…Mérési segédlet

Összeállította: Majdán András, Heszberger Zalán

E segédlet a távoktatásban végzett méréshez is használható.

1. Bevezető

A mérés során megismerkedünk IP alapú asztali és szoftveres telefonokkal, telefonos szolgáltatásokkal, számítógép virtualizációval és a hálózati csomagelemzéssel. Minden hallgató egy virtualizált saját IP telefonközpontot is kipróbálhat. Ahhoz hogy a mérés során megértsék az alkalmazott technikákat, a mérési útmutatóban röviden ismertetett témák elolvasása szükséges. Az ellenőrző kérdések nem véletlenül vannak, megválaszolásuk **erősen ajánlott**.

Beszédkódolási eljárások (kódekek)

A beszédkódolás célja az analóg beszédhangok digitális formába történő átalakítása. A kódolók arra törekednek, hogy minél kisebb sávszélesség mellet minél jobb hangminőséget nyújtsanak.

A beszédkódoló eljárások tulajdonságai:

- **Igényelt sávszélesség**: a kódolt hang sávszélesség igénye. Bizonyos kódekek esetén ez lehet adaptív is, például AMR.
- **Hangminőség**: egy 1-től 5-ig terjedő MOS (Mean Opinion Score) érték segítségével szokták jellemezni. Ezt az értéket nem programok, hanem valódi hallgatói csoportok szubjektív pontozásának átlagolása alapján határozzák meg. A MOS értéke:
    - 5 - Kiváló. Nem vehető észre a kódolás.
    - 4 - Jó. Észlelhető a kódolás, de nem zavaró.
    - 3 - Megfelelő. Kicsit zavaró csak a kódolás.
    - 2 - Gyenge. Zavaró a kódolás.
    - 1 - Rossz: Nagyon zavaró a kódolás
- **Komplexitás**: milyen számítási igényű a kódoló. Beágyazott illetve korlátozott energiával rendelkező rendszerek esetén ez fontos kritérium.
- **Érzékenység a csomagvesztésre**: kihagyható-e a hiányzó minta, van-e olyan predikáció amivel kitalálható a hiányzó részlet
- **Késleltetés**: a kódoló algoritmusa által igényelt idő

Mi a mérés során csak a G.711 (PCM) beszédkódolóval találkozunk. Ezt az ITU-T (ITU Telecommunication Standardization Sector) által meghatározott kódeket minden VoIP eszköz ismeri és ezt használják az ISDN digitális vonal esetén is, így például az IP és ISDN alapú hangkommunikáció között nem szükséges átkódolást alkalmazni. Két típusa van, az Észak-Amerikában használt μ-Law, és a világ más részein (például Európa) használt A-Law.

Paraméterei:

- Sávszélesség: 64 kbit/s
- Késleltetés: 0.125 ms
- MOS érték: 4.1
- Számításigény: 0.52 MIPS (félmillió utasítás/mp)

A G.711 jó hangminőséget nyújt, kis késleltetéssel, alacsony komplexitás mellett. Sávszélesség igényben azonban vannak jóval jobban teljesítő kódekek. Például a régi 2G mobiltelefon hálózatokon használt GSM 06.10 kódeknek elég 7 kbit/s vagy 13 kbit/s sávszélesség is (módtól függően), de MOS értéke csak 3.5-3.9. A G.729-es kódek csak 8 kbit/s sávszélességet igényel és MOS értéke 3.9, azonban nagyon magas számításigényű és szabadalmakkal védett.

RTP:

Az RTP (Real-time Transport Protocol) valós idejú forgalom számára nyújt szállítási szolgáltatást. Az IP telefonok esetén ez hang vagy videó továbbítást jelent. Általában UDP protokoll felett használjuk, unicast vagy multicast címzéssel.

A protokollnak két része van:

- **RTP**: szállítja a való idejű adatot
- **RTCP (RTP Control Protocol)**: felügyeli az átvitel minőségét, információt hordoz a részvevőkről

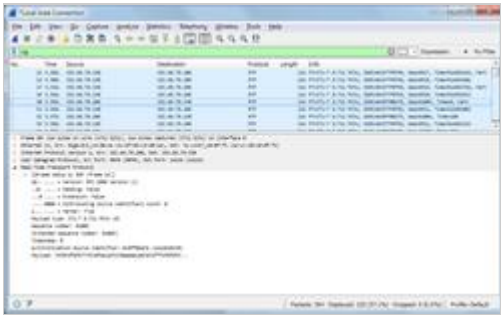
Az RTP csomag fejléce:

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                               ....                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A legtöbb mező ismeretére nincsen szükségünk, ugyanis technikai jellegűek, csak a csomag megfelelő értelmezéséhez szükségesek (verzió, kitöltési bit, van-e fejléc kiterjesztés, stb..) Ezt majd a csomagelemző helyettünk elvégzi. Az RTP továbbá információkat tartalmaz a vevők és adók közötti viszonyról, amit mi nem vizsgálunk

Számunkra a következő mezők fontosak:

- **PT: Payload type. Hasznos teher típusa**. Ez határozza meg az IP hangátvitel esetén a használt kódolót. G.711 esetén két érték lehetséges: 0 -> µ-Law, 8 -> A-Law.
- **Sequence number: Sorszám.** Egyesével növekszik, minden elküldött csomaggal. Ezáltal észlelhető a csomagvesztés, illetve állítható helyre a megfelelő sorrend. Kezdeti értéke véletlen.
- **Timestamp: Időbélyeg.** Az RTP csomag adatrészében található első bájt, monoton és lineárisan növekvő órától vett mintavételezési időbélyege. Kezdeti értéke véletlen.



1. ábra: RTP csomagok elemzése, Payload type: 0

Fontos hogy az RTP nem próbálja a csomagvesztést helyrehozni.

Az RTCP vevő (RR) és adó (SR) jelentésekkel írja le az RTP folyamok minőségét, de kapcsolatbontást és alkalmazásra jellemző adatokat is hordozhat. Általa képet kaphatunk a csomagvesztésről és a késleltetés ingadozásról is.

A mérés során, amikor a sávszélesség igényt kell meghatározni, nem szabad elfelejteni a protokollok rétegezésének többletét (40 bájt összesen):

- IP: 20 bájt
- UDP: 8 bájt
- RTP: 12 bájt

Ebből látható hogy a csomagban tárolt minta időtartamától is függ a sávszélesség igény: minél hosszabb minta van egy csomagban, annál kisebb lesz az overhead és így a sávszélesség igény. Azonban a csomag tárolt minta méretével a késleltetés is nő, hiszen később küldhető csak el.

