024 bytes, 1 Kib = 1024 bits. For data rates, we traditionally use the powers of 10, the kilo and not the kibit.

1. *Overhead components:* If you measured correctly, the last number in the previous exercise - the average bandwidth - does not match the 64 kbps learned from PCM, but the value is more than that. Probably because 64 kbps is the data rate emitted by the codec only, and the bandwidth displayed by Wireshark also includes the packet headers.

There are a lot of headers: from the inside out there are the headers of RTP, UDP, IP and Ethernet. Open any RTP package from the saved packages with Wireshark! It won't be hard, there's a lot of it. If you select different fields, at the bottom, in the status bar, Wireshark shows the length of that field.

Let’s count!

**E3.9** How many bytes (not bits!) are in a whole RTP packet?

The whole RTP packet: 214 bytes

The Ethernet header: 14 bytes

The IP header: 20 bytes

The UDP header: 8 bytes

The RTP PDU, RTP header + Payload (data): 172 bytes

- only RTP header from it: 12 bytes

- only Payload (PCM encoded audio): 160 bytes

1. *Overhead counting:* The Wireshark did not take the header of the Ethernet frames into the counting when determining the bandwidth. So we will count also only with IP, UDP RTP headers.

**E3.10** Let's count based on the answers to the previous two questions. Write your answer to the questions below in the box.

1. If the PCM data rate is 64,000 bits / s (= 8,000 bytes / s) and a packet contains the number of useful bytes shown in the answer to the last subquestion of the previous question, how many packets should be sent per second?
2. Is this really happening? Divide the number of packets shown in the answer to question 3.8 by the length of the stream also shown there. This is the measured values of the number of the RTP packets per second. How much is this?
3. Are the answers to questions a) and b) close values?
4. What is the total length of IP + UDP + RTP headers in bytes?
5. Based on this information, what is the percentage of overhead, with other word “wasted” (transport) work? overhead = total length of headers / total packet length (NOTE! The packet legth is therefore only the packet without the Ethernet header. As well as the total length of the headers is also without the Ethernet header.)
6. How many bits (not bytes!) headers are transferred in total per second?
7. So 64,000 useful bits are transferred per second, plus the amount of "useless" bits given in the previous answer. So what is the total required bandwidth (bit rate)?
8. How does this bandwidth value relate to the last sub-question answer of the 3.8 question?

a:50 packets/s 8000/160

b:50.0540 packets/s 463/9.25

c:yes

d:40 bytes

e:0.2(20%) 40/200

f:16000 bits 50\*40\*8

g:80000 bits/s 64000+16000

h:They are nearly the same because Wireshark measures the bit rate including the headers (i.e of the whole packet) and that is what we did here.

1. *Replay.* A little play with the Wireshark. Wireshark is also able to play sounds for us. Going back to Telephony / RTP / RTM Streams, you can select one, the other, or both streams. Then select Analyze to bring up the already known window, and here select Play Streams to bring up the audio player. If we selected both streams, we will hear one part in the left channel and the other part in the right channel. Cool!
2. *End of this exercise:* Close Wireshark and press Ctrl + C in the command line window where plink.exe or ssh is running, stopping them. You can also close that command window on your computer.

# 4. exercise: SIP trunk or a connection to a remote Asterisk telephone exchange (OPTIONAL EXERCISE)

It would be nice to use our created phone (client) for something really good: to connect us with our friends. Therefore, we created an Asterisk-based telephone exchange on a departmental TMIT server (**voipmeres.tmit.bme.hu**).

Your last exercise is to build a trunk connection between your Asterisk and the Asterisk at the TMIT department. Trunk is a type of connection which can be used between the call centers (telephone exchanges)

If your friend or classmate is measuring at the same time as you, and you both get to this exercise, you can even call each other! Of course, this is not obligatory, without this calling it will make sense as well.

Figure 2 shows the system which has to be implemented. Call numbers are marked in blue, we will write more about them later.



*Figure 2 The architecture of the measurement*

To implement this system, call centers (telephone exchanges) must be configured on both (or all three) sides. Asterisk is already running in the TMIT department.

At first check your correct connection with the BME and TMIT network with next two ping commands

**ping** [**www.bme.hu**](http://www.bme.hu) and **ping voipmeres.tmit.bme.hu** from your computer.

If neither of them work correctly problem is on your side!  
If both go, there is no further action you will able to communicate through the BME network  
If only the **ping voipmeres.tmit.bme.hu** doesn't go, then go to the page:

[**http://voipmeres.tmit.bme.hu:9999/likemyip/**](http://voipmeres.tmit.bme.hu:9999/likemyip/) and press the “Allow my IP” button! Then the ping must go! (don’t share this link with anybody!)

