

Kommunikációs hálózatok 2

IP feletti beszédátvitel (VoIP) mérés távolléti mérés

Mérési jegyzőkönyv v1.98

Mérést végezte: **Rittgasszer Ákos, Z8WK8D**

(a mérést egyedül kell elvégezni)

Mérés időpontja: **2021.03.11.**

Szabályok:

1. A mérési jegyzőkönyvet nevezze át *KH2_VoIP_jegyzokonyv_XYZABC.docx* névre, ahol XYZABC az Ön Neptun kódja!
2. Töltse ki a jegyzőkönyv fejlécét! (A zöld keretes rész fent. A zöld keret maradjon meg!)
3. A feladatok megoldását az adott feladat alá írja a zöld keretbe!
4. Ha elakadt, kérjen segítséget a Teams-ben! Nincs „rossz” vagy „buta” kérdés, kérdezzen bátran!
5. A mérés végén a jegyzőkönyvet töltsse fel a Moodle-ba!

A mérést kidolgozták: Majdán András
Németh Krisztián

BME TMIT, 2021. február – március

E mérés tartalmaz részeket egy régi mérésből, amely kidolgozásában Heszberger Zalán is részt vett.
Ezúton köszönjük a munkáját!

Tartalom

Bevezető	3
1. feladat: az Asterisk előkészítése	4
1./A feladat: az OpenWrt kézi telepítése (IMsc)	5
1./B feladat: a kiadott OpenWrt képfájl beüzemelése	12
2. feladat: Asterisk a helyi gépen	14
3. feladat: Forgalomelemzés protokollanalizátorral	21
Kitérő: egy hibaüzenet, egy kismadár és egy kis angol humor	29
4. feladat: SIP trönk, avagy kapcsolat egy távoli telefonközponttal	30
5. Bónusz feladat: További telefonos alkalmazások (IMSc)	36

Bevezető

A VoIP bonyolult. Bonyolult, mert összetett problémát kell megoldania. Az automata telefonközpontok hajnalán még egyszerű volt a feladat: adott volt egy központ, hozzá néhány előfizető, mindegyikhez egy hívószám, az előfizetők felhívhatták egymást – és ennyi. Ma a telefonszolgáltatás ennél sokkal összetettebb. Hívásátirányítás, konferenciahívás, hangposta, hívások sorba állítása, számlázás, jogosultság-ellenőrzés, magán alközpontok kezelése, emeltdíjas szolgáltatások, ... a sort hosszan lehetne folytatni. A feladat megoldására korábban komplex szoftverek születtek, melyek a telefonközpont-gyártók erre a célra készült célszámítógépein, a tulajdonképpeni központokon futtattak. Ahogy a külön e célra készült átviteli utakat az általános célú IP-re cserélték a szolgáltatók, úgy jelentek meg az általános számítógépen futó telefonközpont szoftverek is. Ezek legismertebbje az Asterisk.



1. ábra. Cisco IP Phone 303
(forrás: Wikimedia Commons)

Az Asterisk bonyolult. Az „Asterisk: The Definitive Guide” c. [könyv negyedik kiadása](#) például 732 oldal hosszú.¹ Ez a fentiek fényében már nem is meglepő, minket mindenesre szép kihívás elé állított, hogy egy olyan mérést rakjunk össze, mely során pár óra alatt hasznos ízelítőt kaphatnak belőle.

A jelenléti mérés során az 1. ábrán látható valódi VoIP telefonkészülékekkel történt a mérés, egy valódi hardveren futó központtal, és sok egymást telefonon hívogató hallgatóval. Hangulatos volt, az biztos. Nem volt könnyű becsempészni ennek a hangulatát egy távolléti mérésbe, ahol alapvetően a hallgatók egyedül ülnek a szobájukban a számítógépük előtt. Se telefon, se hardver a központnak, de még talán partner sincs, akit felhívhatnának.

Azt reméljük, hogy ezeket az akadályokat a lehetőségekhez képest sikeresen vettük, és egy hasznos, érthető és nem utolsó sorban élvezetes mérést sikerült összeraknunk. Ezt azonban ítélik meg Önök! Kezdjük is el a mérést!

¹ Az [ötödik kiadás](#) csak 414 oldal, viszont nem is tárgyal mélységében több fontos témát. A harmadik kiadás ingyen hozzáférhető, mégsem feltétlen ajánljuk, annyira elavult: <http://www.asteriskdocs.org/>

1. feladat: az Asterisk előkészítése

Mivel minden hallgatónak konfigurálnia kell az Asterisk szoftvert, adódott, hogy ezt fel kell telepítse a gépére. Az Asterisk azonban Linuxra készült, a hallgatók többségének pedig nem ez fut gépén, így ismét csak a virtuális gép jelentette a kézenfekvő megoldást, ahogy a LAN mérésnél is. A jelenléti mérésnél egy speciális hardveren és egy speciális Linuxon futott az Asterisk, és ez adta az ötletet, hogy legyen itt is így. Főleg csak a változatosság, az érdekesség kedvéért, illetve, hogy ezzel is tanuljanak. Maga az Asterisk szinte ugyanaz, mint Ubuntun vagy bármelyik másik Linuxon.

OpenWrt

A használt Linux-változatot OpenWrt²-nek nevezik. Ez egy nagyon leegyszerűsített Linux alapú operációs rendszer, amely minimális hardverigényekkel rendelkezik csak. A fő alkalmazási területe az otthoni, irodai kis WiFi routerek (innen a név: Wrt: wireless router). Ilyen szinte mindenkinek van a lakásában, bár újabban sokszor az internetszolgáltató adja. E routereken természetesen fut valami operációs rendszer, ám ebbe ritkán látunk bele. A legtöbb esetben lehetőségünk van azonban lecserélni ezt egy másik oprendszerre, és ezt néhányan meg is teszik. Több ilyen alternatív rendszer is létezik, pl. a DD-WRT³, de talán az OpenWrt a legnépszerűbb.

Ugyan miért tenne ilyet valaki, amikor várhatóan temérdek kényeztetéssel jár, és még a garanciáját is elveszítheti az eszköznek, sőt, ha nem vigyáz, tönkre is teheti? Nos azért, mert ezek a „kis vacak dobozok” jóval többre is képesek, mint WiFi hozzáférési pontként és Ethernet kapcsolóként működni. Ha már van ez a számítógép a lakásunkban, ami éjjel-nappal be van kapcsolva, kis házi-szerver építhető belőle. Egy USB-s háttértárat rádugva (persze csak ha van USB portja) olcsó, de használható NAS-t készíthetünk⁴, de lehet belőle nyomtatószerver, futhat rajta torrent-kliens (!), webszerver, telefonközpont (erről szól a mérés), és persze mi is írhatunk rá saját programot. Igaz, a szűkös memóriakapacitás miatt sokszor erősen válogatnunk kell, mit teszünk fel.

Egy ilyen operációs rendszert fogunk tehát most használni. Rendhagyó módon az első mérési feladat mindjárt kiérdemelte az „IMSc pontokért, nem kötelező” címkét. Nem mintha különösebben nehéz lenne – erről szó sincs! – egyszerűen nem tartozik a VoIP mérés szigorúan vett témakörébe. Az első feladat ugyanis egy OpenWrt telepítése a nulláról, rajta persze az Asteriskkel.

Aki ezt kihagyná, megteheti, számára összeállítottunk egy kész virtuális gép képfájlt (VM image, „véem imidzs”), töltsse le és használja. Megspórol vele egy órát, ennyivel kevesebbet tanul, de ettől még simán lehet ötös a mérése. Ha azonban egy icipicit is felkeltettük az érdeklődését az OpenWrt iránt, vágjon bátran vele az IMSc feladatokba! Mit veszíthet?

² Ld. <https://openwrt.org/>

³ Ld. <https://dd-wrt.com/>

⁴ Igaz, ezt sok router gyári szoftvere is tudja, csak ezért nem érdemes lecserélni.

1./A feladat: az OpenWrt kézi telepítése (IMSc)

Ez a feladatrész nem kötelező. IMSc pontok kaphatók érte, de ez a mérésre adott érdemjegyet nem befolyásolja. A jegyet pusztán a nem IMSc-snek megjelölt feladatok megoldásai alapján adjuk. Ha nem kívánja megoldani, akkor ugorjon az 1./B feladatra a 12. oldalon!

Készítsünk hát egy virtuális OpenWrt rendszert! Ennek persze futnia kell egy virtuális hardveren, ehhez pedig kell egy ún. hypervisor szoftver.

1. *Hypervisor installálása.* Ha valami csoda folytán nem lenne a gépén már legalább egy hypervisor („gép-virtualizáló”) szoftver a gépén, tegyen fel gyorsan egyet! Mi a VirtualBox-ban⁵ próbáltuk ki az alábbiakat, de a VMware Player-ben⁶ és a többi hasonlóban is mennie kell.

2. *OpenWrt letöltése.* Töltse le a 19.07.7-es verziójú OpenWrt-t a <https://openwrt.org/>-ról. E dokumentum írásakor ez a legfrissebb változat, így ezt teszteltük, de ha van már újabb, próbálkozhat azzal is. Direkt linket nem adunk, hogy legyen egy kis kihívás.

Pár segítség: a stabil kiadások (Stable Release builds) között keresse. „Firmware”-nek hívja magát, mert ezekre a routerekre valóban firmware-ként tehető fel. A „target”-ek között egy halom processzort (pontosabban SOC-t, System on a Chip) talál. Ez azért van természetesen, mert a WiFi routerekben ezek vannak. Mi azonban most egy sima x86 alapú PC-t fogunk emulálni, így az x86, azon belül is a 64 bites változat legyen a target. A combined-ext4 képfájlt válassza!

Nem teljesen nulláról installáljuk fel tehát az OpenWrt-t, hanem egy előre elkészített csomagot használunk. Sebaj, valódi routerre is ez az ajánlott legegyszerűbb módszer.

F1A.1 Mi a letöltési link?

Törölje ki ezt a sort, és ide írja a választ! A zöld keret maradjon, hogy könnyebb legyen javítani!

3. *Kitömörítés.* Ez egy tömörített fájl, nyissa ki!. gunzip, 7-Zip, PeaZip vagy hasonló programok segíthetnek, de a Total Commander is ismeri ezt a formátumot.

4. *Képfájlkonverzió.* Ezzel egy nyers képfájlunk lett, a VirtualBox-nak azonban egy VDI típusú fájl kéne. Szerencsére a hozzá adott VBoxManage parancssori programmal könnyen átalakíthatjuk. Használata:

```
"\Program Files\Oracle\VirtualBox\VBoxManage.exe" convertfromraw --format VDI BEMENETI_FÁJL.img KIMENETI_FÁJL.vdi
```

Ha Linux alatt csinálja mindezt, akkor ott elvileg a VBoxManage már a PATH-on van.

5. *VM létrehozása, beállítása.* Hozza létre a virtuális gépet a VirtualBox-ban (Gép/Új)! A „szakértő módban” kicsit több dolgot beállíthatunk, válasszuk azt. A nevet Önre bízuk, a Típusra és Verzióra a „Linux”, „Linux 2.6 / 3.x / 4.x (64-bit)” jó lesz. A Haladó fülön az Osztott vágólap lehet kétirányú, ekkor a copy-paste működik a gazda és vendég oprendszer között: a Windowson kimásolt szöveg a Linuxon beilleszthető és fordítva.

Memória: itt jön be az OpenWrt szépsége, a kis erőforrásigény. 256 MB bőven elég, de 128-cal is mennie kell. A VM (virtuális gép) kikapcsolt állapotában később is módosíthatjuk ezt, de ne ezen

⁵ <https://www.virtualbox.org/>

⁶ <https://www.vmware.com/hu/products/workstation-player.html>

spóroljunk, kapjon 256-ot, a böngészőnk megeszi ennek a többszörösét sajnos. A merevlemez pedig legyen Létező, a VDI fájl, amit az előző pontban készített.

6. További VM beállítások. Indítás előtt kicsit állítgassuk még a leendő telefonközpontunk emulált hardverét: Gép / Konfigurálás menüpont. Rendszer / Alaplap: floppy és optikai meghajtó nélkül megleszünk (vegye ki), a Tárolók közül a régi típusú IDE busz sem kell (bár talán ártani sem árt; szedjük ki azért), elég lesz a SATA. Audio sem kell. Ez meglepő lehet egy telefonközpontnál, de csak annyit jelent, hogy hangkártyát nem emulálunk, ez a virtuális gép nem fog a gépünk hangszóróján zenélni. Ő csak egy központ, ha zenélni akar, elküldi IP-n egy telefonnak. (El is fogja.)

A hálózat viszont kritikus. Vagy a VirtualBox emulál egy helyi NAT-ot és úgy ad egy IP címet (ez van például a LAN mérésnél), vagy a VirtualBox bridge-ként viselkedik, és az OpenWrt DHCP kérést nem megválaszolja, hanem továbbítja a valódi (WiFi) routerünkhöz. Most válasszuk ezt, ugyanis lesznek telefonklienseink, amelyek a NAT mögötti Asterisk központnak nem könnyen tudnának csatlakozási kérést küldeni. Lényeg a lényeg: a Hálózat legyen „Bridge-elt kártya”. Oké, de melyik? Ha pl. laptopról dolgozik, amelyiknek van Ethernet és WiFi csatlakozási lehetősége is, akkor mindenképp azt a kártyát kell bridge-elnie, amelyiken át a laptop az Internetre csatlakozik!⁷ A „haladó” módban még az emulált hálókártya típusát is kiválaszthatjuk. Ha ezt „Paravirtualizált hálózat (virtio-net)”-nak választjuk, akkor egy egyszerűbb emulációt futtat a VirtualBox, ami kisebb terhelést rak a gépünkre. Válasszuk ezt!

