■ KUTATÁS

Peer-to-peer alapú betörésérzékelés

CZIRKOS ZOLTÁN, HOSSZÚ GÁBOR

Budapesti Műszaki és Gazdaságtudományi Egyetem, Elektronikus Eszközök Tanszék
hosszu@nimrud.eet.bme.hu

Lektorált

Kulcsszavak: P2P, betörésvédelem, NIDS, átfedő (overlay) hálózat

Cikkünkben egy új hálózati biztonsági eljárást mutatunk be. A működés során a módszert megvalósító szoftveregyedek a hálózaton egy egyenrangú (Peer-to-Peer, P2P) felépítésű alkalmazási szintű hálózatot hoznak létre, amelyen megosztják egymás között az általuk érzékelt betörési kísérletek adatait. Az összegyűlt tapasztalatok az összes résztvevő biztonságát növelik. A rendszer teljesen decentralizált, ezért instabil hálózat esetén, valamint több gépet egyszerre érő támadások során is működőképes marad. A rendszer megvalósítására a Kademlia P2P átfedőt találtuk a legalkalmasabbnak. Ennek megbízhatóságát, illetve a felette megvalósított broadcast üzenetküldő algoritmust is elemezzük.

1. Bevezetés

Több olyan biztonsági program létezik, amelynek különböző gépeken futó példányai egymással kapcsolatot tartanak [3,4]. Az általunk kidolgozott szoftver újdonsága az, hogy az egyes gépeken futó egyedek az Interneten egy *egyenrangú* (Peer-to-Peer, P2P) átfedő (overlay) hálózatot hoznak létre. A szerveződés önműködő, felhasználói beavatkozást nem igényel. Ez a hálózati felépítés nagy stabilitást biztosít, amelyre az egyes egyedek között a tapasztalatok gyors, megbízható átadása miatt van szükség. A rendszer felépítéséből adódóan a hálózati hibák és a támadások miatt megbízhatatlan hálózaton is működőképes marad.

A szoftvert *Komondornak* neveztük el, hiszen feladatkörében sok mindenben hasonlít a házőrzésben híresen kiváló kutyafajtára.

A cikk a szakirodalom jelenlegi állásával foglalkozó második szakaszában általánosságban bemutatja a P2P átfedőket, valamint a két legelterjedtebb elosztott betörésérzékelő rendszert. Ezekután ismertetjük az általunk kidolgozott Komondor rendszer tervezési szempontjait és működését. A negyedik szakasz az alkalmazott Kademlia átfedő felépítését magyarázza el. A részletek ismertetése után igazoljuk, hogy az átfedő alkalmas az érzékelő rendszer megbízható megvalósítására, végül pedig összefoglaljuk a cikkben közölt állításokat, valamint a Komondor rendszer működésével és hatékonyságával kapcsolatban összegyűlt eddigi tapasztalatokat.

2. Irodalmi áttekintés

2.1. A P2P átfedők és típusaik

Az egyenrangú (P2P) közlési modellen alapuló alkalmazási szintű *átfedő (overlay)* hálózatok lehetnek strukturáltak és nem strukturáltak. Az utóbbi csoportba tartozó átfedőknél egy-egy *egyenrangú (peer)* könnyen nélkülözhető, a hálózat rugalmasan kezeli a kilépéseket és a meghibásodásokat. Az alkalmazásoknál szokásos keresést is az egyes egyedek maguk végzik el, a keresési kéréseket egymásnak továbbítva [7]. Ilyen nem strukturált hordozók a *Gnutella*, a *Freenet* és a *FastTrack* [2].

Az átfedők másik csoportja az úgynevezett *elosztott hash táblázatok* (Distributed Hash Table, DHT). Ezek a hálózatok kulcsérték-párokat tárolnak és egy adott kulcshoz tartozó érték, adat gyors megkeresését teszik lehetővé. A nem strukturált hálózatokkal szemben itt az egyedek közötti kapcsolatok meghatározottak; a hálózat topológiája pontosan definiált. Minden eltárolt adat (fájl) meghatározott helyre, adott egyedhez kerül. Az egyedek egy nagy értékkészletből választott, például 160 bites *csomóponti azonosítóval* (Node IDentifier, NodeID), rendelkeznek. Hasonlóan az egyes adatokhoz (fájlokhoz) is hozzá van rendelve kulcs, ami például a fájl nevének hash értékéből képzett, a NodeID-vel azonos bitszámú (példánkban 160 bites) *fájlazonosító* (File IDentifier, FileID). Minden egyed azokat a kulcsérték-párokat tárolja, amelyek kulcsának valamilyen hash függvény szerinti értéke legközelebb van a saját csomóponti azonosítójához. A NodeID bitjeinek száma megegyezik az egyedek által használt hash függvény kimeneti bitjeinek számával. Így az egyes értékekről a kulcs ismeretében könnyen eldönthető, hol kell keresni azokat. Ezt az eljárást *consistent hashing*nek nevezik [8,9].