SIP:

A áramkörkapcsolt hangátviteli technika szerepét az évezred utolsó éveiben kezdték átvenni az olcsó nagy távolságú csomagkapcsolt hálózatok (például a globális Internet). Az ITU-T által ajánlott csomagkapcsolt hálózati hangátvitel a H.323 protokoll család volt. Ez erősen épített a hagyományos áramkörkapcsolt hálózatok logikájára és sok terminológiát vett át belőle. Bináris protokollként nagyon takarékoskodott a sávszélességgel, azonban implementálása így igen összetett feladat volt.

Az H.323 ajánlás megjelenése környékén kezdett munkálkodni az IETF (Internet Engineering Task Force) szervezet is egy olyan VoIP protokollon, amely az egyszerűséget és átláthatóságot célozta meg. Megpróbálta nem újra feltalálni a kereket, hanem meglévő, az Interneten már elfogadott mechanizmusokat használt. A protokoll neve SIP (Session Initiation Protocol) lett.

A SIP jelzési protokoll, hang csomagokat nem hordoz, de azok szállítási végpontjait meghatározza. A protokoll szöveges (a H.323-al ellentétben), így csomagelemző nélkül is olvasható, megérthető. Szerkezete nagyon hasonló a HTTP protokollhoz, annyira hogy a hibaüzenetek sokszor megegyeznek. A klienseket a SIP URI azonosítja, mely tipikusan név@cím alakú, ahol ha van IP telefonközpont, akkor a cím (IP vagy domain név) a telefonközponté. Felfedezhető hogy ez a címzési forma az email rendszerekből származik.

A SIP alapú rendszerek legfontosabb résztvevői:

- User agent (UA), mely lehet:
    - asztali vagy vezeték nélküli hardveres IP telefon
    - alközpont
    - számítógépes VoIP kliens program, úgynevezett softphone
- Szerver, mely lehet:
    - Proxy server: a hívás felépítésében közvetítői szerepet lát el
    - Redirect server: a továbbítás irányának meghatározására szolgáló alkalmas eszköz
    - Location server: a felhasználók aktuális helyét tárolja
    - Registar: a user agent regisztrációját végzi

A SIP kommunikáció kliens-szerver alapú, azonban a hívó és hívott fél is mindkét szerepben fellép. A szerver oldalon a SIP standard portszáma az 5060-as UDP vagy TCP port, ehhez csatlakozik a kliens oldal.

Fontosabb SIP üzenetek:

- INVITE: összeköttetés kezdeményezés
- ACK: megerősíti a hívásfelépítést
- OPTIONS: lekéri a támogatott szolgáltatásokat
- REGISTER: a felhasználó jelzi hogy hol van. Általában felhasználói azonosító és jelszó szükséges, valamint periodikusan ismételni kell.
- BYE: kapcsolat bontása
- CANCEL: ha egy helyről már válasz érkezett a kérésre, ezzel törölhetjük a többi kérést. Például egy hívás a telefonok egy csoportján csörög, majd felveszi valaki. Ha nem kap CANCEL üzenetet a többi telefon, akkor missed call-ként, vagyis nem fogadott hívásként jelenik meg rajtuk.
- SUBSCRIBE/NOTIFY: feliratkozás eseményekre, értesítés eseményekről. Például jelentést kaphatunk arról, hogy a felhasználó off-line van on-line (presence service).

A SIP kérésekre minden esetben érkezik válaszüzenet, mely a következők lehetnek:

- 1xx - Átmeneti válasz, a feldolgozás folyamatban van, követnie kell egy végleges válasznak. Például: 100 - Trying vagy 180 - Ringing.
- 2xx - Sikeres végrehajtás. Például 200: - OK
- 3xx - Átirányítás. A hívás felépítéséhez további műveletek szükségesek. Például: 302 - Moved temporary
- 4xx - A kért művelet nem végrehajtható, hiba történt. Például: 401 - Unauthorized
- 5xx és 6xx szerver vagy általános hiba

Egy példa INVITE üzenetre::

```
INVITE sip:kitti@172.16.0.75:5060 SIP/2.0
Via: SIP/2.0/UDP 172.16.0.135:5060;branch=z9hG4bK0017530a;rport
Max-Forwards: 70
From: <sip:101@172.16.0.135>;tag=as010dc147
To: <sip:kitti@172.16.0.135>;tag=f7eb8d16-dd11-1910-817c-4437e6aa81bd
Contact: <sip:101@172.16.0.135:5060>
Call-ID: f7eb8d16-dd11-1910-817d-4437e6aa81bd@majdan-THINK
CSeq: 102 INVITE
User-Agent: Asterisk PBX 1.8.32.3
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO,
PUBLISH, MESSAGE
Supported: replaces, timer
Content-Type: application/sdp
Content-Length: 264

v=0
o=root 1007958305 1007958306 IN IP4 172.16.0.153
s=Asterisk PBX 1.8.32.3
c=IN IP4 172.16.0.153
t=0 0
m=audio 16000 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=silenceSupp:off - - - -
a=ptime:20
a=sendrecv
```

Az első sorból látszik, hogy a hívott fél a 172.16.0.75 IP címen lévő "kitti". A Via mező megmutatja, hogy mely SIP proxy szerver továbbítja az üzenetet. A From mezőből kiolvasható, hogy a hívó a "101" a 172.16.0.135-ös IP telefonközpontról. A csomag végén szereplő egy betűs mezők az SDP (Session Description Protocol) blokk részei. Ebből kiolvasható hogy az RTP hangcsomagok G.711 PCM μ-law kódolásúak (PCMU/8000 és az AVP 0 bejegyzésekből). Az hangcsomagok útjából lehet következtetni arra is, hogy direkt RTP kapcsolat épül fel a két fél között, ugyanis az SDP blokkban szereplő 172.16.0.153 IP cím nem egyezik a proxy címével, így ő a média folyamba nem avatkozik bele. Az m mezőből az is kiolvasható hogy a 16000-as RTP portot kell használni a 172.16.0.153 IP címen.

```
SIP/2.0 200 OK
CSeq: 103 INVITE
Via: SIP/2.0/UDP 172.16.0.135:5060;branch=z9hG4bK5d0f2266;rport=5060;
received=172.16.0.135
User-Agent: Ekiga/4.0.2
From: <sip:101@172.16.0.135>;tag=as010dc147
Call-ID: f7eb8d16-dd11-1910-817d-4437e6aa81bd@majdan-THINK
To: <sip:kitti@172.16.0.135>;tag=f7eb8d16-dd11-1910-817c-4437e6aa81bd
Contact: <sip:kitti@172.16.0.75:5060>
Allow:
INVITE,ACK,OPTIONS,BYE,CANCEL,SUBSCRIBE,NOTIFY,REFER,MESSAGE,INFO,PING,PRACK
Content-Length: 218
Content-Type: application/sdp

v=0
o=- 1456043216 3 IN IP4 172.16.0.75
s=Ekiga/4.0.2
c=IN IP4 172.16.0.75
t=0 0
m=audio 5064 RTP/AVP 0 101
a=sendrecv
a=rtpmap:0 PCMU/8000/1
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=maxptime:240
```

Kitti felvette a hívást, a hang csomagokat RTP protokollon a 172.16.0.75 IP cím 5064 portjára kell küldeni.