7. VM indítása. Összeraktuk a virtuális gépünket, vegyünk nagy levegőt és indítsuk el! Az indulás után 5-10 másodperccel már nem jönnek új rendszerüzenetek, de nem épp barátságos, ami ránk vár. Sebaj, üssünk egy Entert és már ott is az üdvözlő kép és a parancssori prompt. (Ha nem tettük még meg, most már nyugodtan kifújhatjuk a levegőt.)

Megjegyzés. E ponton megemlíthjük, hogy a „régi szép időkben” az OpenWrt verziókat koktélokról nevezték el. Volt White Russian, Kamikaze, Backfire, Attitude Adjustment, Barrier Breaker és Chaos Calmer. Ezen a ponton, a bejelentkezési képernyőn az unalmas verziószámon kívül nem csak a koktél nevét írta ki, hanem a receptjét is. Those were the days!

F1A.2 Elindult végre az OpenWrt. Történelmi pillanat! Örökössük meg gyorsan, mint Doki és Marty az órátorony avatását! Kérünk a bejelentkezési képről egy képernyőfotót ide.

Törölje ki ezt a két sort, és ide tegye a képet! A zöld keret maradjon! Itt is és a továbbiakban is a képernyőnek/ablaknak csak a releváns részét szűrja be!

8. Jelszó készítése. A `passwd` paranccsal állítson be gyorsan egy jelszót a root felhasználónak! Írja fel, ha kell, ne felejtse el, szükség lesz még rá!⁸

9. Router funkciók lekapcsolása. Az OpenWrt alapvetően routerként szeretne működni, hiszen erre tervezték. Erre most mégsem lesz szükség, hiszen csak egy végpontként csatlakozik a hálózatra. Ezért kapcsoljuk ki és tiltsuk le a `dnsmasq` és `odhcpd` szolgáltatásokat! (Az előbbi főként DNS caching-et valósít meg, az utóbbi egy DHCP szerver.)

⁷ Ha mobilinternetet használ a gépbe bedugott USB stick-el, vagy PPPoE dialapot használ úgy, hogy nem a router „tárcsáz”, hanem a gépe, az baj, mert ekkor csak egy IP címe van, nincs helyi NAT. Megoldható ez a helyzet is, keresse Majdán Andrást, ő segít összehozni egy port forwardingot a VirtualBox NAT-hoz! Tavaly (2020) amúgy senkinek nem volt ilyen gondja szerencsére.

⁸ Ha „beszorul” a virtuális gépbe, akkor a jobb alsó sarokban kiírt „host key” gomb (alap esetben a jobb oldali Ctrl) után Alt-Tab-ot nyomva kijut.

```
service odhcpd stop
service dnsmasq stop
service odhcpd disable
service dnsmasq disable
```

A `stop` sorok leállítják a futó szolgáltatást, a `disable` sorok pedig a következő rendszerindításkor nem engedik, hogy elinduljanak. A Unixos hagyományok szerint ne várjunk dicséretet: ha egy parancs hiba nélkül lefut, akkor jellemzően semmit nem ír ki a képernyőre. (Ez teszi egyszerűvé a parancsok egymásután fűzését.)

10. Hálózati interfész beállítása. Eddig jó, de még nincs megfelelő IP címünk, csak egy „gyárilag” előre beállított valami (192.168.1.1/24). Ez elképzelhető, hogy működik az otthoni hálózaton (de akkor sem az igazi így), de valószínűbb, hogy nem. Tegyük rendbe úgy, hogy az OpenWrt DHCP-vel kérjen magának IP címet a valódi helyi (WiFi) routertől!

Ehhez az `/etc/config/network` fájlt kell majd szerkesztenünk. Gondot mindössze az okozhat, hogy a gépen nincs még semmi kényelmes szövegszerkesztő, és letölteni sem tudunk, amíg nincs hálózat. No de ez egy IMSc feladat, nincs jogunk kétségbeesni! Van azért a gépen egy szövegszerkesztő a `vi` (szóban angolosan: „víáj”) nevű. Ez a program elég régi és eléggé, khm, másképp működik, mint az összes többi szövegszerkesztő. Sebaj, segítünk.

Pár sorban leírjuk a lényegét, amivel felfegyverkezve már menni kell a dolognak. Arra kell majd figyelni, hogy az OpenWrt-nken jelenleg csak angol kiosztású billentyűzet van telepítve.

- Ún. parancsmódban indulunk. Itt:
 - a kurzormozgató és Del gombok használhatók (Backspace nem)
 - beírni úgy tudunk, hogy előtte a kis `i` betűt lenyomjuk (Insert mode, beszúrásmód)
 - a sor végére írni az `a` betű lenyomása után tudunk (Append – ezzel is beszúrásmódba kerülünk)
 - teljes sor törlésére a `dd` parancs (billentyűkombináció) szolgál
- beszúrásmódban:
 - beírhatunk szöveget
 - mozoghatunk a kurzorral
 - működik a Del és Backspace
 - az Esc leütésére kerülünk vissza a parancsmódba
- parancsmódból a „legalsó sor módba” (last line mode) a `:` gomb lenyomására (magyar billentyűzeten ez a nagy É helyén van!) kerülhetünk át. Itt:
 - `w` mentés
 - `q` kilépés
 - `wq` mentés és kilépés
 - `q!` kilépés, mentés nélkül (a felkiáltójel a Shift+1 helyén van)

Még egyszer: kilépés, mentéssel: „Esc” „:” „w” „q” „Enter”.

Kilépés, mentés nélkül: „Esc” „:” „q” „!” „Enter”⁹

⁹ Erről szól a következő kocka vicc is:

– Már évek óta csak a vi szövegszerkesztőt használom!
– Miért, annyira szereted?
– Nem, csak nem jövök rá, hogy lehet kilépni...

Erről amúgy van egy jópofa oldal a Stack Overflow-n: <https://stackoverflow.blog/2017/05/23/stack-overflow-helping-one-million-developers-exit-vim/>

Szerkesszük hát az `/etc/config/network` fájlt úgy, hogy egy egyszerű IP végpontot készítsünk a routerünkben. Cél, hogy a `config interface 'lan'` szekcióban pontosan ez álljon:

```
config interface 'lan'
    option ifname 'eth0'
    option proto 'dhcp'
```

F1A.3 Küzdöttünk a vi-jal és győztünk, kérünk egy képernyőképet a kész `/etc/config/network` fájlról! (vi-ból kilépés után a `cat /etc/config/network` kiírja)

Törölje ki ezt a két sort, és ide tegye a képet! A zöld keret maradjon! Itt is és a továbbiakban is a képernyőnek/ablaknak csak a releváns részét szúrja be!

11. Most olvassuk újra ezt a konfigurációs fájlt az OpenWrt-vel:

```
service network reload
```

12. IP cím. Nézzük meg, milyen IP címet kaptunk:

```
ip addr
```

Az `lo` interface a loopback interfész, ezt a gépen futó alkalmazások használják egymás közötti kommunikációra port szinten (unix socketek használata nélkül). Az `eth0` interfészt használja a gépünk más számítógépekkel való kommunikációra.

F1A.4 Kérünk egy képernyőképet az előző parancs által kiírt beállításokról!

Ide

F1A.5 Szűrje ki ebből a lényeget! Mi az OpenWrt kapott IP címe és hálózati maszkja?

IP cím:

Netmaszk:

13. *SSH kapcsolódás.* Bár a VirtualBoxon át is használhatjuk szöveges módban az OpenWrt-t, kényelmesebb, ha egy SSH klienssel kapcsolódunk hozzá. Könnyebben lehet például a képernyőt átméretezni, betűméretet állítani, egyszerűbb az Alt-Tab, van magyar billentyűzet. Amúgy is, egy valódi WiFi router esetén is így tennénk. Kerítsünk egy SSH klienst! Mac-en és Linuxon simán megy a paracssorból az `ssh` parancs, sőt már Windows 10-en is, Windowson mégis kényelmesebbnek tűnik egy dedikált alkalmazás használata. A mérés összeállító a putty-ot¹⁰ szeretik, mert kicsi, egyszerű és tud mindent, amit kell, de persze bármilyen más kliens is megteszi. Lépünk be az előző válaszban megadott IP címre SSH-val!

Ha biztonsági figyelmeztetést kap, hogy a szerver kulcsa nincs regisztrálva, ne ijedjen meg. Ez az első alkalom, hogy csatlakozott a géphez, ezért a virtuális gép publikus kulcsa még nem ismert az SSH kliens számára. Következő alkalommal már ez a kiírás nem fog megjelenni. (A kulcs kliensben való eltárolásának a célja a „man in the middle” támadások kivédése, amikor a támadó a saját nyilvános kulcsát adja meg nekünk a szerveré helyett.) Lépjen tovább, csatlakozzon az OpenWrt szerverhez. A felhasználónév `root` (ez a unix világban a rendszergazda neve), a jelszó az, amit a 8. pontban adott.

¹⁰ <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

14. Csomagadatbázis frissítése. Mint a Linux rendszerek általában, az OpenWrt is rendelkezik a legtöbb alkalmazást előre összeállítva tartalmazó központi tárral (repository). Ez tehát egy netes adatbázis, amelyből egyszerűen és biztonságosan letölthetjük és telepíthetjük az alkalmazásokat. Erre való a helyi gépen a csomagkezelő program, az `opkg`. Az adatbázisnak a helyi gépen tárolt „tartalomjegyzékét” kell először frissíteni:

```
opkg update
```

Ha nem sikerült, valami gond van a hálózati eléréssel.

14. Frissíthető csomagok listája. Mint más oprendszereknél, itt is az a helyzet, hogy az előre telepített programok egy részének van újabb verziója. Ezeket frissítsük most!

Az `opkg list-upgradable` parancs megmutatja a frissíthető, már telepített csomagok listáját.

F1A.6 Mely csomagokat lehet frissíteni?¹¹

Ne képernyőképet tegyen ide, hanem most már szöveges listáját a csomagoknak! E sort törölje!

15. Telepített csomagok frissítése. Sajnos nincs olyan parancs, ami az összes fenti csomagot egy csapásra frissítené. Sebaj, a Unix rendszerek lényege, hogy sok kis egyszerű parancsból tudunk egy bonyolultat csinálni. Az előző parancs eredményének első „oszlopa” (minden sor első „szava”) a frissítendő csomagok neve. Ezt a `cut` parancssal ki tudjuk vágni, és az `opkg install` parancsnak átadva (erre van az `xargs`) azokat egyenként frissíteni. Ez tehát a megoldás (futtassa le!):

```
opkg list-upgradable | cut -f 1 -d ' ' | xargs opkg install
```

Higgyék el, csak elsőre ijesztő! Egy kis magyarázat: a pipe („cső”) karakter (`|`) azt jelenti, hogy a bal oldalán található parancs kimenete lesz a jobb oldalon található parancs bemenetele. Tehát az `opkg` kiírja a frissítésre váró csomagok listáját, majd a sorokat szétszedi szóközzel elválasztott mezőkre (`-d` jelentése delimiter = határoló), és csak az első mezőt írja ki (`-f` jelentése field), így minden sor csak a csomagok neveit tartalmazza kimenetként. Az `xargs` csomag elvégzi a megadott parancsot minden egyes sorra, amit bemenetként megkapott. Az `opkg` paramétere `install`. Lehetne `upgrade` is, ez esetben ugyanaz a kettő, de az `install` üzembiztosabbnak bizonyult, legalábbis a korábbi OpenWrt verziókban.

A fenti parancs néha nem fut le hibátlanul, például – ok nélkül – kevés memóriára panaszkodik. Ekkor próbáljuk újra, és esetleg ismét újra. Akkor jó, ha az `opkg list-upgradable` már nem mutat semmit.

16. Szövegszerkesztő. Akinek megtetszett a vi, használhatja, a többieknek viszont végre van lehetősége valami egyszerűbbre váltani. A nano elég közkedvelt választás. A joe némileg oldschoool, de sokan szeretik. Biztos van még sok más, a példákban mi most maradunk a nano-nál.

```
opkg install nano
```

Láthatjuk, hogy nem csak a nano csomagot töltötte le az `opkg`, hanem az általa használt, hozzá szükséges csomagokat – az úgynevezett függőségeit (dependency) – is: ezek a `terminfo` és a `libncurses6` csomagok.

nano gyorstalpaló:

¹¹ putty-ban a szöveget kijelölve a vágólapra másoldóik egyből, nem kell Ctrl-c-t nyomni. Beillesztés: Shift-Insert vagy jobb egérgomb

- mentés: ctrl-o
- kilépés: ctrl-x

17. Hostfile szerkesztése. Az Asterisk keresni fog egy OpenWrt nevű géphez tartozó IP címet. Nem szép tőle, de mondjuk, hogy „nem ezért szeretjük”. Be kell állítsuk hát a helyi gépen, hogy ez az OpenWrt nevű gép bizony mi vagyunk: a 127.0.0.1 IP cím mindig a helyi gépet jelenti. A nano segítségével az `/etc/hosts` fájl első sorát egészítsük ki ilyenné:

```
127.0.0.1 localhost OpenWrt
```

Magyarázat: amikor a domain név feloldása történik IP címre, a lokális feloldó (resolver) először a `/etc/hosts` fájlban belül keresi a nevet és a hozzá tartozó IP címet, majd ha itt nem találja, akkor fordul a DNS szerverekhez. Ilyen módszerrel bármilyen domain-t átirányíthat bármilyen IP címre. Ilyen hosts file létezik a Windows rendszereknél is, a `c:\windows\system32\drivers\etc\` könyvtárban található.

F1A.7 Ellenőrzésképp pingeljük az OpenWrt gépet! Működnie kell.

```
ping -c 3 OpenWrt
```

Másolja ide (szöveggént) a fenti parancs kimenetét!