1. *Preparations:* You will need some plus information to configure. By the time you read this, you have already received an email to your **edu.bme.hu email-address** in which we have provided this personal information:

* Your Neptuncode (you obviously knew this)
* Call number (where you will be reached)
* Your username
* Authentication username
* Authentication password

Find this mail you will need it now!

1. *PJSIP configuration:* Edit your **pjsip.conf** file with **nano** editor.

**nano /etc/asterisk/pjsip.conf**

We will set two things here:

* First a trunk connection to the TMIT Asterisk server where you need to register your own Asterisk call center (telephone exchange).
* Second, you need to set up the TMIT Asterisk as a remote endpoint, where our outgoing calls can go and where an incoming call can come from.

1. *Trunk connection with registration.* Around the 162 line in the pjsip.conf file you can find the “OUTBOUND REGISTRATION WITH OUTBOUND AUTHENTICATION” part, this part is important for us now. Here you can find two sections: **[mytrunk]** and **[mytrunk\_auth]**. We will change some lines here.

Let’s start with the **[mytrunk]** section. Rewrite the header to **[voipmeres-trunk]** Thus, indicating that we are connecting to the server called “voipmeres”. Delete the semicolon “**;**” from begin of lines (including the header line). The **type** and **transport** lines are good and probably understandable as well. We want to register to a remote server and we will use UDP protocol. You need to register because the remote server does not know your IP address.

In the line **outbound\_auth** we specify the name of the section which describes the authentication. Rewrite this name to **voipmeres-trunk\_auth**.  
So this line will be: **outbound\_auth=voipmeres-trunk\_auth**

Note that this is not an **auth** like we had before. Here we have **outbound\_auth** as it belongs to outbound registration!

The **server\_uri** should be **sip:voipmeres.tmit.bme.hu** This does not need more explanation.

The **client\_uri** should be **sip:YOUR\_USERNAME@voipmeres.tmit.bme.hu**, where **YOUR\_USERNAME** is the username from the email what you received! The username format is **studentNNN-pbx** where *NNN* is a number (1, 2 or 3 digits).

The **contact\_user** should be the same username as in the previous line (ONLY the username without the @ and what is after).

The **retry\_interval** tells you in how many second we will try again in case of error. The value 60 will be fine.

The **forbidden\_retry\_interval** is also a variation when the server did not allow us in.

It shouldn't happen, but if it does, don't wait 10 minutes: take this value to 60!

The **expiration = 3600** How often to re-register (after successful registration). The default value is 3600 seconds. = one hour. It is good.

The **line = yes** Good as it is. It is using for binding incoming calls with outgoing registrations

The **endpoint** should be **voipmeres-trunk**

So far is good. Now the **[mytrunk\_auth]** section comes. Quickly rewrite the header to **[voipmeres-trunk\_auth]** based on the above! Let's take out the semicolons! Save often (in nano texteditor Ctrl O)!

The first two lines (**type, auth\_type**) are good this way. You received your authentication username and password via email to your edu.bme.hu address.  
**WARNING:** username here is the "authentication username" and not the "username"! The form of your "authentication name" is ***studentNNN-user***, where ***NNN are numbers***. Enter the received data form the email!

The **realm** should be **voipmeres.tmit.bme.hu**

The registration settings for the outgoing trunk are complette now. Save the settings, but don't exit the nano texteditor yet!

**E4.1** Copy the contents of the just created two sections here!

Put it HERE

1. *Endpoint setting:* We set up the remote Asterisk call center as an endpoint for us. You don't have to go far, the next part in the config file "ENDPOINT CONFIGURED AS A TRUNK, OUTBOUND AUTHENTICATION" is what we need now.

All three sections here carry headers **[mytrunk]** we will replace them with [**voipmeres-trunk]** !

The first two and the fourth lines of the first section (**type, transport, disallow**) are as good as they are. The context should be **from-external** and rewrite **allow** from **ulaw** to **alaw**. We configure an endpoint, UDP is the transport protocol, we describe the rules for incoming calls from here in the **[from-external]** section of the dialplan, and we allow the use only the PCM A legal codec.