Asterisk:

2. ábra: Digium IP telefon

3. ábra: Digium analóg vonali illeszt?kártya

Az Asterisk egy ingyenes, nyílt forráskódú kommunikációs keretrendszer, amelyből IP telefonközpontok, átjárók hozhatóak létre. Mivel mi IP telefonközpontként használjuk ezért így hivatkozok rá, azonban testreszabhatósága miatt számos felhasználási lehetősége van. Az Asterisk projekt első kódját 1999-ben Mark Spencer adta ki, terjedése akkoriban annak volt köszönhető, hogy átjárót kínált az analóg és a digitális világ között. Mindezt nem drága DSP (digitális jelfeldolgozó) kártyákkal érték el, hanem a Zapata Telephony Project-nek köszönhetően az egyre erősebb számítógépes processzorokra hagyatkoztak és csak egy minimális áramkört alkalmaztak az analóg vonali (PSTN) illesztésre.

Az Asterisk Linux alapú rendszer, a kezdetektől fogva annak felépítésére és szolgáltatásaira épít, így Windows-on csak valamilyen virtualizációs technikával futtatható (bár régen léteztek kísérletek a Windows-ra való portolásra).

Az Asterisk esetén kiterjed közösségi támogatás érhető el fórumokon és dedikált VoIP oldalakon (lásd például http://www.voip-info.org/), de több szintű business támogatási lehetőség is van a jelenlegi fejlesztő cégtől, a Digiumtól. Továbbá a Digiumtól és partnereitől lehet olyan alközpontokat, IP telefonokat, ISDN és analóg vonali illesztőkártyákat venni, amelyek kompatibilitása garantált a szoftverrel.

A következő leírások az **Asterisk 1.8** verziójáról szólnak, ezt használjuk majd a laborban is.

Az Asterisk egy moduláris rendszer, a modulok külön-külön tölthetőek be és mind más-más funkcióért felelős. Egy ilyen modul például a SIP csatorna driver (chan_sip.so), amely nélkül nem lenne lehetőség SIP alapú VoIP kommunikációra. A betöltendő modulok listáját a `modules.conf` fájlban kell megadni, amely alapértelmezetten úgy van beállítva hogy minden olyan modult betölt (`autoload=yes`), amelyet mi kifejezetten nem tiltunk meg neki (`noload valami.so`).

A modulok típusa lehet:

- **Applications** : Alkalmazások. Például a Dial() alkalmazás amely összeköt bizonyos csatornákat (hívás felépítés) vagy a Playback() ami képes előre felvett hangok lejátszására.
- **Bridging modules** : Konferencia esetén használatos modulok.
- **Call detail recording (CDR) modules** : Hívás információk tárolására szolgáló modulok (CSV fájlba, adatbázisba, ...)
- **Channel event logging (CEL) modules** : Csatornák eseményeinek mentésére szolgáló modulok
- **Channel drivers** : Csatorna driverek. Egy-egy protokollt illetve csatorna típust támogatnak. Például: SIP, H323, GTalk, Skinny, ISDN, DAHDI (analóg), ...
- **Codec translators** : Különböző kódekek között képesek konvertálni. Például ha az ISDN G.711 kódeket használ és egy ilyen csatornát kell összekötni egy SIP csatornával, ami mondjuk G.729-et használ, akkor a kettő között átkódolás szükséges. Ez processzor intenzív feladat, ezért mi mindenütt ugyanolyan kódeket használunk.
- **Format interpreters** : Különböző fájl formátumok támogatása, amelyek hangot (vagy videót) tartalmaznak. Például hang felvétel vagy tartási zene bejátszás esetén vannak használatban. A laborban a legegyszerűbb wav formátumot használjuk, ebben van tárolva a tartási zene (Für Elise).
- **Dialplan functions** : Az alkalmazásokat egészíti ki a - később tárgyalt - Dialplan-hez köthető funkciókkal. Például lekérhető, illetve beállítható a CallerID (hívó fél száma, egyes esetekben a neve is), de akár egy bizonyos szám tárcsázására Linux shell parancs is futtatható.
- **PBX modules** : A központ beállító moduljai, szkriptelésre lehetőség (LUA)
- **Resource modules** : Külső erőforrások elérésére szolgáló modulok. Például LDAP adatbázishoz való hozzáférés, amely tárolja a felhasználók adatait.
- **Addons modules** : A közösség által írt kiegészítő modulok, amelyek alap esetben nem képezik az Asterisk részét. Ezek használati vagy licencelési okokból különülnek el. Ilyen például az MP3 fájlok lejátszására szolgáló modul.
- **Test modules** : Az Asterisk fejlesztők használják tesztelésre.

Az Asterisk különböző moduljainak bővebb megismertetésére jelen útmutatóban nincsen hely. További információ elérhető az Asterisk: The Definitive Guide, 3rd Edition könyvben a 10-24.oldalon.

Mivel a mérés a SIP VoIP jelzési protokollra épül, ezért a SIP csatorna driver legalapvetőbb beállításait tekintjük át, amely a `sip.conf` fájlban található.