18. Asterisk telepítse. Itt az ideje végre telefonközponttá varázsolni a gépünket. Telepítsük az alap Asterisk csomagot és egy csomó kiegészítő modult! Mindezt egy csapásra (egy sorba írjuk!):

```
opkg install asterisk16 asterisk16-pjsip asterisk16-res-rtp-asterisk
asterisk16-codec-ulaw asterisk16-codec-alaw asterisk16-format-wav
asterisk16-format-pcm asterisk16-bridge-simple
```

A parancs természetesen nem csak a fenti nyolc csomagot telepíti, hanem a függőségeiket is.

19. LuCI. Eddig titkoltuk, de van ám az OpenWrt-nek egy webes interfésze is. Minden WiFi routernek van, nyilván ő sem maradhat ki. Az OpenWrt alapértelmezett webfelületét LuCI-nak hívják, és bár vannak más webes interfészek is hozzá (pl. a magyarul viccesen csengő nevű JUCI)¹², a LuCI bőven jó lesz nekünk. Próbáljuk is ki, írjuk be a böngészőbe a VM-ünk IP címét (ld. 1.5 feladat)!

F1A.8 Lépjen be (név és jelszó mint az SSH belépésnél) és kérünk egy képet a Status / Overview oldalról!

Ide. Csak amit kértünk, ne az egész képernyőjét! Elég csak annyit, amennyi egy képernyőre kifért.

Nézzen szét bátran a LuCI-ban, hogy mi mindent lát itt! Ha a System / Startup-nál jár, ellenőrizze, hogy a `dnsmasq` és az `odhcpd` letiltott (`disabled`) legyen! A webfelület eléggé hasonlít más WiFi routerek webes interfészéhez.

Most ugyan van elég memóriánk és lemezterületünk, valódi routerek esetén azonban sokszor nem ez a helyzet, ezeket ugyanis csak annyi erőforrással látják el, amennyi muszáj a működésükhöz. Ha ez kevésnek ígérkezik, a LuCI-t el is lehet távolítani, minden megoldható parancssorból is.

20. További csomagok. Bár a parancssori csomagtelepítés sem épp megerőltető, megnézzük, hogyan megy ez a webfelületen át. A System / Software-ben írja be a Filter-be, hogy asterisk (vagy asterisk16), és kattintson az Update lists...-re. Közel 300 találatot fog kapni: ennyi Asterisk-kel

¹² <https://openwrt.org/docs/guide-user/luci/webinterface.overview>

kapcsolatos csomag van az OpenWrt repositoryban. Ezek egy részét már telepítette („Installed”), a többségét pedig nem is akarjuk most telepíteni. Néhányat azért mégis. Ezeket telepítse a weben át:

```
asterisk16-app-record  
asterisk16-app-sayunixtime  
asterisk16-codec-gsm  
asterisk16-format-gsm  
asterisk16-sounds
```

Telepítés előtt láthatjuk a függőségeket is, amelyek ez esetben amúgy már mind telepítve vannak.

F1A.9 Kérünk egy képernyőképet a telepített Asterisk csomagokról. Ehhez a filter maradjon, csak alatta az Installed fület válassza ki!

Ide. Ha nem fér ki egy képernyőre, akkor többet tegyen ide egymás alá.

Lépjen ki a LuCI-ból! Mostantól maradunk a szöveges terminálnál.

Kész! Működik az OpenWrt, működik az Asterisk! Esetleg tartson egy rövid szünetet, és utána lépjen 2. feladatra, ahol használatba is veheti!

1./B feladat: a kiadott OpenWrt képfájl beüzemelése

Ezt a feladatrészt csak akkor oldja meg, ha kihagyta az 1./A részt!

Egy virtuális OpenWrt rendszert fogunk használni, aminek persze futnia kell egy virtuális hardveren, ehhez pedig kell egy ún. hypervisor szoftver.

1. *Hypervisor installálása.* Ha valami csoda folytán nem lenne a gépén már legalább egy hypervisor („gép-virtualizáló”) szoftver a gépén, tegyen fel gyorsan egyet! Mi a VirtualBox-ban¹³ próbáltuk ki az alábbiakat, de a VMware Player-ben¹⁴ és a többi hasonlóban is mennie kell.

2. *VM képfájl letöltése.* Töltse le az általunk előre összeállított virtuális gép képfájl (VM image). Mivel OpenWrt-t használunk, e fájl kellemesen kicsi, mindössze 15 MB. E helyekről töltheti le:

- VIK Moodle (<https://edu.vik.bme.hu/>), a feladatkiírás mellől (Asterisk_OpenWrt17.07.7.ova)
- a KommHáló2 tantárgy Teams csoportjában az Általános csatorna Fájlok / Osztályanyagok / Asterisk_OpenWrt17.07.7.ova helyről
- biztos, ami biztos, innen is: http://w3.tmit.bme.hu/kh2/Asterisk_OpenWrt17.07.7.ova

3. *VM elkészítése.* A VirtualBox-ban válassza a Fájl/Gép importálás...-t. Szakértő módban keresse meg a letöltött .ova fájlt. A Névét átírhatja, bár erre később is lesz lehetősége. Importálja a fájlt, ezzel elkészül a virtuális gép.

4. *VM konfigurálása.* A VirtualBoxban az új gépet kiválasztva a Konfigurálás / Hálózat részen a Bridge-el kártya alatti Név részen válassza ki azt a hálózati kártyát, amelyen át a gépe az internetre van kötve. Például ha egy laptopot használ, amelynek van WiFi interfésze és vezetékes Ethernet interfésze is, akkor azt válassza ki, amelyiket a mérés közben használja, amelyen át az Internetre csatlakozik. Ha ezt rosszul csinálja, a VM nem fog hozzáférni a hálózathoz, anélkül meg nem lehet mérni. (Semmi baj, akkor ki kell kapcsolni a VM-et, vissza kell jönni és átállítani, majd a VM-et újra elindítani.)

Megjegyzés: Ha mobilinternetet használ a gépbe bedugott USB stick-el, vagy PPPoE dialaport használ úgy, hogy nem a router „tárcsáz”, hanem a gépe, az baj, mert ekkor csak egy IP címe van, nincs helyi NAT. Megoldható ez a helyzet is, keresse Majdán Andrást, ő segít összehozni egy port forwardingot a VirtualBox NAT-hoz! Tavaly (2020) amúgy senkinek nem volt ilyen gondja szerencsére.

5. *VM indítása, IP cím kikeresése.* Indítsa el a virtuális gépet! Kb. 10 másodperccel az indulás után (amikor már nem jönnek újabb üzenetek) üssön egy Entert, és megkapja a bejelentkező képernyőt. Itt adja ki az

```
ip addr
```

parancsot. Az `lo interface` a loopback interfész, ezt a gépen futó alkalmazások használják egymás közötti kommunikációra port szinten (unix socketek használata nélkül). Az `eth0` interfészt használja a gépünk más számítógépekkel való kommunikációra.

Megjegyzés. Ha „beszorul” a virtuális gépbe, akkor a jobb alsó sarokban kiírt „host key” gomb (alap esetben a jobb oldali Ctrl) után Alt-Tab-ot nyomva kijut.

¹³ <https://www.virtualbox.org/>

¹⁴ <https://www.vmware.com/hu/products/workstation-player.html>

F1B.1 Kérünk egy képernyőképet az előző parancs által kiírt beállításokról!

```
root@OpenWrt:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP qlen
    1000
    link/ether 08:00:27:ad:4d:3f brd ff:ff:ff:ff:ff:ff
    inet 192.168.88.130/24 brd 192.168.88.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fead:4d3f/64 scope link
        valid_lft forever preferred_lft forever
root@OpenWrt:~#
```

F1B.2 Szűrje ki ebből a lényegét! Mi a kapott IP cím és hálózati maszk?

IP cím: 192.168.88.130

Netmaszk: /24

6. *SSH kapcsolódás.* Bár a VirtualBoxon át is használhatjuk szöveges módban az OpenWrt-t, kényelmesebb, ha egy SSH klienssel kapcsolódunk hozzá. Könnyebben lehet például a képernyőt átméretezni, betűméretet állítani, egyszerűbb az Alt-Tab, van magyar billentyűzet. Amúgy is, egy valódi WiFi router esetén is így tennénk. Kerítsünk egy SSH klienst! Mac-en és Linuxon simán megy a paracssorból az ssh parancs, sőt már Windows 10-en is, Windowson mégis kényelmesebbnek tűnik egy dedikált alkalmazás használata. A mérés összeállítói a putty-ot¹⁵ szeretik, mert kicsi, egyszerű és tud mindent, amit kell, de persze bármilyen más kliens is megteszi. Lépünk be az előző válaszban megadott IP címre SSH-val!

Ha biztonsági figyelmeztetést kap, hogy a szerver kulcsa nincs regisztrálva, ne ijedjen meg. Ez az első alkalom, hogy csatlakozott e géphez, ezért a virtuális gép publikus kulcsa még nem ismert az SSH kliens számára. Következő alkalommal már ez a kiírás nem fog megjelenni. (A kulcs kliensben való eltárolásának a célja a „man in the middle” támadások kivédése, amikor a támadó a saját nyilvános kulcsát adja meg nekünk a szerveré helyett.) Lépjen tovább, csatlakozzon az OpenWrt szerverhez.

Felhasználónév: root (ez a unix világban a rendszergazda neve)

Jelszó: V01Pmeres (nagy V-nulla-egy-P-...)

Ezzel véget ért a bevezető rész, fut az OpenWrt és rajta az Asterisk. Itt tart az is, aki maga állította össze a virtuális gépet az OpenWrt beállítgatásával az 1./A részben. Ha a mérés végén még van kedve és ideje, azt is megteheti, hogy újra elkezdi a mérést az 1./A-nál. Ekkor utólag is kitöltheti a jegyzőkönyvnek azt a részét, és így megszerezheti az IMSc pontokat, és ami talán annál is fontosabb, egy kis extra szakmai tapasztalatot. Még egyszer, ez nem feltétele a jeles mérési jegynek. Akárhogy is, most folytassa a következő feladattal!

¹⁵ <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

2. feladat: Asterisk a helyi gépen

Ez a feladat mindenki számára kötelező!

A frissen elkészült telefonközpontunkat most ki szeretnénk próbálni úgy, hogy hívásokat bonyolítunk le rajta keresztül. Ehhez fel kell konfigurálni a központot és kelleni fognak végberendezések is. Ez utóbbiak most virtuálisak lesznek, bár ha valakinek van kéznél egy valódi VoIP-képes telefonkészüléke, azt is bátran kipróbálhatja.

Kezdjük a központ beállításával!

1. A SIP konfigurációs fájl szerkesztése. A mérés során a SIP¹⁶ jelzésprotokollt fogjuk használni. Az Asterisk ennek két implementációját is tartalmazza. Az egyik `chan_sip` néven ismert, a másik `res_pjsip/chan_pjsip`, vagy csak PJSIP néven. A `chan_sip` a korábbi, amely bár még része az Asterisknek, mára elavultnak számít. Épp ezért mi a perspektivikusabb PJSIP-et¹⁷ fogjuk használni.

Ennek a konfigurációs fájlja a `/etc/asterisk/pjsip.conf`. Menjünk be e könyvtárba¹⁸ (`cd /etc/asterisk`) és készítsünk egy biztonsági mentést a fájlról (`cp pjsip.conf pjsip.conf.bak`)!

Megjegyzés. Unixban kevésbé járatosak számára írjuk, hogy ne lepődjenek meg azon, hogy ha egy parancs hiba nélkül lefut, akkor jellemzően semmit nem ír ki a képernyőre. Ez teszi ugyanis egyszerűvé a parancsok egymás után fűzését.

Ezután szerkesszük a fájlt a nano szövegszerkesztővel (`nano pjsip.conf`).

A nano-ban a legfontosabb parancsok:

- mentés: Ctrl-o
- kilépés: Ctrl-x
- hányadik sorban vagyok?: Ctrl-c

A fájl 284. sora környékén látjuk az „ENDPOINT CONFIGURED FOR USE WITH A SIP PHONE” részt. Itt kell beállítani a központhoz csatlakozó végberendezéseket. Első körben két készüléket adunk hozzá, de ha megtetszett, később továbbiakat is hozzávehet.

Ahogy a `;`-vel kikommentezett példában is látható, egy végberendezéshez – melynek azonosítója a példában 6001 – több szakasz is tartozik, melyek fejléce szögletes zárójelek közt szerepel. Ezt követi a szakasz típusa, amely többek között lehet `endpoint`, `auth` és `oar`:

```
[azonosító]
type = típus
```

Az `endpoint` írja le többek között azt, hogy mire képes az eszköz. Az `auth` a hitelesítés módját definiálja, az `aor` pedig arról ad információt, hogyan érhetjük el az eszközünket.

¹⁶ Ez volt az a pont, ahol töröltem a Wordben, hogy a „sip” szót „síp”-ra javítsa...

¹⁷ Biztos mindenkit nagyon érdekel, mint jelent a PJ a PJSIP-ben. Nos az első készítő (Benny Prijono) vezetéknévéből származik. Nem, nincs köze a pizzamához. Ld. <https://www.pjsip.org/about.htm>

¹⁸ Linux rendszereken a mappákat (folder – ez olyan „windowsos” elnevezés) könyvtárnak (directory) szokás nevezni. Amúgy a mappa találóbb, de most maradunk a hagyományoknál.

2. *UDP*. UDP-t fogunk használni, ehhez engedélyezzük a `[transport-udp]` szakaszt a 128. sor környékén: szedjük ki a `;`-t e szakasz fejléce és a három sora elől.

3. *endpoint*. Hozzunk létre egy új szakaszt az első közvetlenül `;` `[6001]`-et tartalmazó sor felett, az azonosítója legyen `student-phone1`. A szakasz típusa legyen `endpoint`. Figyeljék a 6001-es példát a konfigurációs fájlban! (Természetesen a `;` most nem fog kelleni a sor elejére.)

