Debreceni Egyetem
Informatikai Kar

SOHO hálózatok Internet használatának szokás-elemzése

Témavezető:
Gál Zoltán
Igazgató

Készítette:
Czina Ferenc
Programozó matematikus szak

Debrecen
2008.

Tartalom

1. Szakdolgozatom célja 2
2. Az Internet megszületése 2
3. A számítógépes hálózat protokoll adatelemei és működési mechanizmusai 4
   - *3.1. Ethernet keret:* *4*
   - *3.2. Ip csomag felépítése* *5*
   - *3.3. Az UDP szegmens felépítése* *7*
   - *3.4. TCP szegmens felépítése* *8*
   - *3.5. Address Resolution Protocol (ARP) csomag felépítése:* *11*
   - *3.6. Dinamic Host Control Protocol (DHCP)* *12*
   - *3.7. Domain Name System (DNS)* *14*
   - *3.8. Simple Mail Transfer Protocol (SMTP)* *17*
   - *3.9. Post Office Protocol (POP3)* *18*
4. A kiscéges/otthoni (SOHO) hálózatok 19
5. A megfigyelésben résztvevő hálózat bemutatása 21
   - *5.1. Topológia* *21*
   - *5.2. A hálózaton igénybe vehető szolgáltatások* *22*
   - *5.3. A SOHO hálózat felépítése és kapcsolata az internettel* *25*
6. A tanulmányozott SOHO hálózat Internet szolgáltatójának (ISP) hálózata 27
   - *6.1. Internet és VoIP szolgáltatás a koax kábelen* *28*
   - *6.2. A kábelmodem és a fejállomás (CMTS) kommunikációja* *30*
   - *6.3. A DOCSIS protokoll architektúrája* *32*
   - *6.4. A DOCSIS protokoll közeg-hozzáférési mechanizmusa* *32*
7. A hálózati forgalom megfigyelése 33
   - *7.1. Ifconfig parancs* *34*
   - *7.2. ARP parancs.* *34*
   - *7.3. A route print parancs* *34*
   - *7.4. A ping és tracert parancsok* *34*
   - *7.5. A sniffer programok* *34*
8. Az elemző program bemutatása 36
9. A mintavételezés 43
10. Az elemzés eredményei 44
11. Az eredmények értelmezése 53
12. Összefoglalás, következtetések 55

Irodalomjegyzék 57
Köszönetnyilvánítás 58
„A” melléklet: Ábrák listája 59
„B” melléklet: Rövidítések listája 61
„C” melléklet: Elemző program forráskódja 62

1. Szakdolgozatom célja

Korunkban az Internet már-már napjaink megszokott technológiája lett. Interneten tartjuk kapcsolatot szeretteinkkel, barátainkkal, azon keresztül intézzük a banki ügyeinket. Interneten keresztül foglaljuk le a repülőjegyünket, azon keresztül keresünk magunknak új autót vagy mobiltelefont, informálódunk a világ híreiről, osztjuk meg álláspontunkat más emberekkel. Napjainkban rengeteg ember használja az Internetet, de az emberek nagy része egyáltalán nem tudja, mi történik a háttérben, mikor rákattint egy linkre, elindítja a levelező programját vagy megnéz egy online Tv adást. Az egyes – az internetet használó - alkalmazások, operációs rendszerek és kommunikációs protokollok jelentős overhead-et képeznek, gyakorlatilag teljesen elfedik a háttérben futó bonyolult folyatokat.

A hálózati kommunikációs protokoll végzi ebben a tevékenységben a legnagyobb szerepet. Feladatai közzé tartozik a hibaellenőrzés – és adott esetben hibajavítás -, az adategységek kisebb részadategységre való felbontása (azaz fragmentálása), a fogadó oldalon újbóli összeállítása. A hálózatok többsége úgy épül fel, hogy a részadategységek akár egymástól független úton is haladhatnak a célállomás felé, így ezen részadategységeket megfelelő sorrendbe kell állítani a fogadó oldalon melyet szintén a protokoll feladat. Fontos feladata a protokollnak a forgalomszabályozás is, azaz egy gyors adó ne árasszon el csomagokkal egy lassú vevőt. Adott esetben a protokoll végezhet biztonsággal kapcsolatos feladatokat, azaz titkosítás és hitelesítés. Ezen feladatok többsége a felhasználó számára a háttérben, teljesen rejtetten történnek.

Szakdolgozatom célja, hogy monitorozzunk egy otthoni, több gépből álló hálózat internet használatát. Megfigyeljük, hogy mire használjuk az Internetet, milyen belső folyamatok mennek végbe a mélyben, némi képet kapjunk arról, milyen bonyolult és összetett tevékenységeket követel meg a gépektől az, hogy mi le tudjuk hívni a leveleinket, megnézhessük kedvenc weboldalunkat vagy csak csevegjünk egyet a barátunkkal.

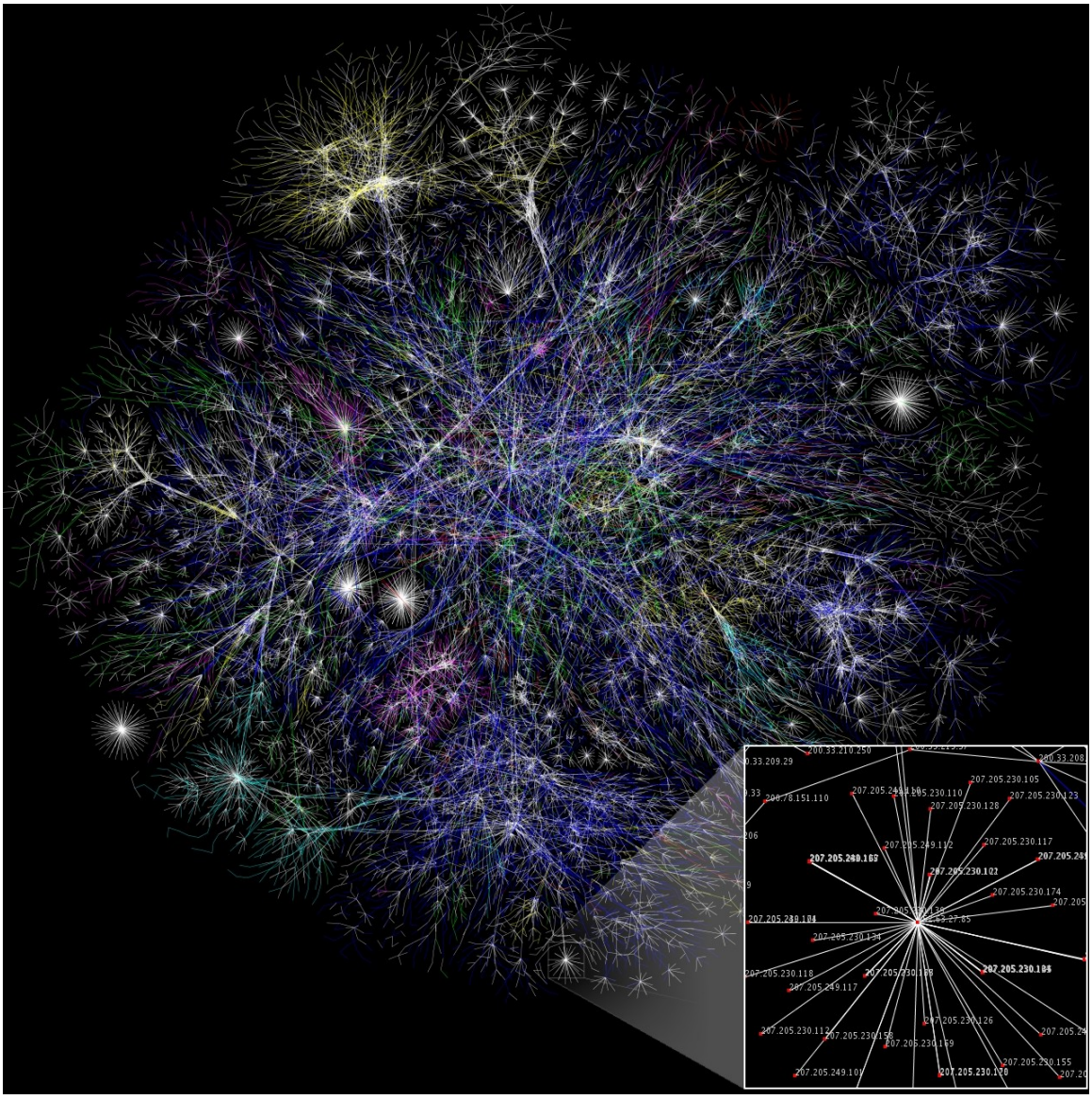
Hosszabb időn keresztül minden egyes, az otthoni hálózatot és az Internetet összekötő gépen átmenő adatcsomagot feljegyzünk, majd egy - általam készített és a szakdolgozatban bemutatott – program bizonyos kimutatásokat fog készíteni ezekből az adatokból.

Megmutatja, hogy milyen hosztokkal és mennyit forgalmaztunk, mely időszakokban milyen mennyiségekben volt hálózati forgalom. Egyáltalán…mire használjuk az Internetet?

2. Az Internet megszületése

Ugyanúgy, mint sok más technológia fejlesztést, az Internetet is a katonai iparnak köszönhetjük. Amerikában a hidegháború idején egy olyan számítógép hálózatot kívántak létrehozni (és ez által összekötni az egyes védelmi csomópontokat), mely egy esetleges atomcsapás esetén is működőképes marad. Így 1969-ben megkezdte működését az ARPANET. Ezután néhány egyetem belátta, milyen jó lenne, ha

összeköttetést tudnának létrehozni egymással ezen a hálózaton keresztül. Majd még néhány egyetem, majd néhány telefonszolgáltató felismerte az ebben rejlő lehetőséget és lehetséges lett az otthoni, telefonvonalon keresztüli internet előfizetés. Lassacskán megszületett az e-mail, az elektronikus hírcsoportok, a WWW mely hatékony információ megosztási eszköznek bizonyult.



1. ábra: Az Internet térképe (2008.04.15.)

A cégek egyre inkább felismerték, hogy az Interneten saját WWW oldal létrehozásával egy hatékony reklámfelületet nyerhetnek. A kereső technológiák fejlődésével az internet fokozatosan a kimeríthetetlen információ forrása lett. Ma már természetes bárkinek, hogy bármilyen információra is legyen szükségünk, azt a Google, Yahoo vagy MSN Search keresőmotorjai segítségével igyekszünk (általában sikeresen) megtalálni.

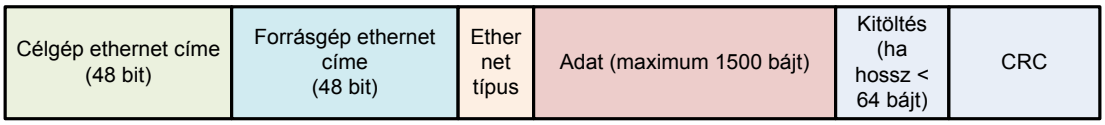
Napjainkban az internetre kötött számítógépek száma havonta 10-15%-al nő. Ez egy óriási mennyiségű, egymással kommunikáló számítógép rendszer (1. ábra). Éppen ezért – mint később látni fogjuk – egy igen kifinomult módszerre van szükség, hogy ezek

az egymástól – sokszor több ezer kilométer – távol lévő gépek hatékony információ cserére legyenek képesek.

3. A számítógépes hálózat protokoll adatelemei és működési mechanizmusai

3.1. Ethernet keret:

Az IEEE által 1983-ban létrehozott 802.3 szabványnak (a ma leggyakrabban használt Ethernet szabványnak) három főbb állomása volt. Az első állomás az úgynevezett klasszikus Ethernet szabvány volt. Ezek a 10Base5, 10Base2 (koaxiális kábelekre), 10Base-T (sodrott érpárra) és 10Base-F (optikai kábelre) megnevezéssel rendelkeztek. Ugyanazon Ethernet keretet használták, de különböző fizikai közegben. Az IEEE 1992-ben összehívta a bizottságot, mivel egy gyorsabb szabványra volt szükség, mely minden szempontból kompatibilis maradt az előzőleg létrehozott Ethernet szabvánnyal. Így jött létre a 802.3u azaz gyors Ethernet: 100Base-T4, 100Base-TX (sodrott érpár) és 100Base-FX (fényvezető szál) megnevezéseket kapta. 1995-ben egy még gyorsabb szabvány létrehozása volt a cél: ez szintén kompatibilis maradt visszafelé az előző szabványokkal. A 802.1z nevet kapta és az 1000Base-SX, 1000Base-LX (fényvezető szál) és 1000Base-CX, 1000Base-T (sodrott érpár) megnevezést kapta.



2. ábra: Ethernet keret felépítése

Az Ethernet keret felépítése a 2. ábrán látható. Ezen nem szerepel, de minden ethernet csomag rendelkezik egy 8 bájtos előtaggal mely az `10101010` mintát tartalmazza. Ez a 'bevezető' rész az adó és vevő szinkronizálásához szükséges.

A célcím egy 6 bájtos cím, melynek legfelső helyi értékű bitje közönséges címeknél 0, csoportcímeknél viszont 1 értékű. A legmagasabb helyi értékű bit szomszédjának szintén különleges jelentése van: amennyiben az 0, a cím 'globális cím' melyet az IEEE jelöli ki azért, hogy a világon ne fordulhasson elő két azonos cím. Ha az érték 1-es, a cím egy helyi cím melyet a helyi hálózat adminisztrátora jelöl ki. A cím három kategóriába tartozhat: lehet `unicast` azaz a címzett egyetlen számítógép. Lehet `multicast` azaz a címzett egy, a hálózaton lévő számítógépek csoportja. Végül lehet `broadcast` azaz az üzenetszórási tartományon belül található összes számítógép (ez a cím a speciális `FF:FF:FF:FF:FF:FF` azaz minden bitje 1).
A típus adja meg, hogy a keretben milyen csomag található (3. ábra). A leggyakoribb természetesen a `0x0800`.

Ezután jön a maximum 1500 bájt hosszúságú adatmező. Az ethernet keretnél nem csak maximum hossz van rögzítve, hanem a minimum is. Egyetlen keret sem lehet rövidebb 64 bájtnál. Amennyiben a hasznos adat mégis rövidebb, az adatrész után

következik a kitöltés, hogy meglegyen a minimum hossz. A keret végén következik a keret CRC ellenőrző kódja.

A legfontosabb ethernet típusok a következőek:

Ethernet típus hexa formában	Ethernet típusa
0800	IPv4 csomag
0806	ARP csomag
8035	RARP csomag
809b	AppleTalk (Ethertalk)
80f3	AppleTalk ARP
8100	IEEE 802.1Q-tagged frame
8137	Novell IPX
8138	Novell
86dd	IPv6 csomag
8847	MPLS unicast
8848	MPLS multicast
8863	PPPoE Discovery
8864	PPPoE Session
888e	EAP over LAN (IEEE 802.1X)
889a	HyperSCSI (SCSI over Ethernet)
88a2	ATA over Ethernet
88a4	EthernetCAT Protocol
88e5	MAC security (IEEE 802.1AE)

3. ábra: Ethernet típus mező lehetséges értékei

A 64 bájt adatrészben van néhány fix érték: biztosan tartalmaz 6 bájt célcímet, 6 bájt forráscímet. 2 bájton ábrázolva van a csomag típusa és végül 4 bájtot foglal el a CRC kód. Tehát a minimum hossz eléréséhez minimum 46 bájtnyi adatnak kell lennie minden ethernet keretben. Ennél kisebb `payload` esetén a Kitöltés mezővel egészítjük ki a szükséges hosszon.

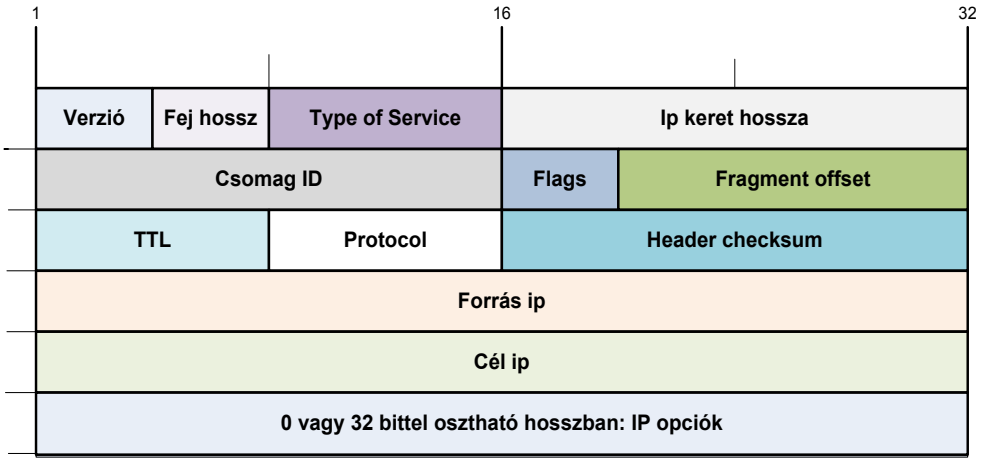
3.2. Ip csomag felépítése

Az IP csomag felépítésében (4. ábra) az első mező a verzió. Jelenleg két lehetséges értéke van: `0x4` IPv4 esetén és `0x6` IPv6 esetén. A Fej hossz mező megmutatja, hogy hányszor 32 bit az IP csomag fejrészének hossza. Ennek értéke minimum 5 (0x0101, ennél rövidebb nem lehet az ip csomag fejrésze) maximum 15 (0x1111, ennél többet nem lehet 4 biten ábrázolni). Ebből következően az IP csomag opció mezője maximum 10×32 bit azaz 40 bájt hosszú lehet.

Az IP csomag 'szolgálat típusa' (Type of Service, TOS) mezője lehetőséget ad különböző szolgáltatási osztályok beállítására. Az 1981 szeptemberében közzétett RFC791 szerint a TOS mező felépítése a következő:

a. Az első 3 bit a prioritásnak felel meg a 000b-től (normál csomag) az 111b-ig (hálózati vezérlő csomag)

b. A következő 3 bitben a hoszt beállíthatja, hogy milyen továbbítási szempontok a fontosak neki: az eső bit a késleltetés majd az átbocsátás végül a megbízhatóság elzésére szolgál.



4. ábra: IP csomag fejrészének felépítése

A következő mezőben a teljes IP csomag bájtban mért hossza van jelölve (IP keret + payload). A mező 16 bitjéből következően egy IP csomag maximális hossza 65535 bájt. Csomag ID ahhoz szükséges, hogy a több részre osztott datagram esetén jelezzük, melyik datagramhoz tartozik az újonnan érkezett darab. Egy datagram minden darabja ugyanazt a csomag ID értéket tartalmazza.

Ezután három jelzőbit következik (5. ábra). Ebből az első bitet jelenleg nem használják. A következő bit (DF) annak jelzésére szolgál, hogy a csomag nem fregmentálható.

Bit	Jelentés
1	Nem használt
2	DF (Do not fragment): Az adott keret nem bontható fel
3	MF (More fragment): Az adott keretet fel lett bontva és további keretek követik

5. ábra: Jelzőbitek (Flags) lehetséges értékei

A következő bit (MF) jelentése: Az IP csomag fregmentálva van, további fregmentek érkezése várható. Ez az IP csomag minden darabjában be kell állítani, kivéve az utolsót. Értelemszerűen az IP csomag utolsó darabja után nincs több darab.

Darabeltolás (Fragment Offset): Megmondja, hogy az aktuális darab hova tartozik a datagramban. Egy datagram minden darabja 8 bájt többszöröse (kivéve az utolsó darabot). Például: Egyik hoszt a másiknak 3000 bájtnyi adatot kíván küldeni. Az adat 3 csomagban lesz elküldve a következő módon:

a. Első csomagban lévő adat: 1480 bájt (ip csomag méret: 1500 bájt – ip fejléc méret: 20 bájt), darabeltolás: 0.

b. Második csomagban lévő adat: 1480 bájt, darabeltolás értéke: 1480/8=185 (0xb9).
c. Harmadik csomagban lévő adat: 40 bájt, darabeltolás értéke: 2960/8=370 (0x0172).

Élettartam (Time to Live, TTL): Az itt található számértéket minden ugrásnál az éppen aktuális router csökkent. Ha a csomagot megkapja egy router, az csökkenti a TTL értékét. Amennyiben az 0-nál nem nagyobb, a csomagot eldobja. Ezzel a mechanizmussal a csomagnak egy maximális élettartamot lehet adni. A 8 biten ábrázolható maximális érték 255, tehát egy csomag maximum ennyi routeren mehet keresztül mielőtt eldobásra kerül. Ez napjainkban elég is, általánosan elmondható hogy a világ legtávolabbi gépe is elérhető 18-30 ugrással.

A következő mező az IP csomagban lévő protocol jelzésére szolgál. A leggyakoribb protokolok a 6. ábrán látható.

Protokoll hexa értéke	Protokoll neve
0x00	IPv6 Hop-by-Hop option
0x01	ICMP
0x02	IGMP
0x04	IP over IP
0x06	TCP
0x11	UDP
0x29	IPv6

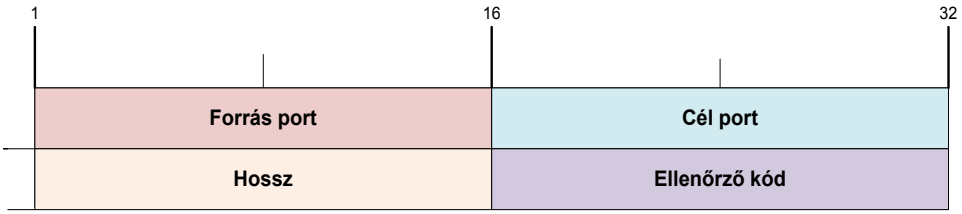
6. ábra: IP csomag protokoll mezőjének leggyakoribb értékei soho környezetben

A fejrész ellenőrző összeg (Header checksum) az IP csomag fejrészének hibaellenőrző kódja. Fontos, hogy a fejrész néhány értéke (például a TTL és esetleg a flags és Fragment offset értékek) routerenként változnak így ezt minden esetben újra kell számolni.
Ezután van feltüntetve a feladó és a cél gép IP címe, majd ezután következik (ha van egyáltalán) az IP csomag opciói. Ennek hossza (ha vannak) 32 bit vagy annak többszöröse. Itt lehetőség van például megjelölni laza vagy szigorú útvonalat, naplózni az adatcsomag útját, jelezni hogy egy titkos csomagról van szó (ezáltal jelezhető egy katonai szempontból kényes információ esetén hogy az adott csomagot ne irányítsák a routerek ellenséges országokon keresztül) vagy naplózható a közbenső állomások feldolgozási időbélyege. Végül a Padding opció kiegészíti 0-al 32 bitre az opció mezőt.

3.3. Az UDP szegmens felépítése

Az Internet két fő protokollja közül az UDP (7. ábra) összeköttetés nélküli. Olyan alkalmazások használnak UDP-t, kommunikációra, amelyek között nem szükséges összeköttetést kiépíteni. Ennél a kommunikációnál nincs garantálva, hogy az elküldött csomag megérkezik a címzetthez. Tehát az esetleges hiányzó csomagok vagy nem okoznak problémát az alkalmazásoknál (például VoIP) vagy a szükséges

hibakezelést/újraküldést a felhasználói folyamatok oldják meg (TFTP). A forrásport az adott hoszton futó küldő alkalmazást azonosítja. A cél port az adott célgépen futó célalkamazást (azaz a célhoszt szállítási entitása innen tudja melyik alkalmazásnak kell továbbítani).



7. ábra: UDP szegmens fejrészének felépítése

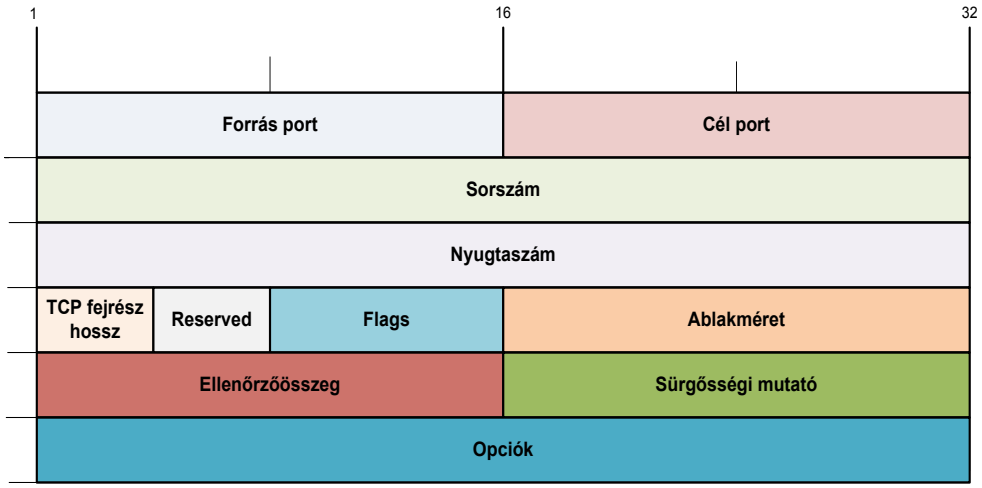
A Hossz a szegmens fej és adatrész hosszának összege. Az ellenőrző kódot nem minden esetben számolják ki (VoIP), ebben az esetben csupa 0-t tartalmaz. Amennyiben számolják, fej és adat együttesen kerül számításra.

3.4. TCP szegmens felépítése

A TCP az UDP-vel ellentétben direkt összeköttetés alapú, megbízható bájtfolyamot biztosítson két végpont között (8. ábra). A maximális mérete (fejléccel) 64 KB de többnyire 1460 bájtos darabokat használ hogy egyetlen ethernet keretbe beleférjen a hozzá tartozó ip fejléccel együtt. A TCP kezeli a szegmensek sorrendbe állítását, hiányzó szegmensek pótlását. Első 32 bitjén a küldő folyamat azonosítója (forrás port) és a cél állomás azonosítója (cél port).
Ezután következik egy 32 bites sorszám (sequence number). Ez biztosítja hogy minden küldött TCP szegmens azonosítható (és sorrendbe rakható) legyen. Fontos tudni, hogy valójában nem a szegmenseket, hanem a bájtokat sorszámozza a protokoll, azaz:

> Lássunk két egymásután küldött TCP szegmenst (a szegmens mérete 1460 bájt). Amennyiben az első szegmens sorszám mezője 0x99a90160 (2577989984 decimálisan) akkor a rögtön utána jövő szegmens sorszáma nem 0x99a90161 (2577989985 decimálisan) hanem 0x99a90714 (2577991444 decimálisan) lesz. És valóban 2577991444 - 2577989984 = 1460.

A nyugtaszám segítségével jelez vissza a célgép a küldő gépnek, hogy néhány a küldött szegmensek közül megérkezett. Fontos, hogy ez a nyugtaszám nem az utoljára fogadott bájt sorszámát, hanem a következőre várt bájt sorszámát ábrázolja. Azaz:
A 0x99a90160-as sorszámmal A géptől B géphez megérkezik egy 1460 bájtos szegmens. B gép ezt szeretné visszaigazolni. Tehát a szegmens mivel 0x99a90160-as sorszámmal rendelkezik, így benne a bájtok a 0x99a90160-tól a 0x99a90713 sorszámú bájtok. De B gép nem az utoljára megkapott bájt sorszámát küldi vissza, hanem az első még nem megkapott bájt sorszámát (0x99a90714). Így gyakorlatilag olyan, mintha a B-ből A-ba küldött ACK mező értéke azt adná meg, mi legyen a következő A-ból B-be küldött csomag sorszáma.



8. ábra: Tcp szegmens fejrészének felépítése

A TCP fejrész hossz megadja, hány 32 bites szóból áll a fejrész. 3 biten maximum 7 ábrázolható, ebből következően a TCP maximális fejrész hosszúsága 7×32 bit. Ebből 5×32 bit rögzített tehát az opciók maximum 2×32 bit hosszúságúak lehetnek. Ezután jön 5 nem használt bit majd 8 jelző bit. Ezek jelentése a 9. ábrán látható.

Bit	Név	Jelentés
1	CWR	Congestion Window Reduced
2	ECE	ECN-Echo
3	URG	Jelzi, hogy a 'Sürgősségi mutató' sürgős adat helyét jelzi
4	ACK	Jelzi, hogy a 'Nyugtaszám' érvényes nyugtát jelöl, ha ez 0 az előbbi mező figyelmen kívül hagyható
5	PSH	Az adat késedelem nélküli továbbítását jelöli, a vevő felé pedig jelzi: ne pufferelje az adatot hanem azonnal továbbítsa az alkalmazás felé
6	RST	A hoszt összeomlása, vagy más okból összezavart összeköttetés helyreálllítására szolgál. Ezenkívül érvénytelen szegmens elutasítására és összeköttetés létesítésének megtagadására is használják.
7	SYN	Összeköttetés létesítésére szolgál
8	FIN	Összeköttetés bontására szolgál

9. ábra: TCP szegmens flag bitjei

A TCP a 'három utas kézfogás' (three-way handshake) technikáját használja a kapcsolat kiépítésére.

a. Amennyiben beérkezik egy TCP szegmens SYN=1 ACK=0 jelzőbitekkel – és 'X' sorszámmal - egy hoszthoz (továbbiakban szerver), ő ellenőrzi, hogy van-e a szegmensben meghatározott cél port alatt olyan futó alkalmazás mely fogadja a kapcsolódási kísérletet. Amennyiben nincs, RST=1 jelzőbittel visszaküld egy szegmenst.

b. Amennyiben fogadja a felhasználói folyamat a kapcsolódási kísérletet, visszaküld egy szegmenst SYN=1 ACK=1 jelzőbittel, melynek ismét van egy sorszáma (például ’Y’) illetve a nyugtaszám a fentebb szerint megbeszélt (ha a kapcsolat létesítésére küldött csomag adatmezője 1 bájtos volt akkor a nyugtaszám ’X+1’).
c. Ezután a kezdeményező gép a már ismert módon nyugtázza a ’b’ pontban beérkező szegmenst.

A kapcsolat lebontására a kapcsolatot bontani kívánó fél küld egy csomagot a másik félnek, melyben a jelzőbitek közül FIN=1. Amennyiben arra megérkezik a nyugta, abba az irányba a kapcsolat lezárul. Amennyiben mindkét irányban bontjuk a kapcsolatot, a már említett nyugtázó csomagban szintén beállíthatjuk a FIN bitet 1-re. Ekkor az eredeti kezdeményező szintén küld egy nyugtát és a kapcsolat leárult. Természetesen időzítők segítenek abban, hogy egy kapcsolat ne legyen fenntartva a végtelenségig (például ha a FIN vagy a FIN nyugtázása elvész vagy egyik gép összeomlik).

Az ablakméret mezőben a küldő gép azt jelzi, hogy hány bájt fogadására képes. Ezzel megakadályozható, hogy egy gyors küldő csomagokkal árasszon el egy olyan gépet, mely csak lassan képes feldolgozni az adatokat. Például:

a. ’A’ gép küld 1460 bájtot ’B’ gépnek ’X’ sorszámmal.
b. ’B’ gép visszaküldi nyugta mezőjében ’X+1’-t de mivel a pufferében csak 700 bájt szabad hely van az ablak méretet 700 bájtra állítja.
c. ’A’ gép megkapja a nyugtát, melyből tudja, hogy az előző csomagot a fogadó megkapta, de most maximum 700 bájtnyi adatot küldhet

Amennyiben az ablakméretet 0-ra veszi a fogadó, a küldő mindaddig nem küldhet csomagot, míg a küldő újra küld egy ugyanolyan nyugtaszámú, de már 0-nál nagyobb ablakméretű csomagot.

Az ablakméret változtatása a küldőt megakadályozza abban, hogy a fogadót elárassza annyi csomaggal, amit már nem tud feldolgozni. Azonban előfordulhat olyan eset, mikor nem a fogadó az, aki kénytelen eldobni a csomagot, hanem valamely közbenső router. Ilyen esetben a router nem tud visszajelezni a küldőnek az ablakméret változtatásával. Ennek a problémának a kezelésére dolgozták ki a TCP torlódásvédelmi (Congestion avoidance) mechanizmusát.

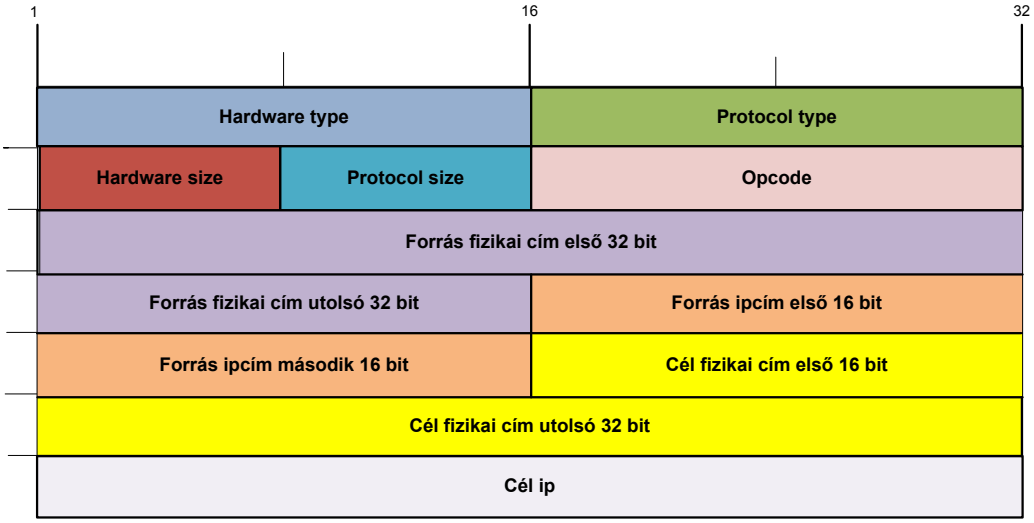
Az eljárás feltételezi, hogy csomagvesztés illetve időtúllépés csak és kizárólag azért történik, mert a hálózaton valahol torlódás alakult ki és ezért a közbenső router kénytelen volt eldobni a csomagot. Hogy a torlódást elkerüljük, a küldő nem csak a vevő által szabályozott ablakméretet tartja nyilván, hanem van egy torlódási ablak is. Mindkét számláló az adó által elküldhető bájtok számát mutatja. Az adó a maximálisan elküldhető bájtok számaként a két érték minimumát tekinti. Amennyiben a torlódási ablak a kisebb, az azt jelenti, hogy a szűk keresztmetszet a hálózaton lévő valamely közbenső gép vagy kapcsolat. Amennyiben a vevő által karbantartott ablak a kisebb, akkor a szűk keresztmetszet a lassú vevő.

Egy összeköttetés létrejötte után a torlódási ablakot az adó a legnagyobb szegmensméretre állítja. Ezután elküld egy szegmenst és ha erre nyugta érkezik, duplájára növeli a torlódási ablakot. Ezt mindaddig teszi, míg vagy eléri a vevő által karbantartott ablakméretet, vagy időtúllépés következik be. Amennyiben bizonyos torlódási ablak méretnél időtúllépés következik be, a torlódási ablakot a felére veszi vissza (tehát az utolsó olyan ablakméretre, melynél még megkapott a vevő minden csomagot). Ez az úgynevezett lassú kezdést biztosító algoritmus (slow start).

Más megoldásban még egy harmadik számláló is van, a torlódási küszöb (treshold). Ennek kezdőértéke 64 kbájt. A normál slow start eljárást használja a mechanizmus, azonban ha időtúllépés következik be, felére csökkenti a torlódási küszöb értékét, a torlódási ablakot pedig a maximális szegmensméretre állítjuk vissza. Abban az esetben, ha a torlódási ablak eléri a torlódási küszöb értékét, már nem exponenciálisan (duplázva) növeljük a torlódási ablakot, hanem lineárisan (szegmensméretenként).
Ha nincs időtúllépés, addig növeljük az ablakméretet, míg elérjük a vevő ablakméretét vagy befut egy ICMP SOURCE QUENCH (lassítást kérő) csomag. Ezt az esetet úgy kezeli az adó, mintha időtullépés következett volna be.

3.5. Address Resolution Protocol (ARP) csomag felépítése:

Ebben az esetben az Arp csomagot tartalmazó Ethernet keret `Típus` mezőjében `0x0806` szerepel. Egy Arp csomag felépítése a 10. ábrán látható.
`Hardware type` mező értéke `Ethernet` esetén 0x0001, a `Protocol type` mező `IPv4` esetén `0x0800`. A `Hardware size` mező értéke megadja a `Fizikai címek` méretét. Ez `Ethernet` esetében 6 byte, tehát a `Hardware size` mező értéke: `0x06`. Ugyanígy a `Protocol size` mező értéke a forrás és cél `protocol cím` méretét adja meg mely `IPv4` esetén `0x04`. Az `Opcode` mező lehetséges értéke a 11. ábrán látható.



10. ábra: ARP csomag felépítése

Opcode	Leírás
0x0001	Arp kérés
0x0002	Arp válasz

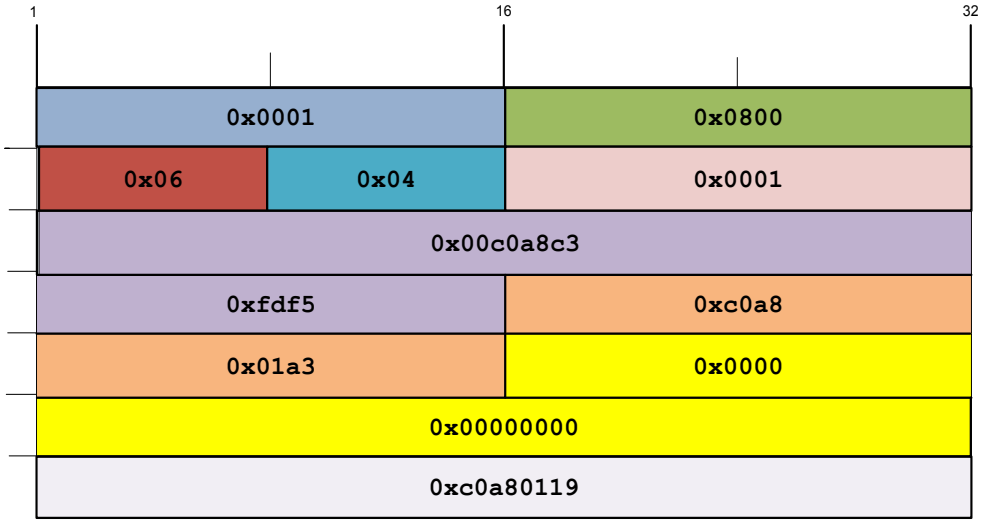
11. ábra: az opcode mező lehetséges értéke

ARP kérelem küldése:

Kérés küldésekor a cél fizikai cím értéke `0x000000000000`. Természetesen az ARP kérést tartalmazó Ethernet csomag cél címének értéke `0xffffffffffff`, azaz egy broadcast üzenetről van szó.
Egy példa csomag (mely a 12. ábrán látható) a következő képen néz ki:
A példában a `192.168.1.163`-es ip címmel és `00:c0:a8:c3:fd:f5` fizikai címmel rendelkező számítógép kívánja lekérni a `192.168.1.25`-ös ip címmel rendelkező számítógép fizikai címét.

