Forgalomemuláló keretrendszer tervezése és fejlesztése

Megyesi Péter
Távközlési és Médiainformatikai Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem
Budapest, Magyarország
megyesi@tmit.bme.hu

Dr. Molnár Sándor
Távközlési és Médiainformatikai Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem
Budapest, Magyarország
molnar@tmit.bme.hu

Absztrakt— **Az Internet egyre nagyobb térhódításával egyre gyorsabb és gyorsabb hálózati eszközöket kell tervezni. Az ilyen nagysebességű eszközök – mint routerek, tűzfalak, mobil átjárók, stb. - tesztelése nehéz feladat. Az internet szolgáltatók ritkán vállalják fel a valós idejű tesztek kockázatát és a rögzített forgalmi adatok terjesztése - elsősorban a felhasználói anonimitás miatt - szintén korlátolt. Ennek a problémának a megoldására egy kutatási és fejlesztési program keretében egy keretrendszert készítettünk az Ericsson Hungary Kft. és a BME TMIT Nagysebességű Hálózatok Laboratóriuma támogatásával. A rendszer képes valós forgalmi méréseket kiértékelni és az eredmények alapján tipikus felhasználói viselkedéseket definiálni. A definiált viselkedéseket a keretrendszer emulálni tudja személyi számítógépek távoli vezérlésével. Ezen emulációk során rögzített hálózati forgalom minták segítségével képesek vagyunk valósághű, nagysebességű forgalmat előállítani.**

Kulcsszavak: Internet, hálózati forgalom, felhasználó emulálás

I. BEVEZETÉS

Az Internet mára az életünk részévé vált. Emberek milliói használják minden pillanatban, miközben az internet szolgáltatók folyamatosan a hálózataik fejlesztésén dolgoznak. Ahogy a forgalom exponenciálisan nő, a hálózati eszközöknek egyre bonyolultabbnak kell lenniük. Az ilyen nagysebességű eszközök tesztelése – például routerek, tűzfalak vagy média átjárók – megoldatlan probléma. Több ezer felhasználó együttes Internet forgalmának struktúrája igen komplex, pedig egy szabványos teszt során a felhasznált adatsornak valósághűnek kell lennie. A valós idejű tesztelés megoldaná ezt a problémát, de a hálózatok üzemeltetői a legritkább esetekben engedélyeznek ilyen méréseket az esetleges meghibásodások kockázatai miatt.

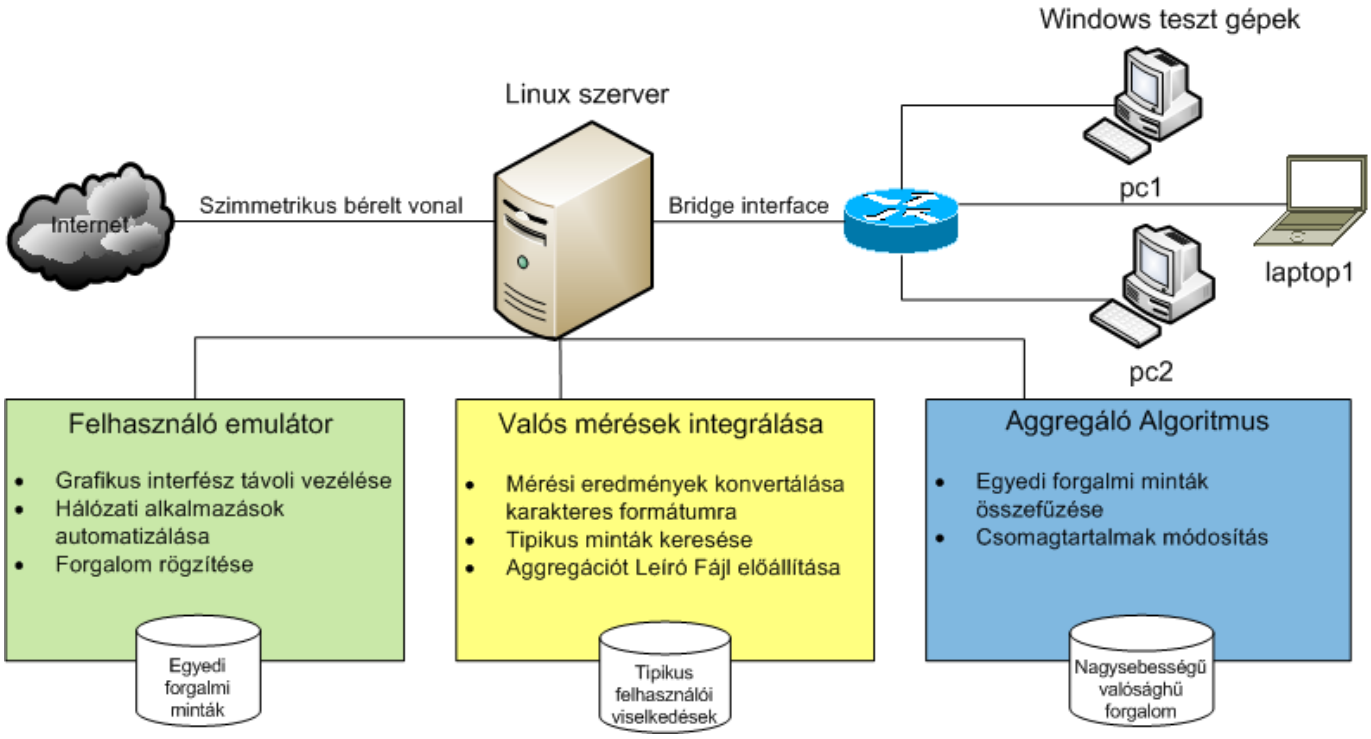
A leggyakrabban használt módszer nagysebességű hálózati eszközök tesztelése során, hogy rögzítik a felhasználók aggregált forgalmát, majd a tesztek során ezeket az adatokat játsszák vissza. Azonban ennek a módszernek az a hátránya, hogy a felhasználói jogvédelem miatt az adatoknak egy hosszadalmas anonimitási eljáráson kell végigmenniük, és a mért adatsor még ezek után sem adható ki harmadik félnek.