A `transport`-ot másolja le a példából, ez hivatkozik az előző pontra.¹⁹ A `context` is kell, ez mondja meg, hogy az innen jövő hívások mely híváscsoportba tartoznak. Ezt ne egy az egyben másolja le, legye az értéke `belso-hivas`, így írja be a fájlba!

A 6001-es példa következő két sora (`disallow`, `allow`) azt mutatja, hogy az eszköz mely kodekeket használatára képes. A példában kizárólag az `ulaw` van engedélyezve, ami a PCM μ -law, vagyis az amerikai PCM változat. Itt Európában használjuk inkább az A-law (`alaw`) változatot! E módosítással szűrje be e két sort!

Ezek után már csak annyi kell, hogy beírja, hogy a hitelesítési és AoR információk hol található. Ezeket a szakaszokat is hamarosan létrehozzuk, addig előlegezzük meg a 6001-es példa alapján, az azonosítót értelemszerűen átírva `student-phone1-re`.

4. *auth*. Új szekció, a szakaszfejléc ugyanaz, a típusa `auth`. A fájlban kicsit lejjebb lévő 6001-es példa alapján tölts ki a további három sort: a hitelesítés felhasználói névvel és jelszóval, amelyeket az ezután következő két sorban ad meg. A felhasználónév legyen `student-user1`, a jelszó mondjuk `4321`. Ezeket később átírhatja, de első körben működjön így! Ezeket az infókat várja tehát az Asterisk központ majd a végberendezéstől.

5. *aor*. Új szekció, a szakaszfejléc ugyanaz, a típusa `aor`. Szép, szép, de mi a csuda az az aor? Address of Record, kicsit bonyolult koncepció, de nagyjából arra való, hogy tudja a központ, hol éri el a végberendezést. Ehhez egy IP cím fog kelleni egyébként, de mivel az változhat, ezért a végberendezés majd a központhoz történő regisztráció során adja azt meg. Így most ide nem kerül be az IP cím. A 6001-es példában szerepel ugyan, de Ön ezt a sort ne másolja át! Pusztán a szakasz típusa kell (`aor`), illetve az, hogy maximálisan hány eszköz csatlakozhat a központhoz, mint `student-user1`. Nos, egyetlen egy, ezért az idevonatkozó sort másolja át a példából! Mondani sem kell, hogy ilyenkor a kommentet jelentő `;`-t mindig törölje a sor elejéről!

Végül érdemes még egy sort ide hozzáadni:

```
remove_existing=yes
```

Ha esetleg létezik egy már beragadt regisztráció (pl. korábban nem jelentkezett ki a lefagyott kliens), akkor ez kidobja az új regisztráció érkezésekor.

6. *Noch einmal!* Ezzel kész a `student-phone1`-hez tartozó konfiguráció. Mi azonban két készüléket szeretnénk, ezért ismételve meg az előző három pontot új szakaszokat beszúrva, csak most mindenhova `student-phone2`-t írjon. A hitelesítéshez tartozó felhasználói nevet is át kell írni (`student-user2`), és a jelszót is (legyen `9876`).

7. *Done*. Mentse el, lépjen ki!

8. *Reload*. Most az jön, hogy a háttérben szolgáltatásként futó Asterisk szoftvert utasítani kell, hogy olvassa újra a beállításait. Ehhez el kell indítani az Asterisk parancssori kezelőfelületét:

¹⁹ `putty`-ban az egérrel kijelölve egy részt az egyből a vágólapra másolódik, nem kell semmit megnyomni. A jobb egérgomb pedig a beillesztés: „nem kérdez – büntet”. A beillesztésre amúgy jó a `Shift+Insert` is.

```
asterisk -r
```

Utána (immár a programon belül) újraolvasatni vele a konfigurációs fájljait:

```
core reload
```

Ezután látjuk az esetleges hibaüzeneteket, ha valamit elszúrtunk a konfigurációban. A hibákat azonban csak egyszer mutatja meg, ha nem mentjük újra a konfigurációt, nem tölti be újra, hiába kérjük. Ráadásul nem minden hiba a mi hibánk, ezekkel például együtt lehet élni:

```
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'
```

Megjegyzés. Bár nem most fog kelleni, jobb híján ide írjuk le, hogyan kaphatunk bővebb információt az Asteriskről. Ahogy láttuk, a programot az `asterisk -r` paranccsal elindítva kapunk bizonyos üzeneteket a működésről, ami jól jöhet hibakereséskor. Még több üzenetet kapunk, ha az Asterisket így indítjuk:

```
asterisk -rvvvv
```

Erről a doksi csak annyit mond „minél több a v betű, annál bővebb lesz a kimenet”. Kösz.

A futó interaktív Asteriskben pedig a következő parancs engedélyezi kimondottan a pjsip modullal kapcsolatban a további részletek megjelenítését:

```
pjsip set logger on
```

Kilépés `exit`-tel vagy `quit`-tal.

Megjegyzés. Az Asteriskbe való belépés nélkül is újraolvasathatjuk a konfigurációt:

```
service asterisk reload
```

vagy így:

```
/etc/init.d/asterisk reload
```

Sőt újra is indíthatjuk az Asterisket:

```
service asterisk restart
```

Ez utóbbi kicsit kegyetlenebb, mert megszakítja az élő hívásokat. Élesben nem szép dolog ilyet tenni, de most „kicsiben” elmegy. Azért választottuk mégis az interaktív belépést, mert csak így láthatjuk az esetleges hibaüzeneteket.

F2.1 Másolja ide a frissen beírt szakaszait a `pjsip.conf`-nak!

```
[student-phone1]
type=endpoint
transport=transport-udp
context=belso-hivas
disallow=all
```



```

allow=alaw
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=belso-hivas
disallow=all
allow=alaw
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
remove_existing=yes

```

9. Dialplan. Csatlakozhatnának már a készülékek a központhoz, valami azonban hiányzik... Nos, telefonálni telefonszámmal szokás, arról pedig még szó sem volt eddig. Ennek az az oka, hogy az nem a SIP-modulhoz tartozik, máshol kell beállítani. Nem messze, az ugyanebben a könyvtárban lévő `extensions.conf` fájlban. Készítsünk egy biztonsági mentést a fájlról (`cp extensions.conf extensions.conf.bak`), majd szerkesszük gyorsan!

Ez egy hosszú és bonyolult fájl, ne is vesszünk el benne. Egyszerűen az elejére írjuk hozzá, amit ide gondolunk. Az első itt is egy szakaszazonosító, mint az imént. A szakasz neve most `belso-hivas` legyen. Ismerős? Ld. a 3. pontot fentebb. Ugyanúgy szögletes zárójelek közé jön!

Alá jönnek a tárcsázási szabályok. Ezek ilyen formátumúak:

```

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

```

Az első sor azt jelenti, hogy ha a hívott mellék (*extension*) 300, akkor első lépésként (1) tárcsázzon (Dial) vagyis csörgesse meg a PJSIP modulban definiált student-phone-1 végberendezést a megfelelő SIP üzenet elküldésével. Ekkor az kicsöng, majd ha felveszik, felépül a kapcsolat. Ha ezzel végzett az

Asterisk, akkor a második sor szerint: még mindig a 300-as hívószámról beszélünk, a következő prioritású feladat (n) az, hogy megszakítsa a beszélgetést (hangup = letenni a telefont).²⁰

Ennek mintájára (még mindig ugyanebbe a szakaszba) írja be, hogy a 301-es mellékre érkező hívás pedig a student-phone2-re menjen! Mentse a fájlt!

F2.2 Másolja ide a frissen beírt első öt sorát az `extensions.conf`-nak!

```
[belso-hivas]
exten => 300,1,Dial(PJSIP/student-phone1)
exten => 300,n,Hangup()
exten => 301,1,Dial(PJSIP/student-phone2)
exten => 301,n,Hangup() 10. Dialplan Reload. A 8. pontban leírt módszerek egyikével olvastassa újra a konfigurációkat!
```

11. SIP kliensek telepítése. So far so good, már csak egy végberendezés kéne. Vagy kettő. Softphone-okat, szoftveres telefonokat fogunk használni. Töltsön le egy-egy ún. SIP kliens telefonszoftvert a számítógépére és mobiljára.

Pontosabban két eszköz kell, ami az otthoni hálózatra van regisztrálva, SIP klienst képes futtatni és van mikrofonja és hangszórója. Triviálisan adódik az Ön számítógépe és telefonja (ha az az otthoni WiFi használja és nem mobilnetet éppen²¹), de lehet más is: családtag vagy lakótárs számítógépe, egy tablet, bármi. Windows alatt használhatjuk például a MicroSIP-et²². Ez nevéhez híven egy elég kicsi program, és van „portable” változata, ekkor nem is kell semmit a gépünkre. A MicroSIP-nél arra érdemes figyelni, hogy bezárva csak elbújik a tálcán, Exit-tel lehet végleg kilépni belőle. Linux és Mac alatt a Linphone²³ biztosan jó lesz, de használhatnak más is.

Kellene fog egy telefonos app is. A MicroSIP honlapjáról is letölthető ilyen, én mégis a ZoiPer nevű klienst használtam Android alatt, de lehet bármi hasonló. A ZoiPer installálása után kicsit agresszívan próbálja a fizetős változatát ránk erőltetni, az ilyen próbálkozásait bátran utasítsuk el (SKIP gomb)!

12. SIP kliens konfiguráció. Konfiguráljuk be az számítógépen SIP klienst! Az Account Name akármi lehet, legyen most `student-phone1`. A SIP szerver és a Domain is az első feladatban meghatározott IP címe az Asterisk VM-nek. A Username az azonosítónk a központ felé: ez lesz a `student-phone1`, ehhez készült a SIP konfiguráció. A Login és Password a hitelesítéshez kell. Itt más a nevünk, ld. a `pjsip.conf`-ban (`student-user1`) és a jelszó is ott van (4321). Beállíthatunk még Display Name-et, ezt fogja a hívott félnek kiírni csengéskor. Minden más maradjon alapértelmezés! (Elvben a PCM A-law kodeket is be kell állítanunk egyedi kodeknek, de amiben néztem, abban ez volt a default.)

Linphone-on a profil beállításakor csak egyféle felhasználónevet lehet megadni. Ez a `student-phone1` legyen. Ezután a regisztrációkor meg fogja kérdezni a hitelesítési felhasználói nevet és jelszót, itt kell majd megadni a `student-user1`-et és a 4321-et.

²⁰ A hangup beírása igazából az Asterisk korábbi verzióiban volt csak kritikus, most ez az alapértelmezés, ha a Dialplanban nincs több tennivaló. Nem baj, azért csak írjuk le ezt a sort.

²¹ Az kell, hogy a SIP klienst futtató eszköz (pl. mobil) pingelni tudja az OpenWrt virtuális gépet. Mivel otthon minden bizonnyal van egy NAT, ezért ha a mobil is és a PC is mögötte van (általában az sem baj, ha a PC Ethernetet használ, nem WiFi-t), akkor valószínű nem lesz gond. Ha a mobil a telefonszolgáltató mobilnetén lóg otthon is, akkor nem fog menni a dolog egyszerűen.

²² <https://www.microsip.org/>

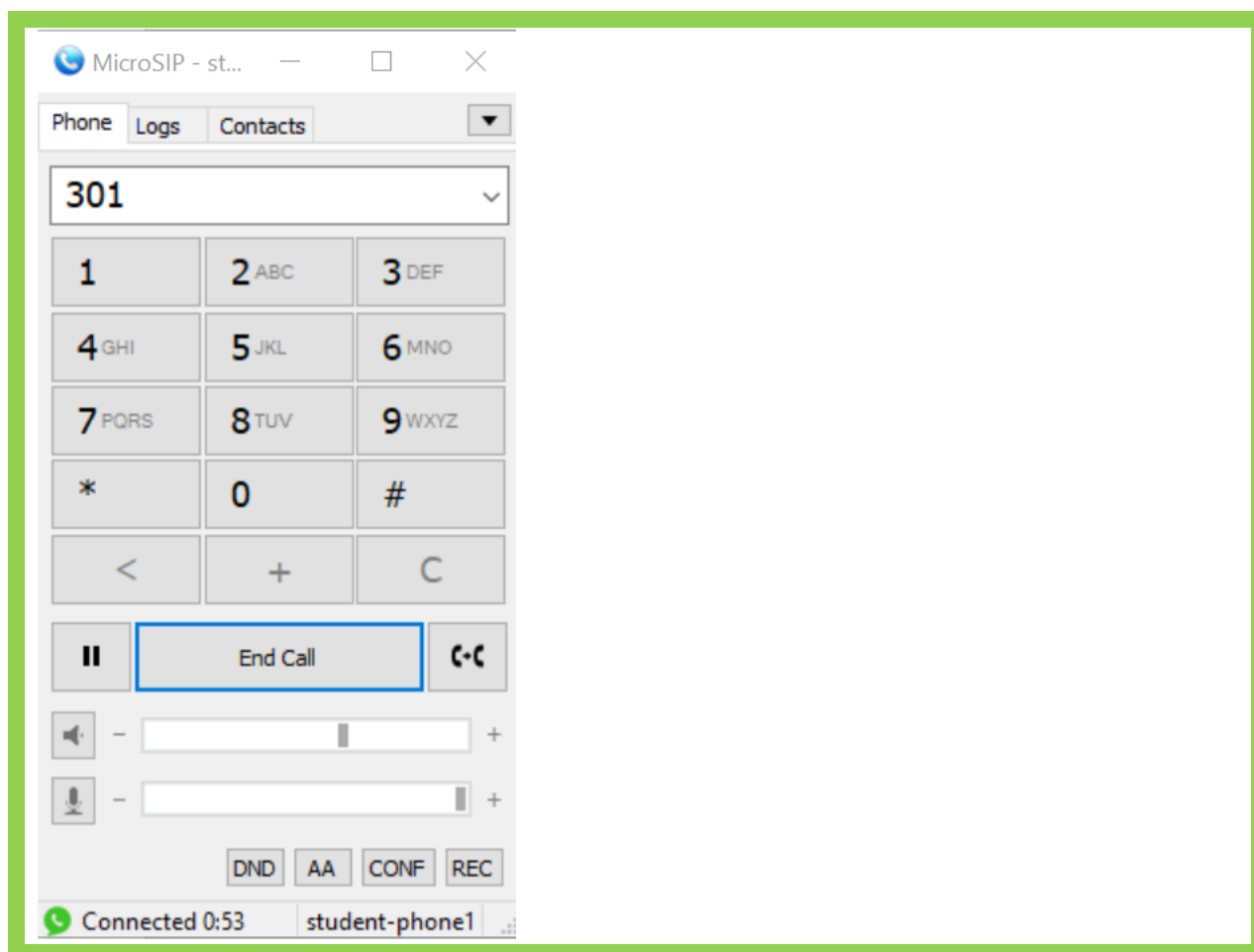
²³ <https://www.linphone.org/>

Ami még érdekes lehet, az az „expiration”, vagy „register refresh”. Ez azt mondja meg, milyen gyakran regisztráljon újra a kliens. Ez a ZoiPer-nél alapból 60 másodperc, ami túl kevés: percenként újregisztrál, ami eltart pár másodpercig, így az idő túl nagy részében nem elérhető a kliens. Ezt vigyük fel bátran úgy 5-10 percre, sőt akár 60 percre is.