ARP válasz:

A lekérdezett ipcím tulajdonosa is megkapja a csomagot. Ő visszaküldi ezt az ARP csomagot, kizárólag annyit változtat rajta hogy a cél fizika cím mezőbe beírja a saját fizikai címét. Ezt a csomagot tartalmazó ethernet csomag már célcíme már unicast.



12. ábra: Egy lehetséges ARP csomag

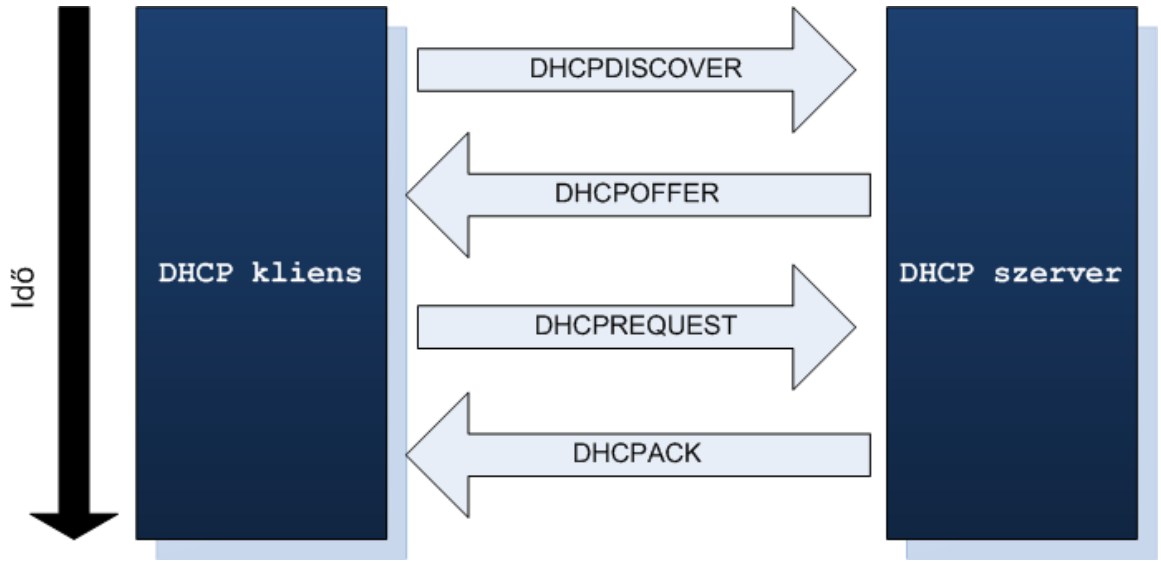
3.6. Dinamic Host Control Protocol (DHCP)

Ahhoz, hogy egy számítógép a hálózaton tudjon kommunikálni, jó néhány fontos konfigurációs paramétert szükséges megadni. Ez kis, néhány gépből álló hálózaton nem túl nehéz. Azonban nagy, több száz vagy ezres gépből álló hálózaton ez bonyolult és fáradtságos munka a hálózati adminisztrátoroktól.

Erre szolgál a DHCP (Dynamic Host Configuration Protocol): A hálózat egy dedikált szerverétől (a DHCP szervertől) a kliensek lekérik a szükséges paramétereket, mint az IP címet, alhálózati maszkot, alapértelmezett átjárót és DNS szervereket.
Az IP címeket a DHCP szerver egy külön IP készletből osztja ki. Azonban ha egy hoszt elhagyja a hálózatot (kikapcsolják, összeomlik) és nem adja vissza az IP címet a DHCP kiszolgálónak, akkor ez a cím tartósan elvész. Ennek megelőzésére az IP címeket egy rögzített időtartamra osztják ki. Ezt a módszert lízingelésnek (leasing) nevezzük. A lízing idő lejárta előtt az IP címet a kliensnek meg kell újítani. A DHCP szerver vagy hozzájárul, hogy az IP címet bizonyos ideig használja (meghosszabbítja a lízing időt) vagy egy új IP címet oszt ki.

A DHCP tranzakciók két udp well-known-port-ot használ: a dhcp szerver a 67-es porton fogadja a kéréseket, a válaszokat pedig a kliens 68-as portjára küldi.
A DHCP kérésnek 4 lépése van (13. ábra):

1. Mivel az első lépésben még nincs IP címe a kliensnek, az IP csomag feladó IP címében 0.0.0.0 szerepel, a címzett mezőben a 255.255.255.255. Az ethernet keret címzett mezőjében az `0xffffffffffff` fizikai cím szerepel, azaz a kliens a DHCPDISCOVER csomagot broadcast üzenettel küldi szét. Mivel a klienseknek ilyenkor még nincs IP címük, a csomagban található egy XID mező mellyel megkülönböztethetőek a gépek a tranzakció ideje alatt. A kliens a csomagban jelzi hogy a DHCP üzenet típusa (0x35) DHCPDISCOVER (0x01), tehát az idevágó rész: 0x350101 (az első 0x01 rész a típus hosszát jelzi, a tényleges csomagtípus a második 0x01). A kliens ebbe a kérésbe berakhatja a kérni kívánt vagy utoljára kapott IP címét illetőleg feltünteti, hogy milyen paraméterekre van szükséges a DHCP szervertől.
2. Erre a csomag egy DHCPOFFER válaszüzenetet küld (ez már unicast), melyben feltünteti egyrészről a csomag típusát (0x350102), feltünteti benne a kérésben kapott XID számot felajánlja neki a kért paramétereket illetőleg közli vele a saját (mármint a DHCP szerver) IP címét.
3. Ezután a kliens ismét egy broadcast üzenetet küld, de immáron ez egy DHCPREQUEST kérés (0x350103) melyben az imént felajánlott adatokat kéri le a DHCP kiszolgálótól. Amennyiben ez nem egy új ip cím lekérése hanem a már korábban lekért IP cím lízing ideje járt le, a kliens egyből ezt az csomagot küldi el (ebben az esetben természetesen nem broadcast üzenettel, mivel már ismeri a DHCP szerver címét a kliens, tehát csak a szerver ip címére küldi a kérést).
4. A DHCP kiszolgáló ezután visszaküld egy DHCPACK csomagot (0x350105), ezzel jelezve, a kliens a megadott lízing ideig használhatja a neki kiadott IP címet.



13. ábra: DHCP kérés folyamata

Természetesen a lekért IP címet a kliens vissza is adhatja, jelezve ezzel a DHCP kiszolgálónak, hogy nem kívánja a továbbiakban használni a számára kiosztott IP címet. Ebben az esetben a kliens egy DHCPRELEASE csomagot küld közvetlen a DHCP kiszolgálónak (unicast-al), melyben jelzi a csomag típusát (0x350107).

3.7. Domain Name System (DNS)

Ahhoz, hogy a hálózaton egy számítógépet megszólítsunk, szükséges hogy tudjuk a keresett számítógép 4 bájtos IP címét. De milyen nehéz lenne megjegyezni saját email címünket, ha annak neve ferenc.czina@64.233.183.27 lenne vagy kedvenc weboldalunk a 217.20.131.2 névre hallgatna.

Éppen ezért találták ki a DNS-t azaz a Domain name system-t vagy körzeti névkezelő rendszert. Egyrészről lehetőség van ezáltal a nehezen megjegyezhető 4 bájtos IP címeknek az ember számára könnyen megjegyezhető szöveges nevet adni. Másrészről a rendszer modulárisan és hierarchikusan van felépítve, emiatt könnyen bővíthető és változtatható a rendszer.

Az Interneten van néhányszáz elsődleges körzet. Ilyenek az `int`, `com`, `edu`, `gov`, `net`. Szintén rendelkezik minden ország egy elsődleges körzetnévvel (Magyarország a `hu`, az USA az `us` míg Japán `jp` körzetnévvel). Az interneten elérhető minden domain név egy fa struktúrába illeszkedik. Ezen fastruktúra gyökerében helyezkednek el az úgy nevezett root-szerverek. A root-szerverek (melyek felsorolása a 14. ábrán láthatóak) ismerik minden elsődleges körzet DNS szerverét (a magyar hu elsődleges körzetbe tartozó neveket a 15. ábrán látható szerverek szolgálják ki).

Név	Helye	Ip cím
A	Észak-Amerika	198.41.0.4
B	Észak-Amerika	192.228.79.201
C	Észak-Amerika 4 városában	192.33.4.12
D	Észak-Amerika	128.8.10.90
E	Észak-Amerika	192.203.230.10

F	Észak-Amerika 8 városában, Európ 12 városában, Ázsia 9 városában, Új-Zéland városában, Dél-Amerika 5 városában, Közel-kelet 3 városában, Ausztrália 1 városában, Közép-Amerika 2 városában, Afrika 2 városában	192.5.5.241
G	Észak-Amerika 1 városában	192.112.36.4
H	Észak-Amerika 1 városában	128.63.2.53
I	Európa 11 városában, Ázsia 9 városában, Észak-Amerika 7 városában, Új-Zéland 1 városában, Afrika 1 városában, Auszrália 1 városában, Közel-Kelet 1 városában	192.36.148.17
J	Észak-Amerika 13 városában, Európa 11 városában, Ázsia 4 városában, Afrika 2 városában, Ausztrália 1 városában, Dél-Amerika 3 városában,	192.58.128.30
K	Európa 10 városa, Közel-Kelet 2 városa, Grönland 1 városa, Ausztrália 1 városa, Észak-Amerika 1 városa, Ázsia 2 városa	193.0.14.129
L	Észak-Amerika 2 városában	199.7.83.42
M	Ászia 2 városában, Európa 1 városában, Észak-Amerika 1 városában	202.12.27.33

14. ábra: A ROOT névszerverek (2007. december)

Domain név	IP cím
sunic.sunet.se	192.36.125.2
ns.nic.hu	193.239.148.62
ns1.nic.hu	193.239.149.3
ns2.nic.fr	192.93.0.4
ns2.nic.hu	193.6.16.1
ns3.nic.hu	195.70.35.250

15. ábra: .hu elsődleges körzet névszerverei

Egy adott elsődleges körzet névszervere ismeri az adott elsődleges körzet alá tartozó domain neveket (például az ns.nic.hu pontosan tudja az `unideb.hu` domainnév címét azonban semmit nem tud a `yahoo.com` domainnév címéről). Amennyiben az elsődleges körzet névszerverétől megkapjuk a domain címét, az még lehetséges nem egy gép, hanem egy újabb névszerver lesz (például: az `unideb.hu` alá tartozó bejegyzés egy újabb DNS kiszolgáló címe, a `domser.unideb.hu` weboldalcím). Ettől a domain fa újabb mélységeibe tudunk lekérdezéseket végrehajtani mindaddig, míg meg nem találtuk az általunk keresett nevet.
Erre látunk példát az alábbi lekérdezésben (16. ábra). Windows környezetben `nslookup` parancs segítségével kértül le a `www.army.mil` weboldal címét.
Az első sorban még az alapértelmezett DNS kiszolgálót használtuk (`192.168.1.25`). Mivel ez a szerver esetünkben csak a SOHO hálózat belső gépeinek IP címét ismeri (és ezelőtt még ezt a címet nem kerestük így nem kerülhetett a DNS kiszolgáló gyorsítótárába), ő nem tudott nekünk válaszolni. Viszont ismer valakit, aki tudja a választ: megadta a root-szerverek neveit (6-25. sor). Ezután a 29. sorban az `i.root-servers.net`-hez intéztük a kérdést: ismeri-e a `www.army.mil` gép IP címét (33. sor). Ő sem tudott pontos választ adni: viszont megadta a `.mil` elsődleges domain-t kiszolgáló szerverek nevét (35-53. sor). Ezután ezen szerverek közül választottunk egyet (54. sor) és kértük: adja meg a keresett cím IP címét. Ő viszont szintén mást ajánlott: kérdezzük meg az army.mil domain DNS szervereitől (59-68. sor). És végül mikor ezen

szerverek közül kérdeztük az egyik kiszolgálót (69-71. sor), már meg tudta válaszolni a kérdésünket (73-76. sor): a keresett `www.army.mil` nevű gép címe: `143.69.249.10`.

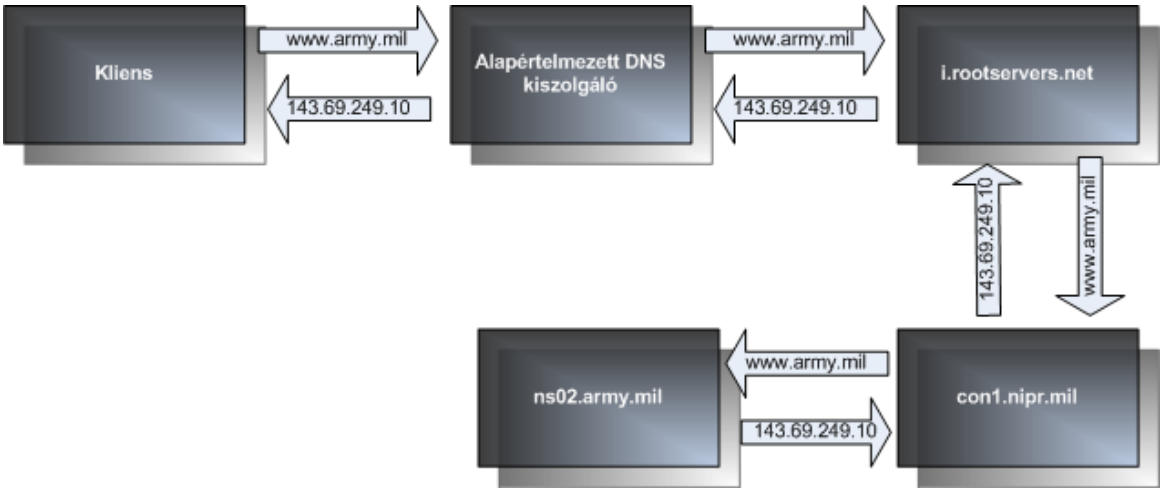
Alap esetben természetesen ez a folyamat teljesen automatikus. Nem nekünk kell a szervereket változtatni, a DNS szerverek a háttérben, egymás közt egyeztetnek ezekről az információkról (17. ábra). Mikor lekérjük a keresett címet, az alapértelmezett DNS kiszolgáló a root-szerverhez fordul, ő a `.mil` elsődleges körzet kiszolgálójához, aki az `army.mil` DNS kiszolgálójához. Ő tudja a választ, aki visszaadja a választ a `.mil` kiszolgálójához...és vissza az úton. Ez a név-cím összerendelés bekerül az alapértelmezett DNS kiszolgáló gyorsítótárába, így még egy ugyanilyen lekérdezés esetén már nem kell ugyanezt a folyamatot végig csinálni. Ezt a fajta lekérdezést hívják rekurzív lekérdezésnek.

```
1  > www.army.mil
2  Kiszolgáló:  [192.168.1.25]
3  Address:  192.168.1.25
4
5  Név:    www.army.mil
6  Served by:
7  - H.ROOT-SERVERS.NET
8
9  - I.ROOT-SERVERS.NET
10
11 - J.ROOT-SERVERS.NET
12       192.58.128.30
13 - K.ROOT-SERVERS.NET
14       193.0.14.129
15 - L.ROOT-SERVERS.NET
16       199.7.83.42
17 - M.ROOT-SERVERS.NET
18
19 - A.ROOT-SERVERS.NET
20       198.41.0.4
21 - B.ROOT-SERVERS.NET
22
23 - C.ROOT-SERVERS.NET
24
25 - D.ROOT-SERVERS.NET
26
27
28
29 > server i.root-servers.net
30 Alapértelmezett kiszolgáló:  i.root-
31 servers.net
32
33 > www.army.mil
34 Név:    www.army.mil
35 Served by:
36 - con1.nipr.mil
37       207.132.116.25
38       mil
39 - con2.nipr.mil
40       199.252.162.234
41       mil
42 - eur1.nipr.mil
43       199.252.154.234
44       mil
45 - eur2.nipr.mil
46       199.252.143.234
47       mil
48 - pac1.nipr.mil
49       199.252.180.234
50       mil
51 - pac2.nipr.mil
52       199.252.155.234
53       mil
54 > server con1.nipr.mil
55 Alapértelmezett kiszolgáló:
56 con1.nipr.mil
57 > www.army.mil
58 Név:    www.army.mil
59 Served by:
60 - NS02.army.mil
61       192.82.113.7
62       army.mil
63 - NS03.army.mil
64       130.114.200.6
65       army.mil
66 - NS01.army.mil
67       140.153.43.44
68       army.mil > server
69 ns02.army.mil
70 Alapértelmezett kiszolgáló:
71 ns02.army.mil
72
73 > www.army.mil
74 Név:    www1.ahp.us.army.mil
75 Address:  143.69.249.10
76 Aliases:  www.army.mil
```

16. ábra: Példa a rekurzív dns lekérdezésre

Egy domain név lehet abszolút és relatív. Amennyiben egy domain név után, a végére rakunk pontot, akkor az egy abszolút név (`www.index.hu.`) ellenkező esetben relatív névről beszélünk (`www.index.hu`).

Minden operációs rendszer rendelkezik egy elsődleges dns utótaggal. Ez linux alapú operációs rendszerben a `/etc/resolv.conf` ’`search`’ sorában, míg windows-nál az `ipconfig` parancs ’ `Kapcsolatspecifikus DNS-utótag`’ sorában látható. Ezt megkaphatjuk a DHCP kiszolgálótól, de mi is beállíthatjuk a gép hálózati beállításai között. Amennyiben egy relatív domain nevet szeretnénk feloldani, az elsődleges dns utótag a keresett domain mögé kerül. Ha az nem található, elindulunk felfele a keresési fában.



17. ábra: Rekurzív névfeloldás

Például: a számítógép elsődleges dns utótagja a `fricci.local`. Ezen a gépen kiadjuk a `www.index.hu` feloldását. A keresés a következő sorban történik:

1. Megkísérli feloldani a `www.index.hu.fricci.local` domain nevet, sikertelenül.
2. Megkísérli feloldani a `www.index.hu.local` domain nevet, sikertelenül.
3. Megkísérli feloldani a `www.index.hu` domain nevet, sikeresen.

Így történhet meg, hogy amennyiben a gép elsődleges dns utótagja a chello.hu, és mi fel szeretnénk oldani a `www` relatív nevet, megkísérli feloldani a `www.chello.hu` domain nevet, mely sikeres lesz.

3.8. Simple Mail Transfer Protocol (SMTP)

Az SMTP protokoll az alkalmazási rétegben helyezkedik el. Segítségével kommunikálnak egymással a levél továbbító szerverek, illetve a levelező kliensek is ezt használják, hogy átadják továbbításra a levelet a felhasználó által használt levél továbbító szervernek. A protokoll egyszerű plain text formátumot használ, így gyakorlatilag ember által olvasható a kommunikáció. Alapértelmezés szerint a 25 TCP portot használja.

Az alábbiakban egy rövid kommunikáció látható telneten keresztül a UPC lakossági smtp kiszolgálójával:

```
fricci@Sentinel:~$ telnet mail.chello.hu 25
Trying 213.46.255.2...
Connected to mail.chello.hu.
Escape character is '^]'.
220 viefep32-int.chello.at ESMTP server (InterMail vM.7.08.02.02 201-
2186-121-104-20070414) ready Wed, 13 Feb 2008 18:01:07 +0100
HELO user.chello.hu
250 viefep32-int.chello.at
MAIL FROM:czina.ferenctamas@chello.hu
250 Sender <czina.ferenctamas@chello.hu> Ok
RCPT TO:ferenc.czina@gmail.com
```

```
250 Recipient <ferenc.czina@gmail.com> Ok
DATA
354 Ok Send data ending with <CRLF>.<CRLF>
Subject:Üdvözlet
Ez egy teszt üzenet e-mail-ben.
.
quit
Connection closed by foreign host.
fricci@Sentinel:~$
```

3.9. Post Office Protocol (POP3)

A Post Office Protocol version 3 (POP3) egy alkalmazás szintű protokoll, melynek segítségével az e-mail kliensek egy meglévő TCP/IP kapcsolaton keresztül letölthetik az elektronikus leveleket a kiszolgálóról. Napjainkban ez a legelterjedtebb protokoll az elektronikus levelek lekéréséhez. A jelenleg használatos harmadik változat (version 3) elődjei a POP, illetve POP2 változatok.

A protokollra eredetileg az időszakosan létrejövő TCP/IP kapcsolatok (pl. dial-up) miatt volt szükség, ugyanis lehetővé teszi a kapcsolódás korlátozott ideje alatt a levelek kezelését a felhasználó gépén, úgy, hogy a levelek összességében akár a szerveren is maradhatnak. A leveleket azután helyben lehet olvasni, szerkeszteni, tárolni stb. A POP3 protokoll kizárólag a levelek letöltésére alkalmas; küldésükre az SMTP protokoll szolgál.

Az alábbiakban egy rövid kommunikáció látható telneten keresztül a UPC lakossági pop3 levelező kiszolgálójával:

```
fricci@Sentinel:~$ telnet pop.chello.hu 110
Trying 213.46.255.2...
Connected to pop.chello.hu.
Escape character is '^]'.
+OK InterMail POP3 server ready.
USER czina.ferenctamas@chello.hu
+OK please send PASS command
PASS fricci001
+OK czina.ferenctamas@chello.hu is welcome here
LIST
+OK 5 messages
1 3555
2 765
3 475
4 764
5 110141
.
RETR 4
+OK 764 octets
Return-Path: <czina.ferenctamas@chello.hu>
Received: from viefep13 ([192.168.13.142]) by viefep13-
int.chello.at InterMail vM.7.08.02.00 201-2186-121-20061213) with
ESMTP id 20080116073441.KRNA6198.viefep13-int.chello.at@viefep13
for <czina.ferenctamas@chello.hu>;
Wed, 16 Jan 2008 08:34:41 +0100
Message-ID: <14264258.1200468881888.JavaMail.root@viefep13>
Date: Wed, 16 Jan 2008 8:34:41 +0100
```

```
From:  <czina.ferenctamas@chello.hu>
To: czina.ferenctamas@chello.hu
Subject:
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 7bit
X-Priority: 3 (Normal)
Sensitivity: Normal
X-Originating-IP: from 80.99.230.91 by web-edge.chello.com; Wed,
16 Jan 2008 8:34:41 +0100

teszt
.
```

4. A kiscéges/otthoni (SOHO) hálózatok

A számítógépek megjelenése után sokáig az egyetemek és katonai rendszerek kiváltsága volt a számítógépes hálózat. Aztán lassanként megjelent majd nemsokára nélkülözhetetlenné váltak a nagyvállalati környezetben is.

A két legfontosabb előnye a kommunikácó és az erőforrás megosztás volt. Többé nem kellett napokat várni, hogy a posta kikézbesítsen egy levelet. Elektronikus megfelelője – az e-mail - néhány másodpercen belül megérkezik a címzetthez. Később az azonnali üzenetküldő szolgáltatások megjelenésével – amilyen az MsnMessenger, Gtalk, ICQ, IRC – a kommunikácó tovább fejlődött. A kapcsolattartás 3 generációja a hang és képtovábbítás, ingyenes telefonálás, élőképes konferenciák.

A második nagy előnye az erőforrás megosztás. Korábban a számítási kapacitás és az egyes perifériák (meghajtók, cd írók, szallagos tároló egységek, nyomtatók) igen drágák voltak. A hálózatoknak köszönhetően elég volt egyetlen eszközt beszerezni, a munkaállomások hozzá tudtak férni hálózaton keresztül ehhez az eszközhöz.

Ezek a funkciók járultak hozzá a hálózat nagyon gyors elterjedésének. Mind több vállalat, kormányzat kötötte hálózatba a gépeket.

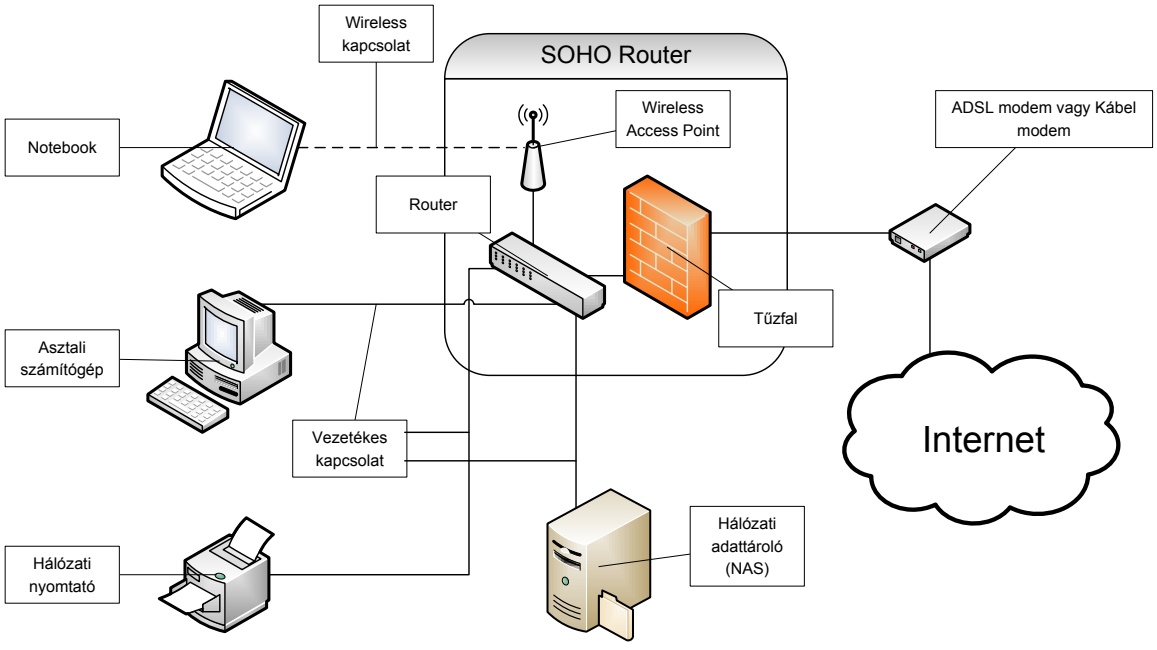
Ahogy az eszközök mind olcsóbbak és egyszerűbbek lettek, megjelentek a direkt az otthoni vagy kiscéges környezetbe szánt eszközök. Új kategória született: a SOHO (Small Office/Home Office) eszközök kategóriája. A SOHO eszközök (mely lehet hálózati eszköz, nyomtató vagy akár hálózati adattároló) jelentése a következő: az eszköz teljesítményben nem versenyezhet nagyvállalati környezetbe szánt társával viszont cserébe sokkal egyszerűbben beállítható, karbantartható és árban mindig jóval a nagyvállalati környezetbe szánt társa alatt van. Az ilyen eszközöket otthonokban vagy kis – néhány 10 fős – irodákban szokták használni.

A SOHO hálózat is innen ered. Az otthonokban, kis irodákban is egyre több hálózat épül ki. Természetesen itt nem áll rendelkezésre szakértő személyzet a karbantartásra, így az eszközök mindig rendelkeznek a könnyű beállíthatóság és karbantarthatóság tulajdonságával – még ha emiatt jó néhány szolgáltatásról le is kell mondani.

Tipikusan egy SOHO hálózat egy SOHO routerből áll, mely egy átjárót jelent egyrészről a helyi internetszolgáltató hálózata és a belső kis irodai hálózat között. Szinte mindig igaz az, hogy az ilyen SOHO routerek switch-el vannak egybe építve, így a számítógépek UTP kábelét bele dugjuk a routerbe (tipikusan 4 csatlakozási lehetőség van erre), majd az internet szolgáltató hálózatát a SOHO router egy dedikált csatlakozójába kötjük – és kész. Ezután ezeket a SOHO routereket valamilyen webes felületen vagy valamelyik gépen futtatható beállító alkalmazás segítségével lehet felkonfigurálni. Ez a konfigurálás is inkább kérdezz-felelek formájú, azaz az alkalmazás egy 'varázslót' indít el, mely lépésről lépésre végig vezeti a tulajdonost a beállítási lépéseken. Egyszerű kérdéseket tesz fel, és a válasz alapján összegyűjtött adatok alapján kvázi önmagát konfigurálja fel.

Az utóbbi időben a vezetéknélküli hálózati eszközök ára drasztikus zuhanásba kezdett. A notebook-ok mára már a háztartásokban kezd egyre inkább elterjedni. Ezen eszközök legnagyobb előnye a hordozhatóság, így szinte mindegyikben beépített wireless – azaz vezeték nélküli – hálózati adóvevő van. Emiatt manapság már igen elterjed, hogy ezek a SOHO routerek wireless access point-ként is működnek. Így egyrészről az otthonokban lévő több számítógép mindegyike rákapcsolódhat az internetre másrészről az utóbbi időben egyre elterjedtebb hordozható számítógépek is vezeték nélkül kapcsolódhatnak a routeren keresztül az internetre.

Egy tipikus SOHO hálózat látható a 18. ábrán. Az internetre a hálózat a szolgáltatótól kapott ADSL modemen vagy Kábel modemen keresztül csatlakozik.



18. ábra: Egy tipikus soho hálózat

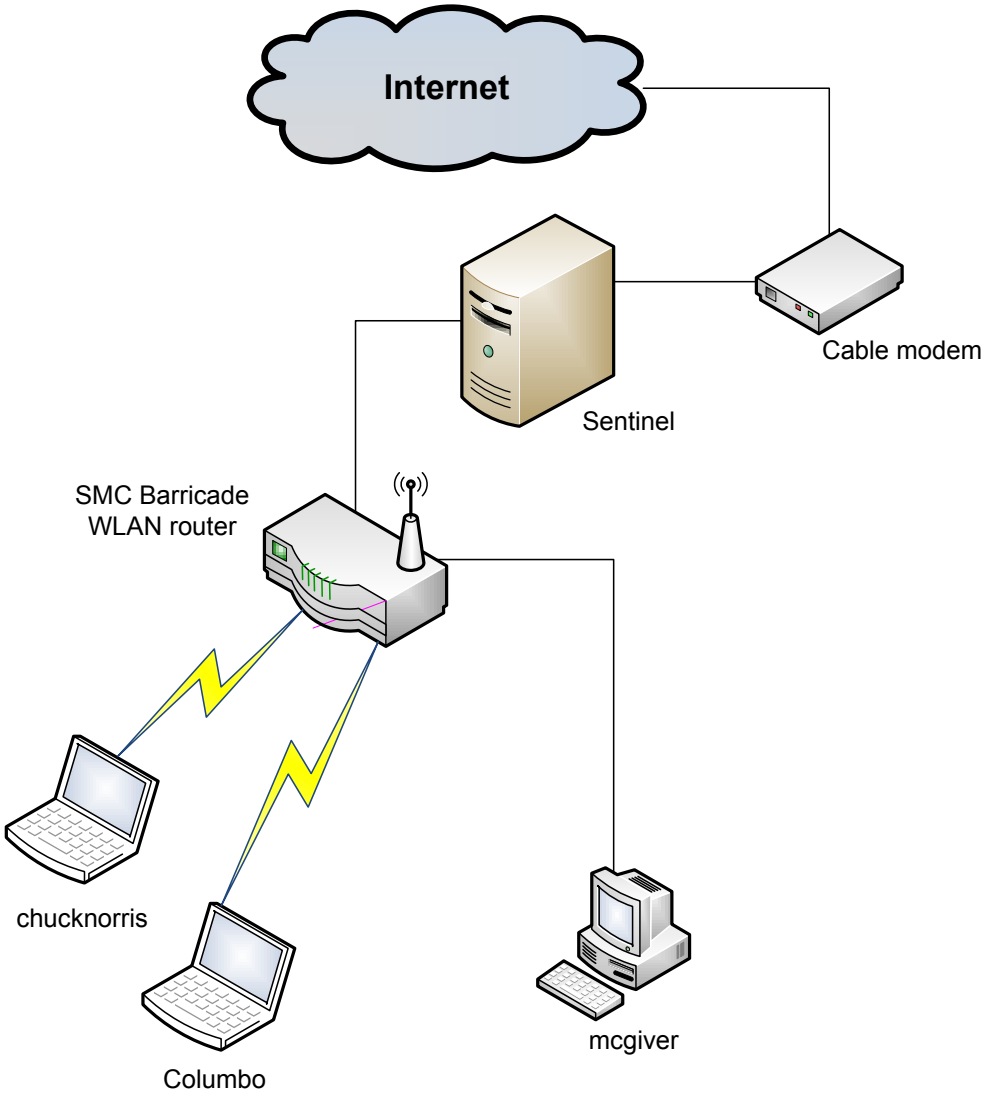
Általában erre csatlakozik a SOHO router mely tipikusan tartalmaz egy tűzfalat, routert és wireless access pointot. Ehhez a routerhez csatlakozik egy vagy több asztali számítógép. Mára már előfordulnak ethernet portra köthető hálózati nyomtatók és

hálózati adattároló eszközök a SOHO hálózatban melyek SMB protokoll segítségével hálózati meghajtóként felcsatolhatók. A routerrel wirelessel szokott kommunikálni a háztartásban lévő egy vagy több hordozható számítógép.

5. A megfigyelésben résztvevő hálózat bemutatása

5.1. Topológia

A SOHO hálózatban egy asztali pc, két notebook, egy router és egy gateway-ként alkalmazott pc vesz részt (19. ábra).



19. ábra: A SoHo hálózat topológiája

Az internetet egy Motorola típusú kábelmodemen keresztül érjük el, mely UTP kábellel van összekötve egy gateway-kén használt Compaq pc egyik hálózati

kártyájával (továbbiakban a gépre *Sentinel* néven hivatkozok). A gép másik hálózati kártyája egy SMC Barricade router LAN1-es csatlakozójába van kötve. Ezen router LAN2-es hálózati csatlakozójára egy asztali pc csatlakozik (továbbiakban *mcgiver* néven hivatkozott).

A router wifi szolgáltatását két notebook veszi igénybe (*columbo* és *chucknorris* néven hivatkozott) WPA-PSK titkosított kapcsolaton keresztül.

5.2. A hálózaton igénybe vehető szolgáltatások

Az elérhető hálózati szolgáltatásokat a Sentinel (192.168.1.25-ös fix ip címen) nyújtja. A gépen található Debian operációs rendszer fut, a rendszer által nyújtott szolgáltatások a következők:

a. DHCP szerver: a SOHO hálózat gépeinek megadja az IP címet (192.168.1.0-s hálózatból), átjárót, DNS kiszolgálókat.
b. Gateway: a SOHO hálózatot (eth0) és az Internetet (eth1) kapcsolja össze
c. SMB server: A SOHO hálózat gépeinek hálózati mappa elérését teszi lehetővé
d. Tűzfal az eth1 irányából: IPTABLES szabályok segítségével csomagellenőrzést végez.

