# Infrastruktúrához igazodó alkalmazások

Doktori értekezés tézisei

**Bilicki Vilmos**

Témavezető:
Dr. Jelasity Márk

Informatika Doktori Iskola
Informatikai Tanszékcsoport
Szegedi Tudományegyetem

**Szeged 2010**

**Bevezető.** Az IP protokoll és az Internet sikere az egyszerűségükben rejlik. A hálózatban rejlő intelligencia a magasabb rétegekben, a hálózat végein valósul meg. Ezt legegyszerűbben a klasszikus homokóra modellel lehet szemléltetni: a homokóra alsó részében elhelyezkedő sok különböző képességű és szolgáltatású második rétegbeli protokoll (3G, WiFi, WiMax, stb.) az IP protokollon, mint a homokóra szűk részén keresztül szolgálja ki a homokóra felső részét jelképező nagyon széles alkalmazás/szolgáltatás halmazt (P2P, Web, Streaming, stb). E megközelítés szerint az alkalmazásfejlesztőknek nem kell az infrastruktúra részleteivel foglalkozniuk, csak a saját üzleti logikájukra kellene koncentrálniuk, figyelmen kívül hagyva a különböző, alacsonyabb rétegekben fellépő problémákat.

A különböző tézis csoportokban arra szeretnénk rámutatni, hogy célszerű figyelembe venni a hálózat adottságait is. A klasszikus, lassan 30 éves megközelítésmód szerint a hálózat gerincét alkotó eszközök nem foglalkoznak a különböző folyamok állapotaival, csak a gyors csomagtovábbításra koncentrálnak. Ezzel szemben a felhasználókhoz közeli, a hálózat szélein lévő eszközök képesek arra, hogy magasabb szintű szolgáltatásokat nyújtsanak a hálózat üzemeltetőjének. A világ azonban sokat változott mióta ezt a megközelítést használjuk: ma már gyakorlatilag minden adatcserével járó kommunikáció az IP protokoll segítségével zajlik. A virtuális és fedő hálózatok elterjedése és széleskörű használata napjaink egy fontos trendje. Az egyszerű közép és okos szélek paradigma fokozatos változása mögött ezen alkalmazások oldaláról jelentkező heterogén igény halmaz, a hálózat kontextus alapú szolgáltatásainak elterjedése (Aktív Hálózat, Jövő Internete) és a hozzáférési hálózatok igen széles skálájának hatása húzódik meg.

A paradigmaváltás lényege röviden: a hálózat egyre intelligensebbé válik, az általa nyújtott szolgáltatások az alkalmazásokhoz idomulnak (pl.: web szolgáltatás forgalomirányítás). Az alkalmazásoknak viszont gyakran alkalmazkodniuk kell az adott hálózati környezethez. A doktori értekezés célja, hogy áttekintést nyújtson erről a területről. Az értekezés két fő részből áll: az első rész a jelenlegi és a lehetséges jövőbeli intelligens hálózat szolgáltatások skálázhatóságát vizsgálja, míg a második rész négy konkrét alkalmazáson keresztül mutat be példákat, a hálózat képességeinek figyelembevételére. Megállapításainkat az alábbi téziscsopotokra bontottuk:

I/1 Folyamok vs. Állapottartó szolgáltatások

II/1 Rejtőzködő botnet-ek

II/2 Kis fokszámú DHT változékony/fluktuáló hálózatban

II/4 SIP tömörítés

II/4 Skálázható elosztott tároló

**I. rész - Hálózati infrastruktúra.** Az első rész arra mutat rá, hogy a hálózatban megtalálható állapottartó szolgáltatások skálázhatósági problémákat vethetnek fel, amivel a hálózati mérnököknek és az alkalmazásfejlesztőknek foglalkozniuk kell. A probléma bemutatását a hálózat struktúrájának és a különböző strukturális rétegekben megtalálható eszközök architektúrájának ismertetésével kezdjük, majd azon, ma is használt szolgáltatásokat ismertetjük, amelyek állapottartást igényelnek. Ezután bemutatjuk a méréseinket és azok eredményeit.

**I/1 Folyamok vs. Állapottartó szolgáltatások.** Az IP hálózatot használó alkalmazások tárháza folyamatosan változik. Íme néhány fontosabb trend:

- A felhasználók által generált IP forgalom négy - ötévente megduplázódik (14 PBájt most, 42 PBájt várható 2014-ben).
- A 3G, mint hozzáférési hálózat egyre népszerűbb lesz (A mobil felhasználók által generált IP forgalom a 2014-ig tartó időszakban várhatóan minden évben megduplázódik).
- Az IPTV és az igény szerinti videó lejátszás (Video-On-Demand) egyre népszerűbbek lesznek (2014-re előreláthatólag a teljes IP forgalom 60%-a lesz videó/TV nézéshez köthető).

- A P2P paradigma széleskörű használata várható. A P2P ökorendszer forgalmának 7%-át nem a fájlcsere, hanem a P2P média folyamok adják már ma is, ez az arány előreláthatólag tovább növekszik.

- Az IPTV sikere valószínűleg hozzájárul az IPv6 és többesküldés technológiák elterjedéséhez.

A fentiek alapján megállapíthatjuk, hogy az IP hálózat használata és jelentősége a jövőben csak fokozódik. A továbbiakban tárgyalt témák megértéséhez néhány alapvető ismeret szükséges az IP ökorendszerről, melyeket a következő szakaszokban foglalunk össze.

Az IP hálózat nem egy monolitikus egység, hanem sok köztes- és végrendszer együttműködő összessége. Az Internetet a hálózatok hálózatának is nevezik. Több mint tízezer autonóm egységből (Autonomous System – AS) áll, melyek más-más szervezetet reprezentálnak. Ezen autonóm egységek a legtöbb esetben publikus vagy magán kapcsolódási pontokban kapcsolódnak egymáshoz. A kapcsolódási pontok és autonóm hálózatok alkotják magát az Internetet és annak virtuális gerincét is. Olyan, tehát mint egy egységes, dedikált Internet gerinc nem létezik. A hálózat konkrét fizikai vagy logikai topológiáját senki sem ismeri pontosan. Az, hogy adott időpontban milyen kommunikáció zajlik az Interneten, még kevésbé látható. Az autonóm rendszerek mérete igen széles skálán mozoghat. Vannak világméretű autonóm rendszerek, de gyakran egy-egy épület, egyetemi részegység is saját autonóm rendszerként kapcsolódik az Internethez. Ellentétben az Internettel, mely kevésbé jellemezhető tervezett struktúrával, az autonóm rendszerek felépítése igen gyakran a jól ismert hierarchikus tervezési mintát követik. A hierarchiában meghatározott rétegekhez alkalmazkodnak az aktív eszközök gyártói is, külön eszköz családokat biztosítva az adott rétegekhez, funkciókhoz. A tervezési gyakorlat az alábbi rétegekre osztja a hálózatot:

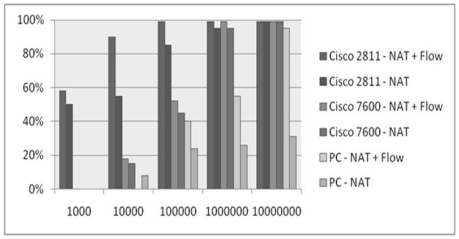
- Hozzáférési réteg: a réteg célja az utolsó mérföldnek nevezett szakasz megvalósítása. Segítségével biztosítják a felhasználók által használt végberendezések hálózati kapcsolatát. A klasszikus tervezési gyakorlat szerint ez a réteg az, ahol az állapottartással összefüggő szolgáltatások megvalósulnak.

- Elosztó réteg: a feladata a földrajzilag lokalizált hozzáférési réteg szigetek összekapcsolása nagyobb régiókba. A klasszikus tervezési gyakorlat szerint itt már ritkábban találhatunk olyan szolgáltatásokat, amelyek állapottartást igényelnek.

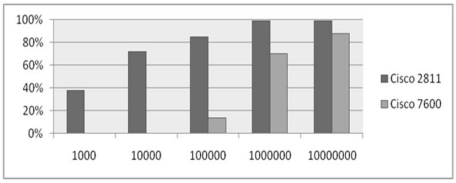
- Gerinc réteg: a réteg célja, hogy a régiókba szerveződő elosztóhálózat-szigeteket egy egységes hálózatba kapcsolja. Ez a réteg biztosítja a kapcsolatokat más autonóm rendszerekkel is. A klasszikus gyakorlat itt a legritkább esetben támogatja az állapottartást igénylő szolgáltatások elhelyezését.

A fentiekben vázolt tervezési minták a klasszikus gyakorlatot testesítik meg. Ez azonban változóban van. Egy trend, amely a változást serkenti a már említett ú.n. Jövőbeli Internet koncepció, amelynek alapötlete a hálózat és a rajta kommunikáló alkalmazások szorosabb együttműködésének igénye. A hálózatnak meg kell értenie az alkalmazásokat és az alkalmazások számára kontextus függő szolgáltatásokat kell nyújtania. (ilyen például a web szolgáltatások számára nyújtott útvonalválasztás). Ebbe az irányba mutat az Aktív Hálózat elképzelés is, mely olyan nyílt, aktív eszközöket vizionál, melyek képesek tetszőleges, harmadik fél által biztosított alkalmazást futtatni. Ahhoz azonban, hogy tanulmányozzuk az állapottartó szolgáltatások problémáit, nem kell új, ma még egzotikusnak számító szolgáltatásokat tanulmányozni, elegendő olyan, ma is jól ismert és gyakran alkalmazott megoldásokat megvizsgálni, mint a forgalomirányítás, tűzfal, hálózati címfordítás, folyam alapú hálózat monitorozás és alkalmazás felismerés. Ezen szolgáltatások közös jellemzője, hogy minden egyes kezelt csomag esetén elvégeznek egy vagy több keresést az általuk karbantartott listákban.