Ezután a mobilos klienst (vagy másik számítógépet, ha olyat telepített egy másik gépre) is állítsuk be hasonlóképpen, csak ott más a felhasználói név és a hitelesítéshez használt név – jelszó páros is.

13. *Most ugrik a majom a vízbe!* Hívjuk fel a számítógépünkről a mobilunkat (301), majd fordítva (300). Vegyük fel, beszélgessünk kicsit magunkkal – de csak ha senki nem látja... Ha van ott valaki, inkább nyújtsuk át neki a mobilt, jobb a békesség. Ha működik, örüljünk! Ha nem, javítsuk ki!

F2.3 Kérünk egy képernyőképet a beszélgetésről (akár mobilos screenshot, akár számítógépes).



14. *Játsszon kicsit!* Mi van, ha önmagát hívja (mármint például a student-phone1 a 300-at)? Mit csinál az Auto Answer és a DND (do not disturb) a MicroSIP-ben? (Sose találná ki...)

15. *Pontos idő.* Ha hiszik, ha nem, van egy telefonszám, amit ha felhívnak, bemondja a pontos időt. Régen nagyon népszerű volt, mára eléggé elfeledték, de létezik. Azaz létezik, ha implementálja a szolgáltatást a szolgáltató²⁴. Ez a 180-as szám. Telenoros mobilról próbáltam, nem működött, de Telekomos vezetékes számról igen. Percre pontos volt, de nem mondta be a másodpercet, mint régen. Nos, ezt a szolgáltatást implementáljuk most mi is.

²⁴ https://nmhh.hu/cikk/159/A_18_kezdoszamu_kozerdeku_tajekoztato_es_tamogato_szolgaltatasszamok_es_hasznalatuk_feltetelei

Ezt írja a lábjegyzetben hivatkozott hatósági oldal: „A közérdekű tájékoztató és támogató szolgáltatások bevezetése nem kötelező, de alkalmazásuk esetén a fent felsorolt számok csak a megadott szolgáltatáshoz használhatók.” Arról, hogy az adott szolgáltatás lehet-e más számon, nem ír, de jobb ezzel nem packázni, tegyük mi is a 180-ra!

Ehhez három bejegyzést kell elkészítsünk az `extensions.conf`-ba:

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

Az első sor megválaszolja a hívást, létrejön a kapcsolat. A többi elég egyértelmű.

Próbáljuk ki: írjuk be a konfigurálba, olvassuk újra a konfigot és nézzük meg, működik-e!

F2.4 Mi az első információ (első szó), amit közöl a gépi hang, ha felhívjuk ezt a számot? Stimmel az időzóna a magyar helyi idővel?

```
A nap neve(Thursday)
```

```
Nem stimmel az időzóna. Greenwich-i időt mondja.
```

(Tipp: ld. az OpenWrt-n kiadott `date` parancs kimenetét is!)

3. feladat: Forgatomelemzés protokollanalizátorral

Ez a feladat mindenki számára kötelező!

A cím kicsit félelmetesen hangzik, de csak Wireshark-kal fogunk vizsgálni picit.

1. *Wireshark install.* Ha nem lenne fenn a gépünkön a Wireshark²⁵, telepítsük fel gyorsan! Ha újra kell indítani a számítógépünket, akkor előtte az OpenWrt-t állítsuk le a `poweroff` paranccsal!

2. *tcpdump.* Sajnálatos módon a VirtualBox bridgelt hálózati megoldása kikerüli a Windows hálózati vermet, ezért nem kezdhethetjük el a csomagok direkt elkapását a hálózati interfészen. (Használhattunk volna a bridge mód helyett NAT módot, de ennél a mérésnél nem akarunk port-átírányítással foglalkozni.) A megoldás az, hogy a VM-en belül kapjuk el a csomagokat, majd átküldjük az elkapott forgalmat SSH-n keresztül a gazda oprendszerben futó a Wireshark program bemenetére. Bonyolultnak hangzik, de segítünk, mert e mérés most nem az SSH átírányításról szól.

Először a virtuális gépen installálja a tcpdump-ot, amely a csomagok elkapására szolgál:

```
opkg update
opkg install tcpdump
```

3. *plink.* Most az SSH átírányításhoz szükséges plink.exe-t töltsse le a Putty weblapjáról²⁶. Nem túl elegáns dolog, de így egyszerű: másolja be ezt a fájlt a Wireshark telepítő könyvtárába (C:\Program Files\Wireshark)! Linuxon és Mac-en a beépített SSH parancs elég lesz, nem kell a plink.

4. *Indítás.* Az alábbi Windowsos parancssor elindítja a csomagelkapást a vendég oprendszerben, és a Wireshark-ot a gazdagépen, illetve megteremti a kettő közötti kapcsolatot. Indítson el egy parancssori ablakot Windowsban és lépjen be a Wireshark könyvtárába:

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

majd jöhet a következő parancs (egy sorba):

```
plink -batch -ssh -pw JELSZÓ root@SAJAT_VM_IP_CÍM "tcpdump -ni eth0 -s 0 -U -w - not port 22" | Wireshark.exe -k -i -
```

A fenti parancsban a jelszót és az IP címet persze írja át, mielőtt lefuttatja.

Megjegyzés. Akit érdekel (de még egyszer, nem ez most a mérés tárgya!):

-s 0	teljes csomagok elkapása
-n	nem akarjuk a címeket nevekre konvertálni
-i eth0	a csomagok elkapása az eth0 hálózati interfészen történik
-U	pufferelés megakadályozása
-w -	nem fájlba írás, hanem a standard kimenetre
not port 22	az SSH csomagjait a 22-es portról nem kapjuk el
	az előző parancs kimenete a következő parancs bemenete lesz (pipe)
-k	a csomagok elkapása rögtön elkezdődik
-i -	csomagok olvasása a standard bemenetről

²⁵ <https://www.wireshark.org/>

²⁶ <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Ezzel a Wireshark elindul és gyűjti a csomagokat.

Linuxon²⁷ és Mac-en ugyanez²⁸:

```
ssh root@SAJAT_VM_IP_CÍM "tcpdump -ni eth0 -s 0 -U -w - not port 22" |  
wireshark -k -i -
```

Ezután be kell majd írni a távoli root jelszót.

5. Elfogás! Most hívja fel a 180-at a számítógépéről, hallgassa meg a pontos időt, azután állítsa le a Wiresharkban a csomagelkapást. A programban meg kell jelenjenek a híváshoz tartozó csomagok.

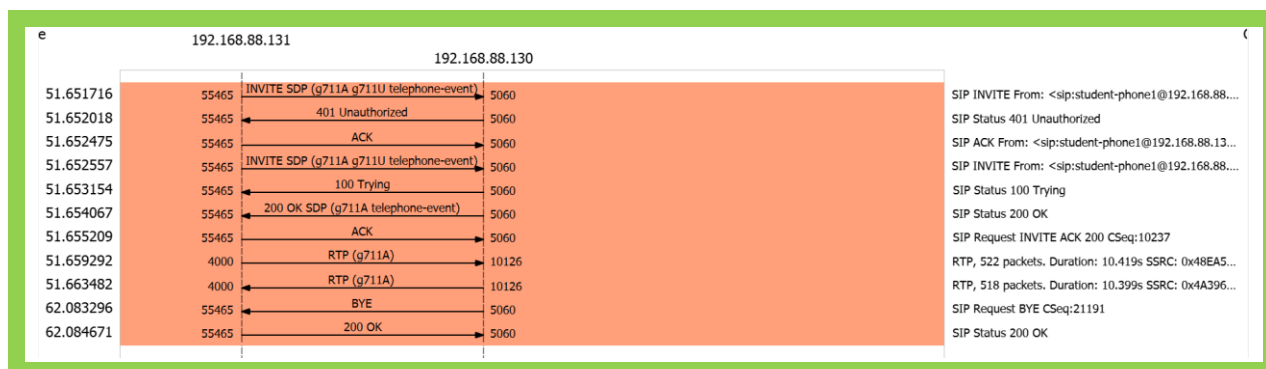
6. Értelmezzük! Állítsa be a Wireshark-ban az ablak tetején a megjelenítési szűrőt sip-re! (Apply a display filter...)

F3.1 Kérünk egy képernyőképet az elkapott csomagokról!

No.	Time	Source	Destination	Protocol	Length	Info
99	34.718470	192.168.88.130	192.168.88.193	SIP	585	Status: 401 Unauthorized
100	34.726901	192.168.88.193	192.168.88.130	SIP	963	Request: REGISTER sip:192.168.88.130;transport=UDP (
101	34.728263	192.168.88.130	192.168.88.193	SIP	572	Status: 200 OK (1 binding)
143	51.651716	192.168.88.131	192.168.88.130	SIP/SDP	1033	Request: INVITE sip:180@192.168.88.130
144	51.652018	192.168.88.130	192.168.88.131	SIP	608	Status: 401 Unauthorized
145	51.652475	192.168.88.131	192.168.88.130	SIP	428	Request: ACK sip:180@192.168.88.130
146	51.652557	192.168.88.131	192.168.88.130	SIP/SDP	1335	Request: INVITE sip:180@192.168.88.130
147	51.653154	192.168.88.130	192.168.88.131	SIP	410	Status: 100 Trying
148	51.654067	192.168.88.130	192.168.88.131	SIP/SDP	970	Status: 200 OK
151	51.655209	192.168.88.131	192.168.88.130	SIP	424	Request: ACK sip:192.168.88.130:5060
1311	62.083296	192.168.88.130	192.168.88.131	SIP	503	Request: BYE sip:student-phone1@192.168.88.131:55465;
1312	62.084671	192.168.88.131	192.168.88.130	SIP	417	Status: 200 OK

7. Flow élmény. Most a Wiresharkban a Telephony menün keresztül válassza ki a SIP Flows-t. Válassza ki a folyamat és kattintson a Flow Sequence-re! Megjelenik a folyamathoz tartozó SIP csomagok időrendbeli ábrázolása.

F3.2 Olyan szép, mentse le a képet!



²⁷ Ha nem megy, célszerű megpróbálni root-ként (`sudo -i` előtte). Linuxon igazából nem is kell semmi ilyesmi parancs, mert Wireshark-ban indulás után csak SSH remote capture-t kell választani és kész. Lehet, hogy ez utóbbi működik Mac-en is, nem próbáltuk. Azt tudjuk, hogy jelenleg a Windowsos Wireshark nem tud ilyet, de sebj.

²⁸ Igazából működik Windows 10 alól is. Ekkor azonban a Wiresharknak a teljes elérési útját be kell írni ("C:\Program Files\Wireshark") a parancsba, vagy abból a könyvtárból kell kiadni e parancsot. Ha először lépünk be így, akkor a kérdésre yes-t kell gépelni a jelszót előtt, de vakon, mert nem látjuk, amit írunk.

8. *Hitelesítés.* Ha minden rendben, akkor a folyamatban az alábbi üzeneteket is látnia kell: INVITE, 401 Unauthorized, INVITE, OK. Ez csak a hívásfelépítés, és az is ennél több üzenetet tartalmaz, de ezek talán a legfontosabbak.

Megjegyzés: Ha több INVITE üzenet is megjelenik, akkor elfelejtette beállítani a G.711 A-law kodek kizárólagos használatát. Ez esetben az első két INVITE üzenet az összes támogatott kodeket tartalmazza és a harmadik INVITE (és a második OK) üzenet tartalmazza a kiválasztott kodeket.

Eddig rendben, de miért látunk két INVITE üzenetet, és mit jelent a „401 Unauthorized” üzenet? Ha jól megnézi, észreveszi, hogy az első INVITE üzenetben a SIP üzenet fejléce nem tartalmaz semmilyen hitelesítési információt. Márpedig autentikáció nélkül nem tud telefonálni, mivel beállította a jelszó kérését a `/etc/asterisk/pjsip.conf` konfigurációs fájlban. Tehát mivel a SIP fejlécben nincs hitelesítési információ, ezért az Asterisk „401 Unauthorized” hibaüzenetet küld vissza. Sőt, azt is megmondja, hogyan várja a hitelesítést.