A mesterségesen előállított forgalmi minták használta megoldást jelentene erre a problémakörre. Mindemellett szükségünk van arra, hogy az előállított minta a lehető legjobban hasonlítson a valós mérésekhez. Ez azt jelenti, hogy a mesterségesen generált forgalomnak hasonló statisztikai jellemzőkkel kell rendelkeznie, mint a valós forgalomnak, mind csomag (például a csomagok közötti időkülönbségek eloszlása és korrelációs struktúrája), mind folyam (például protokoll és folyam méret eloszlás) szinten.

A ma létező forgalomgenerátorok nem alkalmasak erre a feladatra. A legáltalánosabb ilyen alkalmazások általában szerverek, tűzfalak vagy routerek stressz tesztelésére [6], [7] illetve sávszélesség mérésre [8] alkalmasak. Annak ellenére, hogy ezek az eszközök a maximális linksebességen képesek forgalomgenerálásra, a csomagok ugyanolyan időközönként követik egymást, és nem tartalmaznak valódi hasznos adatot. A kifinomultabb forgalomgenerátorok esetén lehetőségünk van beállítani néhány egyszerűbb valószínűségi eloszlást a csomagkövetési időkre illetve a csomagméretre [9], [10], de nem vagyunk képesek beállítani olyan komplex statisztikai értékeket, amik a valós forgalomra jellemzőek.

A másik súlyos hiányossága ezeknek az alkalmazásoknak a generált csomagok hasznos adatmezőjében jelentkezik. Bár néhány szoftver képes egyes protokollok valósághű szimulálására, a keletkezett csomagok hasznos tartalma nem felel meg a valóságnak. A valós csomagtartalom elengedhetetlen feltétele azon forgalomazonosító alkalmazások teszteléséhez, melyek nem csak a csomagok fejlécét, hanem a hasznos tartalmukat is megvizsgálják. Az ilyen eszközöket összefoglaló néven DPI (Deep Packet Inspection) alkalmazásoknak nevezzük. Az Internet Szolgáltatók egyre szélesebb körben alkalmaznak ilyen DPI eszközöket különböző számlázási irányelvek érvényesítésekor, forgalom szabályozáskor valamint, hogy különböző szolgáltatási minőséget biztosítsanak egyes forgalom típusoknak [5]. Így egy alkalmas forgalomemulátornak ki kell elégíteni ezeket a feltételeket is, hogy felhasználható legyen DPI eszközök tesztelésére.