A forgalomirányítás esetén ezt IP keresésnek nevezik. A listákkal kapcsolatos műveletekhez komoly feldolgozási kapacitás szükséges. Ennek szemléltetése érdekében vegyünk egy egyszerű példát: egy elosztó rétegben szolgáló eszköznek 48 darab 1 GBit/s-os interfésze van.Vizsgáljunk egy viszonylag kedvezőtlen esetet, amikor a csomagok rövidek (legyen egy csomag 100 bájtos). Ez esetben az eszköznek 15 ns áll rendelkezésre egy-egy csomag kezelésére. Figyelembe véve a ma hozzáférhető memória modulok elérési idejét, ami 23-55ns nagyságrendbe esik, a probléma nyilvánvalóvá válik. A helyzetet tovább rontja, hogy a listakezelési műveletek nem egy, hanem több műveletet igényelnek (a B-Fa esetén ez $O(log(n))$ ahol n a bejegyzések száma).




1. ábra. Csomag vesztés




2. ábra. Netflow mérés adatvesztés

A tudomány/technika mai állása szerint az ú.n. tartalommal címezhető memóriák (Content Addressable Memory – CAM) jelentik a megoldást a több művelettel járó késleltetések kezelésére. A CAM egy művelettel képes az adott tartalom alapján visszaadni azt a cellát, vagy egyéb adatstruktúrát, ami az adott kulcshoz kötődik. Az IP keresésénél nem pontos egyezés alapján történik a keresés, hanem a leghosszabb egyezés alapján. Ezt támogatja a hármas tartalommal címezhető memória (Ternary CAM – TCAM). A világ azonban nem tökéletes, a CAM, TCAM sokat fogyaszt és drága. Még a legnagyobb teljesítményű hálózati eszközök is csak szerény méretű TCAM-ot tartalmaznak (néhány százezer bejegyzés). Az egyszerűbb eszközök nem tartalmaznak TCAM-ot, a problémát CPU és egyszerű memória segítségével próbálják megoldani.

A skálázódási probléma pontosabb megértése céljából hagyományos PC-vel és aktív eszközökkel is végeztünk többesküldés és egyesküldés folyam alapú méréseket. Az 1. ábrán az egyesküldés mérések eredménye látható. A hálózati címfordítás, illetve a folyam alapú monitorozás skálázhatóságát vizsgáltuk hozzáférés rétegbeli (Cisco2811), gerincbeli (Cisco 7600 + NPEG2) aktív eszközökön és egyszerű PC-n is. A 2 ábra a folyam szám hatását mutatja a folyam alapú monitorozás pontosságára. Mindkét ábrán látható, hogy a folyam szám növelésével az adott szolgáltatás egyre kevésbé látja el feladatát: a folyamszám növelésének eredménye az egyre nagyobb csomagvesztés illetve egyre pontatlanabb mérés. Az eszközök a hardver, szoftver képességeiktől függően különböző mértékben tolerálják a folyamszám növelést. Kihangsúlyoznánk azt a tényt, hogy a méréseket nem az eszközök által kezelhető maximális csomagmennyiséggel végeztük, csupán a hátlap sebességük 10-30%-ával megegyező forgalmat használtunk fel. Az egyesküldés esetén 1 és 10 millió folyam közötti tartományban mértünk, míg a többesküldés esetén 10 és 60 ezer közötti folyamszámmal mértünk. Megállapíthatjuk tehát, hogy az állapottartó szolgáltatások esetén súlyos skálázhatósági problémával szembesülünk.

**1. Tézis.** *A különböző rétegekben elhelyezkedő aktív eszközökön megvalósított állapottartó szolgáltatások teljesítménye szorosan összefügg az általuk kezelt egyesküldés vagy csoportküldés folyamok számával.*

Jól ismert tény, hogy az Interneten megjelenő adatforgalom jelentős része P2P alkalmazásokból származik. E forgalom folyam szintű összetételének tanulmányozása azonban kevésbé vizsgált terület. A különböző P2P protokollok más-más módon oldják meg a társak, erőforrások kezelését, ezért nehézkes az egész P2P ökorendszer folyamlenyomatát egységesen vizsgálni. A szakirodalomban ismert, hogy a Bittorrent alkalmazás folyamlenyomata jelentősen különbözik a web forgalom folyam alapú lenyomatától. Talán nem meglepő, hogy a generált folyamok számában egy nagyságrendi eltérés van. Míg a web célú kommunikáció néhány tíz kapcsolatot tart fenn, addig a Bittorrent sok esetben több százat (az esetek több mint 25%-ában több mint száz kapcsolata volt egy-egy kliensnek). Megállapítható tehát, hogy egyes, igen népszerű, a P2P forgalom jelentős részét adó protokollok igen sok folyamot generálnak. Az ISP-barát P2P kutató közösség azonban ma még csak az átvitt forgalom mennyiségére koncentrál, nem veszi figyelembe e forgalom összetételét. Az 1. tézisben láthattuk, hogy a hálózati mérnökök és alkalmazás fejlesztők számára fontos lehet a folyamok számának figyelembevétele, ennek hangsúlyosan így kellene lennie a szolgáltató-barát P2P megoldások esetén is.

**2. Tézis.** *A szolgáltató-barát P2P alkalmazások esetén fontos szempontnak kellene lennie a generált folyamok számának is.*

Néhány érdekes/fontos kérdés, melyekre választ keresünk a második részben:

1. Mint azt az előzőekben láttuk, az Internet és a rajta zajló kommunikáció nem figyelhető meg egy pontból. Érdekes kérdés lehet, hogy amennyiben egy megfigyelési pontunk van,tudunk-e detektálni egy, az egész Internetre kiterjedő P2P botnet-et az egyébként igen biztató eredményeket produkáló szomszédosságokon alapuló módszerekkel?

2. Az előzőekben kifejtettük, hogy fontos szempontnak kell lennie a generált folyamok számának a szolgáltatóbarát P2P protokollok tervezésénél. Vajon lehet-e kevés folyamot használó (kis fokszámú),de mégis robosztus, elosztott kivonattáblát megvalósító algoritmust alkotni?

3. Szintén az előzőekben láttuk, hogy a 3G, mint hozzáférési hálózat egyre fontosabb lesz. A 3G IP gerinccel rendelkezik, így sok helyen vezetékes közegre tervezett protokollokat alkalmaz. A protokollok adaptálása leginkább a szűkös sávszélesség miatt szükséges. Ennek egy módszere a tömörítés használata. Felvetődik a kérdés, hogy használhatóak-e változtatás nélkül a klasszikus tömörítés algoritmusok?

4. Érdekes kérdés, hogy helyi hálózatok, számítógépes kabinetek viszonylatában hogy lehet, egy szigorú teljesítmény kritériumoknak megfelelő kitüntetett pont nélküli elosztott fálj tároló rendszert építeni?

**Saját eredmények.** E téziscsoportban szereplő tézisek a szerző munkájának eredményei. A többesküldéssel kapcsolatos eredményeket a [2] cikkben publikáltam.

**II. Rész – Infrastruktúrához Igazodó Alkalmazások.** Ebben a fejezetben az előzőekben feltett kérdésekre fogunk válaszolni.

**Kérdés.** amennyiben egy megfigyelési pontunk van,tudunk-e detektálni egy, az egész Internetre kiterjedő P2P botnet-et az egyébként igen biztató eredményeket produkáló szomszédosságokon alapuló módszerekkel?

**II/1 Rejtőzködő botnetek.** Amennyiben automatikusan detektálni és szűrni szeretnénk egy P2P botnet forgalmát,három alapvető forgalomtípusra koncentrálhatunk: terjedés, a botnet által végrehajtott támadások és a fedőhálózat karbantartásához szükséges kommunikáció. Csekély volumene ellenére is a karbantartás által indukált forgalom alapján történő detektálás tűnik a legjobb megoldásnak, mivel ez a legütemezetebb és a fedő hálózat működése szempontjából nélkülözhetetlen.

A munkánkat az motiválja, hogy bár megfelelőnek tűnik egy-egy kapcsolat (folyam) belső *struktúrája* (csomagok tartalma, késleltetése, stb.) alapján detektálni a P2P botnet-et ez adott esetben nehéz esetleg lehetetlen. Az ismert, hogy csomagokra vagy egy-egy folyamra koncentráló módszerekkel nehézkes vagy lehetetlen a botnet forgalmát elkülöníteni. A folyam belső struktúrája teljesen független lehet az aktuális botnet valós működésétől: a csomópontok utánozhatnak más protokollokat, véletlenszerűen viselkedhetnek, de a folyam eloszlás gráfnak (Traffic Dispersion Graph – TDG), ami a csomópontok kapcsolatait írja le, valamilyen korrelációt kell mutatnia a kártékony aktivitásukkal.