Legegyszerűbb egy példa alapú szemléltetés:

```
[100]
type=friend
callerid="Majdan Andras" <100>
host=dynamic
secret=nehezjelszo
context=pbx
canreinvite=no
```

A mezők jelentése:

- **felhasználói azonosító** : a kapcsos zárójelek között van. Az Asterisk könyv nem ajánlja, hogy ez egyben a hívószám is legyen (hiszen ez magát a fizikai telefont azonosítja és később csúnya lehet egy hívószám módosítás), de mi az egyszerűség kedvéért így használjuk. A VoIP kliensben ezt kell beállítani User ID-ként. Valós rendszerben érdemes a telefon fizikai címét (MAC) használni azonosítóként.
- **type**: az Asterisk hogyan értelmezze az innen bejövő SIP kéréseket
  - **peer** : A bejövő kéréseket az IP cím és portszám alapján azonosítja
  - **user** : a SIP üzenet From: fejléce alapján azonosítja, ennek kell egyeznie a felhasználói azonosítóval
  - **friend** : Akár peer, akár user módon azonosíthatja magát
- **callerid** : azt határozza meg, hogy mit írjon ki a hívott fél telefonja, ha innen érkezik a hívás.
- **host** : azt határozza meg, hogy milyen IP címen érthető el az állomás. A dynamic opció azt jelenti, hogy majd ez az IP cím kiderül, amikor regisztrál a kliens az Asterisk szerveren.
- **secret** : a jelszó, amit be kell állítani a VoIP kliensnek.
- **context** : milyen dialplan kontextusba (lásd később) fusson az innen indított hívás
- **canreinvite** : létrehozhat-e direkt RTP hang kapcsolatot a másik féllel (no esetén nem)

A canreinvite beállítás bővebb magyarázatra szorul. A SIP esetén a hívás felépítés úgy lett kitalálva, hogy a hívó fél INVITE üzenetet küld egy SIP proxy szervernek, aki (akár több más proxy szerveren keresztül) eljuttatja ezt az üzenetet a hívott félnek. Ebben az üzenetben van egy SDP rész, amiben a hívő közli hogy, hova és milyen kódolással kell küldenie a hívott félnek a hang csomagokat. Tehát szerepel benne a hívó fél IP címe, RTP portja és a támogatott hang kódekek. Mint látjuk egy megszegi az OSI modell szerinti rétegezést, hiszen egy másik szint (Layer 3) szerinti címzés szerepel benne. A hívott fél "200 OK" üzenetet küld vissza és kiválasztja a számára szimpatikus kódeket, valamint ugyanúgy közli a saját IP címét, RTP portját ahová a hang csomagokat kell küldeni. Ez bizonyos esetekben nem is az ő - SIP jelzésrendszerre szolgáló - IP címe, hanem egy nagyobb IP telefonközpont esetén, például az egyik DSP kártyájának IP címe (és RTP portja). A hangkapcsolat így direkten létrejön a két fél között és csak a jelzésrendszer (SIP) üzenetei haladnak a SIP proxy szerveren keresztül. Az Asterisk ettől a SIP proxy működéstől kicsit eltér. Az INVITE üzenetet nem direkten továbbítja, hanem ő maga hoz létre egy másik viszonyt (szintén INVITE-al) a hívott féllel, így a csomagokban eltérő hívás azonosítókat láthatunk. Az Asterisk így képes eldönteni, hogy engedje-e a direkt RTP kapcsolatot a két fél között. Ha ezt nem akarja akkor a felekkel történő SIP kommunikáció során a saját IP címét és RTP portjait teszi bele az SDP blokkba és ő maga köti össze a hangcsatornát.

Bizonyos telefonközponti szolgáltatások nehezen vagy egyáltalán nem nyújthatóak e nélkül, úgymint:

- **hívások hangjának rögzítése**
- **központi tartási zene**
- **konferencia**

- **kódek konverzió**
- **in-band (hangcsatornán) küldött DTMF felismerése**
- **beszédfelismerés** (bomba, Osama bin Laden, ..)

A SIP csatornáknak természetesen sokkal bővebb beállításaik vannak, például képes az Asterisk maga is regisztrálni egy SIP proxynál és bejövő hívásokat fogadni tőle, valamint továbbítani felé. Ezért kerülöm a mellékállomás, illetve vonal fogalmakat, mert Asterisk esetén ezek között nincs különbség.

Amivel meg kell még ismerkedni egy működő IP telefonközpont létrehozásához, az az `extensions.conf` fájl. Ebben tárolódik a Dialplan, amely a hívások végződtetéséhez szükséges logikát tartalmazza.

Szemléltetés egy példán:

```
[pbx]
exten => 110,1,Dial(SIP/110)
exten => 200,1,Dial(SIP/101&SIP/102)
exten => 300,1,Dial(SIP/300,10)
exten => 300,2,Dial(SIP/301,10)
exten => 555,1,Answer()
exten => 555,2,MusicOnHold()
```

A pbx alatt található szabályok csak a pbx kontextusban lévő csatornákban történő tárcsázásra vonatkoznak. Ha tárcsázza valaki a 110-et, akkor a 110 felhasználói azonosítójú csatorna csörög. Az 1 a szabály prioritását jelenti. Ha tárcsázza valaki a 200-at akkor a 101 és a 102 felhasználói azonosítójú csatorna csörög párhuzamosan, bármelyik felveheti. A 300-as szám tárcsázására a 300 felhasználói azonosítójú csatorna csörög 10 másodpercig, ha pedig addig nem veszik fel a telefon, akkor tovább megy a 2. prioritású szabályra így a 301-es csörög 10 másodpercig. A Dial tehát gondoskodik a csatornák összeköttetéséről.

Az 555-ös szám tárcsázására örök tartási zenét kap, ha be van töltve a musiconhold modul. Vegyük észre, hogy a MusicOnHold és a Dial is csak egy modul a sok közül és akár mi is írhatunk saját modult, amelyet ide beilleszthetünk.

A Dial alkalmazás formátuma: Dial(Technology/Resource&[Technology2/Resource2[&...]],[timeout,[options,[URL]]])

Létezik több rövidítése is a szabályoknak, de mi nem használjuk őket.

Ezzel a beállítással már működne egy IP telefonközpont, de ha ezt minden egyes hívószámra meg kellene csinálni, akkor a beleőszülne az operátor. Ezért használhatunk különböző mintákat a hívószám meghatározásakor:

```
[pbx]
exten => _1XX,1,Dial(SIP/${EXTEN})
exten => _2XX,1,Dial(SIP/200)
exten => _906!,1,Dial(SIP/upc/+36${EXTEN:3})
```

Az első szabály azt mondja hogy minden 1-el kezdődő 3 hosszú hívószám tárcsázza az adott felhasználói azonosítójú csatornát. Tehát ha például a 105-öt hívom, akkor a 105-öt. A második szabály azt mondja, hogy

minden 2-vel kezdődő 3 jegyű szám csörögjön a 200-as felhasználói azonosítójú melléken. Az utolsó szabály azokra az akármilyen hosszú tárcsázott számokra vonatkozik, amely 906-al kezdődnek. Jelen esetben a 9-es a hagyományos telefonközpontok szokásos külső vonal elérése használt prefix, a 06 pedig Magyarországon a körzet előhívó. A hívás a upc azonosítójú csatornának lesz elküldve, úgy hogy a hívószám első 3 tagját levágja (906) és elé illeszti az ITU szerinti +36-os ország előhívót. A upc pedig majd gondoskodik a hívás végződtetéséről.