Egy jobb megoldás érdekébe egy új keretrendszert készítettünk egy kutatási és fejlesztési program keretében az Ericsson Hungary Kft. és a BME TMIT Nagysebességű Hálózatok Laboratóriuma támogatásával. A megvalósított eszköz – melyet a továbbiakban a TET (Traffic Emulation Tool) betűszóval jelölünk – képes emulálni tipikus felhasználói viselkedéseket és rögzíteni az általuk generált hálózati forgalmat. Ezen rögzített minták segítségével a keretrendszer képes összeállítani egy nagysebességű hálózati folyamot, mely



1. ábra. A forgalomemuláló keretrendszer architektúrája.

egyszerre akár több ezer felhasználó forgalmát mimikálja. Úgy terveztük meg az emulátort, hogy képes legyen valós méréseket is feldolgozni, és ennek eredményeit integrálni a keretrendszerbe.

A következő szakaszban bemutatjuk a forgalomemuláló keretrendszer általános felépítését. Ezek után részletesebben ismertetjük az emulátor egyes részegységeinek működését. Végül az 3. fejezetben összefoglaljuk a munkánkat, ahol a jövőbeli kutatási irányokat is bemutatjuk.

II. A FORGLAOMEMULÁLÓ KERETRENDSZER ARCHITEKTURÁJA

Az 1. ábrán látható az általunk megépített forgalomemulátor felépítése. A keretrendszer egy központi Linux szerverből és hozzá csatlakozó Windows alapú kliensgépekből áll. A kliensgépek a szerveren keresztül érik el az Internetet, így az általuk generált hálózati forgalom áthalad a központi gépen. A Windows alapú kliensgépeken népszerű Internetes alkalmazásokat telepítettünk, melyeket előre megírt szkriptekkel automatikusan tudunk vezérelni.

Ahogy az 1. ábra is mutatja, a központi Linux szervernek három fő szolgáltatása van, melyek az alábbiak: a Felhasználó Emulátor, a valós méréseket feldolgozó egység, illetve az Aggregáló Algoritmus. A továbbiakban bemutatjuk az egyes részegységek különböző tulajdonságait.

A. *A Felhasználó Emulátor*

A keretrendszer tervezése során a Felhasználó Emulátor megépítése volt az első lépés. Szükségünk volt egy olyan eszközre, mely képes alkalmazásokat automatikus vezérelni, hogy az általuk generált forgalmi mintákat fölhasználjuk DPI eszközök tesztelésére.

A Felhasználó Emulátor segítségével képesek vagyunk különböző felhasználói viselkedések definiálni egy webes felületen keresztül. Egy felhasználói viselkedés alatt azt értjük, hogy egy eseménysorozatot adunk meg, mely leírja, hogy az adott felhasználó milyen típusú alkalmazásokat futtat, milyen sorrendben illetve mennyi ideig. TET képes ezeket a tipikus viselkedéseket lejátszani egy Windowsos kliensgép grafikus interfészét vezérelve. Ennek a folyamatnak a során központi szerver *TELNET* kapcsolat segítségével bejelentkezik a kliensgépre és megfelelő, előre megírt szkripteket futtatja le a megadott sorrendben. Ezt az eljárást egy Perl nyelven megírt program vezérli, mely a Perl Expect modulját használja a TELNET kapcsolat vezérlésére [11]. Az Expect egy elterjedt eszköz parancssori, interaktív alkalmazások (például *telnet*, *ftp*, *passwd*) automatikus vezérlésére [12].

A Windows alapú grafikus alkalmazások vezérléséért felelős szkriptek AutoIt programozási nyelven lettek megírva és futtatható állománnyá lettek fordítva. Az AutoIt programnyelv segítségével könnyen tudjuk vezérelni a Windows grafikus interfészét, így népszerű Internetes alkalmazásokat tudunk vele futtatatni és szimulálni előre definiált egér és billentyű eseményeket [3]. A futtatható állományokat a vezérlő program a PsExec alkalmazáson keresztül hívja meg, mely azért felelős, hogy az elindított Windows alapú alkalmazások grafikus interfészhez legyenek kötve [13]. E nélkül a grafikus alkalmazások hibásan futnának, a PsExec használatával azonban a kliensgép monitorján (vagy akár egy Távoli Asztali Kapcsolaton keresztül is) végigkövethető az emuláció folyamata.

```
1223378304 ABZABZABZABZABZABZABZABZABZ
1223378304 HIZHIKZHIZHIZHIZHIZHIZHIZHIZ
1223378804 FZFZFZFZFZFZFZFZFZFZFZFZ
```

2. ábra. Forgalmi mérések konvertált formátuma.

```
1223378304.598435|6.pcap|10.1.1.2|10.2.1.9|
1223378305.139341|9.pcap|10.1.1.1|10.2.6.8|
1223378308.882470|3.pcap|10.1.1.2|10.3.3.2|
```

3. ábra. Az Aggregációt Leíró Fájl formátuma.