A sentinel gép hálózati konfigurációja a következő:

```
eth0      Link encap:Ethernet  HWaddr 00:50:04:51:52:A6
          inet addr:192.168.1.25  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::250:4ff:fe51:52a6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:114956 errors:0 dropped:0 overruns:0 frame:0
          TX packets:223188 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9280837 (8.8 MiB)  TX bytes:176185247 (168.0 MiB)
          Interrupt:11 Base address:0x6000

eth1      Link encap:Ethernet  HWaddr 00:E0:4C:00:75:E1
          inet addr:89.132.242.89  Bcast:255.255.255.255 Mask:255.255.254.0
          inet6 addr: fe80::2e0:4cff:fe00:75e1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:307515 errors:0 dropped:0 overruns:0 frame:0
          TX packets:114248 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:181251853 (172.8 MiB)  TX bytes:9192496 (8.7 MiB)
          Interrupt:11 Base address:0x1000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:636 (636.0 b)  TX bytes:636 (636.0 b)

sit0      Link encap:IPv6-in-IPv4
          NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)


      Kernel IP routing table
      Destination     Gateway         Genmask         Flags   MSS Window  irtt
      Iface
      192.168.1.0     *               255.255.255.0   U         0 0          0
      eth0
      89.132.242.0    *               255.255.254.0   U         0 0          0
      eth1
      default         catv-5984f3fe.c 0.0.0.0         UG        0 0          0
      eth1
```

A kliens gépek:

chucknorris: Fujitsu-Siemens típusú laptop Windows XP operációs rendszerrel. A gépen böngésző program (Firefox és Explorer7), Ftp kliens, Torrent kliens, médialejátszók, online játszható játék (World of Warcraft) és különböző irodai programok és programfejlesztő eszközök futnak illetőleg egy adatbázis szerver (MS SQL)

```
===========================================================================
Kapcsolatlista:
0x1 ........................... MS TCP Loopback interface
0x2 ...00 c0 a8 c3 fd f5 ......  Atheros AR5005G Wireless Network Adapter -
Packet Scheduler Miniport
0x3 ...00 e0 4c 00 75 e1 ......  Realtek RTL8139 csalßd˙ PCI gyors Ethernet NIC -
Packet Scheduler Miniport
===========================================================================
===========================================================================
Aktív útvonalak:
     Hálózati cél   Hálózati maszk           Átjáró       Kapcsolat Metrika
          0.0.0.0          0.0.0.0     192.168.1.25   192.168.1.163       25
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1        1
      192.168.1.0    255.255.255.0    192.168.1.163   192.168.1.163       25
    192.168.1.163  255.255.255.255        127.0.0.1       127.0.0.1       25
    192.168.1.255  255.255.255.255    192.168.1.163   192.168.1.163       25
        224.0.0.0        240.0.0.0    192.168.1.163   192.168.1.163       25
  255.255.255.255  255.255.255.255    192.168.1.163               3        1
  255.255.255.255  255.255.255.255    192.168.1.163   192.168.1.163       1
Alapértelmezett átjáró:      192.168.1.25
===========================================================================
Állandó útvonalak:
  Nincs


Windows IP konfiguráció

        Állomásnév. . . . . . . . . . . . . . : chucknorris
        Elsődleges DNS-utótag . . . . . . . . :
        Csomóponttípus  . . . . . . . . . . . : Ismeretlen
        IP útválasztás engedélyezve . . . . . : Nem
        WINS-proxy engedélyezve . . . . . . . : Nem
        DNS-utótag keresési listája . . . . . : fricci.local

Ethernet-adapter Vezeték nélküli hálózati kapcsolat:

        Kapcsolatspecifikus DNS-utótag. . . . : fricci.local
        Leírás. . . . . . . . . . . . . . . . : Atheros AR5005G Wireless Network
Adapter
        Fizikai cím . . . . . . . . . . . . . : 00-C0-A8-C3-FD-F5
        DHCP engedélyezve . . . . . . . . . . : Igen
        Automatikus konfiguráció engedélyezve : Igen
        IP-cím. . . . . . . . . . . . . . . . : 192.168.1.163
```

```
        Alhálózati maszk. . . . . . . . . . . : 255.255.255.0
        Alapértelmezett átjáró. . . . . . . . : 192.168.1.25
        DHCP kiszolgáló . . . . . . . . . . . : 192.168.1.25
        DNS-kiszolgálók . . . . . . . . . . . : 213.46.246.51
                                                213.46.246.52
        Bérleti jog kezdete . . . . . . . . . : 2007. november 22. 8:03:44
        Bérleti jog vége. . . . . . . . . . . : 2007. november 22. 8:13:44

Ethernet-adapter Helyi kapcsolat:

        Adathordozó állapota. . . . . . . . . : Adathordozó leválasztva
        Leírás. . . . . . . . . . . . . . . . : Realtek RTL8139/810x Family Fast
Ethernet NIC
        Fizikai cím . . . . . . . . . . . . . : 00-E0-4C-00-75-E1
```

columbo: ECS típusú laptop Windows XP operációs rendszerrel. A gépen böngésző program (Firefox és Explorer7) és néhány irodai program fut.

```
===========================================================================
Kapcsolatlista:
0x1 ........................... MS TCP Loopback interface
0x2 ...00 11 5b 9b 11 98 ...... VIA-kompatibilis gyors Ethernet-adapter - Packet
Scheduler Miniport
0x10004 ...00 17 31 2e ba e2 ...... ASUS USB Wireless Network Adapter - Packet
Scheduler Miniport
===========================================================================
===========================================================================
Aktív útvonalak:
     Hálózati cél   Hálózati maszk          Átjáró       Kapcsolat Metrika
          0.0.0.0          0.0.0.0     192.168.1.25   192.168.1.200       1
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1       1
      192.168.1.0    255.255.255.0    192.168.1.200   192.168.1.200       1
    192.168.1.200  255.255.255.255        127.0.0.1       127.0.0.1       1
    192.168.1.255  255.255.255.255    192.168.1.200   192.168.1.200       1
        224.0.0.0        240.0.0.0    192.168.1.200   192.168.1.200       1
  255.255.255.255  255.255.255.255    192.168.1.200               2       1
  255.255.255.255  255.255.255.255    192.168.1.200   192.168.1.200       1
Alapértelmezett átjáró:      192.168.1.25
===========================================================================
Állandó útvonalak:
  Nincs


Windows IP konfiguráció

        Állomásnév. . . . . . . . . . . . . . : laptop
        Elsődleges DNS-utótag . . . . . . . . :
        Csomóponttípus  . . . . . . . . . . . : Ismeretlen
        IP útválasztás engedélyezve . . . . . : Nem
        WINS-proxy engedélyezve . . . . . . . : Nem
        DNS-utótag keresési listája . . . . . : fricci.local

Ethernet-adapter Helyi kapcsolat:

        Adathordozó állapota. . . . . . . . . : Adathordozó leválasztva
        Leírás. . . . . . . . . . . . . . . . : VIA Rhine II Fast Ethernet
Adapter
        Fizikai cím . . . . . . . . . . . . . : 00-11-5B-9B-11-98

Ethernet-adapter Vezeték nélküli hálózati kapcsolat:

        Kapcsolatspecifikus DNS-utótag. . . . : fricci.local
        Leírás. . . . . . . . . . . . . . . . : ASUS USB Wireless Network
Adapter
        Fizikai cím . . . . . . . . . . . . . : 00-17-31-2E-BA-E2
        DHCP engedélyezve . . . . . . . . . . : Igen
        Automatikus konfiguráció engedélyezve : Igen
```

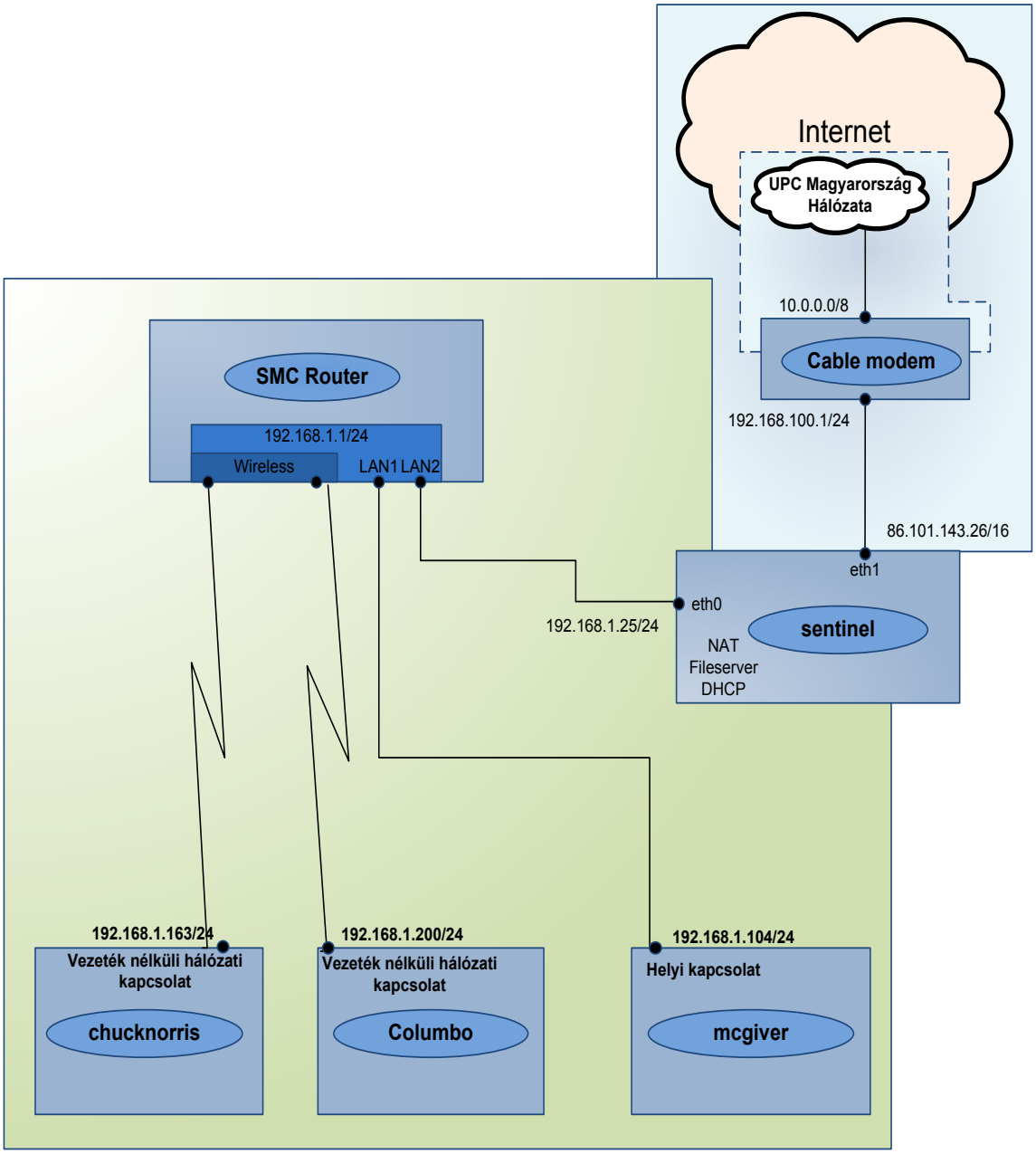
```
        IP-cím. . . . . . . . . . . . . . . . : 192.168.1.200
        Alhálózati maszk. . . . . . . . . . . : 255.255.255.0
        Alapértelmezett átjáró. . . . . . . . : 192.168.1.25
        DHCP kiszolgáló . . . . . . . . . . . : 192.168.1.25
        DNS-kiszolgálók . . . . . . . . . . . : 213.46.246.51
                                                213.46.246.52
        Bérleti jog kezdete . . . . . . . . . : 2007. november 22. 8:01:02
        Bérleti jog vége. . . . . . . . . . . : 2007. november 22. 8:11:02
```

mcgiver: Asztali számítógép melyen Windows XP operációs rendszer fut. A gépen böngésző program (Firefox és Explorer7), különböző irodai programok, adatbázis és média szerver fut.

```
===========================================================================
Kapcsolatlista:
0x1 ........................... MS TCP Loopback interface
0x2 ...00 0d 61 63 63 31 ...... VIA VT6105 Rhine III Fast Ethernet Adapter -
Packet Scheduler Miniport
===========================================================================
===========================================================================
Aktív útvonalak:
     Hálózati cél   Hálózati maszk          Átjáró       Kapcsolat Metrika
          0.0.0.0          0.0.0.0     192.168.1.25   192.168.1.104       20
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1        1
      192.168.1.0    255.255.255.0    192.168.1.104   192.168.1.104       20
    192.168.1.104  255.255.255.255        127.0.0.1       127.0.0.1       20
    192.168.1.255  255.255.255.255    192.168.1.104   192.168.1.104       20
        224.0.0.0        240.0.0.0    192.168.1.104   192.168.1.104       20
  255.255.255.255  255.255.255.255    192.168.1.104   192.168.1.104       1
Alapértelmezett átjáró:      192.168.1.25
===========================================================================
Állandó útvonalak:
  Nincs


Windows IP konfiguráció

        Állomásnév. . . . . . . . . . . . . . : mcgiver
        Elsődleges DNS-utótag . . . . . . . . :
        Csomóponttípus  . . . . . . . . . . . : Ismeretlen
        IP útválasztás engedélyezve . . . . . : Nem
        WINS-proxy engedélyezve . . . . . . . : Nem
        DNS-utótag keresési listája . . . . . : fricci.local

Ethernet-adapter Helyi kapcsolat 2:

        Kapcsolatspecifikus DNS-utótag. . . . : fricci.local
        Leírás. . . . . . . . . . . . . . . . : VIA Rhine III Fast Ethernet
Adapter
        Fizikai cím . . . . . . . . . . . . . : 00-0D-61-63-63-31
        DHCP engedélyezve . . . . . . . . . . : Igen
        Automatikus konfiguráció engedélyezve : Igen
        IP-cím. . . . . . . . . . . . . . . . : 192.168.1.104
        Alhálózati maszk. . . . . . . . . . . : 255.255.255.0
        Alapértelmezett átjáró. . . . . . . . : 192.168.1.25
        DHCP kiszolgáló . . . . . . . . . . . : 192.168.1.25
        DNS-kiszolgálók . . . . . . . . . . . : 213.46.246.51
                                                213.46.246.52
        Bérleti jog kezdete . . . . . . . . . : 2007. november 22. 8:04:29
        Bérleti jog vége. . . . . . . . . . . : 2007. november 22. 8:14:29
```

5.3. A SOHO hálózat felépítése és kapcsolata az internettel

A Soho hálózat felépítése és az egyes csatlakozások IP címei láthatóak a 20 ábrán.



20. ábra: A hálózat felépítése és kapcsolatai

Mint látható, mindhárom számítógép az SMC Routerhez csatlakozik (a mcgiver utp kábel segítségével míg a két hordozható számítógép wireless kapcsolaton keresztül). Ugyanehez a routerhez csatlakzik utp kábelen keresztül a sentinel nevű átjáró egyik hálózati kártyája. Így gyakorlatilag a router mint egy switch és access point egyben, egy hálózatként kezeli a négy gépet. Ezzel együtt a sentinel mint dhcp szerver olyan dhcp paramétereket ad meg a gépeknek, amivel önmagát jelöli meg, mint alapértelmezett átjáró. A sentinel gép másik hálózati kártyája pedig csatlakozik a kábelmodemre.

6. A tanulmányozott SOHO hálózat Internet szolgáltatójának (ISP) hálózata

A hagyományos, antennás televízió és rádió-adás vétele lassacskán teljesen idejét múlttá válik. Régóta megszoktuk, hogy a legtöbb városban egy vagy több olyan cég van, mely a lakásokba behúzott koaxiális kábelen keresztül közvetíti az egyes televízióadók adását. Viszont több év kellett hozzá hogy a technológia eljusson odáig, hogy a kábeltv szolgáltatók a birtokukban lévő a városokat behálózó koaxiális kábelhálózaton keresztül valami mást is elkezdjenek szállítani.

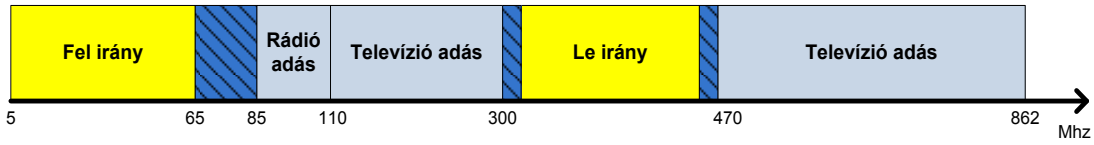
Ahhoz, hogy ezen a kábelhálózaton keresztül IP szolgáltatásokat nyújtson a szolgáltató, meg kellett oldani két problémát:

1. Úgy kell a kábelhálózatán IP szolgáltatást biztosítani, hogy emellett a már megszokott analóg műsorszórás zavartalan legyen.
2. Az IP kommunikáció két irányú, míg a televízió szolgáltatás csak a központtól a előfizetői végpontok felé tartó kommunikáció. Ahhoz hogy egy szolgáltató IP kommunikációs szolgáltatásra is használja a kábelrendszerét, a rendszerben biztosítani kellett a visszirányú kommunikációt.

Az első időkben többféle módszer volt, amivel IP szolgáltatásokat nyújthattak az előfizetőknek. Gyakorlatilag minden kábelmodem gyártó saját technológiát használt. Azért, hogy az eszközgyártók közötti versenyt fokozzák (ezáltal alacsonyabb árat tudjanak kiharcolni), az amerikai szolgáltatók összefogtak, hogy kidolgozzanak egy saját egységes specifikációt. Így született meg a DOCSIS (Data Over Cable System Interface Specification) rendszer. Az amerikai sikert látva az európai szolgáltatók hasonlóan cselekedte, itt némileg módosított szabvánnyal, az EuroDOCSIS szabványban állapodtak meg.

A DOCSIS 1.0 szabvány a sávszélesség kiosztásra a „Best-Effort” vagyis a legjobb elérhető technikát alkalmazta, ami annyit tesz, hogy mindenki annyi sávszélességet kaphat az összesből, amennyi a pillanatnyi terhelésnek megfelelően jut. Garantált sávszélesség és időkritikus szolgáltatások nyújtásához kellett kifejleszteni a (Euro)DOCSIS 1.1-t. A 1.1 első verziója 1999-ben jelent meg és a QoS technológiával egészítette ki az előző változatot. Az átállás az újabb verzióra csupán egy szoftverfrissítést jelent a legtöbb gyártmánynál. Nem ez a helyzet a 2001-ben kihirdetett (Euro)DOCSIS 2.0-val. Itt a fizikai rétegben történt változás miatt teljesen új chip készletekre van szükség. A 2.0 verzióban az előrelépést a visszirány átviteli sebességének felgyorsítása és vele páthuzamosan zavartűrőbbé tétele jelentette. Így már az alapvetően aszimmetrikus KTV hálózatokon is lehet szimmetrikus adatszolgáltatást kínálni. A DOCSIS 3.0-s változatát már felkészítették az IPv6-ra illetve lehetőség van arra, hogy egy előfizető egyidőben több fel- illetve letöltési csatornát használjon (channel bonding).

Az előre és a visszirányra a kábeltévé szolgáltatók két frekvenciasávot különítettek el (21. ábra):
A feltöltési irányba a frekvencia sáv elején egy 60 Mhz széles tartományt, a letöltési irányba pedig egy 170 Mhz széles tartományt. Az IP szolgáltatás tartományok és az analóg szolgáltatás tartományok közzé védősávokat iktatott, hogy véletlen se zavarják egymást.



21. ábra: Frekvencia sávok

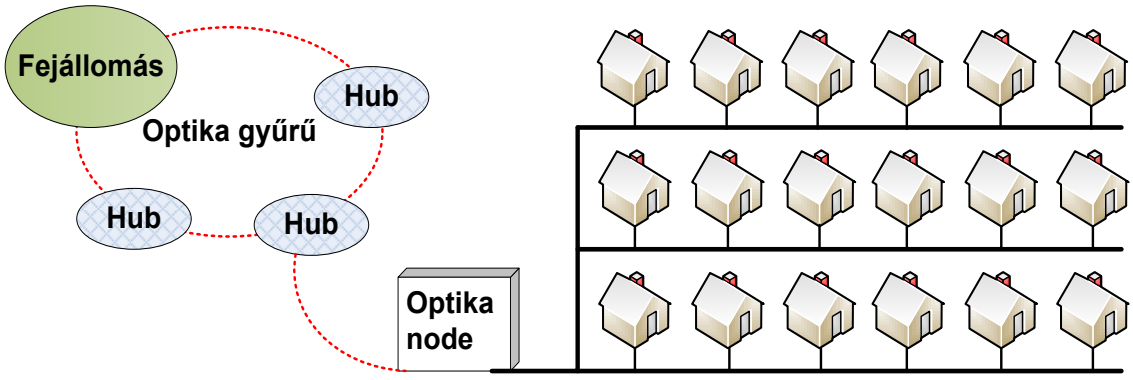
A második problémát úgy sikerült megoldani, hogy a kábelhálózaton elhelyezett azon erősítőket, melyek eddig csak a központból menő analóg jeleket erősítették, korszerű kétirányú erősítőkre cserélték. Ezek a frekvencia tartomány 75 Mhz –től felfelé a leirányba erősítették a jelet, míg a 75 Mhz-től lefele a végponttól a központ felé erősítették. Ezt nevezik a hálózat visszirányúsításának. Így elhárult minden akadály, hogy a kiépített hálózaton IP szolgáltatást nyújtsanak.
A 22. ábrán látható a jelenlegi docsis szabványokon elérhető átviteli sebesség.

Verzió	DOCSIS		EuroDocsis	
	Letöltés	Feltöltés	Letöltés	Feltöltés
1.x	48 Mbit/s	9 Mbit/s	50 Mbit/s	9 Mbit/s
2.0	38 Mbit/s	27 Mbit/s	50 Mbit/s	27 Mbit/s
3.0	152 Mbit/s	108 Mbit/s	200 Mbit/s	108 Mbit/s

22. ábra. DOCSIS sebességei

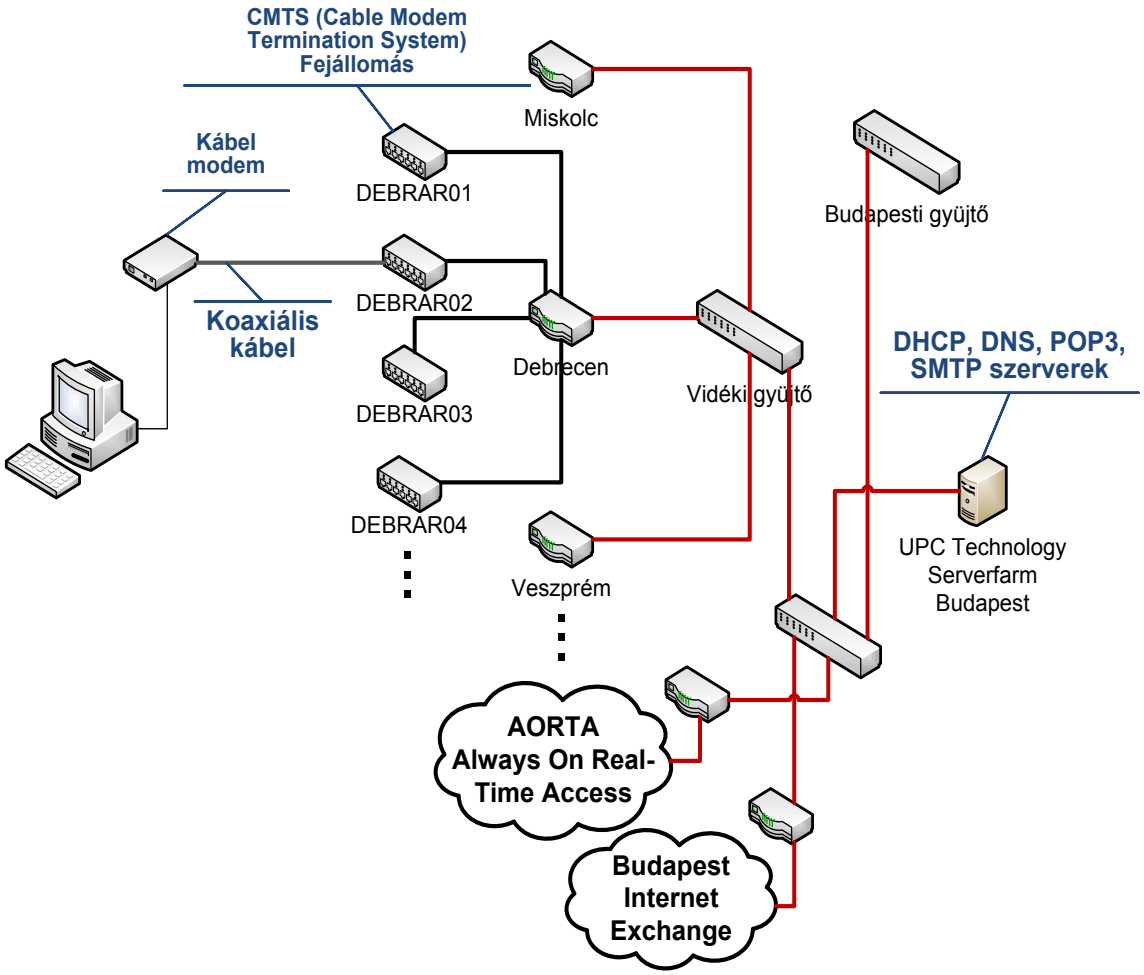
6.1. Internet és VoIP szolgáltatás a koax kábelen

Budapesten és a nagyobb városokban a városi HFC hálózat a 23. ábra szerint van felépítve. A HFC rövidítés a Hybrid Fiber Coax rövidítést takarja: Az IP szolgáltatások a fejállomáson lévő CMTS-en keresztül jutnak el az előfizetői végpontokba. A fejállomás egy – a várost átszelő – többszörösen redundáns optika gyűrűn (Inner ring) keresztül alállomásokkal (másnéven HUB) van összekötve. A HUB-okhoz szintént optikai kábelen keresztül több optikai node csatlakozik (sub ring). Az optikai node-októl immáron a koax kábelen keresztül érkezik a lakásokba az IP szolgáltatás.



23. ábra: HFC hálózat felépítése

A UPC Magyarország IP hálózata látható a 24. ábrán.



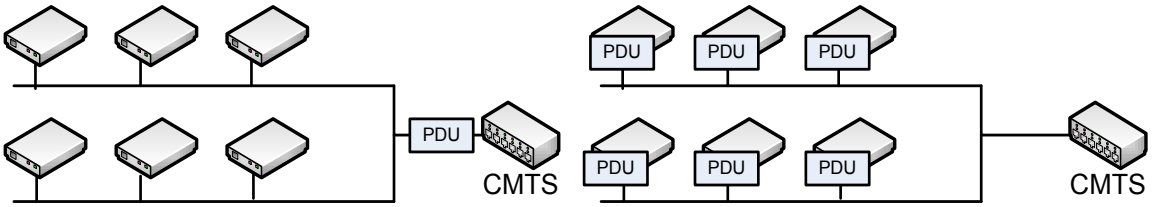
24. ábra: UPC Magyarország hálózatának felépítése

A számítógép RJ 45-ös vagy USB csatlakozón keresztül kapcsolódik a kábelmodemre. A kábelmodem koaxiális kábelen (majd közvetve az optikai node után fényvezető kábelen keresztül) a CMTS –el (Cable Modem Termination System) áll összeköttetésben. A CMTS (városonként illetve Budapesten kerületenként több) a városi routereken keresztül kapcsolódik optikai kábeleken keresztül (az úgynevezett felhordó hálózaton keresztül) a Budapesten lévő Vidéki és Budapesti gyűjtő routerekre - melyek csatlakoznak a BIX illetve az AORTA (a UPC nemzetközi irányú hálózata) routereire - és a UPC belső hálózatára. Ez a belső hálózat (UPC Technology) foglalja magában az IP címeket biztosító DHCP szervereket, DNS szervereket, a modemeknek a konfigurációs beállításokat szolgáltató TFTP szervereket, a VoIP szolgáltatást biztosító telefonközpontot (softswitch), a levelezést biztosító eszközöket és a felhasználók azonosítását végző Ldap szervereket.

6.2. A kábelmodem és a fejállomás (CMTS) kommunikációja

A kábelmodem és a CMTS egymással két irányba kommunikál. Egyrészről a kábelmodem adatokat küld a CMTS-nek (ezt nevezik a feltöltési csatornának vagy vissziránynak) másrészről a CMTS adatokat küld a kábelmodemnek (ezt az úgynevezett letöltési csatornak vagy másnéven előre irány). A kommunikáció mikéntje teljesen eltérő a két irányban:

A letöltési irányban egyetlen adó van, a CMTS. Így semmiféle ütközés figyelésre nincsen szükség. Amennyiben a CMTS felrak egy adatcsomagot a hálózatra, azt minden kábelmodem meg fogja kapni (25. ábra). Természetesen az Ethernethez hasonlóan csak az a kábelmodem fogja feldolgozni, amelyiknek szól az adott csomag (broadcast üzenet esetén ez az összes modem).



25. ábra: előre irányú kommunikáció a kábelTV rendszerben.

Természetesen így minden velünk azonos kábelágon lévő kábelmodem megkapja az összes olyan csomagot amelyet a CMTS a mi kábelmodemünknek küld. Azért, hogy a lehallgatást megakadályozzák, a CMTS és a kábelmodem a teljes kommunikációját titkosítja.

A fő probléma a feltöltési irány. Itt úgy kell megoldani a kommunikációt, hogy gyakorlatilag az összes koax ágon lévő kábelmodem zavarhatja egymást méghozzá oly módon, hogy ezt nem is veszik észre a kábelmodemek. Hiszen ha egy kábelmodem elkezd adni, ezek a csomagok csak a CMTS-hez jut el. Így egy másik kábelmodemnek esélye sincs észrevenni hogy éppen adnak a feltöltési csatornán.

Ez a probléma nagyban hasonlít a korai ALOHANET rendszeren fellépő problémával: hogyan vegyük észre az ütközést, ha nem halljuk a másikat?

A megoldást is az ALOHA-tól vették: a binárisan exponenciális várakozási idő eloszlású, réselt ALOHA algoritmusát használták fel. A feltöltési csatornát az időben miniszeletekre bontjuk. A felfele irányba küldött csomagnak bele kell férnie egy vagy több egymást követő szeletbe.

Azonban ezt a miniszeletet a kábelmodemnek igényelnie kell a CMTS-től. Ezt az igényét az inicializáláskor megkapott MAP csomagban meghatározott időpillanatban jelezheti. Amennyiben a modem küld egy igénylést a CMTS-nek, de nem kap rá nyugtát, az azt jelenti, hogy ütközés történt. Ekkor a modem egy véletlen ideig vár, mielőtt újra próbálkozna (ez a rész az, ami az ALOHA rendszertől vettek kölcsön).

Hogy jobban megérthessük, tekintsük át, hogyan történik a szinkronizáció a modem bekapcsolásánál:

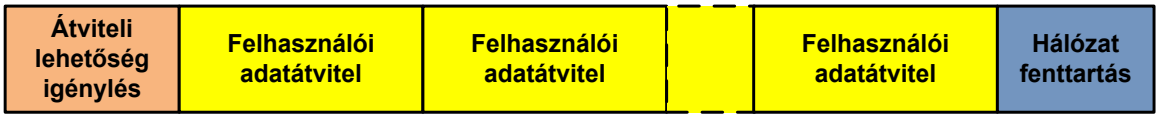
1. A CMTS folyamatosan sugározza a MAP csomagokat melyek tartalmazza a visszirányú csatorna időréseinek kiosztását és a visszirányú csatorna fizikai paramétereit.

2. A kábelmodem a bekapcsolás utáni első ilyen csomagot észlelve, a fizikai paraméterek alapján meghatározott modulációval, frekvencián, a – szintén a MAP csomag által kiosztott – megfelelő időrésben elküldi egy RNG-REQ (Ranging Request) csomagot.
3. A CMTS ezen csomag alapján utasítja a kábelmodemet adási szintjének csökkentésére vagy növelésére szükség szerint.
4. A kábelmodem újabb RNG-REQ csomagot küld, ezt ismételgetik míg a CMTS megfelelőnek találja az adási szintet.
5. A CMTS küld a modemnek egy RNG-CMP csomagot (Ranging Complette), kioszt egy ideiglenes SID azonosítót a modemnek. Ezzel végetért az úgynevezett ranging folyamat. Innentől a regisztrációval folytatja.
6. A kábelmodem DHCP kérésen keresztül kéri az IP címét, a letöltendő konfigurációs file nevét, TFTP szerver címét.
7. Miután ezeket az adatokat megkapta, a kábelmodem valamilyen rendszeridő szerverrel szinkronizálja saját óráját.
8. A modem letölti a konfigurációs beállításait a TFTP szerverről, ezeket a beállításokat végrehajtja majd kéri a CMTS-t a végleges regisztrációra.
9. A CMTS elküldi a kábelmodemnek a végleges SID számot.

Ezzel a kábelmodem feljelentkezett a hálózatra, ezután a modem ethernet csatlakozójára kapcsolt számítógép le tudja kérni a saját IP címét. Amennyiben a számítógép egy – a szolgáltató rendszerében - regisztrált fizika cím, a gép egy publikus IP címet kap. Ellenkező esetben regisztrációra van szükség, tehát a számítógép egy olyan IP címet fog kapni mellyel csak a regisztrációra van lehetőség.

A kábelmodem a szintezés és regisztráció után figyeli az CMTS felől érkező csomagokat. Amennyiben az neki vagy a mögötte lévő számítógépnek szól, azt feldolgozza. Ha adatokat akar küldeni, akkor egy a visszirányú időrés leíró és kiosztó (MAP) üzenetben meghatározott időpillanatban kérhet átviteli lehetőséget. A következő MAP tartalmazni fogja azt, hogy a modemnek mikor és mennyi lehetősége lesz adatokat küldeni. A visszirányú időréseket az előreirányú adatok között átvitt MAP üzenetben rendeljük a modemekhez. Minden visszirányban elküldött csomag tartalmazza a SID számot.

Szintén a MAP üzenet tartalmazza a visszirányú csatorna fizikai paramétereit, így lehetőséget teremt a visszirány paraméterinek dinamikus változtatására (frekvencia, moduláció, sávszélesség,…). Az 26. ábrán látható, egy úgynevezett hálózat fenntartási időrés. Az új modemek ebben az időszeletben tudnak bejelentkezni a rendszerbe, itt az időrésben zajlik a Ranging folyamat is.



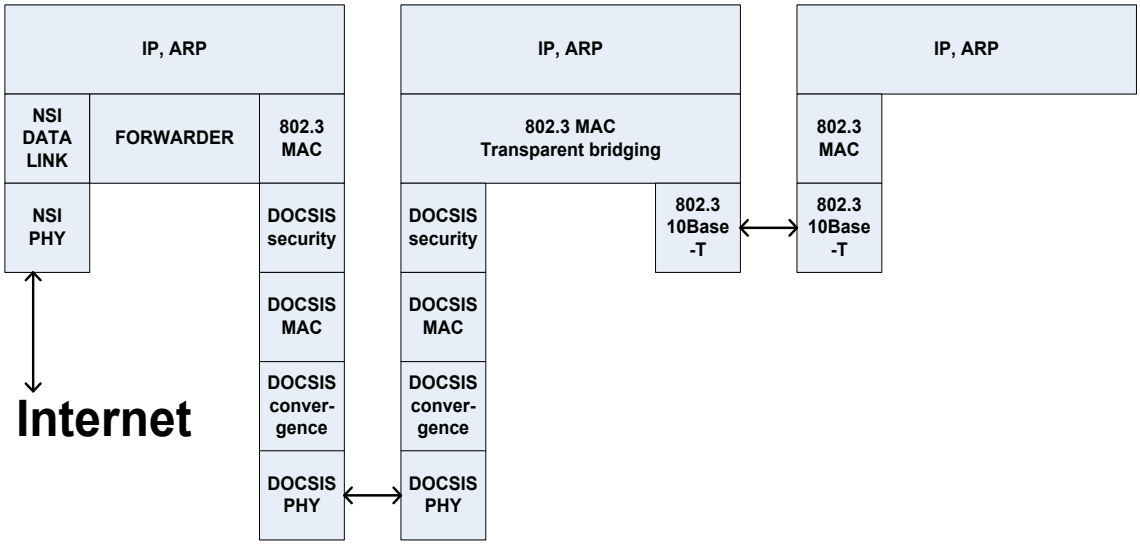
26. ábra: Egy MAP ciklus

Az újabb CMTS rendszerek már az egyes portjai között terheléskiegyenlítést is végez. Ebből kifolyólag kérheti a kábelmodemet hogy inicializálja magát, majd a CMTS az újabb feljelentkezéskor már egy másik frekvencia sávot oszt ki a kábelmodemnek. A UPC hálózaton ez a terheléskiegyenlítés éjszaka történik, mivel ilyenkor rövid időre (néhány másodperc) megszakad az internet szolgáltatás (arra az időre míg a kábelmodem újra inicializálja a kapcsolatot).

6.3. A DOCSIS protokoll architektúrája

A kábelmodem a rá csatlakoztatott személyi számítógéppel egy ethernet csatlakozón keresztül kommunikál. Viszont a CMTS-el való kommunikációra a DOCSIS protokolkészletét használja (27. ábra). A kábelmodem miután átvesz egy ethernet keretet a számítógéptől, transzparens hídként viselkedik. Viszont mielőtt a keretet elküldené a CMTS-nek, elhelyezi az ethernet keret elé a DOCSIS MAC réteg keretét (*28. ábra*).

Ha kap egy csomagot a CMTS-től az előre irányú csatornán, ellenőrzi, hogy neki szól-e a csomag (a csomag SID mezője), ha neki szól, a DOCSIS security réteg visszafejti a titkosítást majd amennyiben egy neki szóló management csomagról van szó, feldolgozza azt. Ha nem, akkor transzparens hídként az ethernet portján helyezi el.

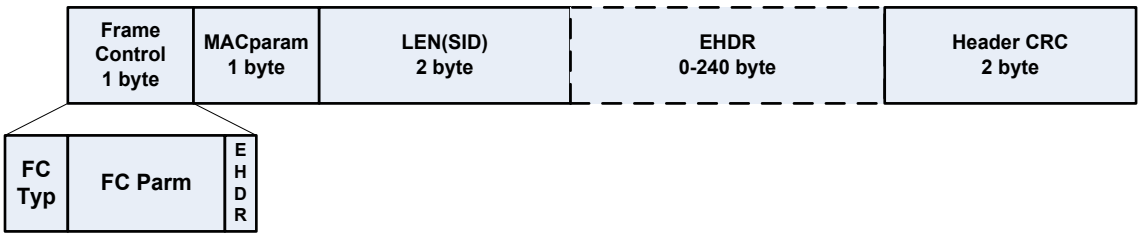


27. ábra: DOCSIS 2.0 Protocol stack

6.4. A DOCSIS protokoll közeg-hozzáférési mechanizmusa

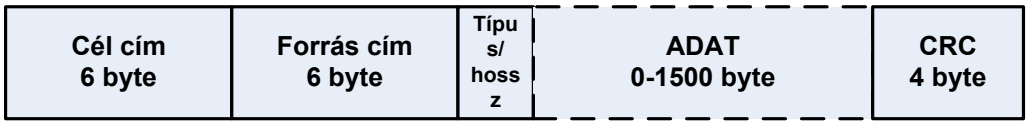
Amennyiben a “Frame Control” mezőben lévő EHDR mező értéke 1, a MACparam mező az EHDR mező hosszát mutatja meg, ellenkező esetben vagy a MAC sorszámát, vagy a kért MINISLOT-ok számát tartalmazza. A LEN vagy SID mező vagy a MAC keret hosszát, vagy (amennyiben ez a modemtől a CMTS-hez küldött ’Request’ csomag) a SID (Service ID) számát. Az SID segítségével azonosítja tulajdonképpen a CMTS a kábelmodemet. Az EHDR mező további opciókat tartalmazhat, ennek hossza 0 és 240 byte között lehet. A fejléc utolsó mezője a fejléc hibaellenőrzésére szolgál.

A fejléc után következik a keret adatrésze mely lehet felhasználói keret (29. ábra) vagy management keret (30. ábra). Felhasználói adat esetén ez tulajdonképpen megegyezik az Ethernet keret felépítésével.

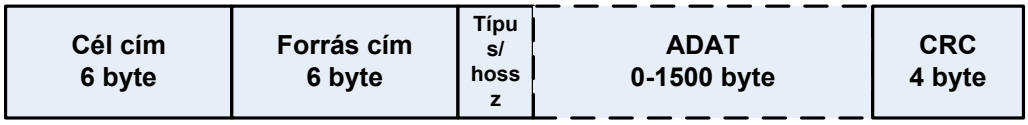


28. ábra: DOCSIS MAC fejléce

A cél cím és a forrás cím iránytól függően egy kábelmodem fizikai cím, egy számítógép mögötti eszköz címe, egy CMTS port cím vagy (ha CMTS által küldött menedzsment csomagról van szó) broadcast cím lehet.



29. ábra: Felhasználói adat



30. ábra: DOCSIS Management keret

Menedzsment csomag esetén a verzió és type (VERS és TYPE) együttesen adja meg a keret típusát. A keret többek között lehet idő szinkronizációs keret, felfele irányt kijelölő csomag, a modem és a CMTS összeszinkronizálását segítő csomag, a modem által küldött felfele irány igénylő csomag.