A TDG egy irányított gráf $G(V, E)$, amit a folyamok $S$ halmazán definiálunk a <srcIP, srcPort, dstIP, dstPort, protocol> formában ábrázolt élek segítéségével. A csomópontok IP címek, míg az élek az IP címek között zajló kommunikációt jellemzik. Mivel azt szeretnénk bemutatni, hogy a lokális információkból származó TDG gráfok segítségével nem lehet detektálni egy megfelelő módszereket alkalmazó botnet-et, ezért kellően optimista feltételezéseket teszünk: a P2P botnet-hez tartozó forgalom elkülöníthető valamilyen módszerrel, azaz csak azokat a folyamokat fogjuk vizsgálni amelyek a botnet-hez tartoznak.Azt persze nem tudjuk, hogy botnet-hez tartozik, csak azt, hogy egy adott alkalmazás rendszerhez tartozik. Megjegyezzük, hogy ez egy nagyon erős feltevés, de bemutatjuk, hogy még ilyen feltevéssel sem tudjuk a megfelelő rejtőzködési megoldásokat alkalmazó botnet-et detektálni.

A következőkben bemutatjuk a TDG alapú detektálási módszert. A P2P botnet logikai struktúráját a Chord topológiájával modellezzük, mely egy gyűrűbe rendezett fedőhálózat és a gyűrűben távoli csomópontokat exponenciális ugrásszámot átívelő közvetlen kapcsolatokkal köti össze (átívelő kapcsolatok). Részletesebben: $N$

csomópont esetén az ID-ket $0, 1, \ldots, N-1$ tartományból választva, az $i$ csomópont a $i-1 \pmod N$ és $i+1 \pmod N$ csomópontokkal kapcsolódva egy gyűrűt alkot. Ezen túl az $i$ csomópont közvetlenül kapcsolódik a $i+2^j \pmod N$ ahol $j = 1, 2, \ldots, (\log_2 N) - 1$ csomópontokhoz az exponenciális ugrásszámot átívelő, ú.n. exponenciális vagy átívelő kapcsolatok segítségével.

Azt azonban fontos megjegyeznünk, hogy a konkrét a hálózatban megjelenő folyamok és a fedőhálózat logikai struktúrája különböző lehet. Két csomópont, $a$ és $b$ akkor vannak összekötve a logikai fedőhálózatban, amennyiben $a$ tud $b$-ről. Ez azonban nem azt jelenti, hogy $a$ valóban kommunikál is $b$-vel. Illusztrálásképpen: $a$ megjegyezheti $b$-t a hibatűrés növelésének érdekében, $a$ akkor is küldhet $b$-nek közvetlen üzenetet, ha nem szomszédok. (a fedőhálózat akkor is működőképes, ha $a$ elfelejti $b$ címét, miután üzenetet küldött neki.) A Kademnia esetén $a$ úgy keresi meg $x$-et, hogy a hozzá vezető úton minden csomópontot megszólít (persze igyekezve a legkevesebb csomóponton keresztül elérni a célt). Ez húzódik meg a Storm botnet által lokálisan generált nagyszámú üzenet mögött.

Röviden, amit láthatunk, azzal szeretnénk modellezni a botnetet. Ezek csak a valós, megfigyelt kommunikációs folyamok, a virtuális struktúráról/fedőhálózatról (ki kit ismer?) nem tudhatunk. A logikai fedőhálózatban szerepelnének a kapcsolatok a 2., 3., stb. gyűrűbeli szomszéddal, valamint a megtanult csomópontokkal is. Ezeket viszont nem tudhatjuk, mivel nem látunk bele a botnetet megvalósító programok belsejébe.

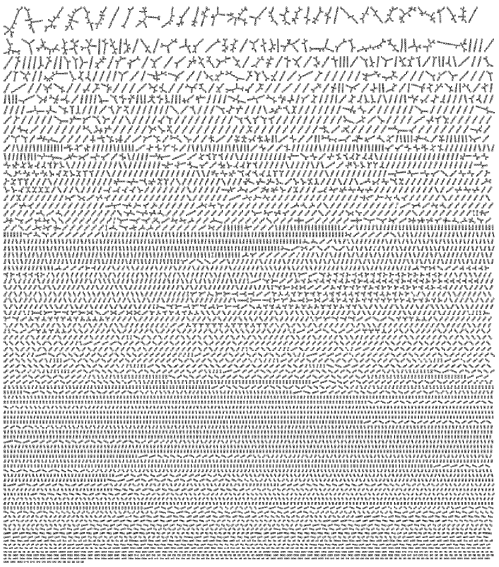
A következőkben két egyszerű módszert ismertetünk, amelyek segítségével a jövőbeni botnet-ek eltüntethetik a forgalmukat. Fontos megjegyezni, hogy a módszerek alkalmazása után a botnet továbbra is működőképes marad, csak a megfigyelési pontokon átmenő folyamainak számát minimalizálja.

- **Átívelő kapcsolatok klaszterekbe rendezése** - A gyűrűben minden csomópont két közvetlen szomszéddal és $\log N$ távoli szomszéddal rendelkezik. Ezeket a kapcsolatokat használja, hogy elérje az ideális ugrásszámot: $O(\log N)$ ahol $N$ a fedőhálózat mérete. A fedőhálózat működéséhez elegendő lenne a közvetlen szomszédok használata is, de ebben az esetben az átlagos ugrásszám $O(N)$-hoz tartana. Van azonban egy köztes megoldás is: lecsökkenthetjük az átívelő távú kapcsolatok számát egy konstans számra úgy, hogy közben az útvonalak hossza sem lesz túl hosszú: $O(\log^2 N)$ vagy akár $O(\log N)$ is lehet. Az ötlet a következő: minden csomópont egyetlen átívelő kapcsolatát használhatja, de ezeket úgy választjuk ki, hogy egy-egy $\log N$ méretű csomópont csoport az egy-egy átívelő kapcsolatával kiadja a teljes hosszú távú kapcsolat teret. Így egymás kapcsolatait használva tudnak kommunikálni. A keresés ugyanúgy megy, mint eddig, csak most, ismerve az adott csoport kapcsolatait, minden csomópont ki tudja választani a számára legmegfelelőbb ugrás kombinációt. Hasonló hatékonyságú útvonalválasztás érhető el akkor is, ha a csoportok nem statikusak, hanem az átívelő kapcsolatok hossza egy adott valószínűség mentén generált. Ez esetben a csomópontnak $\log N$ szomszédját virtuális csoporttársát ismerve kell döntést hoznia. Megjegyezzük, hogy ekkor is csak a dedikált kapcsolatait használja, az ismert, de nem közvetlenül szomszéd csomópontokat a megfelelő szomszédokon keresztül érheti el.

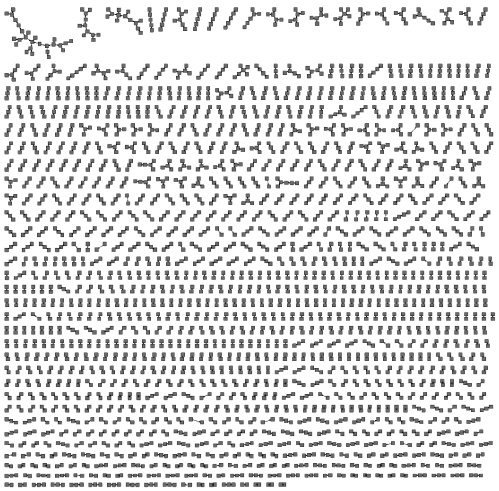
  A folyamok szempontjából tehát minden csomópont két gyűrű (egy kimenő és egy bemenő) illetve egy átívelő kapcsolattal rendelkezik.

- **Lokalizáció** - Úgy is optimalizálhatjuk a gyűrűt, hogy megpróbáljuk minimalizálni a forgalomirányítókon átmenő folyamok számát. Ezt például az azonosítók megfelelő hozzárendelésével érhetjük el. Több ilyen algoritmus is ismert.

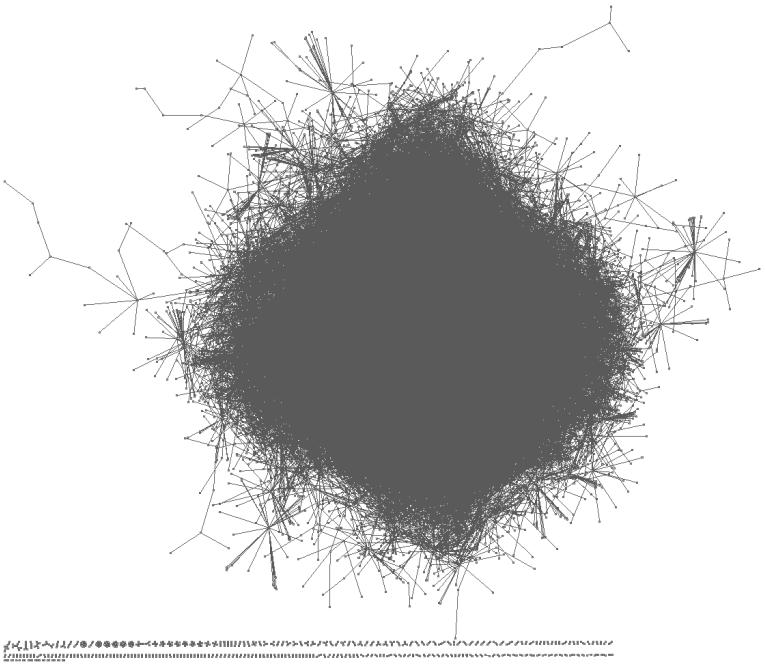
Az Internet egyes pontjaiban lokálisan látható TDG darabok vizsgálatához (i) létrehoztunk egy statikus AS szintű Internet topológia modellt a CAIDA mérési adatok alapján, (ii) a fedőhálózatot (100000 csomópont) ráfeszítettük erre a topológiára, figyelembe véve az adott AS-ek méretét is, (iii) elemeztük a helyi TDG-ket, amelyeket az adott AS-en átmenő folyamok határoztak meg. A legközpontibb elhelyezkedésű AS által látott TDG a 3a ábrán látható. Ebből megállapíthatjuk, hogy az ismertetett két módszer segítségével (kapcsolat klaszter, lokalizáció) véletlenszerű hozzárendelésnél még jól kivehető többszörösen összefüggő komponens (3c ábra) eltűnik (3b ábra). A kevésbé központi elhelyezkedésű AS számára kivehető kapcsolati háló (3b ábra) jóval kisebb, mint a legközpontibb elhelyezkedésű AS-ből látható kapcsolati háló (3c ábra).