Az egyes strukturált hálózatok a kapcsolatok szervezésében, az átfedőn belüli útválasztás algoritmusában, két azonosító közötti távolság függvényben különböznek egymástól.

2.2. Az elosztott betörésérzékelés

A manapság széleskörűen használatos elosztott betörésérzékelő rendszerek általában centralizáltak és csak adatgyűjtésre szolgálnak [4]. A valódi, decentralizált és beavatkozásra is képes alkalmazások csak az utóbbi időben jelentek meg.

A PROMIS nevű védelmi rendszer (és elődje, a Netbiotic) a részben centralizált hálózatot építő JXTA keretrendszert használja az érzékelt támadások adatainak megosztására [12]. A PROMIS rendszerbe beépülő egyedek a többiektől információt kapnak az érzékelt gyanús események számáról és az alapján automatikusan állítják az operációs rendszer és a rendszerben telepített Webböngésző biztonsági szintjét. Ez az eljárás általános védelmet ad a károkozó programok ellen, de egyben csökkentheti is a használhatóságot. A megközelítés hasonló a hétköznapi életből ismert járványok megelőzéséhez.

A Spamwatch nevű levélszemét (spam) szűrő rendszer a Tapestry hálózatra épül [13]. A program egy levelező alkalmazásba épülő bővítmény. Az egyes, felhasználók által levélszemétként megjelölt levelek adatait a rendszer a DHT-ben tárolja; más felhasználóknál így ugyanaz az üzenet automatikusan törölhető. A DHT alkalmazása miatt a lekérdezés gyors és csak kis hálózati forgalmat generál.

3. A kifejlesztett rendszer

A Komondor rendszerben a betörések érzékelését az egyedek elosztottan végzik, egy Kademlia alapú DHT segítségével [1]. A rendszer tervezésekor a következő célokat tartottuk szem előtt:

– stabil átfedő hálózat építése a tapasztalatok megosztására;
– az átfedőn a támadások hírei a lehető leggyorsabban terjedjenek;
– a rendszer decentralizálása, az egyedek nélkülözhetőségének biztosítása;
– a tapasztalatok alapján az egyes egyedek biztonsági réseinek elfedése.

A Komondor szoftver különböző gazdagépeken futó példányai látszólagos, alkalmazási szintű, úgynevezett átfedő (overlay) hálózatba szerveződnek. A tapasztalatok megosztásának sebessége nagyban függ az alkalmazott hálózati modelltől. A decentralizáció és a megbízhatóság biztosításához célszerű a rendszert egyenrangú szoftver egyedek együttműködését megvalósító P2P átfedőre építeni [11], szemben a nagyobb meghibásodási kockázatot jelentő ügyfél-kiszolgáló (client-server) rendszerekkel.

A Komondorban a gyanús események rögzítésére strukturált átfedőt, vagyis egy DHT-t alkalmazunk. A kulcsok a támadók IP címei, az értékek pedig a támadások adatai. Adott támadóról szóló jelentések a közös hash függvény használata miatt egy pontban összegződnek. Ha egy adott Komondor egyed, a hozzá beérkező jelentések elemzése alapján úgy dönt, hogy a jelentésekben szereplő IP címen egy támadó tevékenykedik, akkor szórt üzenetet (broadcast) indít az átfedőn, hogy jelentse a támadás tényét az összes többi Komondor egyednek. Mindenkinek érdeke ugyanis, hogy a felismert támadó ellen védekezni tudjon. A PROMIS rendszertől eltérően a Komondorban a védekezés célzott, csak az adott támadó ellen irányul.