7. A hálózati forgalom megfigyelése

Manapság már az operációs rendszerek szinte teljesen elrejtik a hálózati kapcsolat részleteit. Mikor meghallgatunk egy netes rádiót, megnézzük a leveleinket vagy csupán meglátogatjuk kedvenc web oldalunkat, az operációs rendszer és a felhasználói program (zenelejátszó, e-mail kliens vagy böngésző program) szinte a teljes hálózati tevékenységet a háttérben lekezeli. Mi csak a végeredményt (a zenét, beérkező leveleket vagy a weboldalt) kapjuk „kézhez”.

De természetesen van lehetőség arra, hogy bővebben informálódjunk…mi is történik a számítógépen ilyenkor. Néhány ilyen információ forrás az operációs rendszerben, míg mások külső felhasználói programok.

7.1. Ifconfig parancs

Beépített programok közül a legalapvetőbb információkat az `ipconfig` (unix alapúnál `ifconfig`) parancs szolgáltat. Megadja a számítógépen lévő hálózati csatlakozási pontok számát, annak nevét, alapvető adatait (tcp/ip protokoll esetén az ip címet, alhálózati maszkot, ethernet esetén az ethernet fizikai címét).

7.2. ARP parancs

Az ARP tábla (azaz ip cím és fizikai cím összerendelések) tartalmát jeleníti meg az `arp` parancs. Ezzel a paranccsal törölhetünk is tábla bejegyzéseket, így a törölt sorban szereplő ip címhez tartozó fizikai címet a gép újra le fogja kérni (egy korábban már tárgyal speciális ethernet csomaggal).

7.3. A route print parancs

A csomagirányításról kérhetünk le némi információt a `route print` parancs használatával. Ezzel megtudhatjuk hogy egy adott hálózatot melyik átjárón keresztül érjük el illetve melyik gép az alapértelmezett átjáró, azaz ha a táblában nem szerepel a címzett hálózat akkor annak a gépnek adjuk át a csomagot (mivel valószínűleg ő bővebb információkkal rendelkezik a csomag továbbításáról).

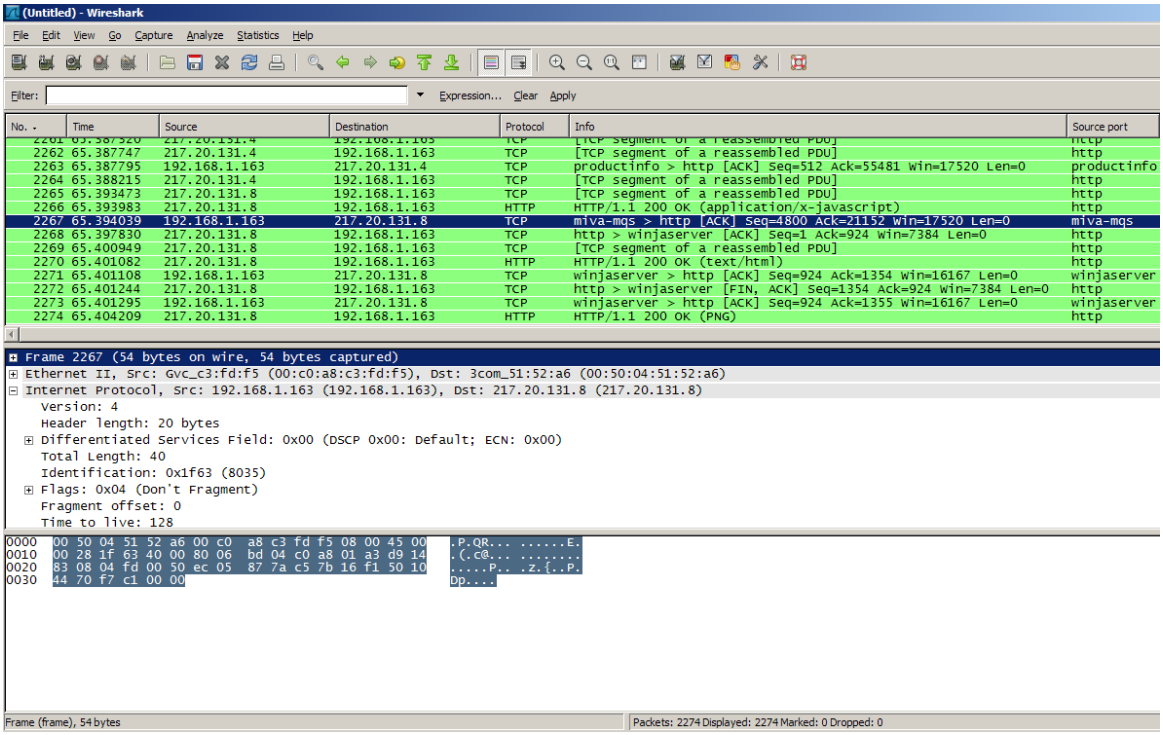
7.4. A ping és tracert parancsok

Két hálózati diagnosztizáló program a `ping` és a `tracert` parancs. Segítségükkel speciális (úgynevezett ECHO-REQUEST) csomagot küldhetünk egy távoli állomásra. Jó cím, működő hálózat (és amennyiben ez a szolgáltatás nincs letiltva a távoli gépen) esetén az állomás visszaküld egy ECHO-REPLY csomagot mely alapján a parancs kimenetén láthatjuk a megtett út idejét. A `tracert` paranccsal ezenfelül még a megtett utat is megláthatjuk, azaz hogy a távoli állomásig milyen köztes állomásokon haladt át a csomag.

7.5. A sniffer programok

Külső hálózati diagnosztizáló program igen sok van. Ezen programok egyik fajtája az úgynevezett SNIFFER programok, melyek rögzítenek minden kimenő és bejövő ethernet keretet és annak tartalmát. Ez alapján tudhatunk meg a legtöbbet egy adott hálózat forgalmáról / működéséről.

Ezen programok egyike az úgynevezett Wireshark nevű program (31. ábra) mely ingyenesen letölthető a világhálóról.



31. ábra: Wireshark program képernyője

A program segítségével megtekinthetjük egy hálózati kártyán áthaladó ethernet kereteket. A program rögzíti a keret feladó és cél címét, méretét és tartalmát. Kiolvashatjuk a benne szereplő protokollt (IP, TCP, UDP, ARP), az IP csomagot jellemző adatokat, TCP, UDP szegmenst jellemző adatokat. Természetesen a program nem csak ezeket a protokollokat ismeri, de egy Soho hálózatban ezek a leggyakrabban előforduló protokollok.

A programban tudunk bizonyos szempontok alapján szűrni illetve korlátozottan de lehetőségünk van bizonyos statisztikák készítésére is.

A megfigyelés

A hálózati forgalom megfigyelése két fő részre osztható. Az első fázis a rögzítés. Ezt a hálózat átjáró gépén lévő tcpdump nevű program végzi. Ő egy szövegfile-ba rögzíti folyamatosan a gép belső hálózati kártyáján áthaladó adatcsomagok bizonyos paramétereit. Egy ilyen rögzítés egy sora látható az alábbiakban:

1193481338.817930 IP 192.168.1.25.22 > 192.168.1.163.1052: tcp 116

Mint látható, a program rögzíti a csomag áthaladásának időbélyegét (első oszlop), a csomag hálózati rétege szerinti protokollját (második oszlop). Ezután következik a forrás ip cím és forrás port majd a cél ip cím és cél port. Ezután a szállítási réteg szerinti protokoll következik, végül a csomag mérete. Ennyi adatból megfelelő eszközökkel igen részletes statisztikai adatokat tudunk kinyerni.

A tcpdump program egy egyszerű bash script-el van indítva, mely biztosítja, hogy az állomány automatikusan nevet kapjon (méghozzá a mérés indításának dátumát + a

méréshez használt hálózati kártya nevét). A mérési adatokat tartalmazó állomány a hálózati meghajtó egy könyvtárába kerül későbbi feldolgozás céljából. A használt bash script az alábbi:

```
#!/bin/bash
tcpdump -i $1 -tt -q -n > /home/samba/$(date +%y%m%d_%H%M%S)_$1
```

Második fázisként egy általam Java nyelven írt program beolvassa ezt a szövegfile-t, feltöli egy adatbázisba és különböző statisztikai adatokat ad meg nekünk.

8. Az elemző program bemutatása

A program, amely beolvassa a tcpdump által rögzített adatokat, egy Java nyelven írt program. Ez a program soronként értelmezi a szöveg állományt, majd egy postgresql adatbázisban rögzíti a sorokat. Azért szükséges az adatbázis használata mert ezután sql lekérdezési utasításokkal és tárolt eljárásokkal olyan adatokhoz juthatunk hozzá, melyhez csak sokkal több kódolással tudnánk megszerezni ha az adatokat a memóriában tárolnánk.

A postgresql adatázis táblája és tárol eljárásai a következők:

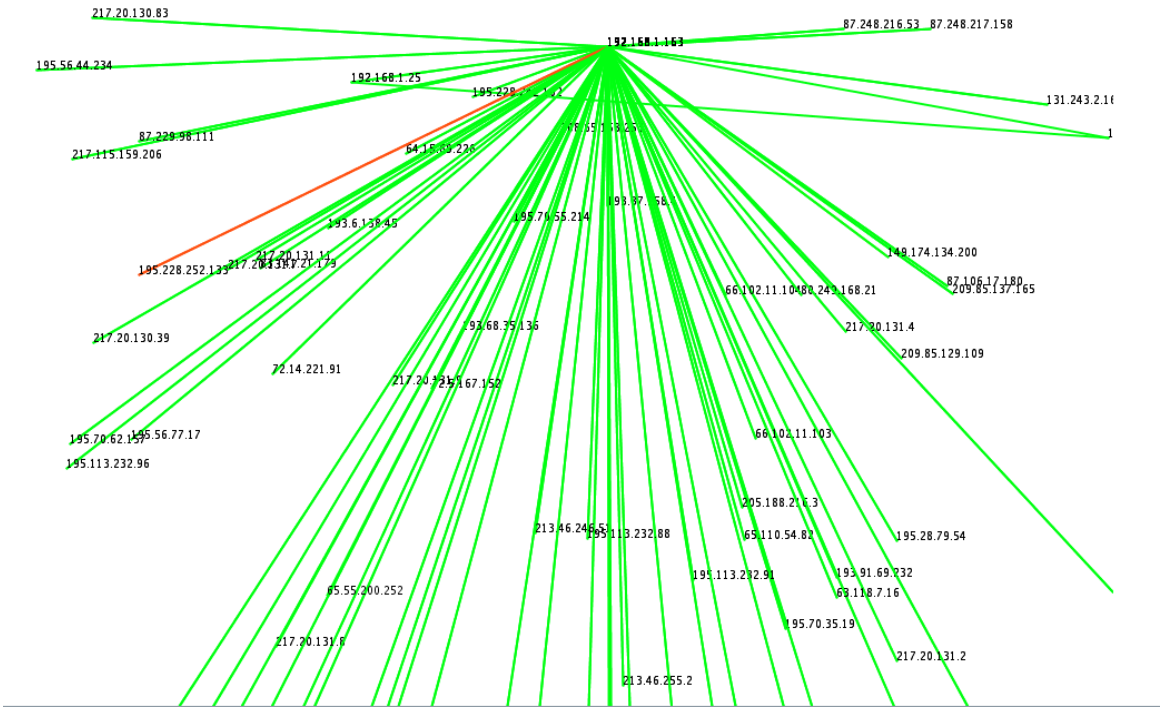
```
--
-- PostgreSQL database dump
--

--
-- TOC entry 1612 (class 1262 OID 16385)
-- Name: HalozatCsomag; Type: DATABASE; Schema: -; Owner: csomagelemzo
--

CREATE DATABASE "HalozatCsomag" WITH TEMPLATE = template0 ENCODING = 'UTF8';

ALTER DATABASE "HalozatCsomag" OWNER TO csomagelemzo;

--
-- TOC entry 19 (class 1255 OID 16410)
-- Dependencies: 4
-- Name: portforgalomtablafeltolt(); Type: FUNCTION; Schema: public; Owner:
-- postgres
--

CREATE FUNCTION portforgalomtablafeltolt() RETURNS void
    AS $$
       insert into portForgalomTabla
              select  date_trunc('minute',  idobelyeg),  forrasport,  sum(meret),
              protokol from csomagok
              where forrasport < 1024
              group by date_trunc('minute', idobelyeg), protokol, forrasport
              order by date_trunc('minute', idobelyeg);

       insert into portForgalomTabla
              select   date_trunc('minute',   idobelyeg),   celport,   sum(meret),
              protokol from csomagok
              where celport < 1024
              group by date_trunc('minute', idobelyeg), protokol, celport
              order by date_trunc('minute', idobelyeg);
       $$
    LANGUAGE sql;


ALTER FUNCTION public.portforgalomtablafeltolt() OWNER TO postgres;
```

```
--
-- TOC entry 20 (class 1255 OID 16412)
-- Dependencies: 4
-- Name: portforgalomtablatorol(); Type: FUNCTION; Schema: public; Owner:
-- postgres
--

CREATE FUNCTION portforgalomtablatorol() RETURNS void
    AS $$delete from portforgalomtabla$$
    LANGUAGE sql;

ALTER FUNCTION public.portforgalomtablatorol() OWNER TO postgres;

--
-- TOC entry 1273 (class 1259 OID 16394)
-- Dependencies: 4
-- Name: csomagok; Type: TABLE; Schema: public; Owner: csomagelemzo; Tablespace:
--

CREATE TABLE csomagok (
    idobelyeg timestamp without time zone NOT NULL,
    forrasip character varying(20) NOT NULL,
    celip character varying(20) NOT NULL,
    forrasport bigint NOT NULL,
    celport bigint NOT NULL,
    meret bigint NOT NULL,
    iphashosszeg bigint NOT NULL,
    iphashszorzat bigint NOT NULL,
    porthashosszeg bigint NOT NULL,
    porthashszorzat bigint NOT NULL,
    ssz bigint NOT NULL,
    protokol smallint
);

ALTER TABLE public.csomagok OWNER TO csomagelemzo;

--
-- TOC entry 1274 (class 1259 OID 16404)
-- Dependencies: 4
-- Name: portforgalomtabla; Type: TABLE; Schema: public; Owner: csomagelemzo;
-- Tablespace:
--

CREATE TABLE portforgalomtabla (
    idobelyeg timestamp without time zone NOT NULL,
    port bigint NOT NULL,
    mennyiseg bigint NOT NULL,
    protokol smallint
);


ALTER TABLE public.portforgalomtabla OWNER TO csomagelemzo;

--
-- TOC entry 1271 (class 1259 OID 16386)
-- Dependencies: 4
-- Name: sszSequence; Type: SEQUENCE; Schema: public; Owner: csomagelemzo
--

CREATE SEQUENCE "sszSequence"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;
```

```
ALTER TABLE public."sszSequence" OWNER TO csomagelemzo;

--
-- TOC entry 1617 (class 0 OID 0)
-- Dependencies: 1271
-- Name: sszSequence; Type: SEQUENCE SET; Schema: public; Owner: csomagelemzo
--

SELECT pg_catalog.setval('"sszSequence"', 1, false);

--
-- TOC entry 1605 (class 2604 OID 16396)
-- Dependencies: 1273 1272 1273
-- Name: ssz; Type: DEFAULT; Schema: public; Owner: csomagelemzo
--

ALTER TABLE csomagok ALTER COLUMN ssz SET DEFAULT
nextval('csomagok_ssz_seq'::regclass);

--
-- TOC entry 1607 (class 2606 OID 16398)
-- Dependencies: 1273 1273
-- Name: prim_key; Type: CONSTRAINT; Schema: public; Owner: csomagelemzo;
-- Tablespace:
--

ALTER TABLE ONLY csomagok
    ADD CONSTRAINT prim_key PRIMARY KEY (ssz);

-- Completed on 2008-02-23 22:55:05 CET

--
-- PostgreSQL database dump complete
--
```

A program két tárolt eljárást használ: az egyik a `portforgalomtablafeltolt` mely feltölt egy ideiglenes táblát. A másik tárolt eljárás a `portforgalomtablatorol` mely szükség esetén törli ezt az ideiglenes táblát. A táblában a portonként csoportosított összforgalom található. Először forrás portonként majd cél portonként. A program csak az 1024-nél kisebb portokat (azaz az úgynevezett Well-knonw-portokat használja.

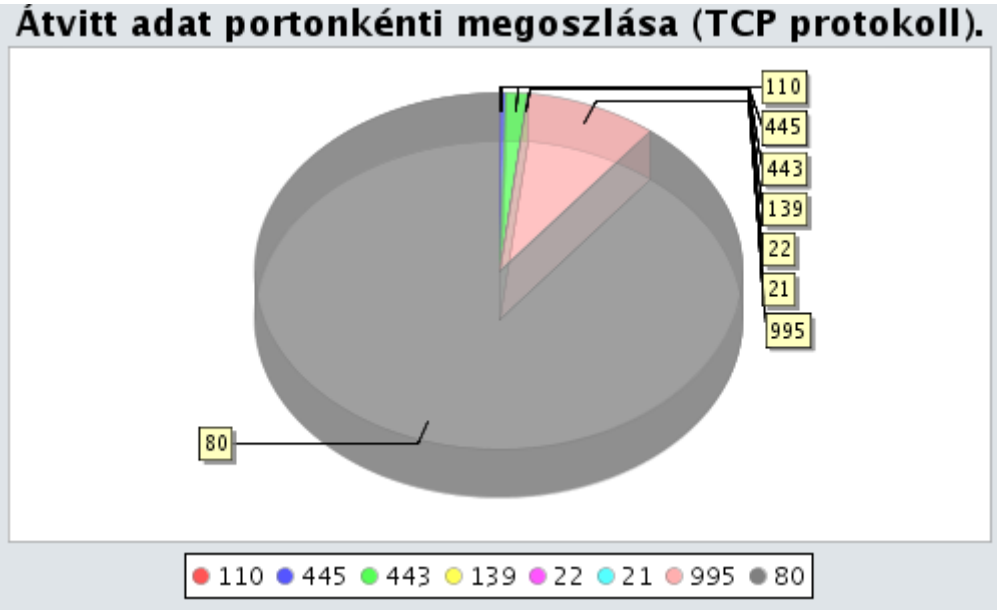
Az adatbázisba való feltöltés után a program megkezdi működését. Épít egy úgynevezett ip térképet. Ezen a mintavétel során észlelt ip címek találhatók. Ha két ip cím között volt forgalom, a két ip címet a térképen összeköti vonallal. A vonalak aszerint vannak színezve, hogy az összforgalom hány százaléka haladt át ezen a vonalon.



32. ábra: A program által kirajzolt IP térkép

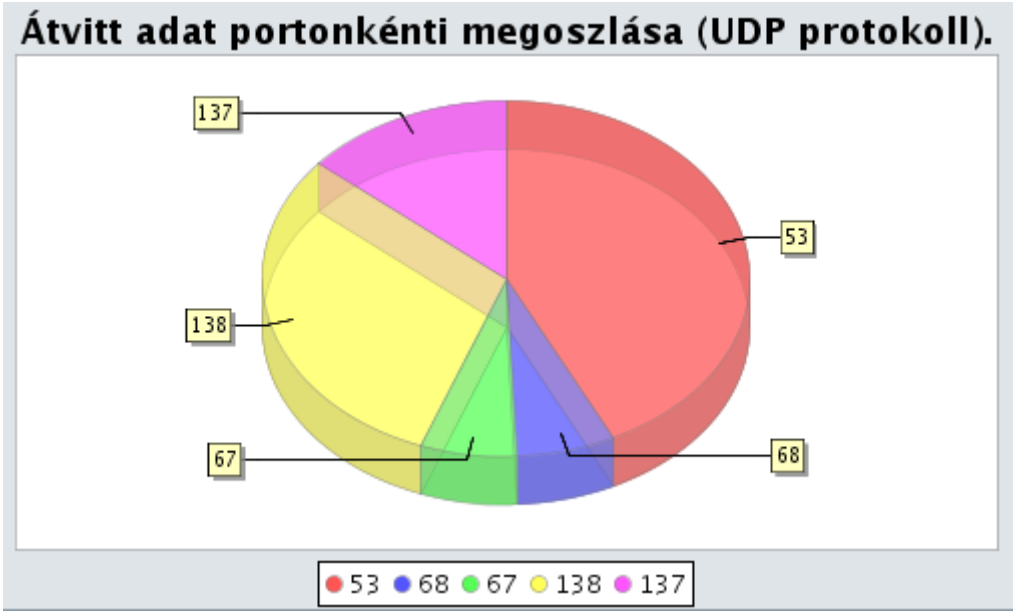
A következő füleken néhány grafikon található, például: A forgalom megoszlása célportok között, ugyanilyen bontásban de protokollonként külön grafikonban, a hálózat forgalma percenkénti bontásban.

Az alábbi képernyőkép egy teszt mérés eredményét ábrázolja. Itt látszik, hogy TCP protokollokból főként a 80-as porton ment át forgalom.



33. ábra: Átvitt adat TCP portonként

Az UDP protokollt érintő forgalmat tekintve az 53-as porton ment át a fő forgalma (amelyet DNS lekérdezésre használunk).



34. ábra: Átvitt adat UDP portonként

A program megad néhány táblázatot melyben a legtöbbet forgalmazott ip címek, az összforgalomhoz képest a különböző ip címek közötti százalékos megoszlást mutatja meg.
Az alábbi képen látszik, hogy ha a forgalom célja szerint csoportosítjuk a forgalmat, akkor a legkisebb forgalom a 213.46.246.52-es ip címre továbbítódott a legevesebb forgalom (132 byte), a sor elején a 192.168.1.163-as ip van (a képen már nem látszik).

celip	forgalom
213.46.246.52	132
149.174.134.200	144
195.228.252.133	148
195.113.232.80	171
193.6.138.45	188
209.85.137.125	220
213.46.255.2	276
195.56.77.17	379
205.188.216.3	405
193.37.158.4	417
195.113.232.96	418
209.221.153.6	427
212.92.23.56	427
66.150.96.119	429
195.113.232.91	477
63.118.7.35	478

35. ábra: Forgalom cél ipcímenként csoportosítva

Ugyanilyen táblázatot készít a program cél ip szerinti csoportosítás helyett forrás ip cím szerinti csoportosításban (azaz megmutatja hogy melyik ip címről érkezett a legtöbb/legkevesebb adat).

A következő táblázat a kapcsolatok közötti mennyiségeket írja ki, azaz: veszi az összes lehetséges kapcsolatot (192.168.1.163 és 213.46.246.52 közötti kapcsolat, 192.168.1.163 és 66.102.11.103 közötti kapcsolat, stb) és az ezen a kapcsolaton átmenő forgalmakat teszi táblázatba.

celip	forrasip	totalme
192.168.1.163	213.46.246.52	264
192.168.1.163	195.113.232.80	460
192.168.1.163	87.106.19.146	722
192.168.1.163	209.221.153.6	732
192.168.1.163	195.113.232.91	819
192.168.1.163	193.6.138.45	820
192.168.1.163	193.37.158.4	833
192.168.1.163	72.5.167.152	946
192.168.1.163	63.118.7.35	984
192.168.1.163	87.248.216.53	1064
192.168.1.163	66.102.11.103	1121
192.168.1.163	213.46.255.2	1236
192.168.1.163	66.150.96.119	1262
192.168.1.163	66.102.9.147	1271
192.168.1.163	209.85.137.125	1281

36. ábra: Forgalom megoszlás cél és forrás ip címenként

A program teljes forráskódja az ’A’ mellékletben található. A program két külső gyártó álltal készített osztály könyvtárt használ:

- A működés közben esetleges hibákat vagy információkat az Apache log4j [10] nevű osztály könyvtára segítségével logolja ki megfelelő állományba: http://logging.apache.org/log4j/1.2/index.html
- A programban található minden grafikon a JFreeChart nevű grafikon készítő osztálykönyvtár segítségével történt. Ezeket a program burkolóosztályokon (csomagelemzo.util.abra csomagban lévő osztályok) keresztül éri el: http://www.jfree.org/jfreechart/

Az elemző program Java SWING technológiát használ a felület előállítására. A program osztály felépítése a 37. ábrán látható. Az események vezérlője a GUI felület, ahol az egyes gombokhoz tartozó eseménykezelők vezérlik a különböző komponenseket.

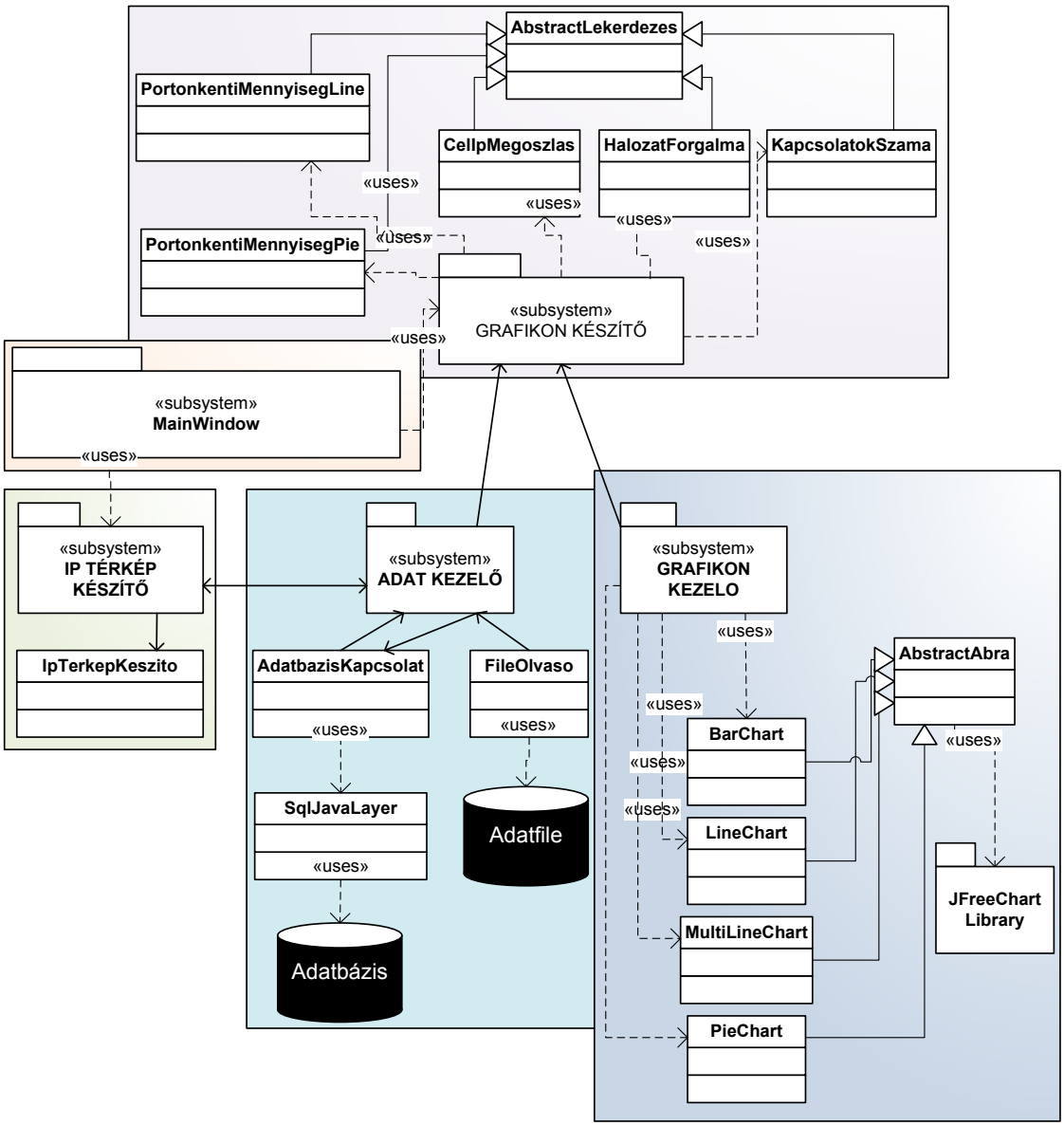
A program ADATKEZELŐ alrendszere végzi az adatfile beolvasását (soronként) és annak tartalmát az adatbázisba is ő tölti fel. Ezenkívül szolgáltatásokat nyújt a két másik alrendszernek (a GRAFIKON KÉSZÍTŐ és az IP TÉRKÉP KÉSZÍTŐ alrendszereknek). Ő látja el a két alrendszert a szükséges adatokkal, hogy azok ezen adatokból el tudják végezni saját feladatukat.

A GRAFIKON KÉSZÍTŐ alrendszer és a JFreeChart grafikon készítő library-ja közzé beékelődik egy GRAFIKON KEZELŐ alrendszer, mely tulajdonképen a JFreeChart library

kezelését könnyíti meg. Itt a használt grafikon típusait megvalósító osztályok találhatók, mindegyik egyetlen szülőosztálytól származik, az ABSTRACT GRAFIKON osztálytól.

A GRAFIKON KÉSZÍTŐ alrendszer az ADATKEZELŐ alrendszeren keresztül lekéri a szükséges adatokat, majd a GRAFIKON KEZELŐ alrendszer szolgáltatásai segítségével elkészíti a kért grafikont melyet visszaad a GUI kezelő számára (aki a megfelelő helyre illeszti a képet).

Az IPTÉRKÉP KÉSZÍTŐ alrendszer szintén az ADATKEZELŐ alrendszert használja a szükséges adatok megszerzéséhez, majd elemzi a kapcsolatokat, elkészíti a szükséges ábrát és szintén a GUI kezelő alrendszerre bízza a kép megjelenítését.



37. ábra: A csomagelemző program felépítése

9. A mintavételezés

A mintavételezés – azaz az sniffer program futtatása – egy héten keresztül történt, 2008. március 28.-a 20 óra és 2008. április 05. 9 óra között. Az időszak alatt a belső hálózaton elhelyezkedő mindhárom gép csak időszakosan volt használva, mindhárom gépet a tulajdonosa használta.

A chucknorris nevű gépet főként fejlesztésre, internetezésre, webes rádió hallgatására, emellett online játék, p2p letöltés és azonnali üzenet küldésre, a columbo nevű gépet szinte kizárólag csak internetezésre, míg a mcgiver nevű gépet internetezésre, azonnali üzenet küldésre használták.

A mintavételező gép folyamatosan ment, de természetesen mikor a három gép közül egyik sem futott, nem észlelt forgalmat. A sniffer program az átjáró belső oldali hálózati kártyáján áthaladó forgalmat figyelte.

Az egy hetes mintavételi intervalum alatt 191,2 Mbyte méretű adatfájl keletkezett, melyet a program megközelítőleg 9 óra alatt dolgozott fel. Ennek a feldolgozási időnek a 17%-a a file-ból a memóriába való beolvasás, 51% a memóriából az adatbázisba való mentés és végül 32%-a a szükséges lekérdezések végrehajtása, a kapott adatok elemzése és a szükséges ábrák elkészítése tette ki.

A mintavétel során 3 adatfájlt kaptunk, egyénként mintegy 750.000 sorral. Egy-egy ilyen file-t a program egyben nem tudott feldolgozni, 700.000 sor után `java.lang.OutOfMemoryError: Java heap space` kivétel váltódott ki a számítógépen (ami egy dual magos 64 bites processzorral és 2 Gbyte memóriával rendelkezett). Ezért a fájlokat split parancs segítségével 350.000 soronként daraboltuk. Így már a program probléma nélkül fel tudta dolgozni az adatokat, részfájlonként kb. 50 perc alatt. A feldolgozás során átlagosan 75-100%-os processzor terheltséget és körülbelül 520 Megabájt memória foglalást okozott a program (38. ábra).

A mintavétel során összesen az adatfájlokba 2.885.257 darab bejegyzés került, melyből a program 2.873.819 darab bejegyzést tudott feldolgozni (azaz a programnak sikerült feldolgozni a bejegyzések 99,6%-át). A mintavétel során nem kerültek feldolgozásra az ARP csomagok, Ipv6 csomagok és egyéb speciális csomagok:

```
1206783437.168974 IP6 :: > ff02::16: HBH ICMP6, multicast listener
    report v2, 1  group record(s), length 28
1206783438.000957 IP6 :: > ff02::1:ffc3:fdf5: ICMP6, neighbor
    solicitation, who  has fe80::2c0:a8ff:fec3:fdf5, length 24
```

38. ábra: A rendszer terheltsége az adatok feldolgozása közben.

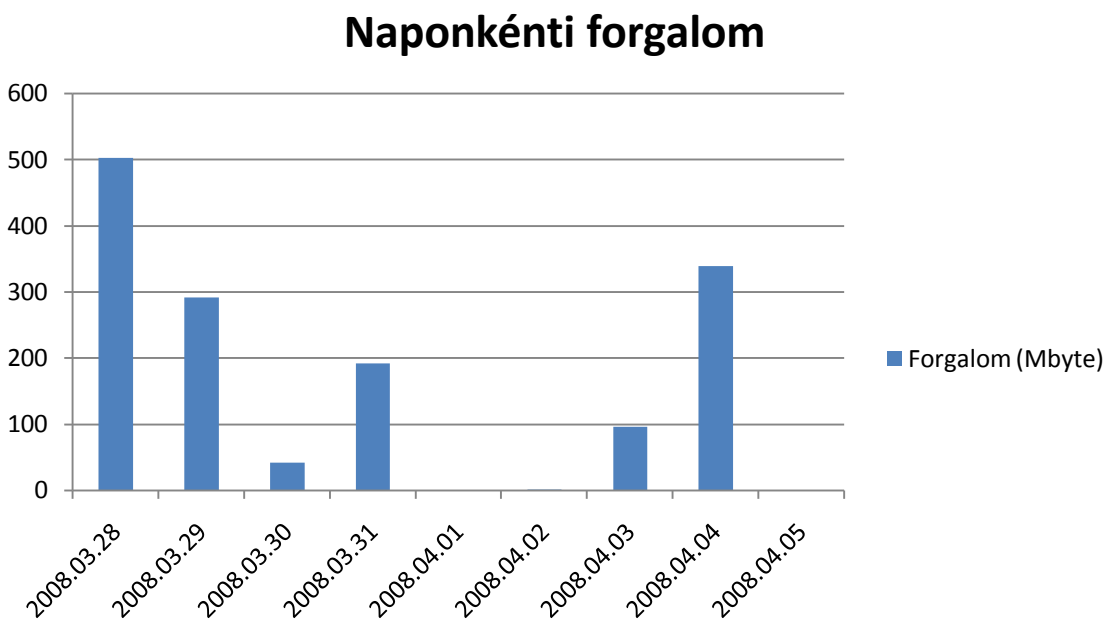
10. Az elemzés eredményei

Az egy hetes mintavétel alatt a SOHO hálózat átjáróján 1,465 Gbyte adat haladt át összesen 2.873.819 db adatcsomagban (a program csak TCP és UDP csomagokat értelmezett, minden más protokollt (ARP, stb) figyelmen kívül hagyott.

A forgalmazott mennyiség naponkénti felbontása a 39. ábrán, a belőle készített grafikon a 40. ábrán látható.

Napok	Forgalom (Mbyte)
2008.03.28.	502
2008.03.29.	291
2008.03.30.	42
2008.03.31.	192
2008.04.01.	-
2008.04.02.	0,77
2008.04.03.	96
2008.04.04.	339
2008.04.05.	0,03
Összesen	1462,8

39. ábra: A Soho hálózat napinkénti forgalma (táblázat).



40. ábra: A Soho hálózat napinkénti forgalma (diagram).

Mint az ábrán látható, 2008. 04. 01.-én nem volt mérhető forgalom a hálózaton, illetve 2008. 04. 05.-én a mérés csak reggel 9 óráig folyt, azonban ekkor nem volt semmilyen számítógép bekapcsolva, így csak hálózati management forgalom volt:

Dátum	Forrás Ip	Cél ip	Méret
2008-04-05 00:40:21	192.168.1.25:138	192.168.1.255:138	216
2008-04-05 08:00:31	0.0.0.0:68	255.255.255.255:67	300

Illetve a mintavétel leállítása ssh-n keresztül történt, így természetesen ott is generálódott minimális forgalom:

Dátum	Forrás ip	Cél ip	Méret
2008-04-05 08:00:48	192.168.1.188:51112	192.168.1.25:22	752

Az első vizsgált kimutatás megmutatja egy táblázatban (41. ábra), hogy ha tekintjük az összes adatcsomag forrás ip-jét, mely ip címek szerepelnek az első 15 helyen a forgalmazott adatmennyiség szerint.

Mint látható, a legtöbb mennyiséget – 373 Megabájtot a stream001.radio.hu. című hoszt forgalmazta, mely az mr2-petofi rádió streamer szervere. Utána második helyen a Soho hálózat egyik gépe, szerepel. A negyedik gép az on-line játékot kiszolgáló gépe. A többi gép valószínűleg p2p forgalomban résztvevői voltak.

Ip cím	Host név	Mennyiség (Mbyte)	Mennyiség (byte)
89.185.228.113	stream001.radio.hu.	373,574865	391721638
195.113.232.96		209,110487	219268238
195.228.242.102	home-322776.b.astral.ro.	125,452921	131546922
192.168.1.188	cds145.frf.llnw.net.	80,6377487	84554808
85.224.92.234	ftp.freepark.org.	57,5148582	60308700
195.70.36.125	ismeretlen de az index.hu tartományába tartozik	35,4851971	37208926
89.185.228.164	63-246-22-42.contegix.com.	32,5106783	34089917
80.249.168.143	mail.newave.hu.	27,1749964	28495049
63.246.22.42	dex164.exmasters.com.	24,6908979	25890283
217.20.131.247	Img.sanomabp.hu.	24,0356569	25203213
195.228.252.133	ua-85-224-92-234.cust.bredbandsbolaget.se.	18,729002	19638782
87.248.217.181	cds145.frf.llnw.net	18,1359444	19016916
89.136.169.61	imgs.adverticum.net.	17,8855219	18754329
192.168.1.163	chucknorris	16,0058041	16783302
217.20.138.66	dex113.exmasters.com.	13,6205778	14282211

41. ábra: Forgalmak forrás ip cím szerint

Ha cél ip cím szerint rakjuk sorrendbe a forgalmat (42. ábra), a lista első két helyén ugyan az a gép szerepel, mely a Soho hálózat egyik gépe. A mintavétel során ip cím változás történt, ezért szerepel egyszer a korábban megismert 192.168.1.163 majd a 192.168.1.188 ip címmel is a gép. A listában szerepel néhány p2p forgalomban érintett gép is illetve a a Soho hálózat felhasználói által levelezésre használt gmail.com szerverei is.