(a) AS174 (központiság: 48904554 átmenő kapcsolat), lokalizált, klaszterezett



(b) AS3491 (központiság: 4460142 átmenő kapcsolat), lokalizált, klaszterezett



(c) AS3491 (központiság: 4460142 átmenő kapcsolat), véletlen

3. ábra. Különböző AS-ekből látható TDG-k. Az AS174 a legközpontibb.

Megállapíthatjuk tehát, hogy még a valószínűtlenül optimista feltételezések mellett (az AS-ek képesek minden rajtuk keresztülmenő forgalmat monitorozni és a folyamokat megfelelően elkülöníteni) sem lehet egy-egy AS-ben összegyűjtött adatok alapján elkészített TDG-ből detektálni a botnet struktúráját. A legközpontibb AS-ek ben ugyan a forgalom egy jelentős része megjelenik, de a kicsi fokszám miatt a megfigyelt TDG néhány csomópontokból álló csoportokra esik szét.

**3. Tézis.** *Lehetséges olyan P2P botnet-et építeni, amelyet nem lehet az Internet egy pontjából (egy AS) TDG segítségével detektálni*

**4. Tézis.** *Lokalizáció és kapcsolat klaszterezés segítségével létrehozható olyan P2P botnet, amelyet nem lehet TDG segítségével detektálni az Internet egy pontjából (egy AS)*

A mélyebb megértés érdekében az alábbi tulajdonságokat elemeztük az egy-egy AS által látott gráf esetén: a csomópontok, az élek valamint a komponensek számát, a legnagyobb komponens méretét, az átlagos fokszámot és az InO-nak nevezett metrikát. az InO azon csomópontok aránya, akik rendelkeznek ki és bemenő kapcsolatokkal is. Az elemzésből kiderült, hogy a csomópont klaszterezésnek köszönhető leginkább a TDG szétesése. A lokalizáció ugyan segít a TDG darabolásában, de hatása jóval kisebb, mint a kapcsolat klaszterezése.

**5. Tézis.** *A lokalizáció hatása csekély, míg a kapcsolat klaszterezés jelentős hatással bír a botnetek rejtőködő képességére.*

A fentiek alól egy kivételt tapasztaltunk: a két legközpontibb AS esetén a lokalizáció hatására a legnagyobb komponens két nagyságrenddel kisebb lett.

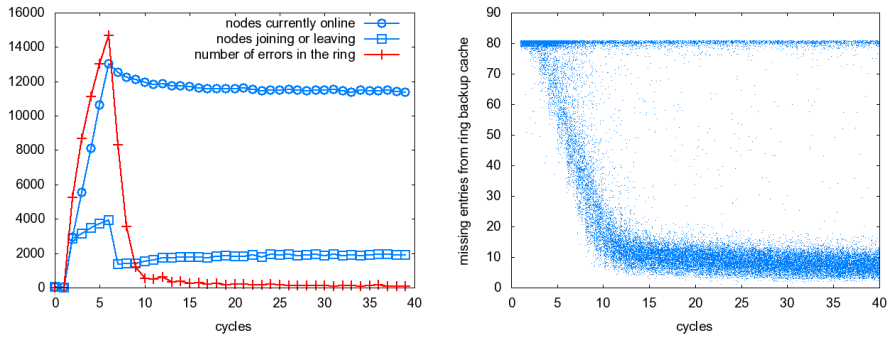
**Összegzés.** A DHT alapú P2P botnet TDG alapú detektálhatóságát elemeztük és bemutattuk, hogy megfelelő eljárások segítségével a botnet el tud rejtőzni. A kapcsolat klaszterezés új, míg a lokalizáció már ismert ötlet. Az is kiderült, hogy a kapcsolat klaszterezés hatása sokkal jelentősebb, mint a lokalizációé.

**Saját eredmények.** A 5-ös tézis a szerző saját eredménye, míg a többi tézis közös munka eredménye. Az itt kifejtett téziseket a [6] konferencia kiadványban publikáltuk.

**Kérdés.** A csomópontok fokszámának jelentőségét már láthattuk a hálózat állapottartó szolgáltatásainál ( 1, 2 tézisek) és a botnetek rejtőzködésénél (3. tézis). Felmerül azonban a kérdés: a hálózatok változékonyságát/fluktuációját hogyan tűri egy ilyen DHT? Lehet-e tehát kis fokszámú, de robosztus és hatékony DHT-t készíteni?

**II/2 Kis fokszámú DHT változékony hálózatban.** A tipikus P2P alkalmazások sok kapcsolatot tartanak fel a robosztus és skálázható működés érdekében. Csak néhány olyan fedő hálózatot ismerünk (pl.: Viceroy, Symphony) amelyeknél a szomszédok száma független a hálózat méretétől, gyakorlatilag konstans. Az továbbra is nyitott kérdés, hogy kis illetve konstans fokszámmal lehet-e hatékony és egyben robosztus fedő hálózatot építeni. A Symphony-val és a Viceroy-al kapcsolatos kutatások a minimális fokszámra korlátozódtak (3, vagy 4 szomszéd). Ez esetben negatív eredmények születtek a skálázhatóság és a robosztusság tekintetében. A következőkben azt szeretnénk bemutatni, hogy lehetséges Symphony-hoz hasonló igen kicsi, konstans fokszámú fedő hálózatot tervezni úgy, hogy közben hatékony és robosztus is legyen. Ezt a célt az alábbi egyszerű megoldások alkalmazásával érhetjük el:

- Előretekintés – A mohó útvonal keresés kiegészíthető előretekintés eljárással, amikor a csomópontok rendelkeznek a szomszédok, illetve adott távolságig azok szomszédjainak ismereteivel vagy annak egy részével. Így jobb lokális döntéseket tudnak hozni, mivel nem csak lokális információkkal rendelkeznek.




4. ábra. A topológia és a tartalék kapcsolatok alakulása $N = 2^{14}$ méretű hálózatra. Az ábrán a jobb oldali pontok az egyes csomópontokhoz tartoznak és véletlenszerűen el vannak tolva a sűrűség ábrázolása érdekében.

- Fokszám korlátozás – A maximális szomszédszám számára felállított szigorú felső korlát betartására a csomópontoknak vissza kell utasítaniuk azokat az átívelő kapcsolatkezdeményezéseket, amelyek a felső korlát felett vannak. Az elutasítás úgy is megoldható, hogy a csomópont a lokális információi alapján javasol egy másik alternatív szomszédot.

- Rétegezés – Mivel az egyes csomópontoknak kevés átívelő kapcsolata van (1-2), a megfelelő választék érdekében célszerű az átívelő kapcsolatokat adott szomszédságban úgy szervezni, hogy minél több különböző hosszúságú kapcsolat legyen. Réteges mintavételezés segítségével az átívelő kapcsolatokat logaritmikus számú intervallumba csoportosítjuk a hosszuk alapján $[e^i/N, e^{i+1}/N]$ $(i = 0, \ldots, [\ln N] - 1)$.

- Rövid kapcsolatok kiszűrése – Érdekes megfigyelés, hogy amennyiben az átlagos útvonal hosszú,érdemes kiszűrni a túl rövid átívelő kapcsolatokat. Bevezettük az $m$ paramétert, mely meghatározza a kihagyandó intervallumokat. Például $m = 2$, akkor az első lehetséges intervallum a $[e^2/N, e^3/N]$ lesz.

Először szimuláció segítségével vizsgáltuk statikus hálózatban az útvonalválasztás hatékonyságát (gyakorlatilag az ugrás számot). Azt találtuk, hogy a fent említett megoldások segítségével a hálózat megfelelő teljesítményt nyújtott. A felsorolt algoritmusok közül a legnagyobb javulást az előretekintés hozta. A fokszám korlátozás szintén jelentős javulást okozott. A rétegezés és rövid kapcsolatszűrés 5-10%-os javulást hozott. Ezen megoldások együttes alkalmazásával egy $N = 2^{20} \approx 1,000,000$ méretű hálózatban 30 ugrásos átlagos úthosszat értünk el úgy, hogy a maximális fokszám 4 volt.

A P2P hálózatok vizsgálatához nem elegendő csak a statikus esettel foglalkozni,nagy gondot kell fordítani a dinamikus eset vizsgálatára is. Analitikus eszközökkel bemutattuk, hogy a Symphony-szerű topológiák skálázhatóvá tehetőek $O(\log N)$ tartalék kapcsolat segítségével (minden meglévő kapcsolathoz), ha a hálózat nem túl változékony,akkor elegendő $O(\log \log N)$ tartalék kapcsolat is.