A mintákat tehát az alulvonás jel jelöli és a következő fontosabb speciális karakterek használhatóak:

- **X** : 0-9-ig szám
- **Z** : 1-9-ig szám
- **N** : 2-9-ig szám
- **[04-6]** : bármely szám vagy tartomány, amely a zárójelben szerepel. Jelen esetben: 0,4,5,6.
- **.** : legalább egy karakter
- **!** : bármennyi karakter

A mérés során a következő fejlett telefonos szolgáltatásokat fogjuk kipróbálni:

- **do not disturb** (DND): Ne zavarj üzemmód. A készülék nem fogadja a hívásokat. A telefonközpont beállításától is függ, hogy milyen módon utasítja el a hívást (akár üzenet is bemondható).
- **call forward**(CFWD): Hívásátirányítás. A készülékre beérkező hívások automatikusan más hívószámra továbbítódnak. Létezik feltételes és feltétel nélküli változata is. Feltételek lehetnek:
    - a készülék foglaltsága
    - a készülék nem elérhető (például nincs a hálózaton)
    - a készülék nem válaszol (bizonyos időn belül nem veszi fel senki)
    - egy testreszabható IP telefonközpont esetén saját feltételek is meghatározhatóak
- **redial**: Utoljára tárcsázott szám újrahívása.
- **hold**: Tartásba rakja a hívást. A tartásba rakott fél általában tartási zenét hall. Eközben a másik fél (például ügyfélszolgálat esetén) a háttérben nyugodtan megbeszélheti az adott kérdést a munkatársaival, hívást fogadhat vagy kezdeményezhet a készülékről. A tartásba rakott felet általában egy villogó gomb jelzi a készüléken és ennek a megnyomásával vissza lehet venni. Gyakran előfordul hogy joga van másnak is átvenni a tartásba rakott felet, amely történhet egy speciális telefonszám tárcsázásával vagy egy kezelő konzolon való gombnyomással.
- **transfer**: Hívásátadás. A beérkező hívás felvétele után az egyik fél dönthet úgy, hogy a másik felet egy másik mellékállomásra kell irányítani. Általában ezt a hívott fél teszi: például ügyfélszolgálat átkapcsol a megfelelő mellékre, miután eldöntötte hogy az ügyfél problémája mely csoporthoz tartozik. Általában a hívásátadás ideje alatt tartási zenét hallgat akit átadnak. Két típusát különböztetjük meg:
    - **supervised transfer** (más néven: announced, attended): Bejelentett hívásátadás. Mielőtt a hívást továbbítja a kezelő, beszél a féllel ahová a hívás továbbítva lesz. Általában amint lerakja a telefont a kezelő, automatikusan összekapcsolódnak a felek és a tartási zene után nincsen kicsengési hang. Előfordulhat azonban hogy egy gombot kell lenyomnia az átadónak, hogy összekapcsolódjanak a felek.
    - **blind transfer** (más néven: unannounced, unsupervised): Vak hívásátadás. A hívás átadásakor nem beszél az átadó fél azzal, akinek átadja a hívást. Általában a fél akit átadnak csak addig hall tartási zenét, amíg beüti a megfelelő hívószámot az átadó a készülékén, utána már kicsengési hangot hall egészen addig míg a hívott fél fel nem veszi.
- **conference**: Konferencia hívás. Több fél is beszélhet egymással egyszerre, az összekapcsolást általában a hívott fél végzi egy gomb lenyomásával vagy egy speciális szám tárcsázásával. A

konferencia résztvevőinek száma, résztvevők eltávolíthatósága, új résztvevő becsatlakoztatása, az új becsatlakozóval való különálló előzetes beszélgetés lehetősége az alkalmazott technológia és beállítás függvénye.
- **call pickup** (más néven: directed pickup): Fel lehet venni a hívást olyan készüléken is, ahová a hívás nem irányult. Ez történhet gombnyomással (például kezelői konzolon) vagy egy speciális hívószám tárcsázásával. Alapvetően két típusa van: egy adott mellékállomásra irányuló hívás átvétele vagy mellékállomások egy csoportjába érkező hívás átvétele. Ez azonban technológia és beállítás függő kategorizálás, hiszen csoporttal is lefedhető minden lehetőség (ha létrehozható 1 tagú csoport).

További információ az Asterisk-ről: http://www.asterisk.org/

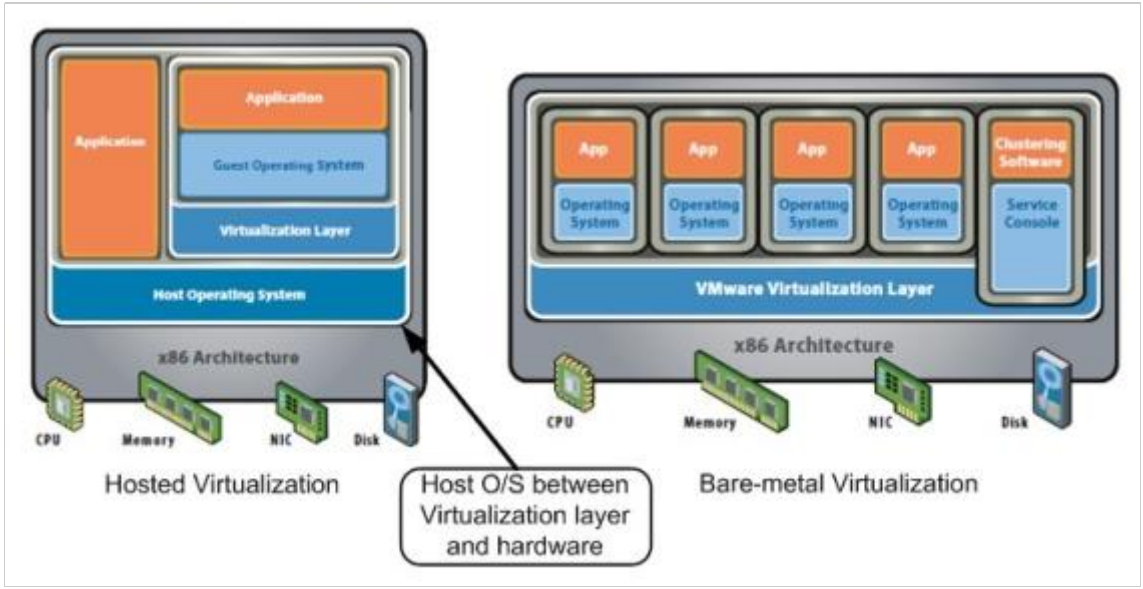
Számítógép hardver virtualizáció:

A korai számítógépek nagyon szűkös erőforrásokkal rendelkeztek, rengeteg alacsony szintű optimalizációt kellett végezni a programokon, hogy megfelelő legyen a futási sebesség és beleférjenek a memóriába. Ráadásul ezek nagyon drága gépek voltak, amelyeken a feladatok egymás után kötegelve futottak és az egyes időszeleteknek megvolt az áruk. A technológia fejlődésével először a nagy mainframe gépeken jelentek meg a különböző virtualizációs technikák, amelyek egyszerű erőforrás virtualizációtól terjedtek egészen egy komplett számítógép virtualizációig. Ennek először biztonsági (ne tudjanak egymásra hatni a különböző alkalmazások) illetve visszafele kompatibilitási (korábban megírt program fusson az új architektúrán is) okai voltak. Később olyan számítási kapacitású gépeket tudtak létrehozni, hogy már a gépek nem voltak teljesen kihasználva. Az gazdaságos üzemeltetés érdekében szükségessé vált, hogy egy szervert több ügyfélnek is ki lehessen adni, vagy ha értékesítésből élt a cég, akkor úgy hirdetni hogy több gépet is helyettesíthet a szerver. Innentől kezdve igazából csak erőforrásokat értékesítenek, amelyek azonban (bizonyos korlátokkal) úgy jelennek meg az ügyfélnél, mintha elkülönült fizikai számítógépek lennének. Ezeknek az erőforrásoknak az elosztása (például hány processzort használhat a virtuális gép -> számítási kapacitás) csak megállapodás kérdése.

Az ezredforduló utáni számítógépek számítási kapacitása, memóriájának mérete olyan szintet ért el, hogy egyszerű PC-n is lehetővé válhatott a virtuális gépek használata. Azonban ezek x86-os architektúrája megnehezítette a virtualizációt végző programok (hypervisor-ok) létrehozását. Sok esetben a futó programok előzetes elemzésére/fordítására volt szükség. Azonban a növekvő igényeket a virtualizációra a fő PC processzor gyártók (AMD, Intel) is láttak és néhány éven belül (2006) kibővítették az instrukciókészletüket olyan utasításokkal, amelyek hardveresen segítették a virtualizációt (AMD-V, VT-x). Innentől kezdve a PC-n a számítógép hardver virtualizáció mindenki számára elérhető vált.

A számítógépet, amelyen a hypervisor fut **host-nak**, vagyis **gazda gépnek** nevezzük, míg a virtualizált gépeket **guest-nek**, azaz **vendégeknek** nevezzük. Ma már létezik **nested virtualizáció** is, amely azt jelenti hogy a virtuális gépben is létrehozhatóak még további virtuális gépek, így lehet egy gép egyszerre gazda és vendég is. Továbbá két kategóriába soroljuk a virtualizációt attól függően, hogy milyen szinten helyezkedik el:

- **bare-metal (Type-1, native)**: a hypervisor direkten a hardveren fut, gyakorlatilag egy mini operációs rendszer
- **hosted (Type-2)** hypervisor: egy operációs rendszeren fut, ennek a szolgáltatásait használja mint bármilyen más program. Mi a laborban ezt használjuk, Windowsból indítjuk el a virtuális gépet.



4. ábra: Bare-metal és hosted virtualizáció

Miért használunk mi egyszerű asztali PC-n virtualizációt?

Az egyetemen (de cégeknél is) gyakran kell olyan programot tesztelni vagy használni, amely az alapvetően használt szoftver környezetben nem futna. Például a laborban Windows 7 operációs rendszer fut a számítógépeken, de az Asterisk IP telefonközpont amit a hallgatók használnak Linux alapú operációs rendszerre van tervezve. Ezért nem érdemes külön számítógépet venni és rá valamilyen Linux disztribúciót telepíteni, hanem elég egy virtuális gépet használni. A virtuális gépbe kell telepíteni a megfelelő operációs rendszert és rá a szoftvert (Asterisk), amit használni szeretnénk.

További előny származik virtualizáció esetén a **snapshot** (pillanatkép) lehetőségéből. Bizonyos pontokon kikapcsolt vagy akár futó állapotban pillanatkép készíthető a virtuális gépről. Erre az állapotra pedig bármikor vissza lehet térni. Például egy gyanús program telepítése vagy egy akár katasztrofális kihatással rendelkező parancs kiadása előtt érdemes egy pillanatképet készíteni. Számítógépes kártevők elemzésére is tökéletes egy ilyen sandbox (homokozó) létrehozása, ahol a kártevő kedvére garázdálkodhat.

Az általunk használt virtuális gépeket sok hallgató használja, azonban nem szeretnénk mindig lemásolni (**clone**) az egész virtuális gépet nekik. Lehetőség van **linked clone** létrehozására, amely csak a különbségeket menti a base image-hez (alapkép) képest. Ezt nagy méretű virtuális gépek esetén érdemes használni, mert így elég egyszer tárolni az alapképet, a hallgatóknál pedig csak ezen a képen az általuk végzett módosítások tárolódnak, az alapkép mindig változatlan marad.

A laborban a VMware cég - nem üzleti felhasználásra - ingyenes Player termékét használjuk, de léteznek teljesen szabad felhasználású hypervisorok is (VirtualBox, Xen, KVM).

OpenWrt

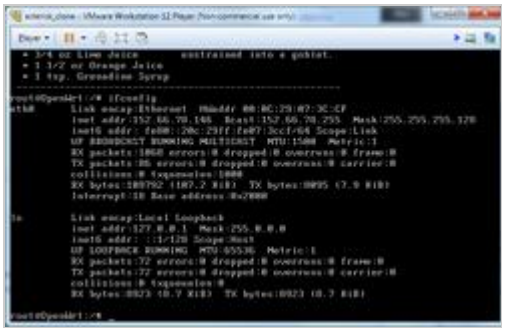
Az OpenWrt egy nyílt forráskódú, ingyenes Linux alapú firmware különböző beágyazott eszközökre (általában routerekre). Korábban csak meglévő routerek képességeit bővítették vele (lecserélve a gyári szofvert), azonban ma már vannak eszközök, amelyek gyárilag OpenWrt -n vagy valamely variánsán alapulnak. A szoftver kezdeti útválasztási képességei gyorsan kibővültek és ma már egy általános platformot kapunk, amelyen a legkülönbözőbb hálózati alkalmazások futtathatók. Mivel beágyazott környezetre lett tervezve, ezért nagyon kevés erőforrás szükséges a futtatásához, hiszen ezek a routerek általában 4-16 MB flash-t (háttértár) és 16-64 MB RAM-ot (memória) tartalmaznak csak. Mindez ideálissá teszi a rendszert virtuális gépben való alkalmazásra

is, mert a gazda gépen akár egy tucat is futtatható belőle. Természetesen nem ugyanaz az OpenWrt fut a különböző architektúrákon, hanem különböző fordítások érhetőek el. Alapvetően ezek használata között azonban nincsen nagy különbség, a telepítés menete és az alkalmazott fájlrendszer viszont eltérhet. A különböző programok - mint más Linux disztribúcióknál is - csomagkezelővel telepíthetőek, így az Asterisk IP telefonközpont is ilyen csomagokból áll. Nem kell letöltéssel, kitömörítéssel, az automatikus indítás beállításával foglalkozni, mindent elintéz a csomagkezelő.