The **outbound\_auth** should be **voipmeres-trunk\_auth** and the **aors** should be **voipmeres-trunk**

Remove the semicolon in front of the line **direct\_media = no** leave the other two NAT options (**force\_rport, ice\_support**) with semmicolon or even delete these two lines.  
And write the following lines:

**from\_user=studentNNN-pbx  
send\_pai=yes  
send\_rpid=yes**

The first line has to be replaced with your own username!. This line sends the username to the remote Asterisk call center, which will use it to identify you. This is very important! The second and third lines send additional information about you which will be useful in case of the caller ID display but this is not currently taken into the account. No problem.

You have already updated the title of the next section. The type remains **aor** you need this line. However, you only need one **contact** with the following content:  
**sip: voipmeres.tmit.bme.hu: 5060** This tells us where the endpoint is.

The type of the final third section is **identify** (remove the semicolon “;” in fron of it). This describes how we authenticate incoming requests on the trunk. Mostly no way, or simply based on IP address. Therefore, rewrite lines 2 and 3 as follows:

**endpoint=voipmeres-trunk  
match=voipmeres.tmit.bme.hu**

It’s done! The next section in the configuration file (“ENDPOINT CONFIGURED AS A TRUNK, INBOUND AUTH AND REGISTRATION”) does not need to be configured now. This is only interesting at the other end of the trunk. We set it up on the remote vipmeres.tmit.bme.hu server.

**E4.2** Copy the contents of the just created three sections here!

Put it HERE

1. *Dialplan.* After saving, close **pjsip.conf** and open **extensions.conf** for editing.

Two things need to be entered: direct outgoing calls to the department’s Asterisk remote call center, and handle incoming calls from there.

1. *Outgoing calls:* The plan is to dial the number 9, as it used to be called, “to get a city line,” after this prefix you have to dial the department’s server numbers. So, if you want to call a friend whose phone number is for example **2566**, you can do it by dialing **92566**

Call numbers on **voip.tmit.bme.hu** Asterisk server are four digits. So, you need to write in the config file that if a five-digit number starts with 9, forward the call on the trunk after cutting the first digit (9). All this need to be in the [from-internal] section, because it is a call initiated from an internal number. You need to write exactly these lines:

**[from-internal]  
exten => \_9xxxx,1,Dial(PJSIP/${EXTEN:1}@voipmeres-trunk)  
exten => \_9xxxx,n,Hangup()**

1. *Incoming calls:* For simplicity, all incoming calls are routed to the same end-device. Let it be the client running on our computer (MicroSIP or something similar). Write these lines into the **extensions.conf** file after the last line of the **[from-internal]** section:

**[from-external]  
exten => *TEL\_NUMBER*,1,Set(CALLERID(num)=9${CALLERID(num)})  
exten => *TEL\_NUMBER*,n,Dial(PJSIP/student-phone1)  
exten => *TEL\_NUMBER*,n,Hangup()**

***TEL\_NUMBER*** rewrite here your externally accessible phonenumber! You have got it by email to your edu.bme.hu mail address. It is a four-digit number. The first digit of it is 2. The first line converts the displayed number: it puts a 9 in front of the caller's incoming number, so in this way we can call it back. In the second line, say toward which endpoint you are directing incoming calls. The third line hangs up. Save the updated file!

**F4.3** Copy the first part of **extensions.conf** file. Everything you've written so far in the Exercise 4!

Put it HERE

1. *Configuration refresh:* Reread the Asterisk configuration file with command in the OpenWrt terminal:

**service asterisk reload**

Or by one of the similar solutions described in Chapter 2 point 8. Reload.

1. *Try the call*  Call **9002** on the remote Asterisk server. To do this, of course, you need to dial 99002 numbers. You can use any of your clients to dial. The best is the computer’s client MicroSIP or other one.

**E4.4** What did you here?

Write it HERE

**E4.5 Optional:** You can also call another student by his/her callnumber, if he/she finished the whole 4 Exercise correctly.

What was your experiment, could you call another student?

Write it HERE

End of the measurement ☺

1. <https://www.virtualbox.org/> [↑](#footnote-ref-0)
2. <https://www.vmware.com/hu/products/workstation-player.html> [↑](#footnote-ref-1)
3. <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> [↑](#footnote-ref-2)
4. PUTTY: Selecting lines with the mouse, these are immediately copied to the clipboard you do not need to do more. And PASTE with the right button on the mouse. you can paste also with the combination Shift+Insert [↑](#footnote-ref-3)