**6. Tézis.** *A Symphony-alapú topológia skálázható azaz,* $\lim_{h\to\infty} p(h, q) > 0$*, amennyiben (i) minden kapcsolathoz van* $O(\log N)$ *tartalék kapcsolat, vagy amennyiben (ii) minden kapcsolathoz, van* $O(\log \log N)$ *tartalék kapcsolat és* $q \leq e^{-2} \approx 0.135$.

A valós hálózati instabilitás hatásának vizsgálata érdekében kísérleteket végeztünk a PeerSim esemény alapú szimulátorra kifejlesztett protokoll segítségével. A célunk egy olyan konkrét protokoll kifejlesztése volt, amely képes a Symphony-topológiát létrehozni és karbantartani a megfelelő tartalék kapcsolatok segítségével. Ennek érdekében a T-Man pletyka-alapú protokollt kiegészítettük a fent említett módszerekkel. A hálózatban résztvevő csomópontok ébrenléti idejét a szakirodalomban található mérések alapján Weibull eloszlással és $k = 0.5$ alak tényezővel modelleztük a gyakran népszerűbb, de kevésbé reális Exponenciális eloszlású ébrenléti időtartam helyett.

A 4. ábrán látható a gyűrű topológia kialakulásának sebessége. Mint láthatjuk, a résztvevő csomópontok folyamatosan változtak (kb. 10% minden ciklusban). A tartalék kapcsolatok hatása szintén jól látható. Annak

ellenére, hogy a legtöbb csomópont jó minőségű tartalék kapcsolatokkal rendelkezik, egyesek nem rendelkeznek ezekkel. Ez nem azért van, mert a protokoll nem működik megfelelően, hanem ezek azok a csomópontok, amelyek nagyon rövid életűek (gyakran csak egy ciklusig maradnak benn), nem tudnak kommunikálni a többiekkel, mert mire megtudna valamit, addigra kilép az ébrenléti állapotból.

**7. Tézis.** *Lehetséges hatékony és nagyon változékony hálózaton is üzembiztos kis fokszámú DHT alapú P2P algoritmust készíteni.*

**8. Tézis.** *A T-Man alapú pletyka-hálózatot kiegészítve, az négy algoritmussal( Előretekintés, Fokszám biztosítás, Rétegezés, Rövid kapcsolatok elkerülése) képes a hatékony és robosztus DHT szolgáltatást nyújtani, amely igen változékony hálózat (Weibull k=0.5) esetén is megőrzi stabilitását.*

**Összefoglaló.** E téziscsoportban bemutattuk azt, hogy lehetséges kis fokszámú, hatékony P2P elosztott kivonat-táblát megvalósító algoritmust tervezni. Először megvizsgáltuk, hogy az általunk javasolt technikák segítségével mennyire lesz skálázható a statikus hálózat majd, a dinamikus esetet vizsgáltuk analitikusan és szimulátorral is. Az analitikus megközelítés segítségével bemutattuk, hogy $O(\log N)$ tartalék kapcsolat segítségével robosztussá tehető a hálózat. Az előző téziscsoportokban felvetett problémákra tehát igen a válasz: lehet rejtőzködő botnet-et gyártani és lehet olyan hatékony P2P algoritmust is fejleszteni, mely figyelembe veszi az állapottartó szolgáltatások skálázhatósági problémáit.

**Saját eredmények.** A téziscsoporthoz tartozó eredményeket a [5] cikkben publikáltuk. A tézisek közül a 8. a szerző saját eredménye, míg a többi közös munka gyümölcse.

**Kérdés.** Az első részben láttuk, hogy a 3G, mint hozzáférési hálózat, egyre fontosabb lesz. A 3G IP gerinccel rendelkezik, így sok helyen a vezetékes közegre tervezett protokollokat alkalmazza. A protokollok adaptálására leginkább a szűkös sávszélesség miatt van szükség. Ennek egy módszere a tömörítés használata. Felvetődik a kérdés, hogy vajon használhatóak-e a klasszikus tömörítés algoritmusok minden változtatás nélkül?

**II/3 SIP tömörítés.** A mobil rendszerek üzemeltetői komoly befektetéseket eszközöltek az infrastruktúra kialakításába. Ahhoz, hogy ez a befektetett összeg megfelelő jövedelmet termeljen, ki kell elégíteni a felhasználók igényeit, akik az infrastruktúra által nyújtott szolgáltatásokat helyszíntől és a konkrét eszközöktől függetlenül szeretnék használni. Szükség van tehát egy olyan protokollra, amely segít a rendelkezésre álló szolgáltatások és azok paramétereinek egyeztetésében. A 3GPP a Viszony Kezdeményező Protokollt (Session Initiation Protocol – SIP) választotta erre a célra. A SIP lett a 3G gerinc rendszer egyik legfontosabb protokollja.

A SIP azonban egy szöveg alapú, igen redundáns protokoll, amely nem feltétlenül ilyen közegre lett szánva. Problémák merültek fel a viszony felépítéssel kapcsolatban, mivel a teljesen IP (all-IP) alapú 3G hálózatban ez túl sok időt vett igénybe. Esetenként 10 másodperces késleltetések is előfordultak. A probléma kezelésére a tömörítés tűnt a legjobb megoldásnak. Ezzel a témával az IETF ROHC (RObust Header Compression) csoportja kezdett foglalkozni, egy virtuális gépet (Universal Decompressor Virtual Machine – UDVM) javasoltak a kitömörítés kezelésére. Ezen túl definiáltak egy külön réteget a jelzés tömörítés kezelésére (SigComp). A konkrét tömörítési protokollokkal nem foglalkoztak.

Mivel a téma viszonylag új,érthető, hogy nem találtunk a SIP tömörítésével foglalkozó műveket. A feladatunk tehát azzal kezdődött, hogy megvizsgáltuk a SIP tömöríthetőségét és a tömörítés elméletet, hogy megfelelő megoldásokat találjunk. Ezzel párhuzamosan kifejlesztettünk egy teszt keretrendszert, ami két SIP ügynököt összekötve a teljes UDVM környezetet megvalósítva tömöríti/kitömöríti a SIP üzeneteket.

A tömörítési algoritmusok vizsgálatakor igyekeztünk az általánostól kezdve, egyre jobban a SIP specifikus megoldások felé haladva, minél jobb eredményt elérni. Az algoritmusokat nem csak a tömörítés hatékonyságával, hanem a tömörítéshez/kitömörítéshez szükséges idő, illetve az ezekhez szükséges memória/CPU alapján is értékeltük:

- Első megközelítésként megpróbáltuk a klasszikus tömörítő algoritmusokat változtatás nélkül alkalmazni a SIP tömörítésére. Az LZ77-et megvizsgálva kiderült, hogy nagyobb tömörített állományokat gyártott, mint az eredeti állomány volt. E mögött kevés szót és az összes karaktert tartalmazó szótár csatolása volt. A Huffman algoritmust is teszteltük, de ekkor még rosszabb, rövid üzenetek esetén is akár két és félszeres méret növekedés volt az eredmény. Itt karakterek kódolását definiáló fa csatolása volt a felelős a rossz eredményért. Kiderült tehát, hogy nem érdemes a meglévő tömörítő eljárásokat változtatás nélkül alkalmazni.

  **9. Tézis.** *Változtatás nélkül a klasszikus tömörítési algoritmusokat nem érdemes SIP tömörítésre használni.*

- A következő körben megpróbáltunk egy mások által már kialakított SIP szótárat alkalmazni a szótár alapú algoritmusokkal. A meglévő szótár alapú algoritmusokat (LZ77, Aritmetikus, Huffman) módosítva lehetővé tettük a SIP szótár használatát. Ezen túl adaptívvá tettük a szótárat egy egyszerű algoritmus segítségével (Előre Mozgatás, Move to Front). Így a népszerűbb szimbólumok előre kerültek. Ezt a szótárat használtuk a tömörítésnél és a kitömörítésnél egyaránt. Tovább módosítottuk az LZ77-et annak érdekében, hogy a hosszt csak egy és ne két bájton tárolja. A valódi áttörést az a módosításunk okozta, amikor a túl rövid egyezéseket (1,2,3 hosszú) nem kódoltuk, hanem átküldtük kódolatlanul. Egy jel segítségével különböztettük meg a kódolt és a kódolatlan részeket. Ezen módosításokkal már 55% és 75% közötti tömörítési arányt tudtunk elérni. Az ismétlődő részletet tovább lehet tömöríteni (Deflate), így további 10% - 12%-os tömörítés növekedést értünk el. Összefoglalásképpen az alábbi eredmények a legfontosabbak a SIP tömörítés szempontjából:

  **10. Tézis.** *A módosított Deflate algoritmus (futáshossz kódolás + kontextus modellezés) teljesít legjobban amennyiben a SIP kapcsolat-felépítés időtartama a döntő.*

  **11. Tézis.** *A módosított LZ77-es algoritmus a leggyorsabb a kitömörítő oldalon, míg a tömörítésnél a kontextus modellezésen alapuló eljárások a leggyorsabbak (pl.: előtag mentes eljárások).*

- Végezetül megvizsgáltuk az üzenet folyam entrópiáját és kiderült, hogy a várakozásoknak megfelelően sokkal jobb, mint az egyes üzeneteké külön-külön (az egyes üzenetek gyakran tartalmaznak azonos információt pl.: címzett). Jobb tömörítés-arányokat érhetünk tehát el, amennyiben az egész folyamot tömörítjük, nem egyes üzeneteket külön-külön. A ROHC ezt dinamikus tömörítésnek nevezi és egy további fontos irány lehet a tömörítés hatékonyságának növelésénél.