Megjegyzés. Nem szigorúan része e mérésnek, de azért pár dolgot leírunk erről. Ezt a megjegyzés bekezdést nyugodtan átugorhatja, akit nem érdekel. Pillantson bele az üzenetbe, ilyesmit fog látni:

Authentication Scheme: Digest – a nevet és a jelszót (és mást is) egy hash-en átküldve titkosítjuk

Realm: "asterisk" – ide jelentkezőnk be

Nonce Value: "1614737669/1cfaa..." – ld. nemsokára

Opaque Value: "571c5e4b44815df1" – szerver küldi és neki kell visszaküldeni. Most ne merüljünk el ebben.

Algorithm: md5 – a hash függvény típusa

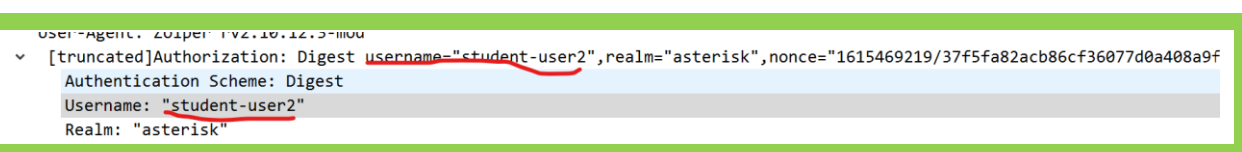
QOP: "auth" – quality of protection

Ez utóbbiról, ill. erről az egészről bővebben, de közérthetően:

https://en.wikipedia.org/wiki/Digest_access_authentication

Most a második INVITE üzenet SIP fejlécében keresse meg a hitelesítési információt tartalmazó részt!

F3.3 Készítsen képet az authorization részről! Jelölje a képen (pl. piros aláhúzással) a felhasználói nevet.



Megjegyzés. A felhasználói név olvasható szöveg, de a jelszó a Digest Authentication Response mezőben már hash-elve van, így ha valaki elkapja a csomagot, nem tudja kiolvasni. Tehát a jelszó nem olvasható szöveg, de támadó azért megteheti, hogy az adott üzenetet megismétli (repeat attack), igaz? Nem, nem tudja megismételni, mivel a jelszó össze van hash-elve egy egyszer használatos kóddal, ez a „nonce” a kérésben. Tehát a következő üzenetben a jelszó már nem úgy lesz titkosítva, mint az előző üzenetben. Ezt nevezik challenge – response hitelesítésnek.

9. *SDP.* A SIP csak a jelzésüzenetek cseréjére szolgál, az adatok – ez esetben: a beszédcsomagok – majd az RTP protokoll felett fognak menni. Az ezzel kapcsolatos információkat egy külön protokoll, a

Session Description Protocol (SDP) segítségével beszélnek meg a felek. Milyen információkról is van szó? A legfontosabbak: a média típusa (hang, video, stb.), az alkalmazott átviteli protokoll (pl. RTP), a média formátuma (pl. az alkalmazott kodek), a média elérhetősége (pl. IP cím, port). Ahogy a Wiresharkban is láthatja, az SDP üzenetei a SIP üzenetekbe vannak beágyazva.

F3.4 Készítsen egy képet a második INVITE SIP üzenet SDP részéről!

```

v Session Initiation Protocol (INVITE)
  > Request-Line: INVITE sip:180@192.168.88.130 SIP/2.0
  > Message Header
  v Message Body
    v Session Description Protocol
      Session Description Protocol Version (v): 0
      > Owner/Creator, Session Id (o): - 3824461638 3824461638 IN IP4 192.168.88.131
      Session Name (s): pjmedia
      > Bandwidth Information (b): AS:84
      > Time Description, active time (t): 0 0
      > Session Attribute (a): X-nat:0
      > Media Description, name and address (m): audio 4000 RTP/AVP 8 0 101
      > Connection Information (c): IN IP4 192.168.88.131
      > Bandwidth Information (b): TIAS:64000
      > Media Attribute (a): rtcp:4001 IN IP4 192.168.88.131
      Media Attribute (a): sendrecv
      > Media Attribute (a): rtpmap:8 PCMA/8000

```

F3.5 Készítsen egy képet a válaszról, azaz az első 200 OK SIP üzenet SDP részéről!

```

> User Datagram Protocol, Src Port: 5060, Dst Port: 55465
v Session Initiation Protocol (200)
  > Status-Line: SIP/2.0 200 OK
  > Message Header
  v Message Body
    v Session Description Protocol
      Session Description Protocol Version (v): 0
      > Owner/Creator, Session Id (o): - 3824461638 3824461640 IN IP4 192.168.88.130
      Session Name (s): Asterisk
      > Connection Information (c): IN IP4 192.168.88.130
      > Time Description, active time (t): 0 0
      > Media Description, name and address (m): audio 10126 RTP/AVP 8 101
      > Media Attribute (a): rtpmap:8 PCMA/8000
      > Media Attribute (a): rtpmap:101 telephone-event/8000
      > Media Attribute (a): fmp:101 0-16
      > Media Attribute (a):ptime:20
      > Media Attribute (a):maxptime:150
      Media Attribute (a): sendrecv
      [Generated Call-ID: 4f60b7e898eb487dbec44af61393e6da]

```

Értelmezzük a látottakat! Ezeknek az üzeneteknek a „Media Description, name and address (m):” részében megtaláljuk a Media Port mezőt, amely tartalmazza mind a két oldal RTP port számát, a Media Format mező pedig a kodektípust tartalmazza. A két oldal IP címei a Connection Address mező „Connection Information (c):” részében található meg. (Wiresharkban a releváns sort előtti kis háromszögre kattintva megkapjuk az adatok részletes magyarázatát.)

F3.6 A kiolvasott adatokat írja le alább az XXXXX-ek helyére!

```
Végberendezés: IP cím: 192.168.88.131 RTP Port: 4000  
Telefonközpont: IP cím: 192.168.88.130 RTP Port: 10126  
A kodek típusa: ITU-T G.711 PCMA
```

Megjegyzés: az 5060 biztosan rossz válasz RTP portnak, mivel a SIP használja az 5060-as portot!

10. Egy kis SIP elmélet. Felmerül a kérdés, hogy miért van szükségünk mindkét oldal IP címére? Miért nem elég csak SIP üzenetben található forrás IP cím ismerete?

Azért, mert a SIP protokoll egyik fő előnye, hogy a jelzési útvonal csak kevés erőforrást (kis sávzélességet) igényel, míg a beszéd átvitele sokkal nagyobb erőforrásigénnyel rendelkezik. Ez lehetővé teszi, hogy egy olcsó, egyszerű „low-end” routert használjon telefonközpontnak, úgy, hogy csak a jelzések mennek át a routeren, és a hangátvitel csak a hívó felek között történik. A mérésnél használt Asterisk verzió ehhez a `direct_media` konfigurációs opciót használja a `pjsip.conf`-ban, aminek az alapértelmezett értéke `yes`. Nagyobb alközponti rendszerek esetén, ahol több ezer SIP melléket (ez az alközponti hívószámot jelenti) is használnak, nem ritka, hogy több DSP kártyát alkalmaznak, ahol minden DSP kártya saját IP címmel rendelkezik. Így ha hívást kezdeményezünk, az INVITE üzenet SDP mezője az egyik DSP kártya IP címét tartalmazza, ezért az alközpont processzora kevésbé lesz terhelve, mivel csak a SIP jelzési protokoll feldolgozásával kell foglalkoznia.

11. RTP vizsgálata. Törölje ki a Wiresharkban felül a `sip`-et a megjelenítési szűrőből! Válassza ki a Telephony menü alatt ezt: RTP / RTP Streams! Válasszon ki a pontos idő bemondásához tartozó RTP folyamatot és kattintson az Analyze-ra!

Megjegyzés: ha minden rendben, a Telephony / RTP / RTP Streams menüt kiválasztva két folyamat fog látni. Az egyik forrása a számítógépe, célállomása az OpenWrt (ld. a Source / Destination Address mezőket), a másikonál épp fordítva. Ez azért van, mert a pontos idő felhívása is egy kétirányú kapcsolatot hoz létre, csak a központ nem foglalkozik azzal, amit közben szóban mondunk neki. Innen már könnyű, melyik folyamatot kell kiválasztania.

F3.7 Válassza ki a központtól a végberendezés felé menő folyamatot és kattintson az Analyze-ra!
Készítsen egy képet a megjelenő ablakról!

Forward		Forward	Reverse	Graph			
		Packet	Sequence	Delta (ms)	Jitter (ms)	Skew	Bandwidth
192.168.88.130:10126 → 192.168.88.131:4000		153	4967	0.00	0.00	0.00	1.60
SSRC	0x4a3969e3	155	4968	20.44	0.03	-0.44	3.20
Max Delta	27.78 ms @ 1123	157	4969	19.94	0.03	-0.39	4.80
Max Jitter	0.91 ms	159	4970	19.51	0.06	0.10	6.40
Mean Jitter	0.50 ms	171	4971	20.75	0.10	-0.65	8.00
Max Skew	-59.80 ms	173	4972	19.43	0.13	-0.07	9.60
RTP Packets	518	175	4973	19.42	0.16	0.51	11.20
Expected	518	181	4974	20.32	0.17	0.19	12.80
Lost	0 (0.00 %)	183	4975	19.98	0.16	0.21	14.40
Seq Errs	0	185	4976	20.28	0.17	-0.08	16.00
Start at	51.663482 s @ 153	187	4977	20.55	0.19	-0.63	17.60
Duration	10.40 s	189	4978	19.34	0.22	0.03	19.20
Clock Drift	-62 ms	191	4979	20.10	0.21	-0.06	20.80
Freq Drift	7952 Hz (-0.59 %)	193	4980	19.87	0.21	0.07	22.40
Reverse		195	4981	19.39	0.23	0.68	24.00
:0 →		197	4982	20.53	0.25	0.15	25.60
:0		199	4983	20.29	0.25	-0.14	27.20
SSRC	0x00000000	201	4984	19.77	0.25	0.09	28.80
Max Delta	0.00 ms @ 0	203	4985	19.94	0.24	0.14	30.40
Max Jitter	0.00 ms	205	4986	20.46	0.25	-0.32	32.00
Mean Jitter	0.00 ms	207	4987	19.48	0.27	0.19	33.60
Max Skew	0.00 ms	209	4988	20.37	0.28	-0.18	35.20
RTP Packets	0	212	4989	19.75	0.28	0.07	36.80
Expected	1	214	4990	19.71	0.28	0.36	38.40
Lost	1 (100.00 %)						

12. *Értelmezzük!* A Wireshark a teljes RTP folyamatot elemzi és különböző statisztikai adatokat ad róla. Ilyenek például a késleltetésingadozás átlaga (mean jitter), az RTP csomagok száma, az elveszített RTP csomagok száma és aránya, a folyam időbeli hossza, valamint a folyam által használt átlagos sávszélesség (kbit/s-ben). Ez utóbbi kivételével a bal oldali összefoglaló táblázatból olvashatjuk ki az adatokat, míg a becsült sávszélességet csomagonként adja meg a Wireshark, így le kell görgetni a csomagok listájának az aljára az állandósult átlag kiolvasásáért.

F3.8 Az előre irányú (Asterisk → végberendezés) folyamból adja meg a következő adatokat!

Késleltetés ingadozás átlaga (mértékegységgel!):	0.5 ms
Az RTP csomagok száma:	518
Az elveszített RTP csomagok aránya (százalékban):	0%
A folyam hossza (időben, mértékegységet is írjon):	10.4 s
Az átlagos sávszélesség (kb/s):	80

Megjegyzés. A „kilo” egy tízes alapú előtag (prefix) az SI rendszerben és ezerrel való szorzásra utal. Így 1 kB = 1000 bájt, 1 kb = 1000 bit, 1 kb/s = 1000 bit/másodperc, stb. A „kibi” egy

kettes alapú előtag, ami $2^{10} = 1024$ -gyel való szorzásra utal. Így 1 KiB = 1024 bájt, 1 Kib = 1024 bit. Az adatsebességeknél hagyományosan a 10 hatványait használjuk, tehát a kilot és nem a kibit.

13. Overhead komponensek. Hmm, itt valami nem stimmel. Ha jól csinálták, akkor az utolsó szám az előző feladatban – az átlagos sávszélesség – nem egyezik meg a PCM-ről tanult 64 kb/s-mal, hanem több annál. Nyilván azért, mert a 64 kb/s az csak a kodek által kibocsátott adatsebesség, a Wireshark által kijelzett sávszélesség pedig tartalmazza a csomagok fejléceit is. Márpedig fejlécből akad nem is kevés: belülről kifelé haladva ott az RTP, az UDP, az IP és az Ethernet fejléce. Nyissunk meg a Wiresharkkal egy tetszőleges RTP csomagot a mentett csomagok közül! Nem lesz nehéz, nagyon sok van belőle. Ha a különböző mezőkre kattint, akkor legalul, a státuszsorban megmutatja a Wireshark az adott mező hosszát, bár kézzel sem nehéz megszámolni a bájtokat.

Tartsuk egy kis számvetést.

F3.9 Egy RTP csomagban mi hány bájt (bájt!, nem bit) hosszú?

A teljes csomag:	214
Az Ethernet keret fejléce:	14
Az IP fejléc:	20
Az UDP fejléc:	8
Az RTP PDU, tehát fejléc és hasznos teher együtt:	172
- ebből RTP fejléc:	12
- ebből hasznos teher (PCM-mel kódolt hang):	160

14. Overhead számítás. Eláruljuk, a Wireshark a sávszélesség meghatározásánál nem számolt az Ethernet keretek fejlécével, így hagyjuk most mi is ki azt, és nézzük csak az IP-t és a hierarchiában felette (a fenti válaszban alatta) lévő dolgokat!