A több helyen történő érzékelés és az adatok összevetése igen hatékony lehet. Tekintsük a következő példát. Adott egy támadó, aki levélszemét (spam) küldése céljából keres helytelenül konfigurált SMTP kiszolgálókat. Eljut egy alhálózatra, amelyen belül több géphez is megpróbál kapcsolódni azért, hogy feltérképezze, hol fut egyáltalán levelező szerver. A támadó által kezdeményezett TCP kapcsolatok a nyitott portok felé felépülnek, aztán meg is szakadnak. Egyetlen felépülő és megszakadó TCP kapcsolat nem jelent önmagában támadást, utalhat hálózati hibára, vagy egy felhasználó által megszakított levélküldésre is. Ha azonban ez a jelenség az alhálózat többi gépén, például a szomszédoknál megismétlődik, az már gyanúra ad alapot. A Komondor rendszerben a támadók IP címe alapján dől el, hogy melyik Komondor egyed lesz felelős a támadó azonosításáért, ezért a hálózati szintű támadások érzékeléséhez a gyanús eseményekről szóló jelentéseket meg kell osztani.

Kutatásunk fő célja a Kademlia P2P átfedő megbízhatóságának vizsgálata és a P2P alapú betörésérzékelés lehetőségeinek megismerése. A jelenleg megvalósult, Linux és Microsoft Windows alapú Komondor implementációk az érzékeléshez a Snort-ot és az operációs rendszer naplófájljait használják; beavatkozáshoz pedig az adott számítógépen működő tűzfalat. A jövőbeli fejlesztések során a felsoroltakon kívül más érzékelő és beavatkozó modulok is elképzelhetőek a Komondor rendszerben.

4. A Kademlia átfedő alkalmazása a Komondorban

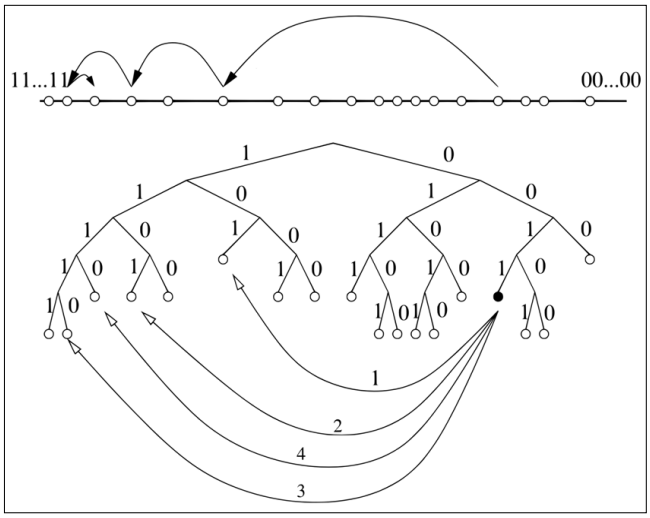
4.1. A Kademlia átfedő felépítése

A Kademlia átfedő megbízhatóságának vizsgálatához és az üzenetszóró algoritmusok bemutatásához ebben a részben vázlatosan ismertetjük a Kademlia átfedő felépítését és működését.

A Kademlia elosztott hash táblázatokat (Distributed Hash Table, DHT) használ, az ilyen típusú átfedő hálózatokat gyakran DHT-knek nevezik. A Kademlia DHT-ben egyedek alkalmazási hálózatbeli címeik szerint egy bináris fával ábrázolhatók [5]. A Kademlia egyedek minden távoli részfából ugyanannyi kapcsolati lehetőséget (IP hálózati címet, port számot) tartanak nyilván; ezeket a listákat *k-vödröknek* nevezik. A listák mérete egy *k* szám, amely rendszerszintű konfigurációs paraméter. Egy nagy hálózat esetén a nagyobb részfákban jóval több egyed van, mint *k*, vagyis a fának a távoli részeiről egy egyednek arányaiban kevesebb tudása van, míg a hozzá legközelebbi egyedekről teljes képe.