A felhasználói viselkedések lejátszása során generálódott hálózati forgalmat rögzítjük, és a szerveren letároljuk *Libpcap* formátumban [4]. A központi szerveren lehetőségünk van a *bridge interfész* sávszélességének korlátozására is, így egy felhasználói viselkedést több féle hozzáférési sebesség mellett is tudunk emulálni. Később az aggregáció során ezekből az egyedileg rögzített mintákból építi fel a keretrendszer a kimeneti nagysebességű folyamot. A távoli vezérlésről és a rögzítés folyamatáról bővebben az [1] alatt található leírás.

Jelenleg a keretrendszer az alábbi 11 féle forgalomtípus emulálására képes, melyek mindegyikéhez egy-egy karaktert rendeltünk:

- Fájl megosztás (A)
- Média lejátszás (B)
- Távoli elérés (C)
- Szoftverfrissítés (D)
- Voice over IP (E)
- Játék (F)
- Azonnali üzenetküldés (G)
- Közösségi oldalak (H)
- Web böngészés (I)
- Fájl letöltés (J)
- E-mail (K)

B. Valós mérések integrálása

Az egyes típusokhoz rendelt karakterek a valós mérések integrálása során kerülnek előtérbe, mely a keretrendszer második részegysége. Miután a Felhasználó Emulátor elkészült, szükségünk volt egy olyan algoritmusra, mely a felhasználói viselkedéseket automatikusan generálja valós forgalmi mérési adatok alapján.

Ennek a folyamatnak a során az emulátor képes különböző forgalomazonosító eszközök által generált riportok átalakítására egy speciális formátummá, melyet a 2. ábrán adtunk meg. Ennek az átmeneti fájlnak minden sorában a bemeneti mérésben megtalálható egyedi felhasználó forgalma van leképezve a következő módon. Minden sor egy időbélyeggel kezdődik, mely megadja, hogy az adott felhasználó mikor kezdett el forgalmazni. Az ez után következő karakterlánc ad információt arról, hogy a felhasználó milyen típusú alkalmazásokat futatott. Ennek során percenkénti időfelbontást alkalmazunk, úgy, hogy az egymás utáni perceket a „*Z”* karakter választja el egymástól.

A 2. ábrán megadott példában így az első sorban leírt felhasználó folyamatosan egy fájlmegosztó klienst futtatott, miközben valamilyen médiatartalmat játszott le az Interneten keresztül; a második felhasználó egyszerre böngészett és közösségi oldalakat nézegetett; valamint a harmadik egy Internetes játékot használt.

A keretrendszer ezek után képes kiszűrni a bemenet alapján tipikus felhasználói viselkedéseket, és ezeket integrálni a Felhasználó Emulátorba. Megvizsgáltuk a lehetőségét, hogy már létező mintakereső algoritmusok segítségével oldjuk meg a gyakori előfordulások keresését, de arra a megállapításra jutottunk, hogy ezek a megoldások nem kezelik kellőképpen rugalmasan az egyes forgalomtípusok közötti esetleges hasonlóságokat vagy éppen nagy eltéréseket. Ezért a keretrendszer egy általunk implementált, speciális pontozó algoritmust használ, melynek teljes leírása megtalálható [1] alatt.

A tipikus minták keresése során a rendszer időben fix hosszúságú mintákat keres (a minta időbeli hosszúsága megállapítható a benne szereplő „Z” karakterek számából). Gyakorlati méréseink során azt tapasztaltuk, hogy négy percnél rövidebb mintákat nem érdemes tipikusnak tekinteni, mert ennél rövidebb idő alatt a forgalom bizonyos statisztikái nem állnak be (például torrent fájl letöltése során a kapcsolódó kliensek száma). Ezen kívül a minták maximális hosszúságára tíz percet határoztunk meg, mert méréseinkből arra következtettünk, hogy az ennél hosszabb minták már nagyon kevésszer fordulnak elő a bemenetben.