**Összegzés.** A kísérletek alapján megállapíthatjuk, hogy nincs optimális tömörítő algoritmus. A hatékonyan tömörítő eljárás gyakran lassú, a gyorsabban működő pedig gyenge a tömörtítés mértékében. Az eredeti igényeket figyelembe véve a módosított Deflate biztosítja a legkisebb késleltetést (1.62 s), viszont ez a leglassabb a tömörítő oldalán. Megemlítjük, hogy a szótár alapú módszereink abban az esetben optimálisak, ha nincsenek különleges karaktereket tartalmazó üzenetek. Az adaptív megoldásunk képes hatékonyan kezelni ezen üzeneteket is.

Összefoglalva tehát: 50% alatti méretre csökkentettük az üzeneteket, a SIP hívásfelépítés 1.5s lett. (Dinamikus tömörítéssel ez tovább javítható 30% tömörítés, 1s késleltetés). Megoldásainkkal igazoltuk a SigComp réteg létjogosultságát és azt, hogy valóban fontos az, hogy a partnerek választhassanak kontextusuknak leginkább megfelelő algoritmust.

**Saját eredmények.** A téziscsoportban leírt eredmények a [4] cikkben voltak publikálva és az elérésükben szerző munkája volt a meghatározó.

**Kérdés.** Érdekes kérdés, hogy helyi hálózatok, számítógépes kabinetek viszonylatában hogy lehet egy szigorú teljesítmény kritériumoknak megfelelő kitüntetett pont nélküli elosztott fájl tároló rendszert építeni?

**II/4 Skálázható tárhely.** Napjaink számítógépes laboratóriumaiban az 500 megabájtnál nagyobb memória és a több gigaherzes-es CPU már átlagosnak számít. A háttértár mérete sok esetben jelentősen meghaladja a valóban felhasznált területet, ez a nem használt terület nem ritkán több tíz gigabájt. Egy tipikus kis-közép vállalkozás néhány tucat, míg egy nagyobb intézmény több száz PC-vel is rendelkezhet. Ha ezt a számot megszorozzuk az előzőekben említett nem használt tárterülettel, akkor igen komoly veszendőbe menő tárkapacitást kapunk (több TBájt). A fent említett tárterületek céljára terveztünk és valósítottunk meg egy demonstrációs alkalmazást a LanStore-t. A tervezési alapelvek közül a fontosabbak:

- Központi, kitüntetett szerver nélküli elosztott rendszer.
- A bevont eszközök terhelése minimális legyen. (mivel napi használatban lévő munkaállomásokat alkalmazunk)
- Helyi hálózatra optimalizált megoldás legyen. Itt olyan megoldásokat tudunk használni, mint a többesküldés.
- Önszervező, önjavító legyen. Mivel a bevont gépeket senki sem menedzseli, komoly fluktuációval kell számolni a résztvevő gépek között.
- Fájl alapú és ne blokk alapú megosztást biztosítson.
- Kutató labor, egyetemi környezethez alkalmazkodjon, ahol az írás ütközések ritkák.
- Hatékonyan használja a tároló kapacitást, de legyen képes tolerálni a várhatóan nagy fluktuációt is.

Az alap elgondolás az, hogy a bevont gépek között a tárolandó fájlokat egyenletesen elkenjük. Ehhez a fájlokat darabokra osztjuk és egy-egy darabot egy-egy eszközre töltünk fel. Mint már említettük, a nagy fluktuációt is tudni kell kezelni. De mekkora ez a fluktuáció? Mivel nem igazán találtunk az irodalomban erre utaló leírást a számítógépes laboratóriumokban mért fluktuációról, megmértük ezt. A várt eredményt kaptuk: a gépek több mint 10%-a cserélődik ki percről percre, de a gépek 50%-a folyamatosan elérhető. Ennek a fluktuációnak a kezelésére két megoldást alkalmaztunk. Az adatok védelmét egy ismert hibajavító kódolás(Reed-Solomon) segítségével értük el, míg a rendszer konzisztenciáját egy ismert, szavazás alapú algoritmus (Paxos) módosításával valósítottuk meg. A következőkben ezt a két feladatot megvalósító modult fejtjük ki:

**Adat redundancia modul.** A szakirodalomban számos megoldás található a megfelelő adat redundancia elérésére. A legegyszerűbbek a tükrözésen alapulóak, ezek azonban nem tudják a tárterületet kellő hatékonysággal használni. A paritás alapú megoldások ugyan hatékonyabban tudják a tárat használni, de csak egy vagy két hibát tudnak javítani. Esetünkben az ú.n. hibákat előre javító (Forward Error Correction - FEC) kódok egy speciális osztálya, a törléses hibákat javító algoritmusok a legmegfelelőbbek. Mivel csak adatvesztés van (egy gép nem válaszol),ezt a hibát (törlést) kell csak javítani. Ezek közül is két osztály van: az egyik ugyan kisebb számítási kapacitást igényel, de a hibát nem tudja garantáltan javítani, míg a másik garantáltan javítja a hibát. Mi az utóbbit választottuk. A Bose-Chaudhuri kódok egy speciális típusát, a Reed-Solomon kódolást. Ez az alábbiakat biztosítja: n darab adathoz gyártunk m darab javító adatot, ezután az m+n darab adatból bármely m db elvesztését tolerálja a rendszer. Ezt egy speciális egyenletrendszer megoldásával, a Vanderomnde mártix segítségével oldja meg. A műveleteket a Galois mezőben hajtja végre, így táblázat keresésre cseréli a komplex számítási műveleteket. Mi a Reed-Solomon kódok Lugu Rizzo féle implementációját használtuk. A fájlokat 64KBájtos darabokra osztjuk és ezekre a szeletekre számítjuk a szükséges redundanciát.

**Paxos implementációnk.** A szavazás alapú konzisztenciát biztosító algoritmusnak a klasszikus Paxos algoritmust választottuk. A Paxos három entitásra épül: Vezető, Konszenzus algoritmus és a Tanuló. Röviden, a Paxos az alábbiak szerint működik: A vezető egy háromfázisú tranzakciót hajt végre a résztvevő csomópontokon és az eredményt elküldi a tranzakció elindítójának. Az alábbi optimalizációkat valósítottuk meg: ahhoz, hogy a rendszer tovább tudjon lépni egyik állapotból a másikba, a csomópontok többségének jelen kell lennie. Egy

| CPU Frekvencia (GHz) | N | K | Sávszélesség (MBit/s) |
|---|---|---|---|
| 1 | 64 | 32 | 40 |
| 2 | 64 | 32 | 80 |
| 3 | 64 | 32 | 120 |
| 3 | 200 | 100 | 38.4 |

1. táblázat. Reed-Solomon teljesítmény

fluktuáló rendszerben előfordulhat, hogy ugyan jelen van a szükséges számú csomópont, de ez folyamatosan változik (a tranzakció alatt is). Például, A küld egy üzenetet B-nek, eközben A újraindul és elfelejti, hogy üzenetet küldött B-nek, így a választ sem fogadja el. A klasszikus Paxos is így működik. Mi megengedjük a kéretlen üzenetek feldolgozását is, így egyes esetekben le tudtuk csökkenteni a szükséges üzenetek számát hatról kettőre. Az algoritmus robosztussága ezzel nem változott.

**Értékelés.** Szimulációk segítségével vizsgáltuk a mért fluktuáció hatását a Paxos implementációnkra. Az eredmények igazolták a várakozásainkat, nagyobb fluktuációnál lassabb volt a tranzakciók előrehaladása, de a rendszer stabil maradt.

A következő vizsgált terület az volt hogy, a számításigényes algoritmus hírében álló Reed-Solomon képes-e nagy terhelés igény nélkül kiszolgálni a kabinetes körülmények között felmerülő igényeket? Ennek validálására méréseket végeztünk, amelyek összegzése az 1. táblázatban látható. Megállapíthatjuk, hogy a használt PC-k kódolási sávszélessége esetenként nagyobb, mint a rendelkezésre álló hálózati átviteli kapacitás, így a Reed-Solomon használható a kívánt körülmények közötti hibajavításra.

**12. Tézis.** *A napjainkban rendelkezésre álló személyi számítógépek számítási kapacitása alkalmas arra, hogy a Reed-Solomon alapú kódoláson alapuló hibajavító megoldásokat használjunk a laboratóriumi változékonyság okozta hibák elfedésére.*

A fenti mérés csak a nyers kódoló kapacitást mutatta be, a rendszer teljesítménye ettől persze eltérhet. A fájlrendszerek tesztelésére az ú.n. Andrew teszt szolgál, mely az adott művelethez szükséges időt vizsgálja. A legtöbb művelet fájlokkal történik, itt viszont fontos a fájlok mérete. Jól ismert, hogy a Unix-os fájlok mérete Pareto, míg a Windows-os fájlok Lognormal eloszlással modellezhetőek. Mi a Pareto modellt alkalmaztuk. A mérés érdekében legyártottunk egy teszt állományt, ami Pareto eloszlású, hosszúságú és véletlen tartalmú fájlokat tartalmazott egy egyenletes eloszlással legyártott könyvtár struktúrában. Ezt a teszt állományt manipuláltuk 10 gépből álló teszt környezetben és egy referenciaként használt Windows fájl megosztáson:

1. Mértük a könyvárak létrehozásához szükséges időt (a) és ezek törléséhez szükséges időt (b). Ezeket 615 véletlen struktúrába helyezett könyvtárral vizsgáltuk.