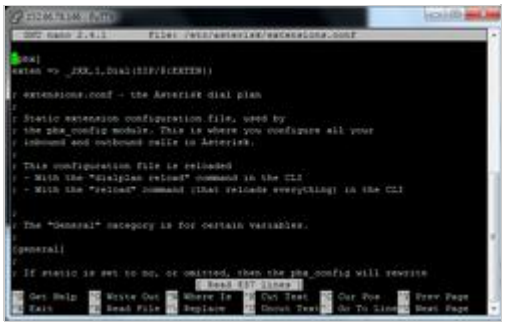
Az OpenWrt rendszer rendelkezik web felülettel (LuCI), de mivel mi nem routerként használjuk, így nem vesszük hasznát. A shell (parancssor) elérése általában hálózatról titkosított csatornán keresztül történik (SSH), de helyileg is elérhető, ha az eszköz rendelkezik valamilyen ezt támogató porttal. Beágyazott rendszereken ez lehet soros port, de lehet akár billentyűzet + képernyő csatlakozási lehetőség is. Más Linux disztribúciókhoz hasonlóan itt is több fajta shell elérhető, az alapértelmezett az ash.

Az általunk használt parancsok:

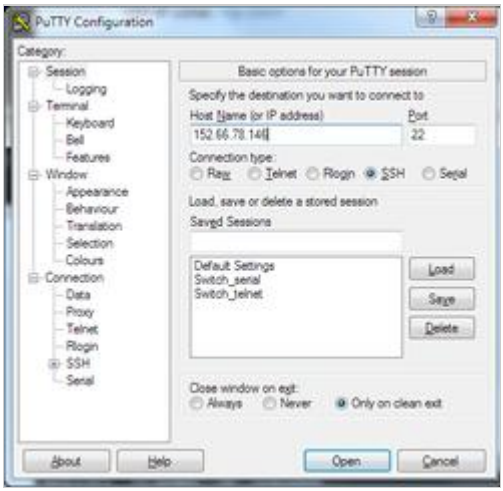
- `cd` Belép egy könyvtárba
- `cd ..` Kilép egy könyvtárból
- `ls` Listázza a fájlokat és könyvtárakat
- `nano` Szerkesztésre megnyitja a megadott fájlt
- `ifconfig` Kiírja a hálózati beállításokat
- `/etc/init.d/asterisk restart` Újraindítja az Asterisk IP telefonközpontot



6. ábra: OpenWrt parancssorában IP cím lekérdezése (virtuális gép)



5. ábra: Nano szövegszerkeszt?



7. ábra: Csatlakozás SSH kapcsolaton PuTTY segítségével

A nano szerkesztőben használandó billentyűkombinációk:

- `ctrl+o` Fájl mentés
- `ctrl+x` Kilépés

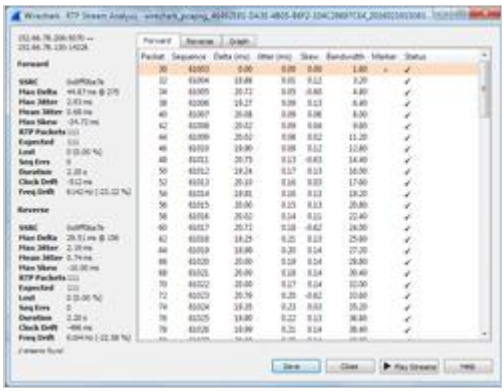
Az Asterisk IP telefonközpont nálunk egy TP-LINK TL-WR1043ND routeren fut, amelyen a gyári firmware-t OpenWrt -re cseréltük. Ez egy nagyon elterjedt otthoni router, 2016-ban bruttó 14-15e Ft körül kapható, 8 MB háttértárral és 64 MB memóriával rendelkezik, a processzor pedig 720 MHz-es (MIPS architektúrájú). Ezen az eszközön az OpenWrt tömörített fájlrendszert alkalmaz, hogy beleférjen a szűkös háttértárba. Ennek ellenére képes 24 db IP telefont (asztali + softphone) kiszolgálni a laborban.

A hallgatók gépén az OpenWrt -nek ehhez képest 50 MB háttértárat, 256 MB memóriát és egy jóval gyorsabb processzormagot (3.3 GHz) virtualizálunk.

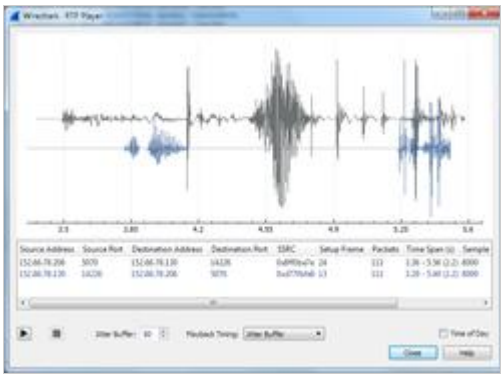
További információ az OpenWrt -ről: https://openwrt.org/

Wireshark

A Wireshark (régebbi nevén Ethereal) egy nyílt forráskódú, ingyenes hálózati csomagelemző. Széles körben alkalmazzák az iparban hálózati problémák megoldására, elemzésére, de protokollok fejlesztésénél, tesztelésénél is hasznos eszköz. Minden elterjedt asztali és szerver operációs rendszerre elérhető. A forgalom figyelését a pcap (packet capture) függvénykönyvtáron keresztül végzi, de lehetőség van előre lementett forgalom elemzésére is. Az utóbbi funkció ott hasznos, ahol nem futtatható Wireshark, de a csomagokat valamilyen programmal (például tcpdump) mégis le lehet menteni. Ilyen például az OpenWrt rendszer, de szinte minden IP telefonközpont, illetve IP média átjáró (például SIPGSM) rendelkezik hasonló funkcióval.