Az útválasztás az *1. ábrán* látható módon történik. (Azokat a részfákat, amelyek csak egyetlen egyedet tartalmaznak, a Kademlia irodalmában nem szokás külön faként ábrázolni, csak levélként. A példában ettől függetlenül az összes egyed azonosítója 5 bites.) Ha a 00110 című egyed az 11100 címűnek üzenne, nem kell mást tennie, mint küldeni egy üzenetet *bárkinek* az 1-es-

sel kezdődő részfában, akik már jobban fogják ismerni az 11-gyel kezdődőek részfáját stb. A lekérdezések sorrendjét a számozott nyilak mutatják. Az üzenetküldés láthatóan *O(log n)* lépésben megvalósul. Az egyedek két azonosító távolságát (hálózati címek vagy hálózati cím és kulcs) a *kizáró vagy* függvénnyel számolják. Minél nagyobb helyiértéken találunk a távolságban 1-est, annál távolabbi részfában van a keresett egyed. Ezért ábrázolható a hálózat bináris fával; ez az XOR topológia. A művelet szimmetriája miatt egy adott egyed szempontjából a bejövő és a kimenő üzenetek eloszlása azonos. Az egyedek útválasztási táblája így az üzenetek hatására automatikusan frissül; a hálózat önmegerősítő.



1. ábra Útválasztás a Kademlia átfedőben

Más DHT rendszerekhez képest szokatlan tulajdonsága a Kademliának az egyedek nagyfokú szabadsága. Az adott kulcs tárolásához a tároló egyed az üzenetet nem „útjára indítja”, hogy majd a megfelelő egyedhez eljutva az érték tárolódjon, hanem ő maga keresi fel az adott kulcshoz legközelebbi egyedet. Ez leegyszerűsíti a replikáció (replication, másodlatolás) kezelését. Egy adott kulcs-érték párt eltárolni szándékozó egyed nem a kulcshoz legközelebbi egyednek, hanem a legközelebbi *k* darab egyednek küldi el az üzenetet. Ezzel a *k* szám megválasztása hatással van egyrészt az áfedő hálózat stabilitására van hatással. Azonban, mint az az alábbiakban látni fogjuk, ha *k*>1 az eltárolt kulcsok elérhetősége is javul. A Kademlia protokoll tulajdonságaiból adódik ugyanis, hogy egy adott egyed az alkalmazási szintű címtartomány megfelelő részeiből igyekszik legalább *k* darab másik egyedet ismerni. Az elérhetőségeket szükség szerint óránként frissíti; a *k* számot úgy kell megválasztani, valószínűtlen legyen, hogy az összes *k* darab ismert egyed egy órán belül elhagyja a hálózatot.

A hálózatot elhagyó egyedek a Kademliában *nem küldik el* az eltárolt kulcsaikat a szomszédjaiknak. Vagyis ha az egyik egyed eltűnik a hálózatból, akkor a benne tárolt kulcsok is vele együtt eltűnnének, hacsak nem tárolták azt is több helyen. Vegyük észre, hogy egy DHT-ben egy adott azonosító elérhetősége azonos egy adott kulcs elérhetőségével. Vagyis a replikáció fokának érdemes ugyanazt a *k* számot választani, amit a fentiekben a stabilitás fokának választottunk. Így gyakorlatilag csak erre az egyetlen egy rendszerszintű konfigurációs paraméterre van szükség.

4.2. A Kademlia átfedő megbízhatósága

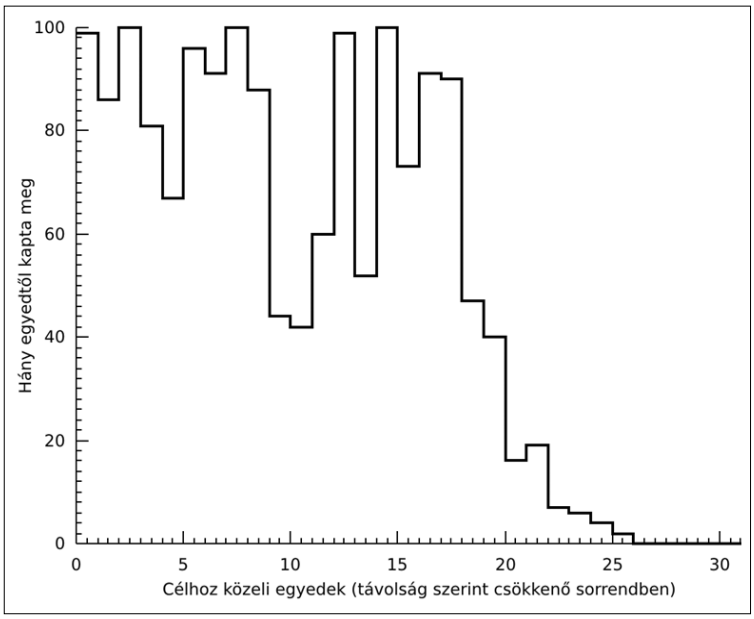
A Komondor rendszerben az átfedőt egy módosított Kademlia protokoll hozza létre. Így a Komondoron végzett mérések egy része a Kademlia tulajdonságait is leírja. A Komondor eddigi futtatásai bebizonyították, hogy a Kademliában a másodlatolásnak jóval nagyobb a szerepe, mint más típusú átfedőkben. Ugyanis az egyes Kademlia egyedek esetenként nem érik el egymást csomagvesztés, csomagszűrés, címfordítás vagy hasonló okok miatt. Ezért lehetséges, hogy egy adott, egyetlen egyednél eltárolt kulcsot egy másik egyed nem képes elérni, mivel nem tud hozzá csatlakozni.