Cél ip cím	Host név	Forgalom (Mbyte)	Forgalom (byte)
192.168.1.163	chucknorris	575,4779825	603432401
192.168.1.188	chucknorris	513,0834732	538007016
192.168.1.200	columbo	90,52646732	94923881
192.168.1.104	mcgiver	49,35768127	51755280
89.136.169.61	home-322776.b.astral.ro.	40,57206154	42542890
85.224.92.234	c-ea5ce055.733-1-64736c10.cust.bredbandsbolaget.se.	16,36180782	17156599
80.202.209.68	68.80-202-209.nextgentel.com.	14,36892986	15066915
209.85.137.18	gmail.com	12,31035233	12908340
209.85.137.83	gmail.com	10,91244316	11442526
85.229.68.129	c-8144e555.28-25-64736c10.cust.bredbandsbolaget.se.	7,790739059	8169182
80.77.113.58	ftp.extra.hu	6,812555313	7143482
80.197.48.76	0x50c5304c.alb2nxx10.adsl-dhcp.tele.dk.	6,396857262	6707591
82.45.4.251	82-45-4-251.cable.ubr04.stav.blueyonder.co.uk.	5,890187263	6176309
84.192.53.139	d54C0358B.access.telenet.be.	5,474689484	5740628
84.71.183.111	user-5447b76f.wfd88b.dsl.pol.co.uk.	5,416806221	5679933

42. ábra: Forgalmak cél ip cím szerint

Ha a belső hálózat ip címei és a külső hálózat gépei közötti kapcsolatokból készítünk listát (43. ábra), egyrészről látjuk, hogy a Soho hálózat három gépe közül a lista első 15 pozicióján csak az egyik gép szerepel, a többi gép nem végzett számottevő forgalmat. Ezen a gépen belül is főleg a rádió stream szerepelt a fő helyen illetve a levelezés szerepel a listán és p2p forgalom főként.

Soho ip	Külső ip	Külső hostnév	Mbyte	byte
192.168.1.163	**209.85.135.103**	gmail.com	14,1462	14833388
192.168.1.163	**209.85.137.83**	gmail.com	14,1462	14833388
192.168.1.163	**195.113.232.96**		16,0157	16793645
192.168.1.163	**80.202.209.68**	68.80-202-209.nextgentel.com.	22,7072	23810246
192.168.1.200	**195.70.36.125**	Img.sanomabp.hu.	24,0732	25242583
192.168.1.188	**89.185.228.164**	dex164.exmasters.com.	24,7048	25904884
192.168.1.188	**80.249.168.143**	mail.newave.hu.	27,2752	28600172
192.168.1.188	**217.20.131.247**	nem található de index tartománya	31,3961	32921212
192.168.1.188	**63.246.22.42**	63-246-22-42.contegix.com.	32,5115	34090821
192.168.1.163	**85.224.92.234**	c-ea5ce055.733-1-64736c10.cust.bredbandsbolaget.se.	35,0908	36795381
192.168.1.188	**195.228.252.133**	ftp.freepark.org	57,545	60340358
192.168.1.163	**87.248.217.181**	cds145.frf.llnw.net.	80,6883	84607785
192.168.1.163	**217.20.138.66**	stream001.radio.hu.	163,031	170950287
192.168.1.163	**89.136.169.61**	home-322776.b.astral.ro.	166,025	174089812
192.168.1.188	**217.20.138.66**	stream001.radio.hu.	210,548	220776095

43. ábra: Forgalmak kapcsolatok (cél-forrás ip párok) szerint.

A 44. ábrán látható hogy mennyi átvitt adatmennyiség folyt át az 1024-nél (Well-Known porton) keresztül. Mint látható, ha csak a Well-Known portokat tekintjük, kiugró érték a 80 forrásporton keresztül folyt át. Ezenkívül megabájt mennyiség csak a 443-as https porton és a 445-ös smb porton folyt át.

Forrás port	Mennyiség
37	28 byte
995	1,1 kbyte
21	2,55 kbyte
123	4 kbyte
139	6,2 kbyte
137	55 kbyte
22	97 kbyte
67	160 kbyte
68	177 kbyte
554	272 kbyte
138	392 kbyte
53	778 kbyte
445	11 Mbyte
443	27 Mbyte
80	584 Mbyte

44. ábra: Átvitt adatmennyiség az 1024-nél kisebb forrás portokon

A 45. ábrán egy ehhez hasonló táblázat látható, csak éppen forrás port helyett cél portonként vannak listázva a well-known portokon átfolyt mennyiségek. Itt a 80-as port jóval kisebb mennyiséget mutat, a 443-as porton viszont közel ugyanannyi adat foly át befele irányba, mint kifelé.

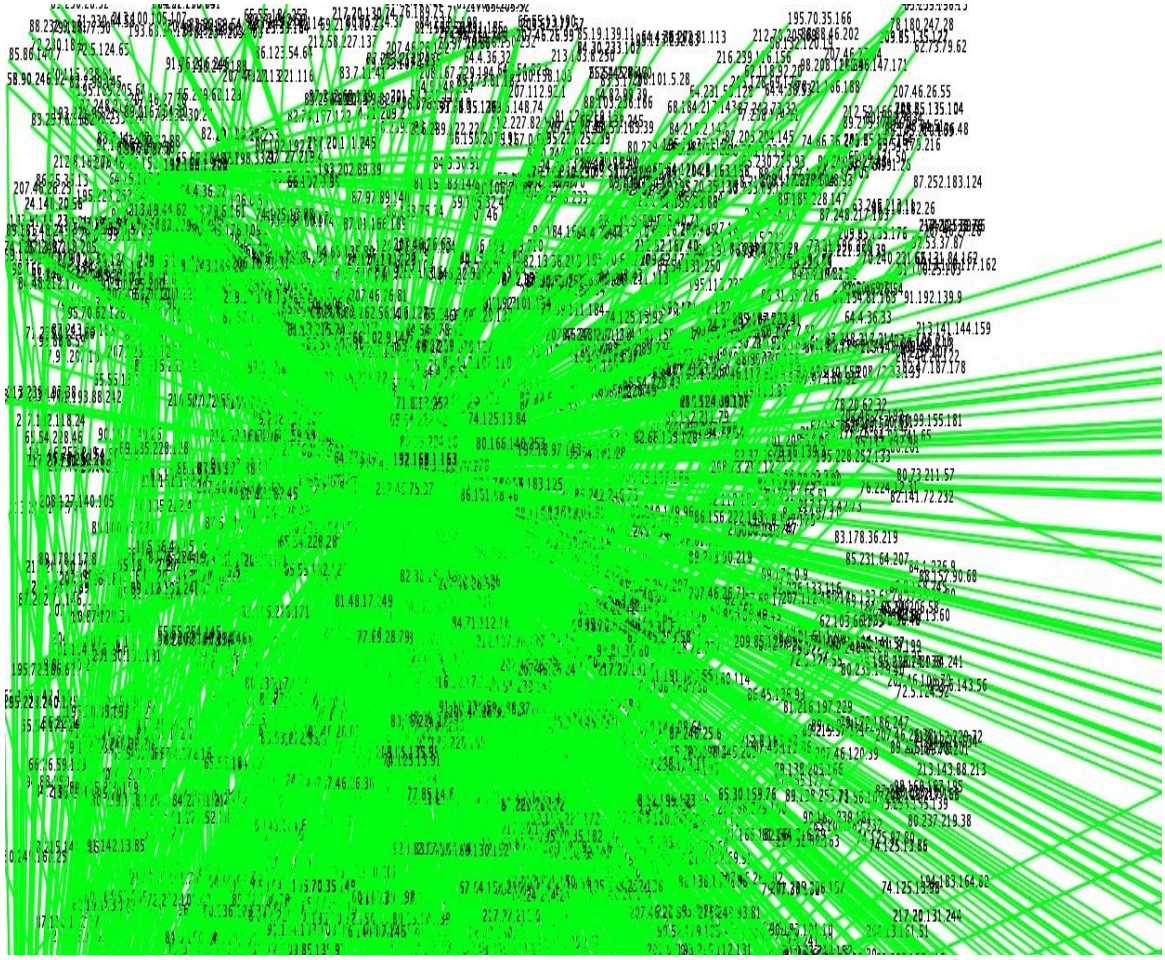
Cél port	Mennyiség
995	368 byte
21	618 byte
123	4 kbyte
139	6 kbyte
554	24 kbyte
22	29 kbyte
137	55 kbyte
53	149 kbyte
68	160 kbyte
67	177 kbyte
138	392 kbyte
445	567 kbyte
80	16 Mbyte
443	25 Mbyte

45. ábra: Átvitt adatmennyiség az 1024-nél kisebb cél portokon

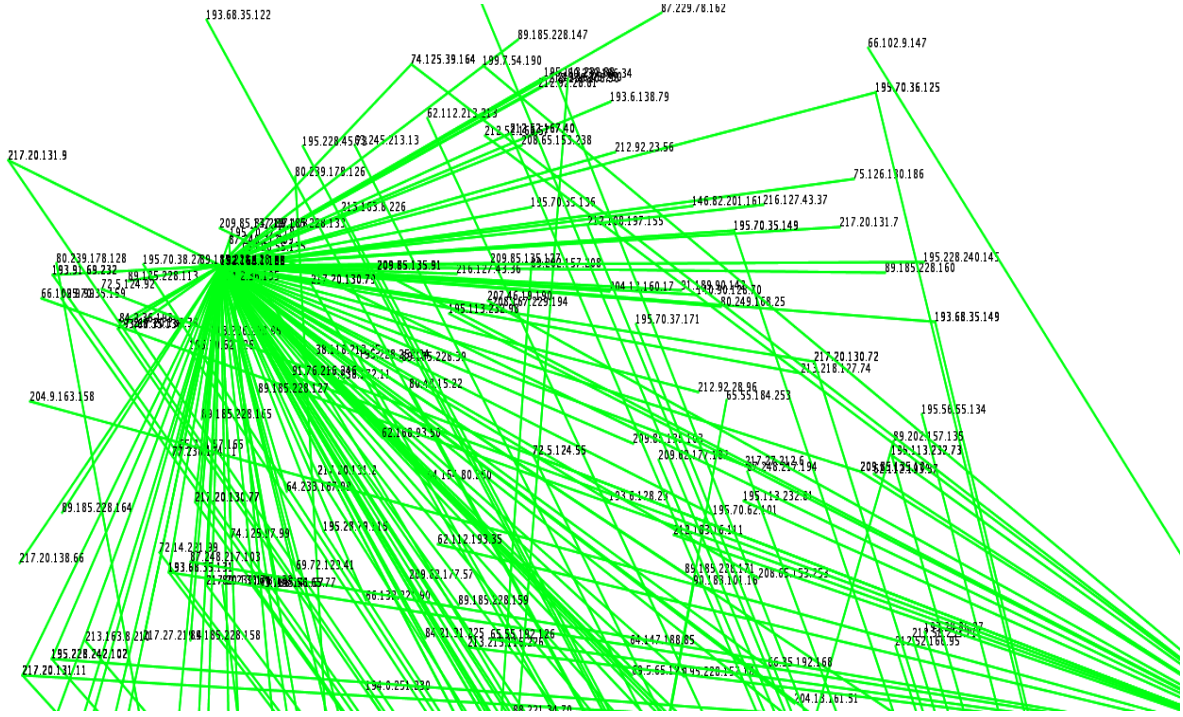
A programnak ezenkívül van egy funkciója, amely ip térképet rajzol a kapcsolatokból. Ez a térkék áttekinthető ábrát készít viszonylag kis számú kapcsolatnál (32. ábra), azonban egy viszonylag hosszú időszak alatti mintavétel esetén a 46. ábrán látható térkép készül. Mivel a csomagok nagy részének esetében vagy a cél vagy a forrás egy bizonyos gép, gyakorlatilag minden kapcsolatot jelképező vonal ugyanabban a pontban találkozik.

Emiatt ez az ábra nem szolgáltat áttekinthető és átfogó képet a Soho hálózat kapcsolatairól.

Tovább szűrve az adatokat, már áttekinthetőbb képet kapunk. Amennyiben például csak a 80-as célporton keresztüli forgalmat tekintjük (47. ábra.), már egy jóval áttekinthetőbb de még mindig igen ’sűrű’ ábrát kapunk.

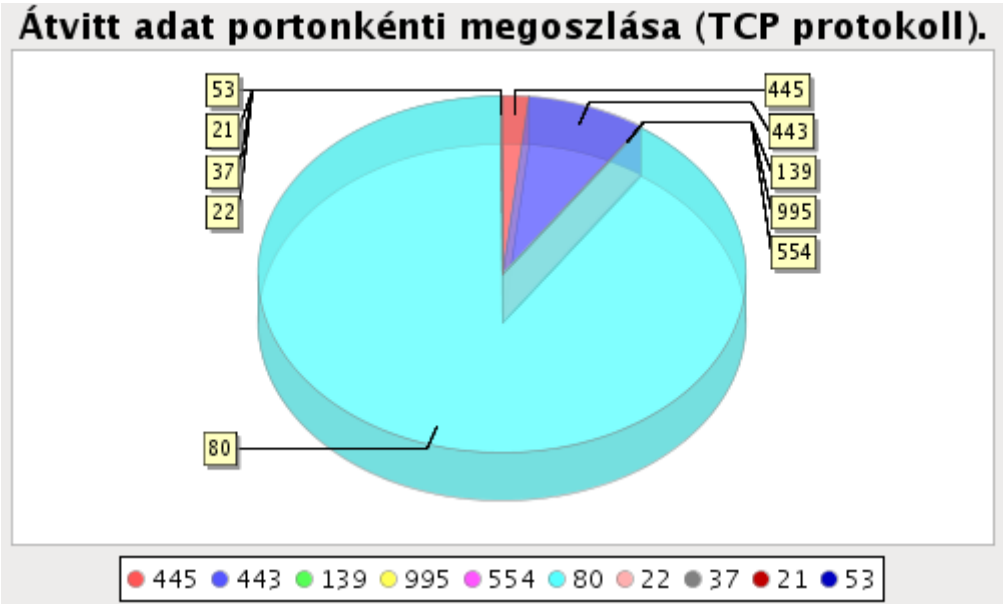


46. ábra: Ip térkép nagy minta esetén



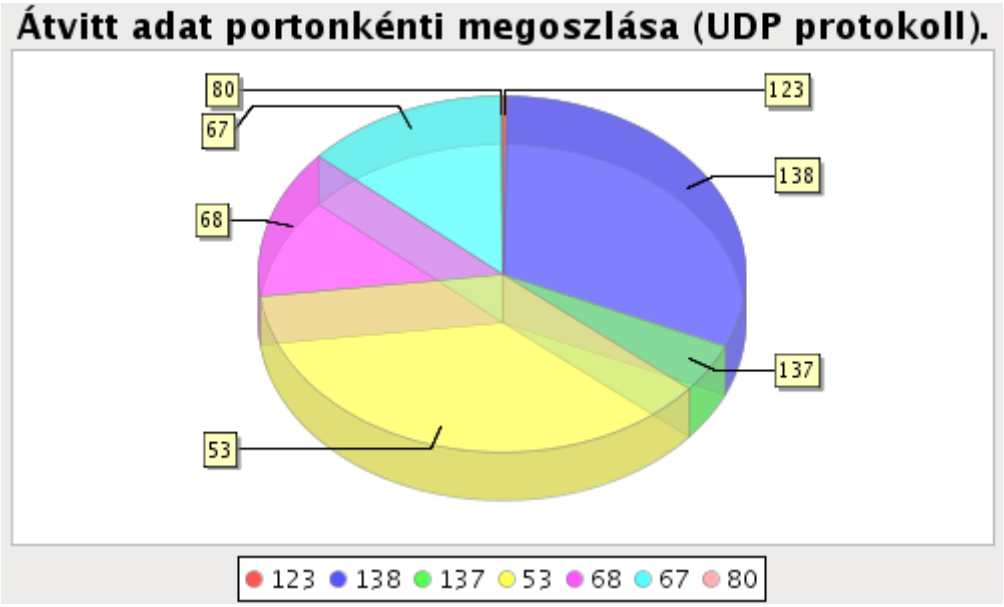
47. ábra: Az ip térkép csak a 80-as portra szűrve

A program által szolgáltatott diagrammok közzül az első (48. ábra) megmutatja, hogy TCP protokoll esetén milyen az 1024-nél kisebb portokon átmenő forgalom (attól függetlenül hogy az cél vagy forrás port). Mint látható, kiugróan magas a 80-as port a többihez képest. Számottevő még a https és az smb porton átmenő forgalom.



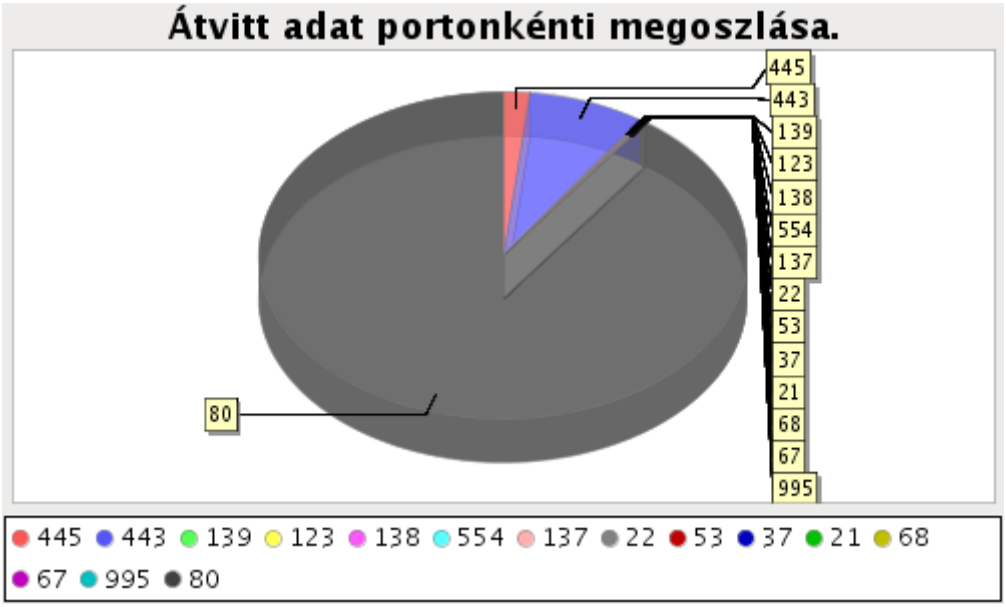
48. ábra: TCP protokollal 1024-nél kisseb portokon átmenő forgalom

Következő ábrán (49. ábra) ugyanilyen kimutatást láthatunk UDP protokoll esetén. Főként az 53-as DNS port és a 138-as NetBios port mutat aktivitást. Ezenkívül megfigyelhető még bizonyos mennyiségű adat a DHCP szerver által használt portokon.



49. ábra: UDP protokollal 1024-nél kisseb portokon átmenő forgalom

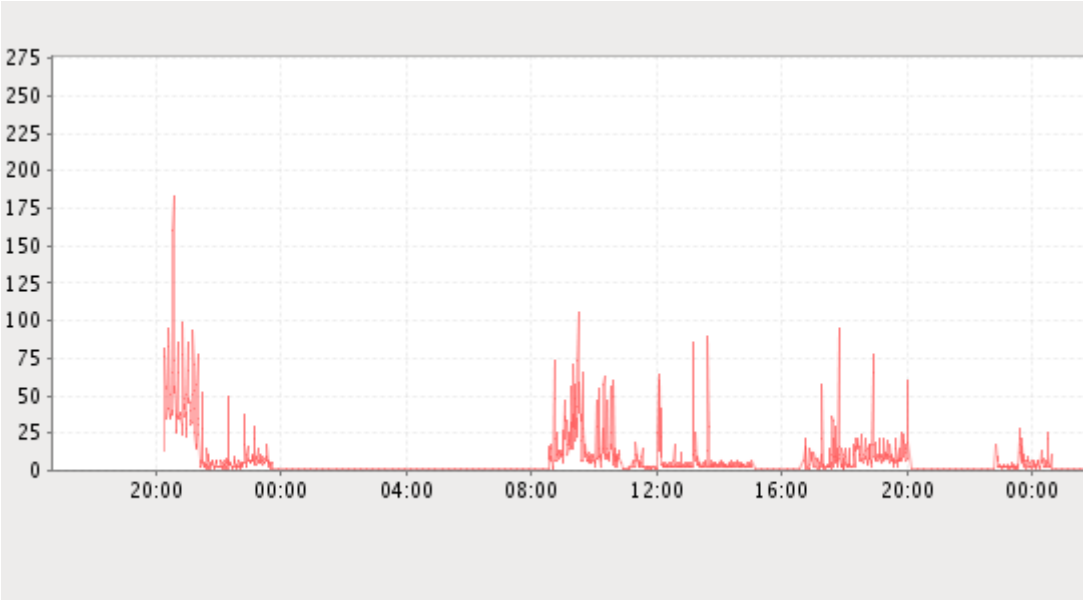
Ha a két ábrát összeolvasztjuk, és iránytól és protokollonként összehasonlítjuk az 1024 alatti portokon átmenő forgalmat (50. ábra), kiugróan a 80 port látszik mint fő forgalmazó.



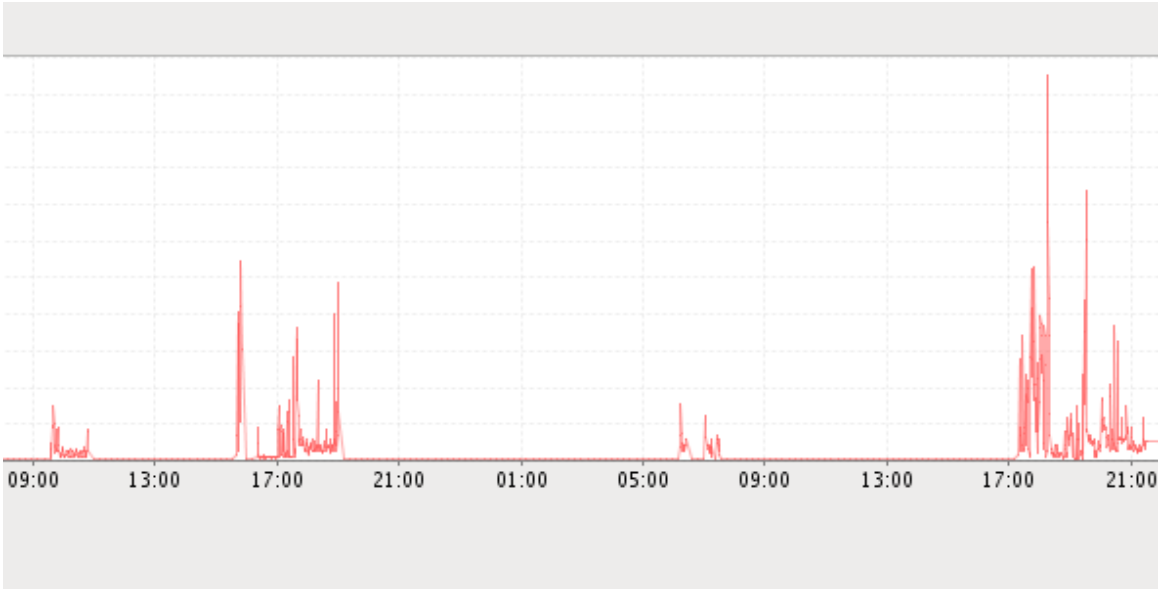
50. ábra: Portokon átvitt adatmennyiség megoszlás protokoltól függetlenül

A program még két ábrát készít. Egyrészről a kapcsolatok számát ábrázoló grafikon, mely x tengely szerint az órákat, az y tengelyen pedig a kapcsolatok számát adja meg. Ez egy igen széles ábra, melyet A4 oldalon nem lehet egyben bemutatni. Ezért itt szétdarabolva, részletek lesznek közölve (51-53 ábra).

Az ábrákon látható, hogy kiugró tüskék vannak, hirtelen felugró kapcsolat számokkal, egyébként az átlagos kapcsolatszám nem mérhető.

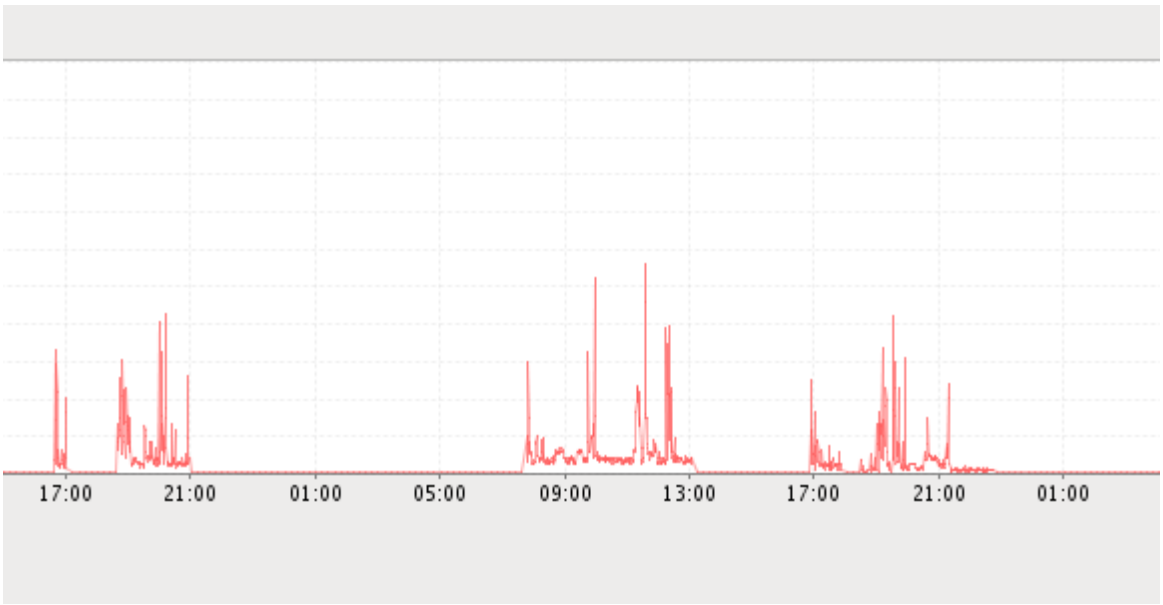


51. ábra: Kapcsolatok száma idősorban I.

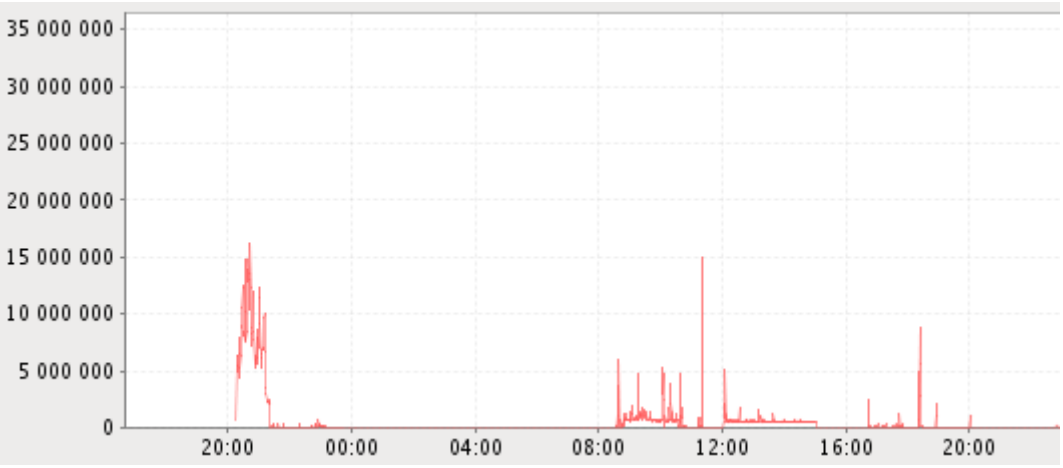


52. ábra: Kapcsolatok száma idősorban II.

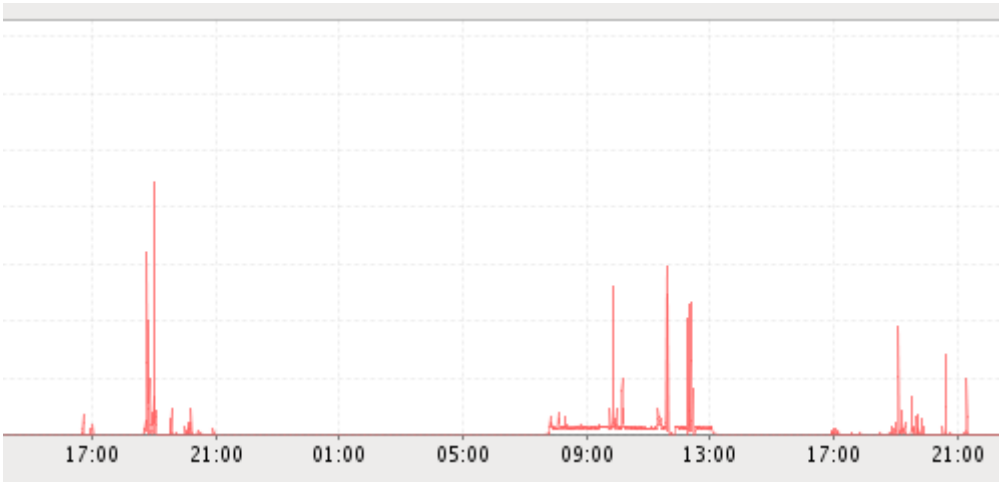
Az 54-as és 55-es ábrán a Soho hálózat sebesség grafikonjának két részlete látható. Mint látszik, a Soho hálózat kapacitása az idő nagy részében egyáltalán nincs kihasználva, csak adott pillanatokban ugrik fel a sebesség. Vegyük észre az 51. ábra és az 54. ábra 20:00 körüli mérési adatait: itt egy p2p letöltés kezdődött el, látható a hirtelen megugró kapcsolat szám majd a sebesség is a maximálisra ugrik. Miután file letöltésre került, a sebesség és a kapcsolatok száma is visszaugrik közel 0-ra.



53. ábra: Kapcsolatok száma idősorban III.



54. ábra: Sebesség a Soho hálózatban (b/s) I.



55. ábra: Sebesség a Soho hálózatban (b/s) II.

11. Az eredmények értelmezése

Amint a 10. fejezetben látható, az idő nagy részében a hálózat nem volt kihasználva. Gyakorlatilag az adatforgalom legnagyobb részét a rádió stream adta. Azokban a táblázatokban, melyekben a legtöbbet forgalmazó hosztok találhatóak (41-42-43. ábra), nem is található gyakorlatilag webkiszolgáló szerver, mint forgalmazó host. A legtöbbet forgalmazó szerver az online stream, a többi pedig különböző p2p kapcsolatban vett részt. Levelező programot a gépek nem használtak, de a levelező kliensként funkcionáló webes felület is szerepel a legtöbbet forgalmazó hosztok között (Ez egyébként a legtöbbet forgalmazó webkiszolgáló hoszt, a gmail.com). Ezen kívül természetesen a Soho hálózat belső gépei megtalálhatóak ebben a táblázatban, hiszen minden forgalom egyik végpontja a 3 gép közzül valamelyik volt.

A három gép közzül természetesen a legtöbbet használt `chucknorris` nevű gép generálta a legtöbb forgalmat. Utána a `mcgiver` nevű majd a `columbo` nevű gépek

forgalmaztak a hálózaton. Ez az adott számítógépen az internet használat gyakoriságát mutatja, a mérések megkezdésénél is ezt az eredményt vártuk.

Az utóbbi időkben a www oldalak megváltoztak. Sokkal több multimédiás tartalom, sokkal több flash, kép, videó került egy-egy oldalra. Ezzel megtöbbszöröződött egy web oldal (mely korábban csupán szövegből, esetleg néhány képből állt) sávszélesség igénye. Mégis, ezzel együtt az átlagos rendelkezésre álló sávszélesség olyannyira megnőtt, hogy immáron ez a megnövekedett mennyiség is a meglevő forgalom között gyakorlatilag meg sem látszódik.

Amennyiben a portok szerinti forgalmat nézzük, láthatjuk, hogy a forgalom nagy része nem is jelenik meg a well-known portok között. A korábbi ’fő’ portok immár nem a legtöbbet forgalmazó portok. A forgalom nagy része az 1024-nél nagyobb portokon megy keresztül (tipikusan p2p és online stream). Az 1024 alatti portok között továbbra is a 80-as port vezet, de mivel web kiszolgáló nem szerepel a top listában, valószínűsíthetően valamilyen 80-as porton áthaladó adat (és nem klasszikus www oldal) letöltése juttatta be a listába ezt a portot.

Ami még érdekes, hogy a top kiszolgálók forgalma listában szerepel a imgs.adverticum.net weboldal is, mely kizárólag online hirdetések, bannerek kiszolgálására szakosodott. Tehát a klasszikus http forgalom, a www oldalak forgalmának is egy igen komoly részét a hirdetések teszik ki.

Az Ip térképen látszik, hogy normál használat mellett egy Soho hálózat nagyon sok kapcsolatot épít ki külső hosztokkal. Mindez a háttérben történik, gyakorlatilag a felhasználó ’tudta’ nélkül. Elég csupán egy weboldalt megnyitni, és a weboldalt tartalmazó szerveren kívül, az oldalon lévő hirdetések, az oldalon lévő multimédiás tartalom, videók külön szerveren érhetőek el, melyre a böngésző a felhasználó tudta nélkül szintén kapcsolódik. A www korai szakaszában, a linkeket az erőforrások transzparans módon történő elosztására hozták létre. Ha egy weboldalon a felhasználó rákattintott egy linkre, a link mögött lévő dokumentum akár lehetett a világ egy másik részén lévő szerveren. Immáron ez annyira teljesült, hogy ’kattintani’ sem szükséges, a multimédiás tartalom, reklámok, flash animációk beillesztésével, az iframe-ekkel a tartalom úgy illeszthető be egyik szerverről a másikba, hogy az az oldal használójának, látogatójának fel sem tűnik.

A sebesség és kapcsolatok száma, mint látható, az idő nagy részében a 0-hoz áll közel. Gyakorlatilag az látható, hogy egy Soho hálózat az idő 95%-ban a rendelkezésre álló forgalomnak csupán néhány %-át használja. Így érhető el, hogy az internet szolgáltatók a hálózatok maximális sávszélességének többszörösét ’adja el’ a lakosságnak, mint internet szolgáltatás sávszélesség. Hiszen hiába rendel valaki 5-10-20 megabites kapcsolatot, az idő nagy részében csak 1-2 megabitet vagy annyit sem használ.

A kapcsolatok száma is erősen hullámzó volt. Az idő nagy részében majdnem 0 volt, aztán néhány esetben (tipikusan a p2p letöltéseknél) a SOHO – Internet közötti kiépített kapcsolatok száma felugrott 100-200 közzé.

Amennyiben az 1024-nél kisebb portok megoszlását nézzük, a TCP protokollnál nem meglepő módon magasan a 80-as port vezet. Azonban az UDP portnál azt láthatjuk, hogy a legnagyobb részben az 53 port szerepel, mely a DNS szolgáltatás portja. Ebből arra következtethetünk, hogy (legalábbis a mérés alatt) az UDP protokoll nem volt túlságosan használva (mely egyébként leginkább a real-time hang és kép átvitelre használatos Soho hálózatban).

12. Összefoglalás, következtetések

Mint az adatokból látható, egy mára általános otthoni Soho hálózat forgalma már megváltozott. Immáron nem csak néhány kapcsolatot, egy-egy www oldal lekérését esetleg levelezést kell használni. A fejlett web2.0 technológiának, multimédiás tartalmaknak köszönhetően nincs szükség levelező kliensre, a korai csak-szöveg felépítésű weboldalak helyett dinamikus, az asztali alkalmazások funkcionalitásával rendelkező weboldalak jönnek létre. Napjainkban elkezdett az asztali és a webes alkalmazások határa összemosódni. Egy asztali alkalmazás használhat css-t, javascript-et a megjelenítéshez, míg egy webes alkalmazás szinte úgy néz ki mint egy asztali alkalmazás. Ezt a határt tovább mosta a Java Web Start technológia, mellyel web-ről letölthető és futtatható asztali alkalmazást jelent. Viszont ahhoz, hogy ezek a technológiák müködjenek, hogy automatikusan, az oldal újratöltése nélkül dinamikusan változzon a weboldal szükség van arra, hogy a háttérben (általában valamilyen javascript kódon keresztül) a böngésző kommunikáljon a szerverrel. Így a régi modemes 'letöltjük a weboldalt majd bontjuk a csatlakozást' technológi végleg elavultá vált. Gyakorlatilag az új webes technológik folyamatos online kapcsolatot követelnek a kliensektől.

Ehhez arra volt szükség, hogy a régi betárcsázós telefonos kapcsolat háttérbe szoruljon és elterjedjenek a folyamatos on-line kapcsolatok, a percdíj nélküli havidíjas internet szolgáltatások (broadband és adsl technológiák). Immáron a mobilszolgáltatók is elkezdték bevezetni általánydíjas internetszolgáltatásaikat az új, fejlett gprs és edge technológiákon keresztül.

Azonban amellett, hogy a webes tartalom sávszélessége az új technológiáknak köszönhetően (Ajax, Web2.0) többszörösére nőtt - köszönhetően az otthoni szélessávú kapcsolatoknak – már sávszélesség szempontjából nem a leguralkodóbb felhasználási mód. Immár a videó konferenciák, online rádió és tv adások, p2p letöltések azok, melyek az igénybevett sávszélesség legnagyobb részét kiteszik.

Nyugodtan kijelenthetjük, hogy a többé az Internet nem egyenlő a weboldalakkal, levelezéssel. Már nem csak írott szöveg mozgatására használjuk, hanem képek, hangok, videók gyakorlatilag a kommunikáció minden formája megtalálható. Ezt az internet szolgáltatók időben felismerték, új otthoni szélessávú technológiájaikkal nem csak kihasználták ezt az új lehetőségeket hanem egyszersmind táptalajul szolgáltak a mégújabb webes lehetőségekhez. Csak idő kérdése, hogy a kereső programok is felismerjék azt, hogy többé nem csak az írott szövegben keressünk hanem keressünk képre (melyre a google-nak vannak próbálkozásai), hangra, videóra. Vannak teszt alatt

olyan kereső programok, melynek egy zene néhány traktusát kell megadni és ő megkísérli megkeresni az adott zenét.