2. A fájlok feltöltéséhez (c) és letöltéséhez szükséges időt. 200 fájlt mozgattunk, ahol a méreteket Pareto eloszlással hoztuk létre (a=1.05, k= 3800). Az össz méret 4,08 MBájt volt.

Az eredményekből (2 tábla) megállapítható, hogy az elosztott tár két nagyságrenddel lassabb volt, mint a Windows fájl megosztás. Ez a fájlok kis méretének tudható be. Mivel minden fájl feltöltését külön tranzakcióként kezeltük a hozzá tartozó szavazással, komoly késleltetést okozott. Ezen kötegelt kezeléssel lehet javítani. A valós teljesítmény nagyobb fájlok esetén mutatkozik. Ezért megvizsgáltuk, hogy amennyiben kötegeljük a fájlokat vagy video archívumként használjuk, hogyan alakul a teljesítménye. Ehhez újra legyártottuk a fájlokat, csak most k értékét 3800000-ra állítottuk. A feltöltés és a letöltés sebessége az (e) és (f) oszlopban látható. Ezekből az eredményekből már látható, hogy hasonló teljesítményre képes, mint Windows fájlmegosztás.

**13. Tézis.** *Az általunk kiegészített Paxos alkalmas arra, hogy a számítógépes laboratóriumokban tapasztalt változékonyság esetén biztosítsa a konzisztenciát.*

| | Lanstore | | Windows fájl megosztás | |
|---|---|---|---|---|
| | Késleltetés | Sávszélesség | Késleltetés | Sávszélesség |
| a | 353 | - | 5.3 | - |
| b | 116 | - | 3.8 | - |
| c | 213 | 0,02 | 3.5 | 1,25 |
| d | 53 | 0,08 | 6.1 | 0,7 |
| e | 262 | 4.02 | 144 | 7,32 |
| f | 240 | 4.39 | 104 | 8,5 |

2. táblázat. Eredmények

**Összefoglaló.** A téziscsoportban bemutattunk egy egyszerű, olcsó és hatékony tároló keretrendszert, ami ugyan már meglévő ötletek alapján valósult meg, egyedi környezetre adaptáltuk. A mérések igazolták a várakozásainkat és a rendszer képes volt a laboratóriumi körülmények között megfelelő teljesítmény produkálására.

**Saját eredmények:.** A téziscsoport eredményeit az alábbi cikkekben publikáltuk: [1], [3]. Az itt megfogalmazott eredményeknél, a laboratóriumi változékonyság mérését kivéve, a szerző hozzájárulása volt meghatározó.

# Összefoglaló

Az értekezés célja, hogy felvázolja, a legfontosabb trendeket, melyek meghatározzák a hálózatok fejlődési irányait és rávilágítson néhány területre, ahol szükséges illetve célszerű az alkalmazásoknak a hálózat képességeihez illeszkedni. Az értekezés két nagy részre, ezen belül öt tézis csoportra illetve tizenhárom tézisre bontva tárgyalja az alkalmazások és a hálózat viszonyait.

Az első részben az IP hálózat és ezen belül az Internet lehetséges fejlődési irányait vázoltuk fel. Itt bemutattuk azt, hogy fontos trend lehet a kontextus alapú kiszolgálást biztosító szolgáltatások megjelenése. Ezen szolgáltatások sokkal komolyabb erőforrás igénnyel fognak rendelkezni, mint a napjainkban is megtalálható állapottartó működést igénylő megoldások. Megvizsgáltuk, hogy az állapotok kezelése milyen nehézségekbe ütközik különböző hálózati rétegeket kiszolgáló aktív eszközökön és mérésekkel prezentáltuk a mai eszközök teljesítménybéli korlátait. Megállapítottuk, hogy az aktív eszközökön kezelt kapcsolatok száma jelenti az állapottartó szolgáltatások kritikus pontját. Ezután bemutattuk azt, hogy a P2P alkalmazások egy része igen sok kapcsolatot tart fenn, ami a fentiek fényében a kiszolgáló hálózaton skálázhatósági problémákat fog felvetni, illetve vet fel már ma is. A megállapításainkat az alábbi tézisekbe foglaltuk össze:

- 1. Tézis: Az állapottartó szolgáltatások teljesítménye a különböző hálózati rétegekben nagymértékben függ a kezelt pont-pont és pont-több pont típusú kapcsolatok számától.
- 2. Tézis: Az ISP barát P2P alkalmazások fejlesztőinek figyelembe kell venniük az alkalmazás által generált kapcsolatok számát is.

A tézis csoportban bemutatott eredmények a szerző eredményei. A csoportküldés teljesítményével kapcsolatos eredményeket a [2] cikkben publikáltuk.

A második részben négy példát mutattunk be az infrastruktúrához illeszkedő alkalmazásokra. A P2P botnetek detektálása egyre több fejtörést okoz a hálózatok működtetőinek és a kutató közösségnek is. Rámutattunk arra, hogy kis fokszámú DHT alapokon készíthető olyan botnet, ami TDG módszerrel gyakorlatilag detektálhatatlan az Internet tetszőlegesen választott pontjából. Ebből az eredményből az következik, hogy a különböző hálózat

üzemeltetőknek együtt kell működniük a botnet-ek felderítésében. Az eredményeket három tézisben fogalmaztuk meg:

- 3. Tézis: Lehetséges olyan P2P botnet-et készíteni, amely nem detektálható TDG módszerrel egyetlen tetszőlegesen kiválasztott autonóm rendszerből sem.
- 4. Tézis: Lokalizáció és kapcsolat klaszterezés segítségével elérhető, hogy a botnet felderíthetetlen lesz a TDG módszerrel, amennyiben csak egy autonóm rendszerből vizsgáljuk.
- 5. Tézis: A kapcsolatok klaszterezése jelentős hatással bír a botnet rejtőzködésére, míg a lokalizáció hatása kevésbé jelentős.

A téziscsoportban megfogalmazott eredményeket a [6] cikkben publikáltuk. Az 5. Tézis a szerző saját eredménye, míg a többi a társszerzővel elért közös eredmény.

Ezek után azt vizsgáltuk meg, hogy lehetséges-e olyan, az előző téziscsoportokban fontosnak tartott, kis fokszámú DHT alapú P2P algoritmus létrehozása, amely skálázható és robosztus is. Elméleti és kísérleti eredmények segítségével bemutattuk, hogy lehetséges, amennyiben néhány általunk javasolt eljárást alkalmazunk. Az eredmények tézisként megfogalmazva:

- 6. Tézis: A Symphony alapú topológia skálázható azaz, $\lim_{h\to\infty} p(h,q) > 0$, amennyiben (i) minden kapcsolathoz van $O(\log N)$ tartalék kapcsolat, vagy amennyiben (ii) minden kapcsolathoz van $O(\log\log N)$ tartalék kapcsolat és $q \leq e^{-2} \approx 0.135$.
- 7. Tézis: Lehetséges hatékony és nagyon változékony hálózaton is üzembiztos kis fokszámú DHT alapú P2P algoritmust készíteni.
- 8. Tézis: A T-Man alapú pletyka-hálózatot kiegészítve az alábbi algoritmusokkal: *Előretekintés, Fokszám biztosítás, Rétegezés, Rövid kapcsolatok elkerülése* a megvalósított DHT képes hatékonyan működni, és igen változékony hálózat (Weibull k=0.5) esetén is megőrzi stabilitását.

A téziscsoporthoz tartozó eredményeket a [5] cikkben publikáltuk. A tézisek közül a 8. a szerző saját eredménye, míg a többi közös munka gyümölcse.

A SIP tömörítés ugyan nem kapcsolódik szorosan a P2P fedőhálózatokhoz és a kapcsolatok számához, de egy másik fontos területhez, a 3G alapú rendszerekben használt protokolloknak viszont szerves része. Ebben a téziscsoportban azt elemeztük, hogyan lehet/érdemes a vezetékes közegre optimalizált SIP protokollt vezetékmentes közegre alakítani. Kiderült, hogy változtatás nélkül nem érdemes a hagyományos tömörítési algoritmusokat alkalmazni. A 3G kapcsolatok felépítésénél nem csak a tömörítés, hanem a késleltetés, a processzor, valamint a memória-igény is fontos szempont. Miután megfelelően módosítottunk néhány jól ismert tömörítési algoritmust, kiderült, hogy érdemes őket használni, viszont nincs optimális megoldás. A legfontosabb megállapításainkat az alábbi tézisekben fogalmaztuk meg:

- 9. Tézis: Változtatás nélkül a klasszikus tömörítési algoritmusokat nem érdemes SIP tömörítésre használni.
- 10. Tézis: A módosított Deflate algoritmus (futáshossz kódolás + kontextus modellezés) teljesít legjobban, amennyiben a SIP kapcsolat-felépítés időtartama a döntő.
- 11. Tézis: A módosított LZ77-es algoritmus a leggyorsabb a kitömörítő oldalon, míg a tömörítésnél a kontextus modellezésen alapuló eljárások a leggyorsabbak (pl.: előtag mentes eljárások).

A téziscsoportban leírt eredményeket a [4] cikkben publikáltuk és elérésükben a szerző munkája volt meghatározó.

Egy másik érdekes példa a hálózati környezethez való alkalmazkodásra egy olyan elosztott fájl tároló rendszer, amely a számítógépes laboratóriumokra jellemző változékonyságban is képes megbízhatóan és hatékonyan működni. A konzisztencia biztosításához a klasszikus Paxos algoritmust és annak néhány általunk javasolt kiegészítését használtuk fel. Az adatok redundanciáját a szintén klasszikus Reed-Solomon algoritmussal biztosítottuk. A téziscsoportban bemutattuk, hogy a fent említett algoritmusokra alapozva lehetséges nagy teljesítményű, skálázható és robosztus elosztott tárolót építeni.