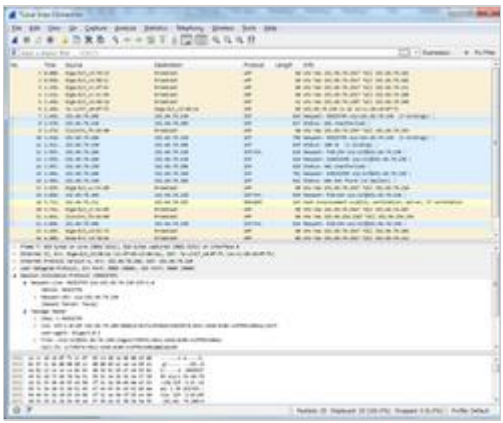


9. ábra: RTP folyam elemzése



10. ábra: RTP lejátszás

A Wireshark nem csak megmutatja a csomagok tartalmát, hanem ha ismeri a protokollt, akkor a csomagot az adott protokoll szabályai szerint részekre bontja és érthető formában strukturáltan megjeleníti. Lehetőség van továbbá bizonyos protokollok esetén csomag folyamokat vizsgálni, a Wireshark képes meghatározni ilyenkor az összefüggő csomagokat és azok összességén elemzéseket végezni. RTP esetén ez odáig megy, hogy a hang akár vissza is játszható a Wireshark felületén.



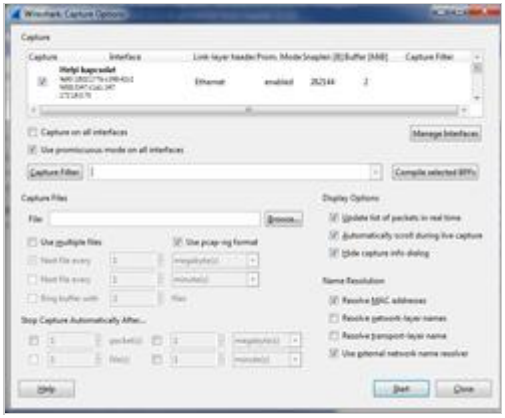
8. ábra: SIP regisztrációs üzenet elemzése

Mielőtt a csomagok élő vizsgálatát megkezdené a Wireshark, ki kell jelölni hogy mely hálózati interfészek forgalmát vizsgálja, valamint hogy csak az azoknak címzett csomagokra van-e szükség. Alap esetben a hálózati kártyák eldobják azokat a csomagokat, amelyek nem broadcast csomagok vagy nem az ő fizikai címük (MAC) szerepel címzettként. A **promiscuous mode** (válogatás nélküli mód) arra szolgál, hogy minden csomagot, amit csak lát a hálózati kártya a "dróton" megkapjunk. Nem minden eszköz képes ilyen módba kerülni, például léteznek olyan vezeték nélküli eszközök, amelyek meghajtóprogramja meggátolja ezt (lásd http://www.aircrack-ng.org/doku.php?id=compatibility_drivers). Felvetődhet a kérdés, hogy mégis miért kapnánk olyan csomagokat, amelyek nem nekünk szólnak. Ez több esetben is előfordul:

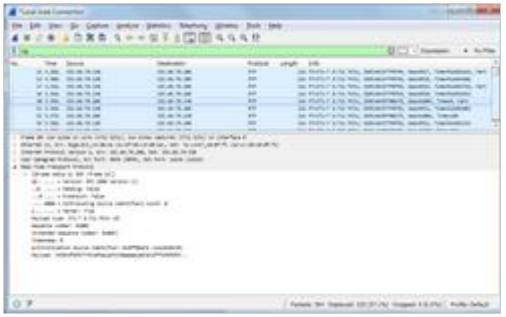
- **Közös fizikai média**: ugyanazon a kábelen vagy vezeték nélküli közegben kommunikálunk. Például régen a számítógépes hálózatokban egyszerű hub-okat használtak, amelyek a fizikai közeg egyesítésén kívül maximum jelerősítést végeztek, így portjaikon nem szelektálták az Ethernet kereteket. Ma is van, aki még ilyen hub-ot használ hálózati elemzésre úgy, hogy beiktatja köztes pontként a vizsgálandó felek

köz, majd harmadikként rádugja a saját gépét, amin fut a Wireshark.

- **Port mirroring**: modernebb hálózati switch-eken beállítható, hogy bizonyos port(ok) forgalmát egy az egyben tükrözze egy megadott portra. Ilyenkor arra portra csatlakoztatja az elemző a számítógépét.



12. ábra: Wireshark capture filter és promiscuous mode beállítási helye



13. ábra: Wireshark display filter beállítás RTP csomagok sz?résére

11. ábra: Wireshark interfész beállítás

A hálózatok elérhető sebességének növekedésével egyre kevesebb figyelem fordítódik a hálózati forgalom minimalizálására, így sokszor felesleges broadcast illetve háttér alkalmazások szűnni nem akaró üzenetei árasztják el a hálózatokat. A hálózati csomagok elemzésekor ezeket szűrni érdemes, hogy csak a minket érdeklő csomagokat lássuk. A Wireshark-ban erre két lehetőség van:

- **Capture filter (szűrés csomag mentés szinten)**: a kiszűrt csomagokat a Wireshark nem kapja meg egyáltalán, erről a pcap függvénykönyvtár gondoskodik
- **Display filter (szűrés megjelenítés szinten)**: a csomagot a Wireshark megkapja, de a szűrő szabályaitól függ hogy megjeleníti-e. Ezen a szűrőn bármikor módosíthatunk, attól függően hogy milyen csomagokat szeretnénk éppen látni. Meg lehet adni konkrét szabályokat is, de a Wireshark képes egy csomag alapján automatikusan ilyen szűrési szabályok létrehozására.

A két szűrési lehetőség teljesen más szintaktikával rendelkezik. Mi a mérésen a Wireshark csomagfolyam elemzési képességét (ahol ő automatikusan tud létrehozni megjelenítési szűrőt) és kézzel megadott megjelenítési szinten való szűrést használunk. A megjelenítési szabályokból is csak kettőt használunk: be kell írni a protokoll nevét, tehát "rtp" vagy "sip", attól függően hogy mely csomagokra vagyunk kíváncsiak.

További információ a szűrési szabályokkal kapcsolatban:

- https://wiki.wireshark.org/CaptureFilters
- https://wiki.wireshark.org/DisplayFilters

További információ a Wireshark-ról: https://www.wireshark.org/