Az Internet háttérbe szorította a levelezést az e-mail-el, a nyomtatott sajtót a hírportáljaival, a lexikonokat a wikipediával, a telefonos szolgáltatásokat a voip és chat programokkal. Már ostromolja a televíziós médiát a digitális televíziós adásokkal (megállítható tv adás, online műsorújság, stb). Elmondhatjuk gyakorlatilag hogy minden kommunikációs formát tud helyettesíteni méghozzá jobban mint elődje.

Elmondhatjuk tehát, hogy az Internet mára már az élet majd minden területén átvette a fő kommunikációs csatornát. Az Internet két fő protokollja, a TCP és az UDP illetve az ezeket hordozó IP protokoll soha nem látott módon kezd elterjedni. Rohamosan kezdünk kifogyni az ip címekből, sávszélességből. A legújabb technológiák néhány hónap alatt elavultá válnak, ahogy sorba kötjük rá a hálózatra számítógépünk után mobiltelefonunkat, videónkat, pda-nkat, hűtőnket, légkondicionálónkat, autónkat.

Már a felhasználók tábora is kezd kiszélesedni. Nem csupán a geek-ek jutnak internetre, nem csupán az informatikusok használják a számítógépet. A 10 évestől a 80 évesig bárkinek lehet otthon szélessávú internet szolgáltatása (és van is), ezzel teljesen új szolgáltatásokra jön létre igény. Ez tette lehetővé a Soho eszközök elterjedését. Az eszközöknél már követelmény a megbízhatóság, alacson ár és könnyű konfigurálhatóság.

És ezért nagyszerű protokoll a Tcp/Ip. Mert mint láttuk, annak ellenére, hogy a felhasználónak elég csak rákattintania, beírnia néhány egyszerű, könnyen megjegyezhető parancsot, a háttérben – transzparans módon – a forgalomirányító, forgalomszabályzó, hibajavító algoritumusok csendben teszik a dolgukat. Láthatatlanul fragmentálják a csomagokat majd sorrendbe teszik és összerakják, értelmezik, az eszközök észrevétlenül kommunikálnak, a torlódást kikerülik, a hibás csomópontok esetén más utat keresnek.

A felhasználónak pedig nincs más dolga, mint próbálni feldolgozni a hihetetlen információ áradatot, amely az interneten megtalálható.

Irodalomjegyzék

[1] Andrew S. Tanenbaum (2004), **Számítógépes hálózatok**, Budapest: Panem.

[2] Gróf Róbert, **CMS alapismeretek**, Internetes forrás

[3] Wikipedia bejegyzései, **http://www.wikipedia.org**, Internetes forrás

[4] Hungarian Unix Portal wiki bejegyzései, **http://wiki.hup.hu**, Internetes forrás

[5] Docsis 3.0 specifikáció (2006), **Operations Support System Interface Specification**, Internetes forrás

[6] Ajay Gummalla (2001), **DOCSIS Overview**, Internetes forrás

[7] Erik Swenson (2002), **Chart a new course with JFreeChart**, Internetes forrás

[8] JDBC Basics: **http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html**, Internetes forrás

[9] Java™ Platform, Standard Edition 6 API Specification: **http://java.sun.com/javase/6/docs/api/**, Internetes forrás

[10] Ceki Gülcü(2002), **Short introduction to log4j**, Internetes forrás

[11] PostgreSQL 8.3.1 Documentation, **http://www.postgresql.org/docs/8.3/static/index.html,** Internetes forrás

Köszönetnyilvánítás

Köszönetet mondok Gál Zoltán témavezetőmnek, hogy lehetőséget biztosított munkám sikeres elvégzéséhez és dolgozatom megírásához. Köszönöm segítőkész támogatását és dolgozatom alapos és kritikus átnézését.

„A" melléklet: Ábrák listája

Ábra sorszáma	Ábra megnevezése
1. ábra	Az Internet térképe
2. ábra	Ethernet keret felépítése
3. ábra	Ethernet típus mező lehetséges értékei (lista kiegészítése)
4. ábra	IP csomag felépítése
5. ábra	IP csomag protokoll mezőjének leggyakoribb értékei soho környezetben
6. ábra	Jelzőbitek (Flags) lehetséges értékei
7. ábra	UDP szegmens felépítése
8. ábra	Tcp szegmens felépítése
9. ábra	TCP szegmens flag bitjei
10. ábra	ARP csomag felépítése
11. ábra	Az opcode mező lehetséges értéke
12. ábra	Egy lehetséges ARP csomag
13. ábra	DHCP kérés folyamata
14. ábra	A ROOT névszerverek (2007. december)
15. ábra	.hu elsődleges körzet névszerverei
16. ábra	Példa a rekurzív dns lekérdezésre
17. ábra	Rekurzív névfeloldás
18. ábra	Egy tipikus soho hálózat
19. ábra	A SoHo hálózat topológiája
20. ábra	A hálózat felépítése és kapcsolatai
21. ábra	Frekvencia sávok
22. ábra	DOCSIS sebességei
23. ábra	HFC hálózat felépítése
24. ábra	UPC Magyarország hálózatának felépítése
25. ábra	Előre irányú kommunikáció a kábelTV rendszerben.
26. ábra	Egy MAP ciklus
27. ábra	DOCSIS 2.0 Protocol stack
28. ábra	DOCSIS MAC fejléce
29. ábra	DOCSIS Felhasználói adata
30. ábra	DOCSIS Management keret
31. ábra	Wireshark program képernyője
32. ábra	A program által kirajzolt IP térkép
33. ábra	Átvitt adat TCP portonként
34. ábra	Átvitt adat UDP portonként

35. ábra	Forgalom cél ipcímenként csoportosítva
36. ábra	Forgalom megoszlás cél és forrás ip címenként
37. ábra	A csomagelemző program felépítése
38. ábra	A rendszer terheltsége az adatok feldolgozása közben.
39. ábra	A Soho hálózat napinkénti forgalma (táblázat).
40. ábra	A Soho hálózat napinkénti forgalma (diagram).
41. ábra	Forgalmak forrás ip cím szerint
42. ábra	Forgalmak cél ip cím szerint
43. ábra	Forgalmak kapcsolatok (cél-forrás ip párok) szerint.
44. ábra	Átvitt adatmennyiség az 1024-nél kisebb forrás portokon
45. ábra	Ip térkép nagy minta esetén
46. ábra	Ip térkép nagy minta esetén
47. ábra	*Az ip térkép csak a 80-as portra szűrve*
48. ábra	TCP protokollal 1024-nél kisseb portokon átmenő forgalom
49. ábra	UDP protokollal 1024-nél kisseb portokon átmenő forgalom
50. ábra	Portokon átvitt adatmennyiség megoszlás protokol függetlenül
51. ábra	Kapcsolatok száma idősorban I.
52. ábra	Kapcsolatok száma idősorban II.
53. ábra	Kapcsolatok száma idősorban III.
54. ábra	Sebesség a Soho hálózatban (b/s) I.
55. ábra	Sebesség a Soho hálózatban (b/s) II.

„B" melléklet: Rövidítések listája

Rövidítés	Magyarázat
Aorta	Always On Real-Time Access
Arp	Address Resolution Protocol
Arpanet	Advanced Research Projects Agency Network
Bix	Budapest Internet eXchange
Cmts	Cable Modem Termination System
Crc	Cyclic Redundancy Check
Dhcp	Dinamic Host Control Protocol
Dns	Domain Name System
Docsis	Data Over Cable System Interface Specification
E-mail	Electronic Mail
Ftp	File Transfer Protocol
Gui	Graphical User Interface
Hfc	Hybrid Fiber Coax
http	Hyper Text Transfer Protocol
Icmp	Internet Control Message Protocol
Ieee	Institute of Electrical and Electronics Engineers
Ip	Internet Protocol
Isp	Internet Service Provider
Lan	Local Area Network
P2P	Peer to Peer
Pc	Personal Computer
Pop3	post office protocol
Rfc	Request for Comments
Smb	Server Message Block
smtp	Simple Mail Transfer Protocol
SOHO	Small Office Home Office
Sql	Structured Query Language
Tcp	Transmission Control Protocol
Tftp	Trivial File Transfer Protocol
TOS	Type of Service
TTL	Time To Live
Udp	User Datagram Protocol
Utp	Unshielded Twisted Pair
Voip	Voice Over Ip
WWW	World Wide Web

„C” melléklet: Elemző program forráskódja

```
001 /*
002  * To change this template, choose Tools | Templates
003  * and open the template in the editor.
004  */
005
006 package csomagelemzo.util;
007 import org.apache.log4j.Logger;
008 import java.sql.*;
009 import csomagelemzo.util.kivetel.*;
010 import java.util.ArrayList;
011 import java.util.GregorianCalendar;
012 /**
013  * Biztosítja a program függő kapcsolatot a program és a programfüggetlen
AdatbazisKapcsolat osztály között.
014  * @author fricci
015  */
016 public class SqlJavaLayer {
017     private Logger logger;
018     private AdatbazisKapcsolat adatbazisMotor;
019
020     /**
021      * Létrehoz egy SqlJavaLayer-t, paraméterül a már kész adatbázis nevét
várja String formátumban.
022      * @param adatbazisNev Az adatbázis neve, amihez kapcsolódni szeretne.
023      */
024     public SqlJavaLayer (String adatbazisNev)
025     {
026         try {
027             logger = Logger.getLogger("csomagelemzo");
028             logger.info("Az adatbázis motor indítása...");
029             adatbazisMotor = new AdatbazisKapcsolat(adatbazisNev);
030         } catch (SajatKivetel ex) {
031             logger.error("Kivétel történt az adatbázis motor indítása
közben!");
032         }
033
034     }
035
036     /**
037      * Célja hogy feltöltse adatokkal az adatbázist. Paraméterül egy
ArrayList-et vár, aminek tartalmát
038      * feltölti az adatbázisba. Visszatér a felvitt sorok számával.
039      * @param csomagok Egy ArrayList<HalozatCsomag> melyben a felviendő
csomagok vannak.
040      * @return A felvitt sorok számával. Normál esetben egyenlő az
ArrayList.size()-al.
041      */
042     public long initDataBase(ArrayList<HalozatCsomag> csomagok)
043     {
044         long felvitel = 0;
045         try
046         {
047             for(HalozatCsomag t : csomagok)
048             {
049                 String insert = new String("INSERT INTO CSOMAGOK (IDOBELYEG,
FORRASIP, CELIP, FORRASPORT, CELPORT, MERET, IPHASHOSSZEG, IPHASHSZORZAT,
PORTHASHOSSZEG, PORTHASHSZORZAT, PROTOKOL) VALUES ('" +
this.getStringDatum(t.getDatum()) +
"','"+t.getForrasIp()+"','"+t.getCelIp()+"',"+t.getForrasPort()+","+t.getCelPort
()+","+t.getCsomagMeret()+","+t.getIpHashOsszeg()+","+t.getIpHashSzorzat()+","+t
.getPortHashOsszeg()+","+t.getPortHashSzorzat()+","+t.getProtokoll()+")");
050                 //logger.info("Az SQL üzenet: "+insert);
051                 //System.exit(1);
052                 felvitel += adatbazisMotor.insertOrUpdate(insert);
053             }
```

```
054         }
055         catch(SajatKivetel sk)
056         {
057             logger.error("Kivétel kezelése!");
058 
059         }
060         finally
061         {
062             return felvitel;
063         }
064     }
065 
066     /**
067      * Egy GregorianCalendar objektumból csinál egy String-et.
068      * @param cal A GregorianCalendar objektum, melyből készít egy Stringet
(év-hónap-nap óra:perc:másodperc formátumban)
069      * @return  String (év-hónap-nap óra:perc:másodperc formátumban)
070      */
071     private String getStringDatum(GregorianCalendar cal)
072         //-------------------------------------------------
073     {   int ora = cal.get(GregorianCalendar.HOUR_OF_DAY); //-- Kikéri int
formátumban
074         String oraString;                                 //-
075         if(ora >= 10)                                     //-- ha 10-nél
nagyobb, egy az egyben
076         {                                                 //-  átalakítja
077             oraString = Integer.toString(ora);            //-
078         }                                                 //-
079         else                                              //-- ha kissebb
mint 10 akkor hozzáadunk az
080         {                                                 //-  elejéhez egy
0-t (pl 9 -> "09")
081             String temp = Integer.toString(ora);          //-
082             oraString = "0"+temp;                         //-
083         }                                                 //-
084         //-------------------------------------------------
085         int perc = cal.get(GregorianCalendar.MINUTE);
086         String percString;
087         if(perc >= 10)
088         {
089             percString = Integer.toString(perc);
090         }
091         else
092         {
093             String temp = Integer.toString(perc);
094             percString = "0"+temp;
095         }
096         int masodperc = cal.get(GregorianCalendar.SECOND);
097         String masodpercString;
098         if(masodperc >= 10)
099         {
100             masodpercString = Integer.toString(masodperc);
101         }
102         else
103         {
104             String temp = Integer.toString(masodperc);
105             masodpercString = "0"+temp;
106         }
107         //Elkészítjük az eredményt
108         String eredmeny = new String("" + cal.get(GregorianCalendar.YEAR) +
"-" + (cal.get(GregorianCalendar.MONTH)+1) + "-" +
cal.get(GregorianCalendar.DAY_OF_MONTH) +" "+ oraString + ":" + percString + ":"
+ masodpercString);
109         return eredmeny;
110     }
111 
112     /**
113      * Töröljük az adatbázis tartalmát.
```

```
114      * @return a törölt sorok száma
115      */
116     public int adatbazisTorol()
117     {
118         try {
119             return adatbazisMotor.insertOrUpdate("DELETE FROM CSOMAGOK");
120         } catch (SajatKivetel ex) {
121             logger.warn("Hiba az adatbázis törlése közben!");
122             return -1;
123         }
124     }
125 
126     /**
127      * Futtat egy <tt>SELECT</tt> utasítást. A parancsot paraméterként kell
átadni, az eredményhalmazzal
128      * visszatér. Amennyiben a lekérdezés során valamilyen hiba történt
(hibás lekérdezés formátum vagy
129      * bármilyen hiba), akkor visszatérési érték <tt>null</tt>.
130      *
131      * @param select Az <tt>SELECT</tt> utasítást tartalmazó String.
132      * @return A lekérdezés eredményhalmaza, vagy <tt>null</tt> ha hiba
történt a lekérdezésben.
133      */
134     public ResultSet selectExecute(String select)
135     {
136         try
137         {
138             return adatbazisMotor.executeSelect(select); //Egyből meghívja
az AdatbazisKapcsolat 'executeSelect' utasítását.
139         }
140         catch (SajatKivetel sk)
141         {
142             logger.error("Hiba a lekérdezés során! Hibakód: " +
sk.getHibaKod() + " Hibaüzenet: " + sk.getHibaUzenet());
143             return null;
144         }
145     }
146 
147 }
001 /*
002  * To change this template, choose Tools | Templates
003  * and open the template in the editor.
004  */
005 
006 package csomagelemzo.util.abra;
007 
008 import java.awt.image.BufferedImage;
009 import java.io.File;
010 import java.io.IOException;
011 import org.apache.log4j.*;
012 import org.jfree.chart.ChartUtilities;
013 import org.jfree.chart.JFreeChart;
014 
015 /**
016  * A grafikonon létrehozásához szükséges osztályok alapja. Minden ilyen
specializált 'grafikon' osztály ebből
017  * az osztályból van származtatva. Ez az osztály végzi a létrehozott
JFreeChart objektum (azaz a grafikon) fájnévvel való
018  * ellátását, képpé alakítását és png-be mentés esetén a File objektum
létrehozását.
019  * @author fricci
020  */
021 public class AbstractAbra
022 {
023     protected String magyarazat;
024     protected String fileNev;
025     protected JFreeChart abra;
026     protected static final Logger logger = Logger.getLogger("csomagelemzo");
```

```
027     private static final int MAX_RANDOM_FILE_NEV = 99999999;
028     public static final int ISMERETLEN_FILEMERET = 500;
029     protected File fileLink;
030     protected BufferedImage image;
031     protected int szelesseg;
032 
033     /**
034      * Létrehoz egy Abstract osztályt, létrehozza egy random generált
számból a kép lehetséges file-nevét (ez csak
035      * akkor kerül felhasználásra, ha valóban mentésre kerül a kép)
036      * @param magyarazat Az ábrához tartozó magyarázó szöveg
037      * @param szelesseg A kép szélesség. Ha az
AbstractAbra.ISMERETLEN_FILEMERET méretet használjuk, akkor az alapértelmezett
500 széles lesz.
038      */
039     protected AbstractAbra(String magyarazat, int szelesseg)
040     {
041         this.image = null;
042         this.szelesseg = szelesseg;
043         this.magyarazat = magyarazat;
044         fileNev = String.valueOf(new
java.util.Random().nextInt(AbstractAbra.MAX_RANDOM_FILE_NEV))+".png";
045     }
046 
047     /**
048      * Paraméter nélküli konstruktor, magyarázat nélküli, 500 széles
grafikonhoz
049      */
050     protected AbstractAbra()
051     {
052         this("", 500);
053     }
054 
055     /**
056      * Lekéri a grafikonból készített BufferedImage objektumot.
057      * @return A grafikonból készített BufferedImage
058      */
059     public BufferedImage getChart()
060     {
061         return this.image;
062     }
063 
064     /**
065      * Beállít egy magyarázatot a grafikonhoz.
066      * @param magyarazat A magyarázatot
067      */
068     public void setMagyarazat(final String magyarazat)
069     {
070         this.magyarazat = magyarazat;
071     }
072 
073     /**
074      * Lekéri a magyarázatot.
075      * @return A grafikonhoz tartozó magyarázat.
076      */
077     public String getMagyarazat()
078     {
079         return this.magyarazat;
080     }
081 
082     /**
083      * Elmenti a grafikont a korábban generált file-nevet adva, png
formátumban. Az átadott paraméter lehet null értékű is, akkor
084      * a ./doc/report/pics (windowsnál a doc\report\pics) könyvtárba menti.
A kép csak egyszer mentődik el, további meghívásnál nem fut le.
085      * @param konyvtar null vagy könyvtár neve ahova mentse (file név nem
kell hogy benne legyen).
086      */
```

```
087     public void saveChart(final String konyvtar)
088     {
089         if(this.fileLink == null)
090         {
091             String utvonal;
092             if(konyvtar == null)
093             {
094                 final String sep = File.separator;
095                 utvonal = "doc"+sep+"report"+sep+"pics"+sep+this.fileNev;
096                 logger.info("Mentem a következő helyre: "+utvonal);
097             }
098             else
099             {
100                 utvonal = konyvtar+this.fileNev;
101             }
102             try
103             {
104                 ChartUtilities.saveChartAsPNG(new File(utvonal), this.abra,
szelesseg, 300);
105             }
106             catch (IOException ex)
107             {
108                 logger.error("Hiba a file mentésénél: " + ex.getMessage());
109             }
110         }
111     }
112 }
01 /*
02  * To change this template, choose Tools | Templates
03  * and open the template in the editor.
04  */
05
06 package csomagelemzo.util.lekerdezesek;
07 import csomagelemzo.util.SqlJavaLayer;
08 import csomagelemzo.util.abra.LineChart;
09 import java.awt.image.BufferedImage;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.util.ArrayList;
13 import org.apache.log4j.*;
14 /**
15  *
16  * @author fricci
17  */
18 public abstract class AbstractLekerdezes {
19     protected static final Logger logger = Logger.getLogger("csomagelemzo");
20     protected final SqlJavaLayer layer;
21
22     protected String lekerdezesString = null;
23     protected String magyarazoSzoveg = "proba";
24     public AbstractLekerdezes(SqlJavaLayer layer, String lekerdezesString)
25     {
26         this.layer = layer;
27         this.lekerdezesString = lekerdezesString;
28         initAbra();
29         magyarazoSzoveg = "proba";
30     }
31
32     abstract protected void initAbra();
33
34     public abstract BufferedImage getKep();
35
36     protected static String getProtokoll(int protokoll)
37     {
38         String eredmeny;
39         switch(protokoll)
40         {
41             case 1:
```

```
42              {
43                  eredmeny = "where protokol = 1 ";
44                  break;
45              }
46              case 2:
47              {
48                  eredmeny = "where protokol = 2 ";
49                  break;
50              }
51              default:
52              {
53                  eredmeny = "";
54                  break;
55              }
56          }
57          return eredmeny;
58      }
59  }
001 /*
002  * To change this template, choose Tools | Templates
003  * and open the template in the editor.
004  */
005
006 package csomagelemzo.util;
007 import java.util.logging.Level;
008 import org.apache.log4j.Logger;
009 import java.sql.*;
010 import csomagelemzo.util.kivetel.*;
011 import java.util.Properties;
012
013 /**
014  *
015  * @author fricci
016  */
017 public class AdatbazisKapcsolat {
018     private Connection conn; //Egy kapcsolat a Derby driver felé
019     private Statement st;
020     private Logger logger;
021     /**
022      * Létrehoz egy belső sql motor felé egy adatbázis kapcsolatot. A
konstruktor
023      * betölti a drivert, elindítja a motort, létrehozza a kapcsolatot.
024      * @param adatbazis Az adatbázis neve amihez kapcsolódni kell
025      * @throws penzforgalom.util.SajatKivetel Amennyiben nem található a
driver illetve más ismeretlen eresetű hiba esetén.
026      */
027     public AdatbazisKapcsolat(String adatbazis) throws SajatKivetel
028     {
029         logger = Logger.getLogger("csomagelemzo");
030         logger.info("Adatbázis motor indítása.");
031         String driver = "org.postgresql.Driver"; //A derby driver neve. A
032         //String driver = "org.apache.derby.jdbc.EmbeddedDriver"; //A derby
driver neve. A
033         String connectionURL = "jdbc:postgresql://localhost/"+adatbazis; //A
derby adatbázishoz a connection string. Az 'adatbazis' változó tartalmazza az
adatbázis nevét.
034         //String connectionURL = "jdbc:derby:"+adatbazis+";create=false";
//A derby adatbázishoz a connection string. Az 'adatbazis' változó tartalmazza
az adatbázis nevét.
035         try
036         {
037             Class.forName(driver);
038             Properties props = new Properties();
039             props.setProperty("user","csomagelemzo");
040             props.setProperty("password","12345");
041             conn = DriverManager.getConnection(connectionURL, props);
042             logger.info("Adatbázis motor elindult.");
043         }
```

```
044         catch(ClassNotFoundException e)
045         {
046             logger.error("Kívétel keletkezett!");
047             throw new SajatKivetel(e, "Nem található a megfelelő
adatbázismotorlib file vagy nem tölthető az be", SajatKivetel.CLASS_NOT_FOUND);
048         }
049         catch (Throwable e)
050         {
051             logger.error("Kívétel keletkezett!");
052             throw new SajatKivetel(e, "Ismeretlen hiba történt az
adatbázismotor kezelése közben", SajatKivetel.ISMERETLEN_HIBA);
053         }
054     }
055 
056     /*
057      * Létrehoz egy statement objektumot.
058      *
059      */
060     private void createStatement() throws SQLException
061     {
062         /*
063          * Először volt itt egy null vizsgálat, azaz csak akkor hozta létre
a
064          * statement-t, ha még nem lett. De valamiért ekkor csak egyszer
lehetett
065          * használni a lekérdezést, másodjára azt mondta hogy nincs a
resultset
066          * megnyitva.
067          */
068         this.st = conn.createStatement();
069     }
070 
071 
072     /*
073      * Leállítja a belső sql motort
074      */
075     protected void serverStop() throws SajatKivetel
076     {
077         try
078         {
079             DriverManager.getConnection("jdbc:derby:;shutdown=true");
//Leállítja a derby adatbázist
080         }
081         catch (SQLException se)
082         {
083             if(!se.getSQLState().equals("XJ015"))  //Az XJ015 kód a sikeres
leállítást jelenti, egyéb esetben kivételt kell dobni.
084             {
085                 logger.error("Kívétel keletkezett!");
086                 throw new SajatKivetel(se, "Nem sikerült az adatbázis motor
leállítása!", SajatKivetel.ADATBAZIS_LEALLITASI_HIBA);
087             }
088         }
089     }
090 
091     /**
092      * Futtat egy SELECT utasítást és visszaadja az eredmény halmazt.
093      * @param select A SELECT utasítást tartalmazó String.
094      * @return Az eredmény halmaz.
095      * @throws penzforgalom.util.SajatKivetel Lekérdezési hiba esetén.
096      */
097     public ResultSet executeSelect(String select) throws SajatKivetel
098     {
099         try
100         {
101             createStatement();  //Létrehozza egy Statement-t
102             return st.executeQuery(select); //Majd futtatja a select-et és
visszatér az eredménnyel
```

```
103         }
104         catch (SQLException e)
105         {
106             logger.error("Kivétel keletkezett!");
107             throw new SajatKivetel(e, "Hiba a lekérdezés közben!",
SajatKivetel.LEKERDEZESI_HIBA);
108         }
109     }
110 
111     /**
112      * Beilleszt egy sort. A beillesztő utasítás (INSERT INTO ...) -t
tartalmazó String utasítást kell átadni.
113      * A függvény visszatér a beillesztett/megváltoztatott sorok számával.
114      * @param insert Az update parancsot tartalmazó String
115      * @return A megváltoztatott sorok számával.
116      * @throws penzforgalom.util.SajatKivetel Bármilyen sql hiba esetén.
117      */
118     public int insertOrUpdate(String insert) throws SajatKivetel
119     {
120         try
121         {
122             createStatement();
123             st.execute(insert);
124             return st.getUpdateCount();
125         }
126         catch (SQLException e)
127         {
128             if(insert.toUpperCase().startsWith("INSERT")) //Kitalálja hogy a
végrehajtandó parancs egy INSERT
129             {
130                 logger.error("Kivétel keletkezett!");
131                 throw new SajatKivetel(e, "Hiba az Ăşj sor beszĂşrása
közben!", SajatKivetel.SOR_BESZURASI_HIBA);
132             }
133             else if(insert.toUpperCase().startsWith("UPDATE")) // vagy egy
UPDATE parancs volt-e
134             {
135                 logger.error("Kivétel keletkezett!");
136                 throw new SajatKivetel(e, "Hiba a sor módosítása közben!",
SajatKivetel.SOR_MODOSITASI_HIBA);
137             }
138             return -1;
139         }
140 
141     }
142 
143     @Override
144     public void finalize()
145     {
146         /*try
147         {
148             this.serverStop();
149         }
150         catch (SajatKivetel sk)
151         {
152             //Semmitteszünk
153         }*/
154     }
155 }
01 /*
02  * To change this template, choose Tools | Templates
03  * and open the template in the editor.
04  */
05 
06 package csomagelemzo.util.abra;
07 
08 import java.util.ArrayList;
09 import java.util.Iterator;
```

```
10 import org.jfree.chart.ChartFactory;
11 import org.jfree.chart.plot.PlotOrientation;
12 import org.jfree.data.category.DefaultCategoryDataset;
13
14 /**
15  * Létrehozható segítségével egy oszlop grafikon.
16  * @author fricci
17  */
18 public class BarChart extends AbstractAbra
19 {
20     /**
21      * Létrehozható segítségével a grafikont jelképező objektum. A
22      *      az adatokat tartalmazó ArrayList felépítése a következő:
23      *      [
24      *          [függőleges tengely értéke][kategória][vízszintes tengely
értéke]
25      *      ]
26      *
27      * @param adatok Az adatok amelyből a grafikon lehet
28      * @param magyarazat A grafikonhoz tartozó magyarázat
29      * @param cim A grafikon címe
30      * @param xTengely Az x tengely neve
31      * @param yTengely Az y tengely neve
32      */
33     public BarChart(final ArrayList adatok, final String magyarazat, final
String cim, final String xTengely, final String yTengely)
34     {
35         this(adatok, magyarazat, cim, xTengely, yTengely, 500);
36     }
37
38     /**
39      * Létrehozható segítségével a grafikont jelképező objektum. A
40      *      az adatokat tartalmazó ArrayList felépítése a következő:
41      *      [
42      *          [függőleges tengely értéke][kategória][vízszintes tengely
értéke]
43      *      ]
44
45      * @param adatok Az adatok amelyből a grafikon lehet
46      * @param magyarazat A grafikonhoz tartozó magyarázat
47      * @param cim A grafikon címe
48      * @param xTengely Az x tengely neve
49      * @param yTengely Az y tengely neve
50      * @param szelesseg A kép szélessége
51      */
52     public BarChart(final ArrayList adatok, final String magyarazat, final
String cim, final String xTengely, final String yTengely, final int szelesseg)
53     {
54         super(magyarazat, szelesseg);
55         DefaultCategoryDataset dataset = new DefaultCategoryDataset();
56         Iterator it = adatok.iterator();
57         while(it.hasNext())
58         {
59             ArrayList sor = (ArrayList)it.next();
60             dataset.addValue((Number)sor.get(0), (Comparable)sor.get(1),
(Comparable)sor.get(2));
61         }
62         this.abra = ChartFactory.createBarChart3D
63                     (cim, // Title
64                      xTengely,              // X-Axis label
65                      yTengely,                 // Y-Axis label
66                      dataset,         // Dataset
67                      PlotOrientation.VERTICAL,
68                      true,                     // Show legend
69                      true,
70                      true
71                     );
72         this.abra.setAntiAlias(true);
```

```
73          this.abra.setTextAntiAlias(true);
74          abra.getPlot().setForegroundAlpha(0.5f);
75          image = abra.createBufferedImage(this.szelesseg, 300);
76      }
77  }
001 /*
002  * To change this template, choose Tools | Templates
003  * and open the template in the editor.
004  */
005
006 package csomagelemzo.util.lekerdezesek;
007
008 import csomagelemzo.util.SqlJavaLayer;
009 import java.sql.ResultSet;
010 import java.sql.SQLException;
011 import java.util.Enumeration;
012 import java.util.Iterator;
013 import java.util.Vector;
014 import javax.swing.table.DefaultTableModel;
015 import org.apache.log4j.Logger;
016
017 /**
018  *
019  * @author fricci
020  */
021 public class CelIpMegoszlas {
022     protected static final Logger logger = Logger.getLogger("csomagelemzo");
023     String lekerdezoString;
024     DefaultTableModel tableModel;
025     public CelIpMegoszlas(SqlJavaLayer layer, int tipus)
026     {
027         switch(tipus)
028         {
029             case 1:
030             {
031                 try
032                 {
033                     lekerdezoString = "select forrasip as ForrasIp,
sum(meret) as Forgalom from csomagok group by forrasip order by sum(meret)";
034                     tableModel =
resultSetToTableModel(layer.selectExecute(lekerdezoString));
035                 }
036                 catch(SQLException e)
037                 {
038
039                 }
040                 finally
041                 {
042                     break;
043                 }
044             }
045             case 2:
046             {
047                 try
048                 {
049                     lekerdezoString = "select celip as CelIp, sum(meret) as
Forgalom from csomagok group by celip order by sum(meret)";
050                     tableModel =
resultSetToTableModel(layer.selectExecute(lekerdezoString));
051                 }
052                 catch(SQLException e)
053                 {
054
055                 }
056                 finally
057                 {
058                     break;
059                 }
```

```
060             }
061             case 3:
062             {
063                 try
064                 {
065                     lekerdezoString =
066                         "select distinct ipTabla.celip as CelIp,
ipTabla.forrasip as ForrasIp, kapcsTabla.mennyiseg  as totalMennyiseg "+
067                         "from "+
068                         "( "+
069                         "  select distinct iphashosszeg, iphashszorzat,
sum(meret) as mennyiseg "+
070                         "  from csomagok "+
071                         "  group by iphashosszeg, iphashszorzat "+
072                         ") as kapcsTabla, "+
073                         "csomagok as ipTabla "+
074                         "where ipTabla.iphashosszeg =
kapcsTabla.iphashosszeg and ipTabla.iphashszorzat = kapcsTabla.iphashszorzat "+
075                         "order by totalMennyiseg";
076                      tableModel =
clearResultSet(resultSetToTableModel(layer.selectExecute(lekerdezoString)));
077
078                 }
079                 catch(SQLException e)
080                 {
081
082                 }
083                 finally
084                 {
085                     break;
086                 }
087             }
088             case 4:
089             {
090                 try
091                 {
092                     lekerdezoString =
093                     "select forrasip, forrasport, celip, celport, sum(meret)
as forgalom "+
094                     "from csomagok "+
095                     "group by forrasip, celip, forrasport, celport "+
096                     "order by forgalom desc";
097                     tableModel =
resultSetToTableModel(layer.selectExecute(lekerdezoString));
098                 }
099                 catch (SQLException e)
100                 {
101
102                 }
103                 finally
104                 {
105                     break;
106                 }
107             }
108             default:
109             {
110
111             }
112         }
113     }
114
115     private DefaultTableModel resultSetToTableModel (ResultSet rs) throws
SQLException
116     {
117         DefaultTableModel model = new DefaultTableModel()
118         {
119             @Override
120             public boolean isCellEditable (int row, int column) //Ezzel
```