Ezen felül a 2. ábrán látható bemeneti mérési eredmények alapján a keretrendszer előállítja az úgy nevezett Aggregációt Leíró Fájlt, mely az Aggregáló Algoritmus bemenete. Az Aggregációt Leíró Fájl formátuma a 3. ábrán látható. Ebben a fájlban minden sornak a következő négy adatot kell tartalmaznia. Az első paraméter egy időbélyeg mikro-szekundum pontossággal, míg a második maga a *Libpcap* formátumú fájl, melynek a tartalmát az algoritmus az aggregációba fűzi. A sorokat úgy kell megadni, hogy az időbélyegek szigorúan monoton növekedjenek. A harmadik paraméterként a Windowsos kliens gép IP címét kell megadni, melyen az adott hálózati forgalmat rögzítettük. A negyedik paraméter az adott felhasználóhoz rendelt egyedi IP cím, mely alapján a megkülönböztethetőek lesznek az aggregátumban.

Az Aggregációt Leíró Fájl előállítása során a keretrendszer megpróbálja a bemeneti mérésben szereplő felhasználók hosszú idejű forgalmát helyettesíteni a rögzített, rövidebb idejű, tipikus viselkedések egymásutánjával. Ehhez az algoritmus az alábbi két lépést végzi: először tipikus viselkedések teljes egyezését keresi, és a lefedhető periódusokat a hozzá rögzített forgalmi mintával helyettesíti. Majd a nem lefedhető tartományokhoz megkeresi a legjobban hasonlító tipikus viselkedést, és a hozzá rögzített forgalmat használja föl. Utolsó lépésként az algoritmus időrendbe helyezi a sorokat, hogy a keletkezett fájl megfelelő formátumba kerüljön.

C. *Aggregáló Algoritmus*

Mint már említettük, az Aggregáló Algoritmus a 3. ábrán található fájl alapján végzi el az aggregációt. Az Aggregációt Leíró Fájlt elő tudjuk állítani mesterségesen is, ekkor ugyan a kimenet nem fog a valósághoz közelíteni, de előfordulhat, hogy egy tesztelés során éppen olyan egyedi mintára van szükség, mely a valóságban nem fordul elő.

Az algoritmus alapvetően kétféle üzemmódban tud működni. Az első, úgynevezett online mód során a csomagokat rögtön a megadott hálózati interfészen továbbítja a program, míg a második, offline mód alatt az algoritmus létrehoz egy aggregált *Libpcap* fájlt a háttértáron, és abba ment a csomagokat.

A program a bementi fájl alapján kétféle módosítást végez egy adott *Libpcap* fájl minden csomagján. Első lépésként minden csomag időbélyegét eltolja azzal az időintervallummal, amivel a *Libpcap* fájlban található első csomag időbélyege az Aggregációt Leíró Fájl első paraméterében megadott időértékhez kerül. Ezen művelet során az egy felhasználóhoz tartozó csomagok között eltelt időintervallumok értékei nem sérülnek az eredetihez képest, mely nagyon fontos tulajdonsága a keretrendszernek, tekintve, hogy a forgalomazonosító algoritmusok egy része ilyen statisztikák alapját működik.

Második lépésként az algoritmus kicserélt a csomagok IP fejlécében a forrás vagy a célcímek közül azt, amelyik megegyezik a bemeneti fájl harmadik paraméterével (tehát a mérés során használt Windowsos kliens IP címével) a negyedik paraméterrel (tehát az adott felhasználóhoz rendelt egyedi IP címmel). A program képes még az IP címek cseréjére a hasznos csomagmezőben is, mind bináris, mind szöveg alapon, ez azonban jelentősen lassítja az algoritmus működését, így opcionálisan használható.

Az Aggregációs Algoritmus teljes specifikációja, valamint teljesítmény tesztelése valós környezetben megtalálható a [2] alatt.

III. Összefoglalás

A cikkben bemutatásra került az általunk tervezett forgalomemuláló keretrendszer, ami képes tipikus felhasználói viselkedése definiálására valós mérési adatok alapján. Az emulátor képes lejátszani ezeket a viselkedések, rögzíteni az általuk generált hálózati forgalmat *Libpcap* formátumban, valamint e minták alapján összeállítani egy valósághű, aggregált hálózati folyamot. A keretrendszer teszt oldala elérhető a [14] alatt megadott linken. Mivel a rendszer folyamatos fejlesztés alatt áll, egyes részei megváltozhatnak a folyamat során.