A replikáció részben megoldást ad a problémára. Ha nem egy konkrét egyednél, hanem a Kademlia egyedek egy tartományában (*k* számú egyednél) tároljuk el a kulcsot, a több egyed közül nagyon valószínű, hogy lesz legalább egy elérhető. Illetve, ha nincs is teljesen egyforma tudomásuk az egyes egyedeknek az adott kulcshoz közeli másik egyedekről, a másodlatolás által kiválasztott intervallumok átfedése biztosítja ezt. (A strukturált hálózatokban a gyakran ki- és belépő egyedek miatt szokott előállni ez az eset; a jelenség neve a *high churn* [10].)

Az előbbi állítás bizonyításához készítettünk egy *Kadsim* nevű szimulátor programot. Bár az elvégzett szimulációk főként a Komondor igényeit tartották szem előtt, de a kapott eredmények általánosak, így minden egyéb Kademlia protokollra épülő átfedő hálózatra is érvényesek. A Kadsim lényege, hogy adott számú egyedhez létrehoz egy kapcsolati mátrixot, amely tulajdonképpen a kapcsolódási lehetőségek gráfjának szomszédsági mátrixa.

Adott egy üzenet, amely csupán egy véletlenszerűen kiválasztott azonosító; a Komondor rendszerben ez a támadó IP címének hash-elt értéke. A Komondor kifejezett igénye, hogy legyen az átfedőben egy olyan egyed, ahol erről az adott támadóról szóló jelentések összefutnak. Ezért a Kadsim azt az esetet modellezi, amikor az átfedőben lévő *összes* egyed érzékel az adott helyről érkező támadást. Mindenki megkeresi azt a másik egyedet, akinek a címe legközelebb van a hash-elt értékhez, nem számítva azokat, amelyek nem elérhetőek. A szokásos, fájltárolásra használatos DHT alkalmazások esetén ez ugyanígy történne; egy adott kulcsot kell megkeresni a kulcshoz közeli egyedek szűk környezetében.

A szimuláció végeztével a program a kulcstól való távolság szerint növekvő sorba rendezi az egyedeket és grafikonon ábrázolja, hogy melyikük hány üzenetet kapott. Ideális esetben, ha minden kapcsolat működik, a grafikon egy lépcső: a kulcshoz legközelebbi *k* darab egyed az összes üzenetet megkapja, a többieknek pedig nem küldenek semmit. Hálózati hibák esetén a görbe ellaposodik és kiszélesedik *(2. ábra)*.



2. ábra
Kulcsok tárolása a Kademlia átfedőben, replikáció: 16-szoros, hibás kapcsolatok: 20%

Például, ha a másodlatolás foka *k=16*, és az egyik egyed nem éri el a 12. és a 15. legközelebbi egyedet, akkor üzenetet fog küldeni a 16. és 17. legközelebbihez is.