- 12. Tézis: A napjainkban rendelkezésre álló személyi számítógépek számítási kapacitása alkalmas arra, hogy a Reed Solomon alapú kódoláson alapuló hibajavító megoldásokat használjunk a laboratóriumi változékonyság okozta hibák elfedésére.
- 13. Tézis: Az általunk kiegészített Paxos alkalmas arra, hogy a számítógépes laboratóriumokban tapasztalt változékonyság esetén biztosítsa a konzisztenciát.

A téziscsoport eredményeit az alábbi cikkekben publikáltuk: [1], [3]. Az itt megfogalmazott eredményeknél a laboratóriumi változékonyság mérését kivéve a szerző hozzájárulása meghatározó.

Végezetül egy összesítés a téziscsoportokról és a megfelelő publikációkról:

| | Tézis csoport | Tézisek | Publikáció |
|---|---|---|---|
| I/1 | Kapcsolatok vs. Állapottartó szolgáltatások | 1. Tézis, 2. Tézis | [2] |
| II/1 | Rejtőző botnet-ek | 3. Tézis,4. Tézis, 5. Tézis | [6] |
| II/2 | Kis fokszámú DHT változékony hálózatban | 6. Tézis, 7. Tézis, 7. Tézis | [5] |
| II/3 | SIP tömörítés | 9. Tézis, 10. Tézis, 11. Tézis | [4] |
| II/4 | Skálázható tároló | 12. Tézis, 13. Tézis | [1] [3] |

3. táblázat. Tézis csoportok, tézisek és publikációk

# Köszönetnyilvánítás

Az értekezéshez és az eredményekhez szükséges munka sok időt és erőfeszítést igényelt. Ez azonban nem csak az én időm volt, hanem a családomé is. Ezért itt szeretném feleségemnek, Andreának valamint fiaimnak, Vilmosnak, Máténak és Andrásnak megköszönni azt a sok türelmet és támogatást, amit számomra nyújtottak. Szeretném megköszönni édesapámnak, édesanyámnak és öcsémnek a támogatást, amivel ezt a munkát elősegítették. Feleségem szülei is sokat tettek azért, hogy a megfelelő pillanatokban a munkára tudjak koncentrálni. A hosszú beszélgetések nélkül, melyeket konzulensemmel Jelasity Márkkal folytattunk és ahol az egyedi intuíciója segítségével mindig jó irányt mutatott, ez a mű nem jöhetett volna létre. Sokat köszönhetek a Szoftverfejlesztés tanszék vezetőjének Gyimóthy Tibornak is, aki hozzájárult ahhoz, hogy kellő időt tudjak a munkára szánni és hasznos tanácsokkal látott el. Szeretnék még köszönetet mondani Fidrich Mártának is, akivel a kutatás sok aspektusát átbeszéltük. A vezetékmentes laboratórium régi és jelenlegi munkatársai is sok segítséget nyújtottak nekem a teljesség igénye nélkül: Roszik György, Bagrij Péter, Sey Gábor, Sógor Zoltán, Dombi József Dániel, Kasza Mikós, Szűcs Vilmos, Béládi Róbert, Végh Ádám és Molnár Gábor. Nekik is köszönettel tartozom.

A dolgozat nyelvi lektorálásában David P. Curley a lehetetlen határidők ellenére is nagyszerű munkát végzett, ezért külön köszönet jár neki.

Hiszek abban, hogy az élet fontosabb pillanataiban nem véletlenül dőlnek el a dolgok, hanem egy mindenható erő szándéka szerint történnek. Boldog vagyok, hogy a céljaim kijelölésében és azok elérésében is számíthattam az Ő segítségére.

*Bilicki Vilmos, 2010. Április*

# Hivatkozások

[1] Vilmos Bilicki. Lanstore: a highly distributed reliable file storage system. In *.NET Technologies2005 conference proceedings(.NET'05)*, volume 3, pages 47–57, Plzen, Czeck Republic, 2005. University of West Bohemia. http://wscg.zcu.cz/ROTOR/Journal/Archive/2005$_Vol_3.pdfpage = 59$.

[2] Vilmos Bilicki. Testing and verifying an ipv6 based multicast network. In *ICCGI '06: Proceedings of the International Multi-Conference on Computing in the Global Information Technology*, pages 3–13, Washington, DC, USA, 2006. IEEE Computer Society.

[3] Vilmos Bilicki and József Dombi. Building a general framework for the consistency management of distributed applications. In *.NET Technologies 2006 conference proceedings(.NET'06)*, volume 3, pages 55–63, Plzen, Czeck Republic, 2006. University of West Bohemia. http://dotnet.zcu.cz/NET$_2006/Papers_2006/!Proceedings_Full_Papers_2006.pdf$.

[4] Manta Fidrich, Vilmos Bilicki, Zoltan Sogor, and Gabor Sey. Sip compression. *Periodica Polytechnica, Electrical Engineering.*, 47(1-2):37–56, 2003. http://www.inf.u-szeged.hu/ bilickiv/research/CSCS2002.ps.

[5] Márk Jelasity and Vilmos Bilicki. Scalable p2p overlays of very small constant degree: An emerging security threat. In *SSS '09: Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 399–412, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] Márk Jelasity and Vilmos Bilicki. Towards automated detection of peer-to-peer botnets: On the limits of local approaches. In *2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09)*, pages 1–8. USENIX, 2009. http://www.usenix.org/events/leet09/tech/.

**Társszerzői nyilatkozat**

Kijelentem, hogy ismerem *Bilicki Vilmos* PhD fokozatra pályázó *Infrastructure aware applications* című disszertációját. A disszertációban szereplő és a

- M. Fidrich, V. Bilicki, Z. Sógor, G. Sey.: SIP compression, Periodica Polytechnica Electrical Engineering, 2003 47/1-2

cikkben publikált eredményekre vonatkozóan kijelentem, hogy a pályázó hozzájárulása volt a meghatározó

Szeged, 2010. április 12.

Fidrich Márta

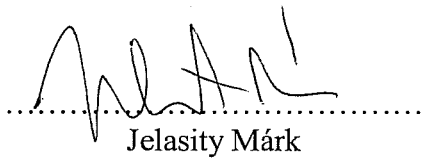
**Társszerzői nyilatkozat**

Kijelentem, hogy ismerem *Bilicki Vilmos* PhD fokozatra pályázó *Infrastructure aware applications* című disszertációját. A disszertációban szereplő és a

- Márk Jelasity and Vilmos Bilicki. Scalable p2p overlays of very small constant degree: An emerging security threat. In SSS '09: Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems, pages 399–412, Berlin, Heidelberg, 2009. Springer-Verlag.
- Márk Jelasity and Vilmos Bilicki. Towards automated detection of peer-to-peer botnets: On the limits of local approaches. In 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09), pages 1–8. USENIX, 2009. http://www.usenix.org/events/leet09/tech/.

cikkekben publikált eredményekre vonatkozóan kijelentem, hogy: az 5-ös és a 8-as tézis a szerző munkájának eredménye volt míg a két cikkhez kapcsolódó többi tézis közös munka eredménye volt (3,4,6,7)

Szeged, 2010. április 8.



Jelasity Márk

**Társszerzői nyilatkozat**

Kijelentem, hogy ismerem *Bilicki Vilmos* PhD fokozatra pályázó *Infrastructure aware applications* című disszertációját. A disszertációban szereplő és a

- Vilmos Bilicki and József Dombi. Building a general framework for the consistency management of distributed applications. In .NET Technologies 2006 conference proceedings(.NET'06), volume 3, pages 55–63, Plzen, Czeck Republic, 2006. University of West Bohemia. http://dotnet.zcu.cz/NET_2006/Papers_2006/!Proceedings_Full_Papers_2006.pdf.

cikkben publikált eredményekre vonatkozóan kijelentem, hogy:

- a kabinetben megtalálható gépek stabilitásának vizsgálata az én munkám eredménye

A többi a cikkben megtalálható eredmény esetében Bilicki Vilmos hozzájárulása volt a meghatározó

Szeged, 2010. április 8.

.....

Dombi József Dániel

**Társszerzői nyilatkozat**

Kijelentem, hogy ismerem *Bilicki Vilmos* PhD fokozatra pályázó *Infrastructure aware applications* című disszertációját. A disszertációban szereplő és a

– • M. Fidrich, V. Bilicki, Z. Sógor, G. Sey.: SIP compression, Periodica Polytechnica Electrical Engineering, 2003 47/1-2

cikkben publikált eredményekre vonatkozóan kijelentem, hogy a pályázó hozzájárulása volt a meghatározó

Szeged, 2010. április 8.

.....

Sógor Zoltán

**Társszerzői nyilatkozat**

Kijelentem, hogy ismerem *Bilicki Vilmos* PhD fokozatra pályázó *Infrastructure aware applications* című disszertációját. A disszertációban szereplő és a

- M. Fidrich, V. Bilicki, Z. Sógor, G. Sey.: SIP compression, Periodica Polytechnica Electrical Engineering, 2003 47/1-2

cikkben publikált eredményekre vonatkozóan kijelentem, hogy a pályázó hozzájárulása volt a meghatározó

Budapest, 2010. április 8.

.....

Sey Gábor