módosítom az DefaultTableModel alapértelmezett isCellEditable függvényét. ÃŤgy a cellák nem módosíthatóak leszenk

```
121             {
122                 return false;
123             }
124         };
125         Vector oszlopNevek = new Vector();
126         String oszlopNev = null;
127         int oszlop = 1;
128         int max = rs.getMetaData().getColumnCount();
129         while(oszlop <= max)
130         {
131             oszlopNev = rs.getMetaData().getColumnName(oszlop);
132             oszlopNevek.add(oszlopNev);
133             oszlop++;
134         }
135         Iterator felsorolo = oszlopNevek.iterator();
136         while(felsorolo.hasNext())
137         {
138             model.addColumn((String)felsorolo.next());
139         }
140         while (rs.next())
141         {
142             Object[] tomb = new Object[max];
143             oszlop = 0;
144             while(oszlop < max)
145             {
146                 tomb[oszlop] = rs.getString(oszlop+1);
147                 oszlop++;
148             }
149 
150             model.addRow(tomb);
151         }
152         return model;
153     }
154 
155     private DefaultTableModel clearResultSet(DefaultTableModel set)
156     {
157        String forrasIp = "";
158        String celIp = "";
159        String forgalom = "";
160        Vector adatok = set.getDataVector();
161        int i = 0;
162        while(i < adatok.size())
163        {
164            Vector adatSorTemp = (Vector)adatok.get(i);
165            if(adatSorTemp.get(2).equals(forgalom) &&
adatSorTemp.get(0).equals(forrasIp) && adatSorTemp.get(1).equals(celIp))
166            {
167                adatok.remove(adatSorTemp);
168                celIp = (String)adatSorTemp.get(0);
169                forrasIp = (String)adatSorTemp.get(1);
170                forgalom = (String)adatSorTemp.get(2);
171                continue;
172            }
173            celIp = (String)adatSorTemp.get(0);
174            forrasIp = (String)adatSorTemp.get(1);
175            forgalom = (String)adatSorTemp.get(2);
176            i++;
177        }
178        return set;
179     }
180 
181     public DefaultTableModel getTable()
182     {
183         return this.tableModel;
184     }
185 
```

```
186 }
01 /*
02  * CsomagElemzoApp.java
03  */
04
05 package csomagelemzo;
06
07 import org.jdesktop.application.Application;
08 import org.jdesktop.application.SingleFrameApplication;
09 import org.apache.log4j.*;
10 /**
11  * The main class of the application.
12  */
13 public class CsomagElemzoApp extends SingleFrameApplication {
14
15     /**
16      * At startup create and show the main frame of the application.
17      */
18     @Override protected void startup() {
19         show(new CsomagElemzoView(this));
20     }
21
22     /**
23      * This method is to initialize the specified window by injecting
resources.
24      * Windows shown in our application come fully initialized from the GUI
25      * builder, so this additional configuration is not needed.
26      */
27     @Override protected void configureWindow(java.awt.Window root) {
28     }
29
30     /**
31      * A convenient static getter for the application instance.
32      * @return the instance of CsomagElemzoApp
33      */
34     public static CsomagElemzoApp getApplication() {
35         return Application.getInstance(CsomagElemzoApp.class);
36     }
37
38     /**
39      * Main method launching the application.
40      */
41     public static void main(String[] args) {
42
43          Logger  logger = Logger.getLogger("csomagelemzo");
44          PropertyConfigurator.configure("log4j.properties");
45          logger.info("Program inicializálása és indítása.");
46          launch(CsomagElemzoApp.class, args);
47
48
49     }
50 }
001 /*
002  * CsomagElemzoView.java
003  */
004 package csomagelemzo;
005
006 import java.util.concurrent.ExecutionException;
007 import java.util.logging.Level;
008 import org.jdesktop.application.Action;
009 import org.jdesktop.application.ResourceMap;
010 import org.jdesktop.application.SingleFrameApplication;
011 import org.jdesktop.application.FrameView;
012 import org.jdesktop.application.TaskMonitor;
013 import java.awt.event.ActionEvent;
014 import java.awt.event.ActionListener;
015 import javax.swing.Timer;
016 import javax.swing.Icon;
```

```
017 import javax.swing.JDialog;
018 import javax.swing.JFileChooser;
019 import javax.swing.JFrame;
020 import org.apache.log4j.Logger;
021 import csomagelemzo.util.*;
022 import csomagelemzo.util.abra.*;
023 import csomagelemzo.util.lekerdezesek.CelIpMegoszlas;
024 import csomagelemzo.util.lekerdezesek.HalozatForgalma;
025 import csomagelemzo.util.lekerdezesek.KapcsolatokSzama;
026 import csomagelemzo.util.lekerdezesek.PortonkentiMennyisegLine;
027 import csomagelemzo.util.lekerdezesek.PortonkentiMennyisegPie;
028 import java.beans.PropertyChangeListener;
029 import java.util.ArrayList;
030 import org.jdesktop.application.Task;
031 import javax.swing.ImageIcon;
032
033 /**
034  * The application's main frame.
035  */
036 public class CsomagElemzoView extends FrameView {
037
038     private Logger logger = null;
039     private SqlJavaLayer layer = null;
040     TaskMonitor taskMonitor = null;
041
042     public CsomagElemzoView(SingleFrameApplication app) {
043         super(app);
044         logger = Logger.getLogger("csomagelemzo");
045         initComponents();
046         ResourceMap resourceMap = getResourceMap();
047         int messageTimeout =
resourceMap.getInteger("StatusBar.messageTimeout");
048         messageTimer = new Timer(messageTimeout, new ActionListener() {
049
050             public void actionPerformed(ActionEvent e) {
051                 statusMessageLabel.setText("");
052             }
053         });
054         messageTimer.setRepeats(false);
055         int busyAnimationRate =
resourceMap.getInteger("StatusBar.busyAnimationRate");
056         for (int i = 0; i < busyIcons.length; i++) {
057             busyIcons[i] = resourceMap.getIcon("StatusBar.busyIcons[" + i +
"]");
058         }
059         busyIconTimer = new Timer(busyAnimationRate, new ActionListener() {
060
061             public void actionPerformed(ActionEvent e) {
062                 busyIconIndex = (busyIconIndex + 1) % busyIcons.length;
063                 statusAnimationLabel.setIcon(busyIcons[busyIconIndex]);
064             }
065         });
066         idleIcon = resourceMap.getIcon("StatusBar.idleIcon");
067         statusAnimationLabel.setIcon(idleIcon);
068         progressBar.setVisible(false);
069         // connect action tasks to status bar via TaskMonitor
070         taskMonitor = new TaskMonitor(getApplication().getContext());
071         PropertyChangeListener forgoIkon = new
java.beans.PropertyChangeListener() {
072
073             public void propertyChange(java.beans.PropertyChangeEvent evt) {
074                 String propertyName = evt.getPropertyName();
075                 if ("started".equals(propertyName)) {
076                     if (!busyIconTimer.isRunning()) {
077                         statusAnimationLabel.setIcon(busyIcons[0]);
078                         busyIconIndex = 0;
079                         busyIconTimer.start();
080                     }
```

```
081                     progressBar.setVisible(true);
082                     progressBar.setIndeterminate(true);
083
084                 } else if ("done".equals(propertyName)) {
085                     busyIconTimer.stop();
086                     statusAnimationLabel.setIcon(idleIcon);
087                     progressBar.setVisible(false);
088                     progressBar.setValue(0);
089
090                 } else if ("message".equals(propertyName)) {
091                     String text = (String) (evt.getNewValue());
092                     statusMessageLabel.setText((text == null) ? "" : text);
093                     messageTimer.restart();
094                 } else if ("progress".equals(propertyName)) {
095                     int value = (Integer) (evt.getNewValue());
096                     progressBar.setVisible(true);
097                     progressBar.setIndeterminate(false);
098                     progressBar.setValue(value);
099                 }
100             }
101         };
102         taskMonitor.addPropertyChangeListener(forgoIkon);
103         this.createDatabaseLayer();
104     }
105
106     @Action
107     public void showAboutBox() {
108         if (aboutBox == null) {
109             JFrame mainFrame =
CsomagElemzoApp.getApplication().getMainFrame();
110             aboutBox = new CsomagElemzoAboutBox(mainFrame);
111             aboutBox.setLocationRelativeTo(mainFrame);
112         }
113         CsomagElemzoApp.getApplication().show(aboutBox);
114     }
115
116
117
118     /** This method is called from within the constructor to
119      * initialize the form.
120      * WARNING: Do NOT modify this code. The content of this method is
121      * always regenerated by the Form Editor.
122      */
123     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
124     private void initComponents() {
125
126         mainPanel = new javax.swing.JPanel();
127         tabPanel = new javax.swing.JTabbedPane();
128         ipTerkep = new javax.swing.JPanel();
129         jScrollPane1 = new javax.swing.JScrollPane();
130         rajzLabel = new javax.swing.JLabel();
131         jPanel1 = new javax.swing.JPanel();
132         jLabel1 = new javax.swing.JLabel();
133         jScrollPane2 = new javax.swing.JScrollPane();
134         forrasIpSzerint = new javax.swing.JTable();
135         jPanel2 = new javax.swing.JPanel();
136         jScrollPane3 = new javax.swing.JScrollPane();
137         celIpSzerint = new javax.swing.JTable();
138         jPanel3 = new javax.swing.JPanel();
139         jScrollPane4 = new javax.swing.JScrollPane();
140         ipKozottiKapcsolat = new javax.swing.JTable();
141         jPanel4 = new javax.swing.JPanel();
142         jScrollPane5 = new javax.swing.JScrollPane();
143         jPanel5 = new javax.swing.JPanel();
144         portonkentiMennyisegLinePie1AbraLabel = new javax.swing.JLabel();
145         portonkentiMennyisegLinePie2AbraLabel = new javax.swing.JLabel();
146         portonkentiMennyisegLinePie3AbraLabel = new javax.swing.JLabel();
```

```
147         portonkentiMennyisegLineAbraLabel = new javax.swing.JLabel();
148         kapcsolatokSzamaAbraLabel = new javax.swing.JLabel();
149         halozatForgalmaAbraLabel = new javax.swing.JLabel();
150         menuBar = new javax.swing.JMenuBar();
151         javax.swing.JMenu fileMenu = new javax.swing.JMenu();
152         jMenuItem1 = new javax.swing.JMenuItem();
153         javax.swing.JMenuItem exitMenuItem = new javax.swing.JMenuItem();
154         adatbazisMenu = new javax.swing.JMenu();
155         adatbazisTorlesItem = new javax.swing.JMenuItem();
156         javax.swing.JMenu helpMenu = new javax.swing.JMenu();
157         javax.swing.JMenuItem aboutMenuItem = new javax.swing.JMenuItem();
158         statusPanel = new javax.swing.JPanel();
159         javax.swing.JSeparator statusPanelSeparator = new
javax.swing.JSeparator();
160         statusMessageLabel = new javax.swing.JLabel();
161         statusAnimationLabel = new javax.swing.JLabel();
162         progressBar = new javax.swing.JProgressBar();
163
164         mainPanel.setName("mainPanel"); // NOI18N
165
166         tabPanel.setName("tabPanel"); // NOI18N
167
168         ipTerkep.setAutoscrolls(true);
169         ipTerkep.setName("ipTerkep"); // NOI18N
170         ipTerkep.addComponentListener(new java.awt.event.ComponentAdapter()
{
171             public void componentShown(java.awt.event.ComponentEvent evt) {
172                 ipTerkepComponentShown(evt);
173             }
174         });
175
176         jScrollPane1.setName("jScrollPane1"); // NOI18N
177
178         org.jdesktop.application.ResourceMap resourceMap =
org.jdesktop.application.Application.getInstance(csomagelemzo.CsomagElemzoApp.cl
ass).getContext().getResourceMap(CsomagElemzoView.class);
179         rajzLabel.setText(resourceMap.getString("rajzLabel.text")); //
NOI18N
180         rajzLabel.setName("rajzLabel"); // NOI18N
181         jScrollPane1.setViewportView(rajzLabel);
182
183         javax.swing.GroupLayout ipTerkepLayout = new
javax.swing.GroupLayout(ipTerkep);
184         ipTerkep.setLayout(ipTerkepLayout);
185         ipTerkepLayout.setHorizontalGroup(
186
ipTerkepLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
187             .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 526, Short.MAX_VALUE)
188         );
189         ipTerkepLayout.setVerticalGroup(
190
ipTerkepLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
191             .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 417, Short.MAX_VALUE)
192         );
193
194
tabPanel.addTab(resourceMap.getString("ipTerkep.TabConstraints.tabTitle"),
ipTerkep); // NOI18N
195
196         jPanel1.setName("jPanel1"); // NOI18N
197         jPanel1.addComponentListener(new java.awt.event.ComponentAdapter() {
198             public void componentShown(java.awt.event.ComponentEvent evt) {
199                 jPanel1ComponentShown(evt);
200             }
201         });
202
```

```
203         jLabel1.setText(resourceMap.getString("jLabel1.text")); // NOI18N
204         jLabel1.setName("jLabel1"); // NOI18N
205
206         jScrollPane2.setName("jScrollPane2"); // NOI18N
207
208         forrasIpSzerint.setModel(new javax.swing.table.DefaultTableModel(
209             new Object [][] {
210
211             },
212             new String [] {
213
214             }
215         ));
216         forrasIpSzerint.setName("forrasIpSzerint"); // NOI18N
217         jScrollPane2.setViewportView(forrasIpSzerint);
218
219         javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
220         jPanel1.setLayout(jPanel1Layout);
221         jPanel1Layout.setHorizontalGroup(
222
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
223             .addGroup(jPanel1Layout.createSequentialGroup()
224                 .addContainerGap()
225
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
226                     .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 502, Short.MAX_VALUE)
227                     .addComponent(jLabel1))
228                 .addContainerGap())
229         );
230         jPanel1Layout.setVerticalGroup(
231
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
232             .addGroup(jPanel1Layout.createSequentialGroup()
233                 .addContainerGap()
234
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
235                     .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 318,
javax.swing.GroupLayout.PREFERRED_SIZE)
236                     .addComponent(jLabel1))
237                 .addContainerGap(87, Short.MAX_VALUE))
238         );
239
240
tabPanel.addTab(resourceMap.getString("jPanel1.TabConstraints.tabTitle"),
jPanel1); // NOI18N
241
242         jPanel2.setName("jPanel2"); // NOI18N
243
244         jScrollPane3.setName("jScrollPane3"); // NOI18N
245
246         celIpSzerint.setModel(new javax.swing.table.DefaultTableModel(
247             new Object [][] {
248
249             },
250             new String [] {
251
252             }
253         ));
254         celIpSzerint.setName("celIpSzerint"); // NOI18N
255         jScrollPane3.setViewportView(celIpSzerint);
256
257         javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
```

```
258         jPanel2.setLayout(jPanel2Layout);
259         jPanel2Layout.setHorizontalGroup(
260 
jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
261             .addGroup(jPanel2Layout.createSequentialGroup()
262                 .addContainerGap()
263                 .addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 502, Short.MAX_VALUE)
264                 .addContainerGap())
265         );
266         jPanel2Layout.setVerticalGroup(
267 
jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
268             .addGroup(jPanel2Layout.createSequentialGroup()
269                 .addContainerGap()
270                 .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 309,
javax.swing.GroupLayout.PREFERRED_SIZE)
271                 .addContainerGap(96, Short.MAX_VALUE))
272         );
273 
274 
tabPanel.addTab(resourceMap.getString("jPanel2.TabConstraints.tabTitle"),
jPanel2); // NOI18N
275 
276         jPanel3.setName("jPanel3"); // NOI18N
277 
278         jScrollPane4.setName("jScrollPane4"); // NOI18N
279 
280         ipKozottiKapcsolat.setModel(new javax.swing.table.DefaultTableModel(
281             new Object [][] {
282 
283             },
284             new String [] {
285 
286             }
287         ));
288         ipKozottiKapcsolat.setName("ipKozottiKapcsolat"); // NOI18N
289         jScrollPane4.setViewportView(ipKozottiKapcsolat);
290 
291         javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
292         jPanel3.setLayout(jPanel3Layout);
293         jPanel3Layout.setHorizontalGroup(
294 
jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
295             .addGroup(jPanel3Layout.createSequentialGroup()
296                 .addContainerGap()
297                 .addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 502, Short.MAX_VALUE)
298                 .addContainerGap())
299         );
300         jPanel3Layout.setVerticalGroup(
301 
jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
302             .addGroup(jPanel3Layout.createSequentialGroup()
303                 .addContainerGap()
304                 .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 312,
javax.swing.GroupLayout.PREFERRED_SIZE)
305                 .addContainerGap(93, Short.MAX_VALUE))
306         );
307 
308 
tabPanel.addTab(resourceMap.getString("jPanel3.TabConstraints.tabTitle"),
jPanel3); // NOI18N
309 
310         jPanel4.setName("jPanel4"); // NOI18N
```

```
311
312         jScrollPane5.setName("jScrollPane5"); // NOI18N
313
314         jPanel5.setName("jPanel5"); // NOI18N
315
316
portonkentiMennyisegLinePie1AbraLabel.setText(resourceMap.getString("portonkenti
MennyisegLinePie1AbraLabel.text")); // NOI18N
317
portonkentiMennyisegLinePie1AbraLabel.setName("portonkentiMennyisegLinePie1AbraL
abel"); // NOI18N
318
319
portonkentiMennyisegLinePie2AbraLabel.setText(resourceMap.getString("portonkenti
MennyisegLinePie2AbraLabel.text")); // NOI18N
320
portonkentiMennyisegLinePie2AbraLabel.setName("portonkentiMennyisegLinePie2AbraL
abel"); // NOI18N
321
322
portonkentiMennyisegLinePie3AbraLabel.setText(resourceMap.getString("portonkenti
MennyisegLinePie3AbraLabel.text")); // NOI18N
323
portonkentiMennyisegLinePie3AbraLabel.setName("portonkentiMennyisegLinePie3AbraL
abel"); // NOI18N
324
325
portonkentiMennyisegLineAbraLabel.setText(resourceMap.getString("portonkentiMenn
yisegLineAbraLabel.text")); // NOI18N
326
portonkentiMennyisegLineAbraLabel.setName("portonkentiMennyisegLineAbraLabel");
// NOI18N
327
328
kapcsolatokSzamaAbraLabel.setText(resourceMap.getString("kapcsolatokSzamaAbraLab
el.text")); // NOI18N
329         kapcsolatokSzamaAbraLabel.setName("kapcsolatokSzamaAbraLabel"); //
NOI18N
330
331
halozatForgalmaAbraLabel.setText(resourceMap.getString("halozatForgalmaAbraLabel
.text")); // NOI18N
332         halozatForgalmaAbraLabel.setName("halozatForgalmaAbraLabel"); //
NOI18N
333
334         javax.swing.GroupLayout jPanel5Layout = new
javax.swing.GroupLayout(jPanel5);
335         jPanel5.setLayout(jPanel5Layout);
336         jPanel5Layout.setHorizontalGroup(
337
jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
338             .addGroup(jPanel5Layout.createSequentialGroup()
339                 .addContainerGap()
340
.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
341                     .addComponent(portonkentiMennyisegLinePie1AbraLabel)
342                     .addComponent(portonkentiMennyisegLinePie2AbraLabel)
343                     .addComponent(portonkentiMennyisegLinePie3AbraLabel)
344                     .addComponent(portonkentiMennyisegLineAbraLabel)
345                     .addComponent(kapcsolatokSzamaAbraLabel)
346                     .addComponent(halozatForgalmaAbraLabel))
347                 .addContainerGap(488, Short.MAX_VALUE))
348         );
349         jPanel5Layout.setVerticalGroup(
350
jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
351             .addGroup(jPanel5Layout.createSequentialGroup()
```

```
352                 .addContainerGap()
353                 .addComponent(portonkentiMennyisegLinePie1AbraLabel)
354                 .addGap(18, 18, 18)
355                 .addComponent(portonkentiMennyisegLinePie2AbraLabel)
356                 .addGap(18, 18, 18)
357                 .addComponent(portonkentiMennyisegLinePie3AbraLabel)
358                 .addGap(18, 18, 18)
359                 .addComponent(portonkentiMennyisegLineAbraLabel)
360                 .addGap(18, 18, 18)
361                 .addComponent(kapcsolatokSzamaAbraLabel)
362                 .addGap(18, 18, 18)
363                 .addComponent(halozatForgalmaAbraLabel)
364                 .addContainerGap(289, Short.MAX_VALUE))
365         );
366 
367         jScrollPane5.setViewportView(jPanel5);
368 
369         javax.swing.GroupLayout jPanel4Layout = new
javax.swing.GroupLayout(jPanel4);
370         jPanel4.setLayout(jPanel4Layout);
371         jPanel4Layout.setHorizontalGroup(
372 
jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
373             .addGroup(jPanel4Layout.createSequentialGroup()
374                 .addContainerGap()
375                 .addComponent(jScrollPane5,
javax.swing.GroupLayout.DEFAULT_SIZE, 502, Short.MAX_VALUE)
376                 .addContainerGap())
377         );
378         jPanel4Layout.setVerticalGroup(
379 
jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
380             .addGroup(jPanel4Layout.createSequentialGroup()
381                 .addContainerGap()
382                 .addComponent(jScrollPane5,
javax.swing.GroupLayout.DEFAULT_SIZE, 393, Short.MAX_VALUE)
383                 .addContainerGap())
384         );
385 
386 
tabPanel.addTab(resourceMap.getString("jPanel4.TabConstraints.tabTitle"),
jPanel4); // NOI18N
387 
388         javax.swing.GroupLayout mainPanelLayout = new
javax.swing.GroupLayout(mainPanel);
389         mainPanel.setLayout(mainPanelLayout);
390         mainPanelLayout.setHorizontalGroup(
391 
mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
392             .addComponent(tabPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
534, Short.MAX_VALUE)
393         );
394         mainPanelLayout.setVerticalGroup(
395 
mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
396             .addComponent(tabPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
477, Short.MAX_VALUE)
397         );
398 
399         menuBar.setName("menuBar"); // NOI18N
400 
401         fileMenu.setText(resourceMap.getString("fileMenu.text")); // NOI18N
402         fileMenu.setName("fileMenu"); // NOI18N
403 
404 
jMenuItem1.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyE
vent.VK_O, java.awt.event.InputEvent.CTRL_MASK));
405         jMenuItem1.setText(resourceMap.getString("jMenuItem1.text")); //
```

```
NOI18N
406         jMenuItem1.setName("jMenuItem1"); // NOI18N
407         jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
408             public void actionPerformed(java.awt.event.ActionEvent evt) {
409                 jMenuItem1ActionPerformed(evt);
410             }
411         });
412         fileMenu.add(jMenuItem1);
413
414         javax.swing.ActionMap actionMap =
org.jdesktop.application.Application.getInstance(csomagelemzo.CsomagElemzoApp.cl
ass).getContext().getActionMap(CsomagElemzoView.class, this);
415         exitMenuItem.setAction(actionMap.get("quit")); // NOI18N
416         exitMenuItem.setName("exitMenuItem"); // NOI18N
417         fileMenu.add(exitMenuItem);
418
419         menuBar.add(fileMenu);
420
421         adatbazisMenu.setText(resourceMap.getString("adatbazisMenu.text"));
// NOI18N
422         adatbazisMenu.setName("adatbazisMenu"); // NOI18N
423
424
adatbazisTorlesItem.setText(resourceMap.getString("adatbazisTorlesItem.text"));
// NOI18N
425         adatbazisTorlesItem.setName("adatbazisTorlesItem"); // NOI18N
426         adatbazisTorlesItem.addActionListener(new
java.awt.event.ActionListener() {
427             public void actionPerformed(java.awt.event.ActionEvent evt) {
428                 adatbazisTorlesItemActionPerformed(evt);
429             }
430         });
431         adatbazisMenu.add(adatbazisTorlesItem);
432
433         menuBar.add(adatbazisMenu);
434
435         helpMenu.setText(resourceMap.getString("helpMenu.text")); // NOI18N
436         helpMenu.setName("helpMenu"); // NOI18N
437
438         aboutMenuItem.setAction(actionMap.get("showAboutBox")); // NOI18N
439         aboutMenuItem.setName("aboutMenuItem"); // NOI18N
440         helpMenu.add(aboutMenuItem);
441
442         menuBar.add(helpMenu);
443
444         statusPanel.setName("statusPanel"); // NOI18N
445
446         statusPanelSeparator.setName("statusPanelSeparator"); // NOI18N
447
448         statusMessageLabel.setName("statusMessageLabel"); // NOI18N
449
450
statusAnimationLabel.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
451         statusAnimationLabel.setName("statusAnimationLabel"); // NOI18N
452
453         progressBar.setName("progressBar"); // NOI18N
454
455         javax.swing.GroupLayout statusPanelLayout = new
javax.swing.GroupLayout(statusPanel);
456         statusPanel.setLayout(statusPanelLayout);
457         statusPanelLayout.setHorizontalGroup(
458
statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
459             .addComponent(statusPanelSeparator,
javax.swing.GroupLayout.DEFAULT_SIZE, 534, Short.MAX_VALUE)
460             .addGroup(statusPanelLayout.createSequentialGroup()
461                 .addContainerGap()
462                 .addComponent(statusMessageLabel)
```

```
463
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 350,
Short.MAX_VALUE)
464                 .addComponent(progressBar,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
465
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
466                 .addComponent(statusAnimationLabel)
467                 .addContainerGap())
468         );
469         statusPanelLayout.setVerticalGroup(
470
statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
471             .addGroup(statusPanelLayout.createSequentialGroup()
472                 .addComponent(statusPanelSeparator,
javax.swing.GroupLayout.PREFERRED_SIZE, 2,
javax.swing.GroupLayout.PREFERRED_SIZE)
473
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
474
.addGroup(statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.BASELINE)
475                     .addComponent(statusMessageLabel)
476                     .addComponent(statusAnimationLabel)
477                     .addComponent(progressBar,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
478                 .addGap(3, 3, 3))
479         );
480
481         setComponent(mainPanel);
482         setMenuBar(menuBar);
483         setStatusBar(statusPanel);
484     }// </editor-fold>//GEN-END:initComponents
485
486     //Lefut ha rákattintuok a megnyitás gombra
487     private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jMenuItem1ActionPerformed
488         logger.info("File/Megnyitásra kattintva");
489         Task task;
490         this.createDatabaseLayer();
491         final javax.swing.JFileChooser fileMegnyitas = new
javax.swing.JFileChooser();
492         logger.info("Megnyitandó file kiválasztása.");
493         int returnVal = fileMegnyitas.showOpenDialog(menuBar);
494         if (returnVal == JFileChooser.APPROVE_OPTION) {  //Ha érvényes a
kiválasztás (nem a cancel gombra kattintottunk)
495             logger.info("A következő file kerül megnyitásra: " +
fileMegnyitas.getSelectedFile().getName());
496             task = new Task(this.getApplication()) {
497
498                 @Override
499                 protected Object doInBackground() {
500                     ArrayList csomagok;
501                     //Elindítja a fileOlvas-s folyamatát
502                     FileOlvaso fileOlvas = new
FileOlvaso(fileMegnyitas.getSelectedFile());
503                     csomagok = fileOlvas.getCsomagok();
504                     long csomagokSzama = csomagok.size();
505                     //Feltölti az adatbázist
506                     long eredmeny = layer.initDataBase(csomagok);
507                     if(csomagokSzama != eredmeny)
508                     {
509                         logger.warn("VIGYĂ?ZZ! Nem sikerült minden sort
felvinni az adatbázisba");
510                     }
```

```
511                     layer.selectExecute("select
portforgalomtablafeltolt()");
512                     return layer;
513                 }
514             };
515             task.addPropertyChangeListener(new
java.beans.PropertyChangeListener() {
516 
517                 public void propertyChange(java.beans.PropertyChangeEvent
evt) {
518                     String propertyName = evt.getPropertyName();
519                     if ("started".equals(propertyName)) {
520                         if (!busyIconTimer.isRunning()) {
521                             statusAnimationLabel.setIcon(busyIcons[0]);
522                             busyIconIndex = 0;
523                             busyIconTimer.start();
524                         }
525                         progressBar.setVisible(true);
526                         progressBar.setIndeterminate(true);
527                     } else if ("done".equals(propertyName)) {
528                         busyIconTimer.stop();
529                         statusAnimationLabel.setIcon(idleIcon);
530                         progressBar.setVisible(false);
531                         progressBar.setValue(0);
532                     } else if ("message".equals(propertyName)) {
533                         String text = (String) (evt.getNewValue());
534                         statusMessageLabel.setText((text == null) ? "" :
text);
535                         messageTimer.restart();
536                     } else if ("progress".equals(propertyName)) {
537                         int value = (Integer) (evt.getNewValue());
538                         progressBar.setVisible(true);
539                         progressBar.setIndeterminate(false);
540                         progressBar.setValue(value);
541                     }
542                 }
543             });
544             task.execute();
545             if(layer == null)
546             {
547                 logger.warn("A layer null referencia!");
548             }
549             /*try {
550                 layer = (SqlJavaLayer) task.get(1);
551             } catch (InterruptedException e) {
552                 logger.warn("Futtatási hiba!");
553             } catch (java.util.concurrent.ExecutionException e) {
554                 logger.warn("Futtatási hiba!");
555             }*/
556             this.adatInit();
557         }
558 
559     }//GEN-LAST:event_jMenuItem1ActionPerformed
560 
561     /**
562      * Létrehoz egy SqlJavaLayer objektumot ha még az nem létezik. Ha
létezik, nem tesz semmit.
563      * Ennek eldöntésére azt vizsgálja, a layer referencia null értékű-e
vagy sem.
564      */
565     private void createDatabaseLayer()
566     {
567         try {
568             if (layer != null) {
569                 return;
570             }
571 
572             Task task = new Task(this.getApplication()) { //Létrehozunk egy
```

```
Ăşj névtelen Task osztályt
573
574                 @Override
575                 protected Object doInBackground() { //Felülírjuk a
doInBackground() függvényt, mely majd el fog indulni
576                     logger.info("SQL Motor indítása");
577                     SqlJavaLayer layer2 = new SqlJavaLayer("HalozatCsomag");
578                     logger.info("SQL Motor indítása...OK!");
579                     layer = layer2;
580                     return layer2;
581                 }
582             };
583             // Ezzel biztosítjuk hogy majd jelezze a gui-n hogy háttérben
munka folyik (progress bar, busyIcon, stb...)
584             task.addPropertyChangeListener(new
java.beans.PropertyChangeListener() {
585
586                 public void propertyChange(java.beans.PropertyChangeEvent
evt) {
587                     String propertyName = evt.getPropertyName();
588                     if ("started".equals(propertyName)) {
589                         if (!busyIconTimer.isRunning()) {
590                             statusAnimationLabel.setIcon(busyIcons[0]);
591                             busyIconIndex = 0;
592                             busyIconTimer.start();
593                         }
594                         progressBar.setVisible(true);
595                         progressBar.setIndeterminate(true);
596                     } else if ("done".equals(propertyName)) {
597                         busyIconTimer.stop();
598                         statusAnimationLabel.setIcon(idleIcon);
599                         progressBar.setVisible(false);
600                         progressBar.setValue(0);
601                     } else if ("message".equals(propertyName)) {
602                         String text = (String) (evt.getNewValue());
603                         statusMessageLabel.setText((text == null) ? "" :
text);
604                         messageTimer.restart();
605                     } else if ("progress".equals(propertyName)) {
606                         int value = (Integer) (evt.getNewValue());
607                         progressBar.setVisible(true);
608                         progressBar.setIndeterminate(false);
609                         progressBar.setValue(value);
610                     }
611                 }
612             });
613
614             //Futtatjuk
615             task.execute();
616
617             //Normál esetben ez nem kell. Azonban ennél a futásnál ha nem
rakjuk bele, rögtön elindul egy külön szállban majd folytatódik az ezt hívó szál
futása
618             //viszont ez néha tĂşl hamar következik be és még nem indul el
az sql motor mikor már az őt hívó eljárás azt már használni akarja.
619             //így a folyamat blokkolódik míg le nem fut a doInBackground
620             task.get();
621         } catch (InterruptedException ex) {
622
java.util.logging.Logger.getLogger(CsomagElemzoView.class.getName()).log(Level.S
EVERE, null, ex);
623         } catch (ExecutionException ex) {
624
java.util.logging.Logger.getLogger(CsomagElemzoView.class.getName()).log(Level.S
EVERE, null, ex);
625         }
626     }
627
```

```
628     //Lefut, ha rákattintunk az adatbázis törlése gombra
629     private void
adatbazisTorlesItemActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_adatbazisTorlesItemActionPerformed
630         this.createDatabaseLayer();
631         logger.info("Az adatbázis törlése");
632         Task task = new Task(this.getApplication()) {
633                @Override
634                protected Object doInBackground() {
635                    int eredmeny = layer.adatbazisTorol();
636                    layer.selectExecute("select portforgalomtablatorol()");
637                    if (eredmeny == 0)
638                    {
639                        logger.warn("Nincs törölhető adat az
adatbázisban!");
640                    }
641                    else if(eredmeny > 0)
642                    {
643                        logger.info("Sikeresen töröltünk " + eredmeny +"
darab adatot az adatbázisból!");
644                    }
645                    else
646                    {
647                        logger.error("Adatbázis törlési hiba");
648                    }
649
650                    return new Integer(eredmeny);
651                }
652            };
653            task.addPropertyChangeListener(new
java.beans.PropertyChangeListener() {
654
655                public void propertyChange(java.beans.PropertyChangeEvent
evt) {
656                    String propertyName = evt.getPropertyName();
657                    if ("started".equals(propertyName)) {
658                        if (!busyIconTimer.isRunning()) {
659                            statusAnimationLabel.setIcon(busyIcons[0]);
660                            busyIconIndex = 0;
661                            busyIconTimer.start();
662                        }
663                        progressBar.setVisible(true);
664                        progressBar.setIndeterminate(true);
665                    } else if ("done".equals(propertyName)) {
666                        busyIconTimer.stop();
667                        statusAnimationLabel.setIcon(idleIcon);
668                        progressBar.setVisible(false);
669                        progressBar.setValue(0);
670                    } else if ("message".equals(propertyName)) {
671                        String text = (String) (evt.getNewValue());
672                        statusMessageLabel.setText((text == null) ? "" :
text);
673                        messageTimer.restart();
674                    } else if ("progress".equals(propertyName)) {
675                        int value = (Integer) (evt.getNewValue());
676                        progressBar.setVisible(true);
677                        progressBar.setIndeterminate(false);
678                        progressBar.setValue(value);
679                    }
680                }
681            });
682
683            task.execute();
684            this.adatInit = false;
685     }//GEN-LAST:event_adatbazisTorlesItemActionPerformed
686
687     private void ipTerkepComponentShown(java.awt.event.ComponentEvent evt)
{//GEN-FIRST:event_ipTerkepComponentShown
```

```
688         //TODO Ez most ideiglenesen kikötve...
689         adatInit();
690 
691     }//GEN-LAST:event_ipTerkepComponentShown
692 
693     private boolean adatInit = false;
694     private void adatInit()
695     {
696         if(adatInit != true)
697         {
698             IpTerkepKeszito keszit = new IpTerkepKeszito(this.layer);
699             rajzLabel.setIcon(new
ImageIcon(keszit.rajzoljIpKapcsolatokat()));
700             CelIpMegoszlas forrasIpSzerintTabla = new CelIpMegoszlas(layer,
1);
701             forrasIpSzerint.setModel(forrasIpSzerintTabla.getTable());
702             CelIpMegoszlas celIpSzerintTabla = new CelIpMegoszlas(layer, 2);
703             celIpSzerint.setModel(celIpSzerintTabla.getTable());
704             CelIpMegoszlas ipKozottiKapcsolatTabla = new
CelIpMegoszlas(layer, 3);
705             ipKozottiKapcsolat.setModel(ipKozottiKapcsolatTabla.getTable());
706             PortonkentiMennyisegPie abra = new
PortonkentiMennyisegPie(this.layer, PortonkentiMennyisegPie.TCP);
707             portonkentiMennyisegLinePie1AbraLabel.setIcon(new
ImageIcon(abra.getKep()));
708             PortonkentiMennyisegPie abra2 = new
PortonkentiMennyisegPie(this.layer, PortonkentiMennyisegPie.UDP);
709             portonkentiMennyisegLinePie2AbraLabel.setIcon(new
ImageIcon(abra2.getKep()));
710             PortonkentiMennyisegPie abra3 = new
PortonkentiMennyisegPie(this.layer, PortonkentiMennyisegPie.MIND);
711             portonkentiMennyisegLinePie3AbraLabel.setIcon(new
ImageIcon(abra3.getKep()));
712             HalozatForgalma abra4 = new HalozatForgalma(this.layer);
713             halozatForgalmaAbraLabel.setIcon(new ImageIcon(abra4.getKep()));
714             KapcsolatokSzama abra5 = new KapcsolatokSzama(this.layer);
715             kapcsolatokSzamaAbraLabel.setIcon(new
ImageIcon(abra5.getKep()));
716             adatInit = true;
717         }
718     }
719 
720     private void jPanel1ComponentShown(java.awt.event.ComponentEvent evt)
{//GEN-FIRST:event_jPanel1ComponentShown
721 
722     }//GEN-LAST:event_jPanel1ComponentShown
723 
724     // Variables declaration - do not modify//GEN-BEGIN:variables
725     private javax.swing.JMenu adatbazisMenu;
726     private javax.swing.JMenuItem adatbazisTorlesItem;
727     private javax.swing.JTable celIpSzerint;
728     private javax.swing.JTable forrasIpSzerint;
729     private javax.swing.JLabel halozatForgalmaAbraLabel;
730     private javax.swing.JTable ipKozottiKapcsolat;
731     private javax.swing.JPanel ipTerkep;
732     private javax.swing.JLabel jLabel1;
733     private javax.swing.JMenuItem jMenuItem1;
734     private javax.swing.JPanel jPanel1;
735     private javax.swing.JPanel jPanel2;
736     private javax.swing.JPanel jPanel3;
737     private javax.swing.JPanel jPanel4;
738     private javax.swing.JPanel jPanel5;
739     private javax.swing.JScrollPane jScrollPane1;
740     private javax.swing.JScrollPane jScrollPane2;
741     private javax.swing.JScrollPane jScrollPane3;
742     private javax.swing.JScrollPane jScrollPane4;
743     private javax.swing.JScrollPane jScrollPane5;
744     private javax.swing.JLabel kapcsolatokSzamaAbraLabel;
```

```
745     private javax.swing.JPanel mainPanel;
746     private javax.swing.JMenuBar menuBar;
747     private javax.swing.JLabel portonkentiMennyisegLineAbraLabel;
748     private javax.swing.JLabel portonkentiMennyisegLinePie1AbraLabel;
749     private javax.swing.JLabel portonkentiMennyisegLinePie2AbraLabel;
750     private javax.swing.JLabel portonkentiMennyisegLinePie3AbraLabel;
751     private javax.swing.JProgressBar progressBar;
752     private javax.swing.JLabel rajzLabel;
753     private javax.swing.JLabel statusAnimationLabel;
754     private javax.swing.JLabel statusMessageLabel;
755     private javax.swing.JPanel statusPanel;
756     private javax.swing.JTabbedPane tabPanel;
757     // End of variables declaration//GEN-END:variables
758     private final Timer messageTimer;
759     private final Timer busyIconTimer;
760     private final Icon idleIcon;
761     private final Icon[] busyIcons = new Icon[15];
762     private int busyIconIndex = 0;
763     private JDialog aboutBox;
764 }
001 package csomagelemzo.util;
002 import java.io.*;
003 import java.util.ArrayList;
004 import org.apache.log4j.Logger;
005
006 /**
007  * Beolvassa az adott file tartalmát és elkészíti az abban szereplő csomagok
tároló osztályát.
008  * @author fricci
009  */
010     public class FileOlvaso
011     {
012         private BufferedReader fileReader;
013         private Logger logger;
014         /**
015          * Létrehoz egy Ăşj FileOlvaso objektumot. Paraméterként a
megnyitandó File objektumot vár.
016          * @param fileNev A megnyitandó File objektum.
017          */
018         public FileOlvaso(File fileNev)
019         {
020         try {
021             logger = Logger.getLogger("csomagelemzo");
022             this.fileReader = new BufferedReader(new FileReader(fileNev));
023         } catch (FileNotFoundException ex) {
024             logger.info("Nem sikerült a file-t megnyitni!");
025         }
026
027
028         }
029
030         /**
031          * Ez a függvény végzi szövegfile feldolgozását. Egy
ArrayList<HalozatCsomag> objektummal tér vissza.