Ha a hálózati hibák eloszlása egyenletes, akkor nem lesz a hálózatnak olyan pontja, ahol az összes támadási jelentés összegződik, bármilyen magasra választjuk is a replikáció fokát. Ilyen esetben a Kademlia kifejezetten rossz választás lenne. A valóságos hálózatok, így az Internet is, viszont szerencsére nem ilyenek: a hibák általában nem egyenletesen oszlanak el. Például vannak olyan számítógépek, amelyek publikus IP címmel rendelkeznek, azokhoz közvetlenül lehet kapcsolódni; akik pedig címfordító mögött vannak, azokhoz nem. Az eloszlás sokféle a lehet, a Kadsim egy hatványfüggvény szerinti hibaeloszlást modellez. Nem egyenletes eloszlás esetén a célhoz közeli egyedek közül lesz olyan is, aki képes fogadni üzeneteket.

A szimuláció azt mutatja, hogy már a nem túl nagyfokú, például *k=8*-as replikáció is igen nagy valószínűséggel biztosítja, hogy van olyan egyed, amelyiknél az összes jelentés összegyűlik *(3. ábra)*. Ez száz egyedhez képest talán soknak tűnik a megszokott P2P alkalmazásokban, de az egyedek számának növelésével *nincs szükség* a növelésére. A kiválasztott egyedek közül nagy valószínűséggel lesz olyan, amelyik mindenki által elérhető.

4.2.1. A hálózati hibák matematikai modellezése

A DHT-kben az egyes egyedek a hálózathoz csatlakozáskor véletlenszerűen választanak maguknak egy azonosítót az igen nagy címtartományból, illetve a hash függvények kimenete is véletlen számnak tekinthető. Ezért az eltárolandó adatokhoz látszólag véletlenszerűen választ a hálózat felelős egyedet. Ez a tulajdonság lehetővé teszi az átfedő egyszerű matematikai modellezését.

Egy adott *m* azonosítójú egyedhez tartozó hálózati hibák *h(m)* számát a következő, (1) függvény adja meg:

$$h(m) = c \cdot \left(\frac{m}{n}\right)^{\alpha}, \qquad (1)$$

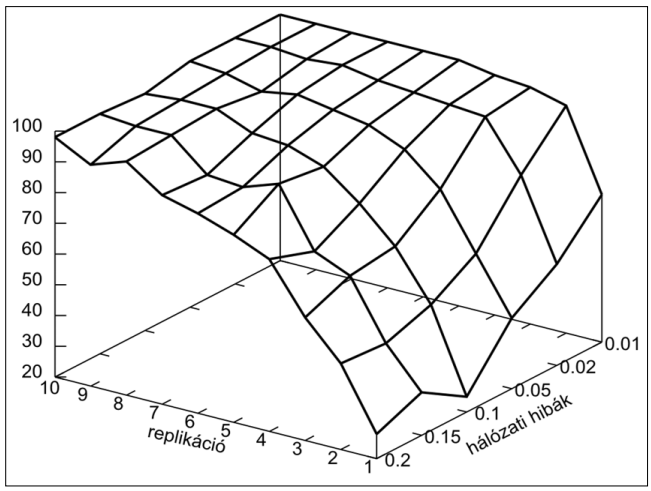
ahol *n* az összes lehetséges egyedek száma (*0≤m<n*). α a hálózati hibák eloszlását határozza meg, α=2 négyzetes eloszlás esetén. *c* a legnagyobb hálózati hibaarányt megadó állandó. Ezeket a paramétereket a modellezett átfedő alapját képező fizikai hálózaton végzett mérések alapján lehet meghatározni.

Az (1) a hibák valódi számának becslését adja és értéke nem feltétlenül egész szám. Ezzel szemben az egyedek hibáinak valódi száma, vagyis *n·h(m)* nyilvánvalóan csak egész szám lehet. Nagyobb számú hibák esetén az ebből adódó különbség elhanyagolható. Az (1) alapú becslés azonban nem alkalmazható kevés számú hiba esetén, vagyis abban az esetben, ha *n* h(m)* az értelmezési tartomány jelentős részén majdnem 0. Ugyanis a valóságban nincsen például 0,3 hiba, csak 0 vagy 1.

Mivel az átfedő a hibákkal teletűzdelt Interneten működik, nem várhatjuk el tőle, hogy tökéletes legyen. Meghatározhatunk viszont egy számszerűen megfogalmazott igényt, például elvárjuk az átfedőtől, hogy az esetek 99%-ában kikereshető legyen az eltárolt adat. Ha adott a megengedhető hibák β=1% aránya, a kikeresés sikerének valószínűsége 1-β, ha a *h(m)*≤β egyenlőtlenség fennáll a kiválasztott egyedre. Ezek azok az egyedek, akik az előírt aránynál több helyről elérhetőek.