A jövőben a kutatást és fejlesztést több irányban folytatjuk. Elkezdtük a keretrendszert implementálását okostelefon platformra. Ennek során Android felhasználókat szeretnénk emulálni hasonló elven. Ennek megvalósításához egy olyan keretrendszerre van szükségünk, mely képes Android alapú hálózati alkalmazásokat automatikusan vezérelni, és az általuk generált forgalmat rögzíteni. Mivel az okostelefonok manapság kezdenek elterjedni, ilyen típusú forgalomhoz kapcsolódó mérési eredményből kevés fellelhető.

Már folyamatban van a keretrendszer kimenetének statisztikai elemzése, melynek során azt vizsgáljuk, hogy az aggregált folyam mennyire egyezik meg a valóságban mért adatokkal. Előfordulhat, hogy ennek során nem kapjuk vissza a megfelelő eredményeket, így a jövőben célunk, hogy megtaláljuk azt a multiplexálási algoritmust, melynek segítségével egyedi forgalmi mintákat egy nagysebességű és valósághoz közeli statisztikákkal rendelkező aggregátummá lehet összefésülni.

Ezen felül szeretnénk megvalósítani az Internet forgalomelemzésének egy új megközelítését, melynek során alulról fölfelé építkezünk. Az eddigi forgalmi elemzések általában egy nagysebességű aggregátumból indultak ki, melyre megpróbáltak valamilyen matematikai modellt illeszteni. Nekünk lehetőségünk van a keretrendszer segítségével arra, hogy tetszőleges méretű aggregátumot állítsunk elő, és így megvizsgáljuk a forgalom statisztikáinak skálázódását többféle szinten. Ebből a megközelítésből akár fény derülhet olyan tulajdonságokra is, melyek eddig rejtve maradtak az Internet forgalmi elemzései során.

Köszönetnyílvánítás

A cikk írói ezúton szeretnék köszönetüket kifejezni Dr. Szabó Gézának, az Ericsson Hungary, Traffic Lab munkatársának, aki nagyban hozzájárult, hogy ez a projekt megvalósuljon.

References

[1] Megyesi Péter, "Design and develepment of a traffic emulator, MSc Diplomamunka", Budapesti Műszaki és Gazdaságtudományi Egyetem, Távközlési és Médiainformatikai Tanszék, 2011

[2] Csatári Bálint, "Framework for Comparison of Traffic Classification Algorithms", MSc Diplomamunka, Budapesti Műszaki és Gazdaságtudományi Egyetem, Távközlési és Médiainformatikai Tanszék, 2011

[3] "AutoIt", http://www.autoitscript.com/site/autoit (2012.04.02. állapot)

[4] "TCPDUMP/LIBPCAP public repository", http://www.tcpdump.org/ (2012.04.02. állapot)

[5] G. Szabó, Z. Turányi, L. Toka, S. Molnár, A. Santos, "Automatic Protocol Signature Generation Framework for Deep Packet Inspection", VALUETOOLS 2011, ENS, Cachan, France, 2011

[6] "Seagull Traffic Generator", http://gull.sourceforge.net/ (2012.04.02. állapot)

[7] "Traffic Emulator, Nsauditor", http://www.nsauditor.com/docs/html/tools/Traffic%20Emulator.htm (2012.04.02. állapot)

[8] "Iperf", http://iperf.sourceforge.net (2012.04.02. állapot)

[9] "Traffic Generator tool, University of Southern California", http://www.postel.org/tg/ (2012.04.02. állapot)

[10] A. Botta, A. Dainotti, A. Pescape, "Multi-protocol and multi-platform traffic generation and measurement", INFOCOM 2007 DEMO Session, Anchorage (Alaska, USA), 2007

[11] ”Perl Expect modul” , http://search.cpan.org/~rgiersig/Expect-1.15/Expect.pod (2012.05.01. állapot)

[12] ”Expect programozási nyelv”, http://expect.sourceforge.net/ (2012.05.01. állapot)

[13] ”PsExec”, http://technet.microsoft.com/en-us/sysinternals/bb897553 (2012.05.01. állapot)

[14] ”Forgalom emuláló keretrendszer demo oldal”, http://megyesi.tmit.bme.hu/ (2012.05.01. állapot)