032          * @return HalozatCsomag-okat tartalmazó ArrayList
033          */
034         public ArrayList<HalozatCsomag> getCsomagok()
035         {
036             logger.info("Csomagok gyűjtése.");
037             try
038             {
039             ArrayList<HalozatCsomag> csomagok = new
ArrayList<HalozatCsomag>();
040             String egySor;
041             int sor = 1;
042             while ((egySor = fileReader.readLine()) != null) //amíg van sor
a file-ban, minden sorra csináljuk meg
043             {
```

```
044
045                 //logger.info("Ăšjabb sor olvasása: "+sor+". sor");
046                 if ((egySor.trim()).length() != 0)  //trimeljük, ha Ăşgy is
hosszabb a sor hossza 0-nál
047                 {
048                     HalozatCsomag csomag = sorFeldolgoz(egySor);
//feldolgozzuk a sort
049                     if (csomag != null)
050                         csomagok.add(sorFeldolgoz(egySor));  //Ha
létrehozott egy csomagot, hozzáadjuk.
051                 }
052                 sor++;
053             }
054             fileReader.close();
055             logger.info("Csomagok gyűjtése befejeződött.");
056             return csomagok;
057             }
058             catch (Exception e)
059             {logger.info("Hiba történt a csomagok gyűjtése közben.");
060             return null;
061             }
062         }
063
064         /**
065          * Feldolgoz egy sor-t. Paraméterül egy String-et kap, ami a
szövegfile egy sora.
066          * @param egySor A szövegfile egy sora
067          * @return A sorból készített HalozatCsomag objektum
068          */
069         private HalozatCsomag sorFeldolgoz(String egySor)
070         {
071             try
072             {
073             HalozatCsomag egyCsomag;
074             String[] szavak = egySor.split(" ");
075             if (szavak[1].equals("arp")) return null;
076             String rawIdoBelyeg = szavak[0].substring(0, 10);
077             long idoBelyeg = Long.parseLong(rawIdoBelyeg);
078             idoBelyeg *= 1000;
079             rawIdoBelyeg = szavak[0].substring(11, 14);
080             idoBelyeg += Long.parseLong(rawIdoBelyeg);
081             String forrasIp = new String(getIp(szavak[2])[0]);
082             int forrasPort = Integer.parseInt((getIp(szavak[2])[1]));
083             String celIp = new String(getIp(szavak[4])[0]);
084             String rawCelPort = (getIp(szavak[4])[1]);
085             rawCelPort = rawCelPort.replace(':', ' ');
086             rawCelPort = rawCelPort.trim();
087             int celPort = Integer.parseInt(rawCelPort);
088             String rawProtokol = szavak[5];
089             byte protokol = 0;
090             short csomagMeret = 0;
091             if (rawProtokol.equals("tcp"))
092             {
093                 protokol = 1;
094                 csomagMeret = Short.parseShort(szavak[6]);
095             }
096             else if (rawProtokol.equals("UDP,"))
097             {
098                 protokol = 2;
099                 csomagMeret = Short.parseShort(szavak[7]);
100             }
101             else return null;
102             egyCsomag = new HalozatCsomag(idoBelyeg, protokol, forrasIp,
celIp, forrasPort, celPort, csomagMeret);
103             return egyCsomag;
104             }
105             catch (Exception e)
106             {
```

```
107                 e.printStackTrace();
108                 logger.info("Hiba történt a sor feldolgozása közben");
109                 return null;}
110         }
111 
112         /**
113          * Egy String-ből felbontja az IP-t. A szövegfile-ban ip.port
formában van (pl: 192.168.1.1.80). Ezt bontja fel két elemü String tömbre:
114          * String[0] = 192.168.1.1
115          * String[1] = 80
116          * @param rawIp ip.port formában a felbontani kívánt ip
117          * @return Két elemű tömb amiben az ip és a port külön lett
választva.
118          */
119         private  String[] getIp(String rawIp)
120         {
121             try
122             {
123                 String[] eredmeny = new String[2];
124                 int pontHelye = rawIp.lastIndexOf('.');
125                 eredmeny[0] = rawIp.substring(0, pontHelye);
126                 eredmeny[1] = rawIp.substring(pontHelye + 1);
127                 return eredmeny;
128             }
129             catch (Exception e)
130             {
131                 logger.error("Hiba a következő sorban: "+rawIp);
132 
133                 System.exit(1);
134                 return null;
135             }
136         }
137     }
001 /*
002  * To change this template, choose Tools | Templates
003  * and open the template in the editor.
004  */
005 
006 package csomagelemzo.util;
007 import java.util.logging.Level;
008 import org.apache.log4j.Logger;
009 import java.util.Date;
010 import java.util.GregorianCalendar;
011 
012 /**
013  * Egy hálózati csomagot reprezentál, csak tároló osztály, egyéb funkciója
nincs.
014  * @author fricci
015  */
016 public class HalozatCsomag {
017         private GregorianCalendar datum;
018 
019     public String getCelIp() {
020         return celIp;
021     }
022 
023     public int getCelPort() {
024         return celPort;
025     }
026 
027     public short getCsomagMeret() {
028         return csomagMeret;
029     }
030 
031     public GregorianCalendar getDatum() {
032         return datum;
033     }
034 
```

```
035     public String getForrasIp() {
036         return forrasIp;
037     }
038 
039     public int getForrasPort() {
040         return forrasPort;
041     }
042 
043     public long getIpHashOsszeg() {
044         return ipHashOsszeg;
045     }
046 
047     public long getIpHashSzorzat() {
048         return ipHashSzorzat;
049     }
050 
051     public long getPortHashOsszeg() {
052         return portHashOsszeg;
053     }
054 
055     public long getPortHashSzorzat() {
056         return portHashSzorzat;
057     }
058 
059     public byte getProtokoll() {
060         return protokoll;
061     }
062         private byte protokoll; //Tcp: 1; Udp: 2
063         private String forrasIp;
064         private String celIp;
065         private int forrasPort;
066         private int celPort;
067         private short csomagMeret;
068         private long ipHashOsszeg;
069         private long ipHashSzorzat;
070         private long portHashOsszeg;
071         private long portHashSzorzat;
072         private Logger logger;
073 
074         /**
075          * Létrehoz egy hálózat csomagot.
076          * @param datum A csomag rögztésének dátuma
077          * @param protokoll A protokoll típusa (Tcp: 1, Udp: 2)
078          * @param forrasIp A csomag forrás ip-je
079          * @param celIp A csomag cél ip-je
080          * @param forrasPort A csomag forrás portja
081          * @param celPort A csomag cél portja
082          * @param csomagMeret a Csomag mérete
083          */
084         public HalozatCsomag(long datum, byte protokoll, String forrasIp,
String celIp,
085             int forrasPort, int celPort, short csomagMeret)
086         {
087             logger = Logger.getLogger("csomagelemzo");
088             /* Ez a rész végzi a long-ot  DateTime-ra való konvertálást,
089              */
090             this.datum = new GregorianCalendar();
091             this.datum.setTime(new Date(datum));
092             this.protokoll = protokoll;
093             this.forrasIp = forrasIp;
094             this.celIp = celIp;
095             this.forrasPort = forrasPort;
096             this.celPort = celPort;
097             this.csomagMeret = csomagMeret;
098 
099             //this.ipHashOsszeg = this.forrasIp.hashCode() +
this.celIp.hashCode();
100             this.ipHashOsszeg = 0;
```

```
101             this.ipHashOsszeg = this.getHashSzam(forrasIp) +
this.getHashSzam(celIp);
102             //this.ipHashSzorzat = this.forrasIp.hashCode() *
this.celIp.hashCode();
103             this.ipHashSzorzat = 0;
104             this.ipHashSzorzat = (this.getHashSzam(forrasIp) *
this.getHashSzam(celIp));
105             //logger.info("iphashszorzat "+this.ipHashSzorzat);
106
107             this.portHashOsszeg = this.celPort + this.forrasPort;
108             this.portHashSzorzat = this.celPort * this.forrasPort;
109         //    System.exit(2);
110
111
112         }
113
114         private long getHashSzam(String ip)
115         {
116             String[] szavak = ip.split("\\.");
117             try
118             {
119                 long eredmeny = 0;
120                 ip = ip.trim();
121                //logger.info("Long.valueOf(szavak[3])
"+Long.valueOf(szavak[3]));
122                 eredmeny = Long.valueOf(szavak[3]);
123                // logger.info("Long.valueOf(szavak[2])*1000
"+Long.valueOf(szavak[2])*10);
124                 eredmeny += Long.valueOf(szavak[2])*10;
125                // logger.info("Long.valueOf(szavak[1])*1000000
"+Long.valueOf(szavak[1])*100);
126                 eredmeny += Long.valueOf(szavak[1])*100;
127                // logger.info("Long.valueOf(szavak[0])*1000000000
"+Long.valueOf(szavak[0])*1000);
128                 eredmeny += Long.valueOf(szavak[0])*1000;
129                // logger.info("Az ip cím: "+ip + " ,hozzá a hash:
"+eredmeny);
130                 return eredmeny;
131             }
132             catch (IndexOutOfBoundsException ex)
133             {
134                 logger.error("Index tĂşlfutási hiba");
135                 logger.error("Bemenő paraméter: " + ip);
136                 logger.error("Tömb mérete: "+szavak.length);
137                 for(String st : szavak)
138                 {
139                    logger.error(st);
140                 }
141                 System.exit(1);
142                 return 0;
143             }
144         }
145
146 }
01 /*
02  * To change this template, choose Tools | Templates
03  * and open the template in the editor.
04  */
05
06 package csomagelemzo.util.lekerdezesek;
07 import csomagelemzo.util.SqlJavaLayer;
08 import csomagelemzo.util.abra.LineChart;
09 import java.awt.image.BufferedImage;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.util.ArrayList;
13 import org.apache.log4j.*;
14 /**
```

```
15  *
16  * @author fricci
17  */
18 public class HalozatForgalma extends AbstractLekerdezes{
19     LineChart abra ;
20     public HalozatForgalma(SqlJavaLayer layer)
21     {
22         super(layer,
23             "select date_trunc('minute', idobelyeg), sum(meret) from csomagok
" +
24             "group by date_trunc('minute', idobelyeg) " +
25             "order by date_trunc('minute', idobelyeg)"
26             );
27     }
28
29     @Override
30     protected void initAbra() {
31         try
32         {
33             final ResultSet  eredmeny =
layer.selectExecute(lekerdezesString);
34             final ArrayList adatok = new ArrayList();
35             int sorokSzama = 300;
36             while(eredmeny.next())
37             {
38                 ArrayList sor = new ArrayList();
39                 sor.add(eredmeny.getTimestamp(1).getTime());
40                 sor.add(eredmeny.getInt(2));
41                 adatok.add(sor);
42                 if(sorokSzama<6500)
43                 sorokSzama++;
44             }
45             this.abra = new LineChart(adatok, this.magyarazoSzoveg, "Ă?tvitt
adat", "Byte-ok", "Idő", sorokSzama, true);
46             eredmeny.close();
47             //this.abra.saveChart(null);
48         }
49         catch(SQLException e)
50         {
51             logger.error("Hiba a lekérdezés közben: "+e.getSQLState());
52         }
53     }
54
55     @Override
56     public BufferedImage getKep() {
57         return this.abra.getChart();
58     }
59
60
61 }
01 /*
02  * To change this template, choose Tools | Templates
03  * and open the template in the editor.
04  */
05
06 package csomagelemzo.util;
07 import java.awt.Point;
08 import java.util.ArrayList;
09 /**
10  * Az IpTerkep egy "címét" reprezentáló osztály, csak tárolásra alkalmas, nem
végez
11  * semmilyen tevékenységet.
12  * @author fricci
13  */
14 public class IpTerkepKapcsolat {
15     private Point koordinata;
16     private ArrayList<IpTerkepKapcsolat> kapcsolatok; //Ezzel a csomóponttal
kapcsolatban lévő csomópontok.
```

```
17      private String ipCim;
18
19      /**
20       * Csomópontok létrehozása.
21       * @param ipCim A csomópont ip címe String formátumban
22       * @param koordinata A térképen található csomópont koordinátája.
23       */
24      public IpTerkepKapcsolat(String ipCim, Point koordinata)
25      {
26          this.koordinata = koordinata;
27          this.ipCim = ipCim;
28          this.kapcsolatok = new ArrayList();
29      }
30
31      /**
32       * Csomópont létrehozása, koordináta helyett külön x és y értéket adunk
át a paraméterként.
33       * @param ipCim A csomópont ip címe String formátumban
34       * @param x koordináta X értéke
35       * @param y koordináta Y értéke
36       */
37      public IpTerkepKapcsolat(String ipCim, int x, int y)
38      {
39          this(ipCim, new Point(x, y));
40      }
41
42      /**
43       * Hozzáad ehhez a csomóponthoz egy másik csomópontot mint kapcsolatot.
44       * @param kapcsolat A kapcsolatként kiépítendő csomópont.
45       */
46      public void addKapcsolat(IpTerkepKapcsolat kapcsolat)
47      {
48          this.kapcsolatok.add(kapcsolat);
49      }
50
51      /**
52       * Lekéri a kapcsolatokat tartalmazó ArrayList-et
53       * @return Ezzel a csomópontokkal kapcsolatban lévő csomópontokat
tartalmazó ArrayList
54       */
55      public ArrayList<IpTerkepKapcsolat> getKapcsolat()
56      {
57          return this.kapcsolatok;
58      }
59
60      @Override
61      public String toString()
62      {
63          return this.ipCim;
64      }
65
66      /**
67       * Lekéri a csomópont koordinátáját.
68       * @return
69       */
70      public Point getKoordinata()
71      {
72          return this.koordinata;
73      }
74
75  }
001  /*
002   * To change this template, choose Tools | Templates
003   * and open the template in the editor.
004   */
005
006  package csomagelemzo.util;
007
```

```
008 import java.awt.BasicStroke;
009 import java.awt.Color;
010 import java.awt.Font;
011 import java.awt.Graphics2D;
012 import java.awt.Point;
013 import java.awt.RenderingHints;
014 import java.awt.image.BufferedImage;
015 import java.sql.ResultSet;
016 import java.sql.SQLException;
017 import java.util.ArrayList;
018 import java.util.Enumeration;
019 import java.util.Hashtable;
020 import java.util.Iterator;
021 import java.util.Random;
022 import org.apache.log4j.*;
023
024 /**
025  * Kirajzolja az adatbázisban lévő ip címek IP térképét. A konstruktor
paraméterének egy SqlJavaLayer objektumot vár
026  * amin keresztül lekéri a szükséges adatokat. A létrehozott IpTerkepKeszito
objektumnak egyetlen függvénye van, a paraméter
027  * nélküli RajzoljIpKapcsolatokat mely visszatér a megrajzolt ip térképet
tartalmazó BufferedImage-el.
028  * @author fricci
029  */
030 public class IpTerkepKeszito {
031     private SqlJavaLayer layer;
032     private BufferedImage kep;
033     private Hashtable forgalomTabla;
034     private long osszForgalom;
035     private Logger logger;
036     Graphics2D g;
037
038
039 /**
040  * Elkészíti az IpTerkepKeszito osztály egy objektumát. Paraméterként egy
SqlJavaLayer objektumot
041  * vár, amin keresztül lekérheti a szükséges adatokat az adatábzisból
042  * @param layer Az adatbázishoz való csatlakozáshoz kell az SqlJavaLayer
043  */    public IpTerkepKeszito(SqlJavaLayer layer)
044     {
045         logger = Logger.getLogger("csomagelemzo");
046         this.layer = layer;
047         forgalomTabla = new Hashtable();
048         osszForgalom = 0;
049         osszForgalom = getOsszMennyiseg();
050         kep = new BufferedImage(1000, 1000, BufferedImage.TYPE_INT_ARGB);
051         g = kep.createGraphics();
052         kep.setRGB(100, 100, 100);
053         g.setBackground(new Color(0,0,0,0));
054         RenderingHints beal = new
RenderingHints(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
055         beal.put(RenderingHints.KEY_RENDERING,
RenderingHints.VALUE_RENDER_QUALITY);
056         g.setRenderingHints(beal);
057         BasicStroke stroke = new BasicStroke(2.0f, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND, 10.0f);
058         g.setStroke(stroke);
059     }
060
061     //Belső függvény, lekéri az adatbázisban lévő csomagok összméretét
062     private long getOsszMennyiseg()
063     {
064         try {
065             long valaszLong;
066             String parancs = "select sum(meret) from csomagok";
067             ResultSet valasz = layer.selectExecute(parancs);
```

```
068             if (valasz.next()) {
069                 valaszLong = Long.parseLong(valasz.getString(1));
070 
071                 valasz.close();
072                 return valaszLong;
073             }
074             return 1;
075         } catch (SQLException ex) {
076             logger.error("Hiba a lekérdezés során: " + ex.getSQLState());
077         } catch (NumberFormatException ex) {
078             logger.warn("Hiba a mennyiség lekérése közben. Ez normális
amennyiben az adatbázis üres volt");
079         }
080         return -1;
081     }
082 
083     //Belső függvény, mely két ip cím köztötti vonal kapcsolat színét adja
vissza. A szín függ
084     //attól, hogy az összes forgalomból mennyi folyt át ezen a kapcsolaton
085     private Color getVonalSzin(String bejegyzes, String kapcsolat)
086     {
087         Color vonal;
088         double szazalek =
(Double)(((Hashtable)forgalomTabla.get(bejegyzes)).get(kapcsolat));
089         int szinErtek = (int)szazalek;
090         int zold;
091         int voros;
092         if (szinErtek <= 50)
093         {
094             zold = 255;
095             voros = (255 / 100)*szinErtek;
096         }
097         else
098         {
099             voros = 255;
100             zold = 255 - (255 / 100) * szinErtek;
101         }
102         vonal = new Color(voros, zold, 17);
103         return vonal;
104     }
105 
106     /**
107      * A tulajdonképeni munkát végző eljárás. Az SqlJavaLayer alapján
108      * @return
109      */
110     public BufferedImage rajzoljIpKapcsolatokat()
111     {
112         try
113         {
114         String parancs = "select distinct ipTabla.celip, ipTabla.forrasip,
kapcsTabla.mennyiseg from ( select iphashosszeg, iphashszorzat, sum(meret) as
mennyiseg from csomagok   group by iphashosszeg, iphashszorzat ) as kapcsTabla,
csomagok as ipTabla where ipTabla.iphashosszeg = kapcsTabla.iphashosszeg and
ipTabla.iphashszorzat = kapcsTabla.iphashszorzat";
115         // TODO Valamikép jelezni a felhasználói felületen, hogy a háttérben
munka folyik
116         //Jelezni hogy dolog van
117         ResultSet valasz = layer.selectExecute(parancs);
118         //idáig
119         Hashtable tabla = new Hashtable();
120         Random veletlen = new Random();
121         while (valasz.next())
122         {
123             String elsoIp = valasz.getString(1);
124             String masodikIp = valasz.getString(2);
125             IpTerkepKapcsolat forrasBejegyzes;
126             IpTerkepKapcsolat celBejegyzes;
127             if (!forgalomTabla.containsKey(elsoIp))
```

```
128             {
129                 double szazalek =
((double)valasz.getInt(3))/((double)osszForgalom)*100;
130                 Hashtable t = new Hashtable();
131                 t.put(masodikIp, szazalek);
132                 forgalomTabla.put(elsoIp, t);
133             }
134             else
135             {
136                 if
(!((Hashtable)forgalomTabla.get(elsoIp)).containsKey(masodikIp))
137                 {
138                     double szazalek = ((double)valasz.getInt(3)) /
((double)osszForgalom) * 100;
139                     ((Hashtable)forgalomTabla.get(elsoIp)).put(masodikIp,
szazalek);
140                 }
141             }
142                 if (!tabla.containsKey(elsoIp))
143                 {
144                     //létrehozom
145                     forrasBejegyzes = new IpTerkepKapcsolat(elsoIp, new
Point(veletlen.nextInt(1000), veletlen.nextInt(1000)));
146                     tabla.put(elsoIp, forrasBejegyzes);
147                 }
148                 else
149                 {
150                     //lekérem
151                     forrasBejegyzes = (IpTerkepKapcsolat)tabla.get(elsoIp);
152                 }
153                 if (!tabla.containsKey(masodikIp))
154                 {
155                     //létrehozom
156                     celBejegyzes = new IpTerkepKapcsolat(masodikIp, new
Point(veletlen.nextInt(1000), veletlen.nextInt(1000)));
157                     tabla.put(masodikIp, celBejegyzes);
158                 }
159                 else
160                 {
161                     celBejegyzes = (IpTerkepKapcsolat)tabla.get(masodikIp);
162                     //lekérem
163                 }
164                 celBejegyzes.addKapcsolat(forrasBejegyzes);
165
166             }
167
168             Font drawFont = new Font("Curier New",Font.PLAIN, 10);
169             g.setFont(drawFont);
170             Enumeration kulcsok = tabla.keys();
171             while (kulcsok.hasMoreElements())
172             {
173                 IpTerkepKapcsolat egyBejegyzes =
(IpTerkepKapcsolat)tabla.get(kulcsok.nextElement());
174                 ArrayList<IpTerkepKapcsolat> bejegyzesKapcs =
egyBejegyzes.getKapcsolat();
175                 Iterator bejegyzesFelsorolas = bejegyzesKapcs.iterator();
176                 while (bejegyzesFelsorolas.hasNext())
177                 {
178                     IpTerkepKapcsolat egyKapcsolat =
(IpTerkepKapcsolat)bejegyzesFelsorolas.next();
179                     g.setColor(getVonalSzin(egyKapcsolat.toString(),
egyBejegyzes.toString()));
180                     g.drawLine(egyBejegyzes.getKoordinata().x,
egyBejegyzes.getKoordinata().y, egyKapcsolat.getKoordinata().x,
egyKapcsolat.getKoordinata().y);
181                     g.setColor(Color.BLACK);
182                     g.drawString(egyBejegyzes.toString(),
egyBejegyzes.getKoordinata().x, egyBejegyzes.getKoordinata().y);
```

```
183                     g.drawString(egyKapcsolat.toString(),
egyKapcsolat.getKoordinata().x, egyKapcsolat.getKoordinata().y);
184                 }
185
186             }
187                 valasz.close();
188             }
189             catch (SQLException e)
190             {
191                 logger.info("Hiba a lekérdezés közben");
192             }
193             return kep;
194         }
195
196 }
01 /*
02  * To change this template, choose Tools | Templates
03  * and open the template in the editor.
04  */
05
06 package csomagelemzo.util.lekerdezesek;
07 import csomagelemzo.util.SqlJavaLayer;
08 import csomagelemzo.util.abra.LineChart;
09 import java.awt.image.BufferedImage;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.util.ArrayList;
13 import org.apache.log4j.*;
14
15 /**
16  *
17  * @author fricci
18  */
19 public class KapcsolatokSzama extends AbstractLekerdezes{
20
21     private LineChart abra;
22
23     public KapcsolatokSzama(SqlJavaLayer layer)
24     {
25         super(layer,
26                 "select date_trunc('minute', idobelyeg),
count(date_trunc('minute', idobelyeg)) " +
27     " from " +
28     " ( " +
29         " select " +
30         " distinct date_trunc('minute', idobelyeg) as idobelyeg, " +
31         " iphashosszeg, " +
32         " iphashszorzat, " +
33         " porthashosszeg, " +
34         " porthashszorzat " +
35         " from csomagok " +
36     ") " +
37     " as altabla " +
38     " group by altabla.idobelyeg "+
39     " order by altabla.idobelyeg "
40                 );
41     }
42
43     @Override
44     protected void initAbra() {
45         try
46         {
47             final ResultSet  eredmeny =
layer.selectExecute(lekerdezesString);
48             final ArrayList adatok = new ArrayList();
49             int sorokSzama = 300;
50             while(eredmeny.next())
51             {
```

```
52                  ArrayList sor = new ArrayList();
53                  sor.add(eredmeny.getTimestamp(1).getTime());
54                  sor.add(eredmeny.getInt(2));
55                  adatok.add(sor);
56                  if(sorokSzama<6500)
57                  sorokSzama++;
58              }
59              this.abra = new LineChart(adatok, this.magyarazoSzoveg,
"Kapcsolatok száma", "Kapcsolatok", "Idő", sorokSzama, true);
60              eredmeny.close();
61              //abra.saveChart(null);
62          }
63          catch(SQLException e)
64          {
65              logger.error("Hiba a lekérdezés közben: "+e.getSQLState());
66          }
67      }
68
69      @Override
70      public BufferedImage getKep() {
71          return this.abra.getChart();
72      }
73  }
001
002  /*
003   * To change this template, choose Tools | Templates
004   * and open the template in the editor.
005   */
006
007  package csomagelemzo.util.abra;
008  import java.sql.Timestamp;
009  import java.util.ArrayList;
010  import java.util.Iterator;
011  import org.jfree.chart.ChartFactory;
012  import org.jfree.chart.plot.PlotOrientation;
013  import org.jfree.data.xy.IntervalXYDataset;
014  import org.jfree.data.xy.XYDataset;
015  import org.jfree.data.xy.XYSeries;
016  import org.jfree.data.xy.XYSeriesCollection;
017  /**
018   * Egy vonal grafikont reprezentál.
019   * @author fricci
020   */
021  public class LineChart extends AbstractAbra
022  {
023
024      private XYSeries dataset;
025
026      /**
027       * Létrehoz egy Ăşj vonal grafikont.
028       * az adatok-at tartalmazó ArrayList felépítése a következő:
029       * [
030       *      [függőleges] [vízszintes tengely értéke]
031       * ]
032       *
033       * @param adatok az adatok
034       * @param magyarazat A grafikonhoz tartozó magyarázat
035       * @param cim A grafikon címe
036       * @param xTengely Az x tengely címe
037       * @param yTengely Az y tengely címe
038       */
039      public LineChart(final ArrayList adatok, final String magyarazat, final
String cim, final String xTengely, final String yTengely, final boolean
idoTengely)
040      {
041          this(adatok, magyarazat, cim, xTengely, yTengely, 500, idoTengely);
042      }
043
```

```
044     /**
045      * Létrehoz egy Ăsj vonal grafikont.
046      * az adatok-at tartalmazó ArrayList felépítése a következő:
047      * [
048      *      [függőleges] [vízszintes tengely értéke]
049      * ]
050      *
051      * @param adatok az adatok
052      * @param magyarazat A grafikonhoz tartozó magyarázat
053      * @param cim A grafikon címe
054      * @param xTengely Az x tengely címe
055      * @param yTengely Az y tengely címe
056      * @param szelesseg A grafikon szélessége
057      * @param idoTengely True ha időtengely szerepel vízszintes irányban
058      */
059     public LineChart(final ArrayList adatok, final String magyarazat, final
String cim, final String xTengely, final String yTengely, final int szelesseg,
final boolean idoTengely)
060     {
061         super(magyarazat, szelesseg);
062         dataset = new XYSeries(cim);
063         Iterator it = adatok.iterator();
064         while(it.hasNext())
065         {
066             ArrayList sor = (ArrayList)it.next();
067             dataset.add((Number)sor.get(0), (Number)sor.get(1));
068         }
069         XYDataset xyDataset = new XYSeriesCollection(dataset);
070         if(idoTengely == true)
071         {
072             abra = ChartFactory.createTimeSeriesChart
073                         (
074                             cim,
075                             xTengely,
076                             yTengely,
077                             xyDataset,
078                             true,
079                             true,
080                             true
081                         );
082 
083         }
084         else
085         {
086             abra = ChartFactory.createXYAreaChart
087                          (cim,  // Title
088                           xTengely,           // X-Axis label
089                           yTengely,           // Y-Axis label
090                           xyDataset,          // Dataset
091                           PlotOrientation.VERTICAL,
092                           true,                // Show legend
093                           true,
094                           true
095                          );
096         }
097         abra.setAntiAlias(true);
098         abra.setTextAntiAlias(true);
099         abra.getPlot().setForegroundAlpha(0.5f);
100         image = abra.createBufferedImage(this.szelesseg, 300);
101     }
102 }
001 /*
002  * To change this template, choose Tools | Templates
003  * and open the template in the editor.
004  */
005 
006 package csomagelemzo.util.abra;
007 
```

```
008 import java.util.ArrayList;
009 import java.util.Iterator;
010 import org.jfree.chart.ChartFactory;
011 import org.jfree.chart.plot.PlotOrientation;
012 import org.jfree.data.category.DefaultCategoryDataset;
013 import org.jfree.data.time.TimeSeries;
014 import org.jfree.data.time.TimeSeriesCollection;
015
016 /**
017  * Több vonalas vonalgrafikont készít.
018  * @author fricci
019  */
020 public class MultiLineChart extends AbstractAbra
021 {
022
023     /**
024      * Létrehoz egy több vonalas vonalgrafikont. Az adatokat a következő
módon kell átadni neki:
025      *      ArrayList felépítése:
026      *      [
027      *          [függőleges tengely értéke][kategória][vízszintes tengely
értéke]
028      *      ]
029      *
030      * @param adatok amiből a grafikont készíti
031      * @param magyarazat A grafikonhoz tartozó magyarázat
032      * @param cim A grafikon címe
033      * @param xTengely Az x tengely címe
034      * @param yTengely Az y tengely címe
035      */
036     public MultiLineChart(ArrayList adatok, String magyarazat, String cim,
String xTengely, String yTengely, boolean idoTengely)
037     {
038         this(adatok, magyarazat, cim, xTengely, yTengely, 500, idoTengely);
039     }
040
041     /**
042      * Létrehoz egy több vonalas vonalgrafikont. Az adatokat a következő
módon kell átadni neki:
043      *      ArrayList felépítése:
044      *      [
045      *          [függőleges tengely értéke][kategória][vízszintes tengely
értéke]
046      *      ]
047      *
048      * @param adatok amiből a grafikont készíti
049      * @param magyarazat A grafikonhoz tartozó magyarázat
050      * @param cim A grafikon címe
051      * @param xTengely Az x tengely címe
052      * @param yTengely Az y tengely címe
053      * @param szelesseg A grafikon szélessége
054      */
055     public MultiLineChart(ArrayList adatok, String magyarazat, String cim,
String xTengely, String yTengely, int szelesseg, boolean idoTengely)
056     {
057         super(magyarazat, szelesseg);
058         DefaultCategoryDataset dataset = new DefaultCategoryDataset();
059         Iterator it = adatok.iterator();
060         while(it.hasNext())
061         {
062             ArrayList sor = (ArrayList)it.next();
063             dataset.addValue((Number)sor.get(0), (Comparable)sor.get(1),
(Comparable)sor.get(2));
064         }
065         if(idoTengely == true)
066         {
067
068
```

```
069
070             /*
071             abra = ChartFactory.createTimeSeriesChart
072                         (
073                             cim,
074                             xTengely,
075                             yTengely,
076                             dataset2,
077                             true,
078                             true,
079                             true
080                         );*/
081         }
082         else
083         {
084             this.abra = ChartFactory.createAreaChart
085                          (cim,  // Title
086                           xTengely,           // X-Axis label
087                           yTengely,           // Y-Axis label
088                           dataset,          // Dataset
089                           PlotOrientation.VERTICAL,
090                           true,                // Show legend
091                           true,
092                           true
093                          );
094         }
095         this.abra.setAntiAlias(true);
096         this.abra.setTextAntiAlias(true);
097         abra.getPlot().setForegroundAlpha(0.5f);
098         image = abra.createBufferedImage(this.szelesseg, 300);
099     }
100 }
01 /*
02  * To change this template, choose Tools | Templates
03  * and open the template in the editor.
04  */
05
06 package csomagelemzo.util.abra;
07
08 import java.util.ArrayList;
09 import java.util.Iterator;
10 import org.jfree.chart.ChartFactory;
11 import org.jfree.data.general.DefaultPieDataset;
12
13 /**
14  * Létrehoz egy körcikk grafikont.
15  * @author fricci
16  */
17 public class PieChart extends AbstractAbra
18 {
19
20     /**
21      * Létrehoz egy körcikk grafikont. Az adatokat a következő formában kell
neki
22      * átadni:
23      *      [
24      *          [Kategória neve][Százalékos értéke]
25      *      ]
26      *
27      * @param adatok Az adat, amiből a körcikket csinálja
28      * @param magyarazat
29      * @param cim
30      */
31     public PieChart(ArrayList adatok, String magyarazat, String cim)
32     {
33         super(magyarazat, 500);
34         final DefaultPieDataset pieDataset = new DefaultPieDataset();
35         final Iterator it = adatok.iterator();
```

```
36          while(it.hasNext())
37          {
38              ArrayList adat = (ArrayList)it.next();
39              pieDataset.setValue((Comparable)adat.get(0),
(Number)adat.get(1));
40          }
41          this.abra = ChartFactory.createPieChart3D(cim, pieDataset, true,
false, false);
42          abra.setAntiAlias(true);
43          abra.setTextAntiAlias(true);
44          abra.getPlot().setForegroundAlpha(0.5f);
45          image = abra.createBufferedImage(this.szelesseg, 300);
46      }
47
48      public void setCim(String cim)
49      {
50          this.abra.setTitle(cim);
51      }
52
53  }
01  /*
02   * To change this template, choose Tools | Templates
03   * and open the template in the editor.
04   */
05
06  package csomagelemzo.util.lekerdezesek;
07
08  import csomagelemzo.util.SqlJavaLayer;
09  import csomagelemzo.util.abra.MultiLineChart;
10  import java.awt.image.BufferedImage;
11  import java.sql.ResultSet;
12  import java.sql.SQLException;
13  import java.util.ArrayList;
14
15  /**
16   *
17   * @author fricci
18   */
19  public class PortonkentiMennyisegLine extends AbstractLekerdezes{
20      private MultiLineChart abra;
21      public PortonkentiMennyisegLine(SqlJavaLayer layer)
22      {
23          super(layer,
24                  "select date_trunc('minute', idobelyeg), port, sum(mennyiseg)
from portforgalomtabla " +
25                  "group by date_trunc('minute', idobelyeg), port " +
26                  "order by date_trunc('minute', idobelyeg)"
27                  );
28      }
29
30      @Override
31      protected void initAbra() {
32          try
33          {
34              final ResultSet  eredmeny =
layer.selectExecute(lekerdezesString);
35              final ArrayList adatok = new ArrayList();
36              int sorokSzama = 300;
37              while(eredmeny.next())
38              {
39                  ArrayList sor = new ArrayList();
40                  sor.add(eredmeny.getTimestamp(1).getTime());
41                  sor.add(eredmeny.getInt(2));
42                  sor.add(eredmeny.getInt(3));
43                  adatok.add(sor);
44                  if(sorokSzama<6500)
45                  sorokSzama++;
46              }
```

```
47              this.abra = new MultiLineChart(adatok, this.magyarazoSzoveg,
"Ă?tvitt adat portonként az idő függvényében.", "Kapcsolatok", "Idő",
sorokSzama, false);
48              eredmeny.close();
49              //abra.saveChart(null);
50          }
51          catch(SQLException e)
52          {
53              logger.error("Hiba a lekérdezés közben: "+e.getSQLState());
54          }
55      }
56
57      @Override
58      public BufferedImage getKep() {
59          throw new UnsupportedOperationException("Not supported yet.");
60      }
61
62  }
01  /*
02   * To change this template, choose Tools | Templates
03   * and open the template in the editor.
04   */
05
06  package csomagelemzo.util.lekerdezesek;
07
08  import csomagelemzo.util.SqlJavaLayer;
09  import csomagelemzo.util.abra.MultiLineChart;
10  import csomagelemzo.util.abra.PieChart;
11  import java.awt.image.BufferedImage;
12  import java.sql.ResultSet;
13  import java.sql.SQLException;
14  import java.util.ArrayList;
15
16  /**
17   *
18   * @author fricci
19   */
20  public class PortonkentiMennyisegPie extends AbstractLekerdezes{
21      private PieChart abra;
22      private int protokoll;
23      public PortonkentiMennyisegPie(SqlJavaLayer layer)
24      {
25          super(layer,
26                  "select port, sum(mennyiseg) from portforgalomtabla "+
27                  "group by port"
28              );
29      }
30
31      public PortonkentiMennyisegPie(SqlJavaLayer layer, int protokoll)
32      {
33          super(layer,
34                  "select port, sum(mennyiseg) from portforgalomtabla "+
35                  AbstractLekerdezes.getProtokoll(protokoll) +
36                  "group by port"
37              );
38          this.protokoll = protokoll;
39          this.initPieAbra();
40
41      }
42
43
44
45      @Override
46      protected void initAbra()
47      {
48          //Nem csinálunk semmit
49      }
50
```

```
51      protected void initPieAbra() {
52          try
53          {
54              final ResultSet  eredmeny =
layer.selectExecute(lekerdezesString);
55              final ArrayList adatok = new ArrayList();
56              int sorokSzama = 300;
57              while(eredmeny.next())
58              {
59                  ArrayList sor = new ArrayList();
60                  sor.add(eredmeny.getInt(1));
61                  sor.add(eredmeny.getInt(2));
62                  adatok.add(sor);
63                  if(sorokSzama<6500)
64                  sorokSzama++;
65              }
66              if(protokoll == PortonkentiMennyisegPie.TCP)
67              {
68                  this.abra = new PieChart(adatok, this.magyarazoSzoveg,
"Ă?tvitt adat portonkénti megoszlása (TCP protokoll).");
69              }
70              else if(protokoll == PortonkentiMennyisegPie.UDP)
71              {
72                  this.abra = new PieChart(adatok, this.magyarazoSzoveg,
"Ă?tvitt adat portonkénti megoszlása (UDP protokoll).");
73              }
74              else
75              {
76                  this.abra = new PieChart(adatok, this.magyarazoSzoveg,
"Ă?tvitt adat portonkénti megoszlása.");
77              }
78              eredmeny.close();
79              //abra.saveChart(null);
80          }
81          catch(SQLException e)
82          {
83              logger.error("Hiba a lekérdezés közben");
84          }
85      }
86
87      @Override
88      public BufferedImage getKep() {
89          return this.abra.getChart();
90      }
91
92      public static final int TCP = 1;
93      public static final int UDP = 2;
94      public static final int MIND = 3;
95  }
001 /*
002  * To change this template, choose Tools | Templates
003  * and open the template in the editor.
004  */
005
006 package csomagelemzo.util.kivetel;
007 import org.apache.log4j.Logger;
008 /**
009  *
010  * @author fricci
011  */
012 public class SajatKivetel extends Throwable {
013      private final Throwable kivetel; //A befoglalt kivétel
014      private final String hibaUzenet; //A String formátumĂş hibaüzenet
015      private final int hibaKod ; //A hibaüzenet int kódja
016      private static final Logger logger = Logger.getLogger("csomagelemzo");;
017      /**
018       * Lekérdezi a hibakódot
019       * @return A hibakód int alakban
```

```
020      */
021     public int getHibaKod()
022     {
023         return hibaKod;
024     }
025 
026     /**
027      * Lekérdezni a hibaüzenetet.
028      * @return A hibaüzenet String alakban
029      */
030     public String getHibaUzenet()
031     {
032         return hibaUzenet;
033     }
034 
035     /**
036      * Lekérdezni a befoglalt kivétel osztályt.
037      * @return a befoglalt osztály.
038      */
039     public Throwable getKivetel()
040     {
041         return kivetel;
042     }
043 
044     /**
045      * Saját kivétel létrehozására szolgáló konstruktor.
046      * @param kivetel Egy kivétel osztály melyet befoglalunk.
047      * @param hibaUzenet Hibaüzenet String formában.
048      * @param hibaKod Hibaüzenet int kódja
049      */
050     public SajatKivetel(Throwable kivetel, String hibaUzenet, int hibaKod)
051     {
052 
053         this.kivetel = kivetel;
054         this.hibaUzenet = hibaUzenet;
055         this.hibaKod = hibaKod;
056         logger.error("***Hiba történt! A saját kivétel adatai:");
057         logger.error("Kivétel: "+this.kivetel.toString());
058         logger.error("Hiba üzenet: "+this.hibaUzenet);
059         logger.error("Hiba kód: "+this.hibaKod+"***");
060     }
061 
062     /**
063      * Saját kivétel létrehozására szolgáló konstruktor. Nincs befoglalt
kivétel!
064      * @param hibaUzenet Hibaüzenet String formában.
065      * @param hibaKod Hibaüzenet int kódja
066      */
067     public SajatKivetel(String hibaUzenet, int hibaKod)
068     {
069         this.kivetel = new Exception();
070         this.hibaUzenet = hibaUzenet;
071         this.hibaKod = hibaKod;
072         logger.error("*** Hiba történt! A saját kivétel adatai:");
073         logger.error("Kivétel: "+this.kivetel.toString());
074         logger.error("Hiba üzenet: "+this.hibaUzenet);
075         logger.error("Hiba kód: "+this.hibaKod+" ***");
076     }
077 
078     /*
079      * 1001 ismeretlen hibák, általános hibák
080      * 1003 az adatbázissal kapcsolatos hibák
081      * 1002 java osztályokkal kapcsolatos hibák
082      * 1004 program és adatbázis közötti hiba
083      */
084     /**
085      * Adatbázis leállítási hiba esetén
086      */
```

```
087     public static final int ADATBAZIS_LEALLITASI_HIBA = 1003001;
088     /**
089      * Valamilyen hiba történt az sql lekérdezés során.
090      */
091     public static final int LEKERDEZESI_HIBA = 1003002;
092     /**
093      * Nem sikerült betölteni egy fontos osztályt.
094      */
095     public static final int CLASS_NOT_FOUND = 1002001;
096     /**
097      * Ismeretlen, közelebbről nem meghatározható hiba történt.
098      */
099     public static final int ISMERETLEN_HIBA = 1001001;
100     /**
101      * Valamilyen hiba történt egy sor beszĂşrása közben.
102      */
103     public static final int SOR_BESZURASI_HIBA = 1003003;
104     /**
105      * Valamilyen hiba történt egy sor módosítása közben.
106      */
107     public static final int SOR_MODOSITASI_HIBA = 1003004;
108     /**
109      * Hiba az adatok adatbázisba történő felvitele közben
110      */
111     public static final int ADAT_FELVITELI_HIBA = 1004001;
112 }
```