F3.10 Számoljunk kicsit az előző két kérdésre adott válasz alapján! Írja be a keretbe az alábbi kérdésekre adott válaszát!

- Ha 64.000 bit/s a PCM adatsebessége (= 8000 bájt/s), és egy csomagban az előző kérdés utolsó alkérdésére adott válaszban látható darab hasznos bájt szerepel, akkor hány csomagot kell küldeni másodpercenként?
- Valóban ez történik? Ossa el a 3.8 kérdésre adott válaszban látható csomagok darabszámát a szintén ott látható folyam hosszával. Ez a mért értékek a csomagok számának másodpercenként. Mennyi ez?
- Az a) és b) kérdésre adott válaszok hihetően közel vannak egymáshoz?
- Mennyi az IP+UDP+RTP fejlécek összhossza bájtban?
- Ezek alapján hány százalék az overhead, vagyis a „kádba veszett” (szállítási) munka? overhead = fejlécek összhossza / teljes csomag hossza (FIGYELEM! A teljes csomag tehát csak az IP csomag, azaz az Ethernet keret fejléce nélkül értendő, ahogy a fejléc összhossza is.)
- Másodpercenként összesen hány bit (nem bájt!) fejlécet viszünk át?
- Másodpercenként tehát 64 000 hasznos bitet viszünk át, plusz az előző válaszban megadott mennyiségű „haszontalan” bitet. Mennyi akkor a teljes szükséges sávszélesség (bitsebesség)?
- Hogy viszonyul ez a 3.8. kérdés utolsó alkérdésére adott válaszhoz?

- a: 50
b: 49.8
c: Igen

```
d: 200  
e: 20%  
f: 16000  
g: 80000  
h: Megegyezik a kető
```

15. Huhh + replay. Remélhetőleg a fenti számolgatás azzal a jóleső érzéssel zárult, hogy minden stimmel, körbeértünk, kerek a világ. Engedjünk hát meg magunknak egy kis mókát! A Wireshark arra is képes, hogy az egymás utáni RTP csomagokból kivadászva a hangokat lejátssza azt nekünk. A Telephony / RTP / RTM Streams-be visszalépve kijelölhetjük az egyik, másik vagy mindkét folyamat. Ezután az Analyze-ra kattintva előjön a már ismert ablak, és itt a Play Streams-ra kattintva előjön a hanglejátszó.²⁹ Ha mindkét folyamat kijelöltük, akkor a bal csatornában az egyik, jobb csatornában a másik felet halljuk. Cool!

16. Az utolsó oltsa le a villanyt! Zárja be Wiresharkot és nyomja meg Ctrl+C kombinációt a parancssori ablakban, ahol fut a plink vagy az ssh, hogy leállítsa azt. Be is zárhatja azt az ablakot.

²⁹ Ha nagyon régi a Wireshark a gépén, akkor nem fog menni. Ez esetben frissítse! A 3.4.3 verzióval már megy.

Kitérő: egy hibaüzenet, egy kismadár és egy kis angol humor



Bírák még? Egy kis kikapcsolódás. Nem fontos, átugorhatják.

Az Asterisk néha ezzel a hibaüzenettel örvendeztet meg minket:

```
[ dátum, idő] WARNING[15713]: asterisk.c:3401 canary_thread: The canary is  
no more. He has ceased to be! He's expired and gone to meet his maker!  
He's a stiff! Bereft of life, he rests in peace. His metabolic processes  
are now history! He's off the twig! He's kicked the bucket. He's  
shuffled off his mortal coil, run down the curtain, and joined the bleeding  
choir invisible!! THIS is an EX-CANARY. (Reducing priority)
```

```
[ dátum, idő] WARNING[15713]: asterisk.c:1776 set_priority_all: Unable to  
set regular thread priority on main thread
```

Az első gondolat nyilván a „WTF?!”, meg az, hogy jó, hogy szorult némi humor a programozókba. A második jó esetben az, hogy „mégis, ez mi lehet?”. (Komolyan gondolt WTF...)

Nos, az Asterisk egy Linuxon oprendszeren fut. Persze ezen a Linuxon más programok is futhatnak ugyanekkor. Mivel a telefonnál fontos a rövid válaszidő, ezért úgy indítják el ez a szoftvert, hogy a legmagasabb legyen a prioritása. Ugyanakkor ők is érezték, hogy ebből baj lehet, mert teljesen kiszoríthat vele együtt futó más szoftvereket a processzorról. Ezért az lett a megoldás, ami elég fura amúgy, hogy elindítottak egy külön "kanári" szoftvert, ami "csiripel", azaz 5 másodpercenként beír egy csip-csip-csip (/var/run/asterisk/alt.asterisk.canary.tweet.tweet.tweet) nevű fájlba. Ezt nézi az Asterisk, és ha azt látja, hogy egy ideje nem frissült a fájl, akkor „meghalt a kanári”. Ez baj, mert akkor az Asterisk túl sok processzoridőt eszik, és ezért megpróbálja csökkenteni a saját prioritását. Ez az egész amúgy több sebből vérzik, és az adott példában nem is sikerül neki.

Minden bizonnyal onnan az analógia, hogy régen a bányákban is használtak kanárimadarakat szénmonoxid-érzékelőnek. Ha megdőglött a kanári, tanácsos volt sietősen távozni.

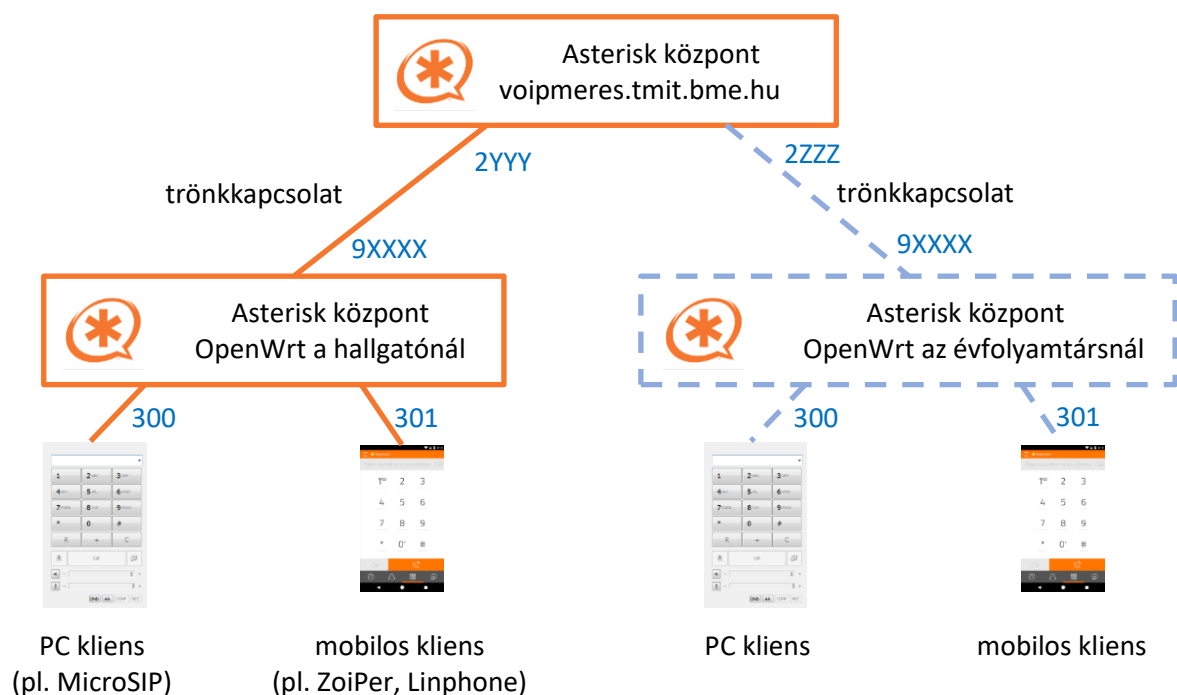
Amúgy a hibaüzenet javarészt egy Monty Python idézet, igaz ott a „kék norvég” egy papagáj volt: <https://youtu.be/vZw35VUBdzo?t=168>.

4. feladat: SIP trönk, avagy kapcsolat egy távoli telefonközponttal

Ez a feladat mindenki számára kötelező!

A mérés kidolgozása során arra gondoltunk, jó lenne a mérésbeli telefonunkat valami igazán jó dologra használni: arra, hogy összekössön minket a barátainkkal. Létrehoztunk ezért mi is egy Asterisk alapú telefonközpontot egy tanszéki szerveren (voipmeres.tmit.bme.hu). Az Ön utolsó kötelező feladata az lesz, hogy a két központ között egy ún. trónkkapcsolatot hozzon létre. Ez ugye a központok közötti összeköttetés neve. Angolul trunk, a 2. előadáson volt erről szó. Ha Önnel egy időben valamely évfolyamtársa, barátja is mér, és mindketten eljutnak odáig, akkor akár egymást is felhívhatják majd! Ez persze nem kötelező, enélkül is lesz értelme a feladatnak és enélkül is lehet jelesre teljesíteni.

A 2. ábrán tüntettük fel a megvalósítani kívánt rendszert. Kékkel a hívószámokat jelöltük, ezekről később írunk bővebben.



2. ábra. A mérési elrendezés

A rendszer megvalósításához mindkét (mindhárom) oldalon fel kell konfigurálni a telefonközpontokat. A mi részünket korábban megcsináltuk, most Önön a sor.

1. *Előkészületek.* A konfiguráláshoz szükség lesz némi személyre szabott információra. Mire ezt olvassa, már kapott egy emailt, amelyben a következő adatokat adtuk meg Önről, Önnek:

- Neptun kód (ezt nyilván tudta)
- Hívószám (ahol Önt el lehet majd érni)
- A felhasználói neve
- Az azonosításhoz szükséges név
- Az azonosításhoz szükséges jelszó

Keresse meg ezt a levelet, most szükség lesz rá!

2. PJSIP konfiguráció. Nyissa meg ismét nano-val a `/etc/asterisk/pjsip.conf`-ot. Itt most két dolgot fogunk beállítani.

- Először egy trónkapcsolatot a BME-s szerver felé, ahova be kell regisztrálnia a saját Asterisk központjának.
- Másodszor pedig a távoli központot mint végpontot állítjuk be, ahova mehet tőlünk kimenő hívás és ahonnan érkezik hozzánk bejövő hívás.

3. Trónkapcsolat regisztrációval. A `pjsip.conf`-ban a 162. sor környékén találja meg az „OUTBOUND REGISTRATION WITH OUTBOUND AUTHENTICATION” részt, ami most érdekes számunkra. A fájlban szerepel egy példa két szakasszal: `[mytrunk]` és `[mytrunk_auth]`. Ezekből indulunk ki, ezeket módosítjuk picit.

Kezdjük a `[mytrunk]`-kel! Írjuk át a fejléct `[voipmeres-trunk]`-re, így jelezve, hogy a „voipmeres” nevű szerverhez kapcsolódunk vele. Szedjük ki a teljes szakasz elől (beleérve a fejléct) a kommentet jelentő pontosvesszőt! A `type` és `transport` sorok jók ahogy vannak, és talán érthetőek is. Be szeretnénk regisztrálni egy távoli szerverhez, és UDP-t fogunk használni. Regisztrálni amúgy azért kell, mert a távoli szerver nem tudja az IP címünket előzetesen.

Az `outbound_auth`-ban adjuk meg, hogy mi az azonosítást leíró szekció neve. Írjuk át `voipmeres-trunk_auth`-ra. Figyeljük meg, hogy ez nem `auth`, mint eddig, hanem `outbound_auth`, hiszen a kimenő irányú regisztrációhoz tartozik.

A `server_uri` legyen `sip:voipmeres.tmit.bme.hu`, ehhez nem kell sok magyarázat.

A `client_uri` legyen `sip:FELHASZNÁLÓNÉV@voipmeres.tmit.bme.hu`, ahol a felhasználónevet emailben megkapta és `studentNNN-pbx` alakú, ahol *NNN* egy (1, 2 vagy 3 jegyű) szám.

A `contact_user` legyen ugyanaz a felhasználónév, mint az előző sorban (csak a felhasználónév, a `@` és az azutáni rész nem kell).

A `retry_interval` azt mondja meg, hogy hiba esetén hány másodperc múlva próbáljuk újra. Jó lesz a 60.

A `forbidden_retry_interval` ennek egy változata, amikor a szerver nem engedett be minket. Nem kéne előforduljon, de ha mégis, ne várjunk 10 percre: vegye le ezt is 60-ra!

`expiration=3600`: Mennyi időként regisztráljunk újra (sikeres regisztráció után). Jó az alapértelmezett 3600 mp. = egy óra.

`line=yes`: Jó ahogy van. A bejövő hívások és kimenő regisztrációk összerendelésére szolgál.³⁰

Az `endpoint` legyen `voipmeres-trunk`.

Eddig jó, most jön a `[mytrunk_auth]` rész. Gyorsan írjuk át a fejléct az előzőek alapján `[voipmeres-trunk_auth]`-ra! Szedjük ki a pontosvesszőket! Mentsünk gyakran (nano-ban: Ctrl-o)!

³⁰ Ha esetleg valakit mélyebben érdekelne, itt talál részletesebb leírást: <https://www.asterisk.org/the-pjsip-outbound-registration-line-option/>

Az első két sor (type, auth_type) jó így. A jelszót és a nevet emailben kapta. Figyelem, a username most az „azonosításhoz szükséges név”, nem a „felhasználói név”, amelynek alakja studentNNN-user. Írja be a kapott adatokat!

A realm legyen voipmeres.tmit.bme.hu.

Ezzel a kimenő trönk regisztrációs beállításai készen vannak. Mentse a beállításokat, de még ne lépjen ki a szövegszerkesztőből!

F4.1 Másolja ide az imént elkészített két szakasz tartalmát!

```
[voipmeres-trunk]
type=registration
transport=transport-udp
outbound_auth=voipmeres-trunk_auth
server_uri=sip:voipmeres.tmit.bme.hu
client_uri=sip:student546-pbx@voipmeres.tmit.bme.hu
contact_user=student546-pbx
retry_interval=60
forbidden_retry_interval=60
expiration=3600
line=yes
endpoint=voipmeres-trunk

[voipmeres-trunk_auth]
type=auth
auth_type=userpass
password=eFk4f7P7SUN
username=student546-user
realm=voipmeres.tmit.bme.hu
```

3. Végpont beállítása. Most jön annak a beállítása, hogy a távoli telefonközpont egy végpont számunkra. Nem kell messze menni, a konfigurációs fájlban most következő „ENDPOINT CONFIGURED AS A TRUNK, OUTBOUND AUTHENTICATION” rész az, ami nekünk kell.

Mindhárom itteni szakasz [mytrunk] fejléccet visel, ezeket cseréljük ki gyorsan [voipmeres-trunk]-re!

Az első szakasz első kettő és negyedik sora (type, transport, disallow) jó is ahogy van. A context legyen [kulso-hivas], az allow-t pedig írja át ulaw-ról alaw-ra. Sok magyarázat ezekhez nem kell: végpontot konfigurálunk, UDP a szállítási protokoll, a dialplan-ban a [kulso-hivas] szakaszban írjuk le az innen bejövő hívásokra vonatkozó szabályokat, és kizárólag PCM A-törvényű kodek használatát engedélyezzük.

Az outbound_auth legyen voipmeres-trunk_auth az aors pedig értelemszerűen voipmeres-trunk.

A három NAT-os opcióból csak a direct_media=no elől vegye ki a pontosvesszőt, a másik kettő (force_rport, ice_support) maradjon kikommentezve, sőt akár ki is törölheti azokat. Írja viszont be még ezeket a sorokat:

```
from_user=studentNNN-pbx
send_pai=yes
send_rpid=yes
```


Az első sor (értelemszerűen a saját felhasználónévére cserélve!) elküldi a távoli központnak a felhasználói nevét, amely alapján az beazonosítja. Ez fontos. A második és harmadik további adatokat küld Önnek, amelyek a hívószám-kijelzésnél jöhetnek jól³¹, ezt azonban a távoli központ jelenleg nem veszi figyelembe. Sebaj.

A következő szakasz címét már frissítette. A típus marad `aor`, ez a sor kell. A `contact`-ból viszont csak egy kell, ezzel a tartalommal: `sip:voipmeres.tmit.bme.hu:5060`. Ezzel mondjuk meg, hol található a végpont.

A harmadik, egyben utolsó szakasz típusa `identify` (komment jöhet ki előle!). Itt írja le, hogy a trónkőn bejövő kéréseket hogyan hitelesítjük. Leginkább sehog, azaz egyszerűen IP cím alapján. Ha onnan jött, jó. Ezért a 2. és 3. sorokat írja át ilyenre:

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

Kész! A konfigurációs fájlban lévő következő résszel („ENDPOINT CONFIGURED AS A TRUNK, INBOUND AUTH AND REGISTRATION”) nem kell most foglalkozni. Ez csak a trónkő másik végén érdekes, mi be is állítottuk a `voipmeres.tmit.bme.hu` szerveren.

F4.2 Másolja ide az imént elkészített három szakasz tartalmát!

```
[voipmeres-trunk]
type=endpoint
transport=transport-udp
context=kulso-hivas
disallow=all
allow=alaw
outbound_auth=voipmeres-trunk_auth
aors=voipmeres-trunk
;
;A few NAT relevant options that may come in handy.
direct_media=no ;of these options.
from_user=student546-pbx
send_pai=yes
send_rpid=yes

[voipmeres-trunk]
type=aor
contact=sip:voipmeres.tmit.bme.hu:5060

[voipmeres-trunk]
type=identify
endpoint=voipmeres-trunk
match=voipmeres.tmit.bme.hu
```

4. Dialplan. Mentés után zárja be a `psip.conf`-ot és nyissa meg szerkesztésre ugyanitt az `extensions.conf`-ot!

Két dolgot kell beírni: a kimenő hívásokat irányítsa a tanszéki Asterisk központhoz, illetve az onnan bejövőket kell kezelje.

³¹ Bővebben, akit érdekel: <https://wiki.asterisk.org/wiki/display/AST/Asterisk+16+Configuration+res+psip>

5. *Kimenő hívások.* Az a terv, hogy a 9-es szám tárcsázásával kapunk – ahogy régen hívták – „városi vonalat”, azaz ez után a prefix után kell a tanszéki szerver hívószámait tárcsázni. Ha tehát majd a barátját szeretné felhívni, akinek a hívószáma mondjuk 2566, akkor 92566 tárcsázásával érheti ezt el.

Megjegyzés. Hagyományosan a 9-es számjegy tárcsázása után egy új tárcsahangra kellett várni, amelyet a távoli központ generált. Erre most SIP használata mellett nem lesz szükség, így egybe lehet írni a 9-et az utána következő számjegyekkel.

A voip.tmit.bme.hu-n négyjegyűek a hívószámok. Azt kell tehát Önnek beírni a konfigurációs fájlba, hogy ha 9-essel kezdődik egy ötjegyű hívószám, akkor a hívást küldje tovább a trónkón, az első számjegy (9) levágása után. Mindez a [belso-hivas] szakaszba, hiszen ez egy belső számról indított hívás. Hogy gyorsítsunk az amúgy is hosszú mérésen, megadjuk, mit kell beírni:

```
exten => _9XXXX,1,Dial(PJSIP/${EXTEN:1}@voipmeres-trunk)
exten => _9XXXX,n,Hangup()
```

Rövid magyarázat: a _9XXXX egy minta, amelyre az ötjegyű, 9-cel kezdődő számok illeszkednek. Az 1 a prioritás (a két sorból, ami e számokra vonatkozik, ezt vegye előre), az \${EXTEN:1} pedig a hívószám, megfosztva az első számjegytől: ezt küldjük tovább. A letevés (hangup) már ismerős.

6. *Bejövő hívások.* Az egyszerűség kedvéért minden bejövő hívást ugyanarra a végberendezésre irányítunk. Ez legyen mondjuk a számítógépünkön futó kliens (MicroSIP vagy valami hasonló). Később persze átírhatja, de most legyen így. Ezeket írja be az extensions.conf-ba a [belso-hivas] szakasz utolsó sora után:

```
[kulso-hivas]
exten => TELSZÁM,1,Set(CALLERID(num)=9${CALLERID(num)})
exten => TELSZÁM,n,Dial(PJSIP/student-phone1)
exten => TELSZÁM,n,Hangup()
```

TELSZÁM helyett írja be az Ön kívülről elérhető hívószámát. Emailben kapta, négyjegyű, az első számjegye 2. Az első sor a kijelzett számot alakítja át: a hívó bejövő hívószáma elé egy 9-est tesz, hiszen így tudjuk visszahívni. A második sorban mondjuk meg, hova (mely végpontra) irányítjuk a bejövő hívásokat. A harmadik megszakítja a hívást és az újabb Asterisk verziókban elhagyható.

F4.3 Másolja ide az extensions.conf első részét: mindent, amit eddig a mérés kezdete óta beleírt!

```
[belso-hivas]
exten => 300,1,Dial(PJSIP/student-phone1)
exten => 300,n,Hangup()
exten => 301,1,Dial(PJSIP/student-phone2)
exten => 301,n,Hangup()

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

exten => _9XXXX,1,Dial(PJSIP/${EXTEN:1}@voipmeres-trunk)
exten => _9XXXX,n,Hangup()

[kulso-hivas]
exten => 2545 aM,1,Set(CALLERID(num)=9${CALLERID(num)})
exten => 2545 aM,n,Dial(PJSIP/student-phone2)
exten => 2545 aM,n,Hangup()
```

7. *Config refresh*. Olvastassa újra az Asterisk konfigurációs fájlokat a `service asterisk reload` parancsot kiadva vagy a 2. fejezet 8. pontjában leírt valamelyik hasonló megoldással.

8. *Próbálja ki!* Hívja fel az egyetemi szerveren a 9001-es hívószámot. Ehhez persze 99001-et kell tárcsáznia. A tárcsázáshoz bármelyik kliensét használhatja.

F4.4 Mit hallott?

Sípszó. Tanár Úr monológja. Örömóda.

9. *„Játsszunk most együtt, amíg csak lehet!”³²* Ha van kedved, játszhat még egy kicsit az elkészült rendszerrel. Pár ötlet ehhez:

- Ha van valaki, akivel egyszerre végzi a mérést, és ő is eljutott idáig, akkor most felhívhatják egymást.
- Játsszhat azzal is, hogy a bejövő hívásokat a mobilos kliensre irányítja át.
- Vagy megnézheti a SIP regisztrációt is Wiresharkban, ez elég tanulságos. Ehhez lépjen ki teljesen a MicroSip-ből (háromszög menü és Exit), indítsa el a csomagelkapást és indítsa újra el a MicroSIP-et.
- Az Asteriskben be lehet állítani sokféle érdekes szolgáltatást, egy-kettőre mutatunk példát az IMSc feladatok között a következő fejezetben. Nem nehezek azok sem.

A mérés kötelező része ezzel véget ért!

³² <https://www.youtube.com/watch?v=Q26itrSbQ0s> (nem Rickroll!)

5. Bónusz feladat: További telefonos alkalmazások (IMSc)

Ez a feladat nem kötelező. IMSc pontok kaphatók érte, de ez a mérésre adott érdemjegyet nem befolyásolja. A jegyet pusztán a nem IMSc-snek megjelölt feladatok megoldásai alapján adjuk.

Már így is elég hosszú ez a doksi, de van pár jópofa lehetőség, amit szerettünk volna még megmutatni. Ha gondolja, próbálja ki, nem fog túl sokáig tartani.

1. Echo with extras. Próbálja ki a következőkkel bővíteni az `extensions.conf` `belso-hivas` szakaszát:

```
exten => 312,1,Answer()  
same => n,Wait(1)  
same => n,Playback(beep)  
same => n,Record(samplefile:gsm)  
same => n,Wait(1)  
same => n,Playback(beep)  
same => n,Playback(samplefile)  
same => n,Hangup()
```

A `same` használata arra jó, hogy ne kelljen a hívószámot újra és újra leírni.

Mentse és töltse be újra a fenti konfigurációt! Most a SIP kliensből hívja fel a 312 számot és beszéljen egy keveset a sípszó után, majd nyomja meg a # gombot, hogy leállítsa a felvételt. Egy sípszó után az Asterisk visszajátssza a felvett hanganyagot.

Az Asterisk-ben tárolt hangok (például a sípszó: `beep` és a felvett hanganyag: `samplefile`) a következő könyvtárban található meg: `/usr/share/asterisk/sounds/`. A fájlok kiterjesztése `.gsm`, mivel ezt a kodektípust használják, ez ugyanis elég jól tömörített, és az OpenWrt általában olyan routereken fut, amelyek nagyon korlátozott memóriakapacitással rendelkeznek.

2. DND. A fentiek alapján bővítse a számozási tervet a 313-as hívószámmal, amelyet felhívva bemondja egy közepesnél kevésbé kedves női hang, hogy „do not disturb”. Töltse be újra a beállításokat és próbálja ki!

F5.1 Másolja ide a konfigfájl releváns sorait!

Ide.

3. „Légy vidám, vagány akár egy srác! / Fújd meg a tülköt, trombitálj, valamit játssz!” Játsszunk hát valamit! Az lesz a feladat, hogy a hívó félnek be kell ütni egy négyjegyű titkos kódot, majd ezután lejátszunk neki a virtuális gép Linux verzióját Morse kódolással. Van értelme? Nem sok. Vicces? Hááát.... Nosza!

Először fel kell telepíteni pár további csomagot (Asterisk modult) a gépünkre. Ezt írja be egy sorba:

```
opkg install asterisk16-app-authenticate asterisk16-func-shell asterisk16-  
app-morsecode
```

Ezután újra kell indítani az szolgáltatást. A konfiguráció újratöltése most nem elég, mert új moduljaink is vannak:

```
service asterisk restart
```

Ezután oldja meg egyedül a következő feladatot. Írjon be olyan sorokat az extensions konfigurációs fájlba, amelyek a következőket csinálják: ha felhívja a 314 számot, a központ válaszol és egy legfeljebb négyjegyű autentikációs kódot kér. Ha megfelelő a kód (a jó kód legyen 9999), akkor Morse kódolással lejátssza a VM Linux verzióját majd befejezi a hívást. Ha a Morse kódolást nehéznek találja debuggolni, akkor használhatja az egyszerűbb SayAlpha() parancsot is.

Az elvégzéshez szükséges információk a következő oldalakon találhatók:

[https://wiki.asterisk.org/wiki/display/AST/Application Authenticate](https://wiki.asterisk.org/wiki/display/AST/Application+Authenticate)

[https://wiki.asterisk.org/wiki/display/AST/Asterisk+16+Application Morsecode](https://wiki.asterisk.org/wiki/display/AST/Asterisk+16+Application+Morsecode)

<https://www.voip-info.org/asterisk-func-shell/>

[https://wiki.asterisk.org/wiki/display/AST/Application SayAlpha](https://wiki.asterisk.org/wiki/display/AST/Application+SayAlpha)

Segítség:

`uname -r` shell parancs kimenete a Linux kernel verziója. Ha egy függvény paramétereként akarja használni, akkor a következő a szintaxis: `$(SHELL(parancs))`

F5.2 Másolja ide a konfigfájl releváns sorait!

Ide.

Most már tényleg vége!