A szokásos 128 vagy 160 bites azonosítók nagy száma miatt a címtartomány folytonosnak tekinthető. Mivel az egyedek véletlenszerűen kiválasztott azonosítókkal rendelkeznek, illetve a hash függvények kimenete is látszólag véletlenszerű és egyenletes eloszlású, *m/n* tulajdonképpen egy [0,1) intervallumból sorsolt véletlen számnak vehető. Ha az egyenlőtlenséget megoldjuk *m/n*-re, megkapjuk azon egyedek számát, amelyek megfelelnek a (2) kritériumnak.

3. ábra
Sikeres kikeresések százaléka a Kademlia átfedőben



$$\frac{m}{n} \le \sqrt[\alpha]{\frac{\beta}{c}} \qquad (2)$$

Legyen egy adott kikeresés sikerének valószínűsége *P'*. Mivel $0 \le m/n < 1$, és véletlenszerűen kiválasztott, a (3) egyenlőség fennáll *P'*-re.

$$P' \le \sqrt[\alpha]{\frac{\beta}{c}} \qquad (3)$$

Ha az átfedő másodlatolást (replikációt) is alkalmaz, az adatok *k* különböző helyen tárolódnak. Vagyis *k*-szor választhatunk egy [0,1) közötti véletlen számot. Ha a *k* alkalomból legalább egyszer teljesül a fenti egyenlőtlenség, a kikeresés sikeres. Kiszámítva az összes kikeresés sikertelenségének valószínűségét és azt 1-ből kivonva kapjuk (4)-et.

$$P = 1 - \left(1 - P'\right)^k \qquad (4)$$

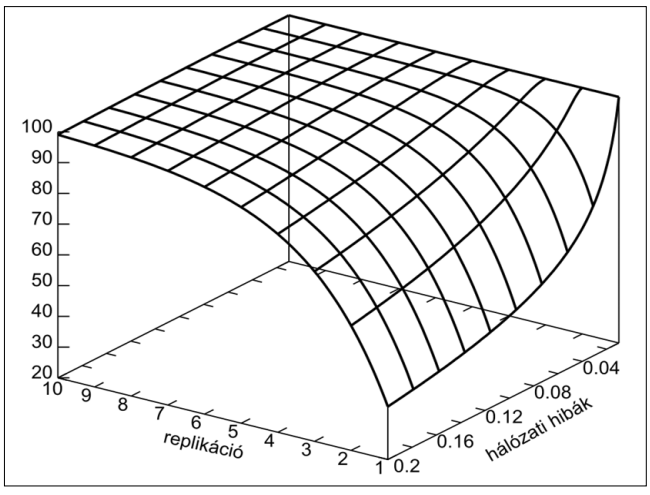
A (4) képlet azt a valószínűséget adja meg, hogy egy adott kikeresés sikeres lesz, a megadott számú hálózati hibák ellenére. Behelyettesítve a hibák számát és a kikeresések elvárt helyességének arányát, meghatározható belőle a szükséges replikáció mértéke.

A *4. ábra* adott hibaarány és replikáció mérték függvényében mutatja a kikeresés helyességének valószínűségét 1%-os megengedett hiba mellett. Látható, hogy még magas, 10%-os legnagyobb hibaarány esetén is elegendő a *k*=5-ös replikáció, hogy nagy valószínűséggel (*P*=80%) biztosítsuk a helyes működést. Ha az átfedőt csak néhány tíz egyed alakította ki, akkor a *k*=5 nagy számnak tűnhet, de nem szabad elfelejteni, hogy a meghatározott *k* érték bármilyen nagyszámú egyedre érvényes. A képlet a szimulációhoz hasonló eredményeket ad. Igen kis hibaarányok esetén mutatkozik eltérés, ahogyan az várható is volt az (1)-beli egyszerűsítés miatt.

4.3. Broadcast üzenetek P2P átfedőkben

A broadcast (egytől mindenkinek típusú) üzenetek küldése a P2P átfedőkben ritka, a résztvevő egyedek nagy száma miatt. Általában nem is terveznek olyan algoritmust, amely az ilyen típusú üzenetek szórását hivatott megoldani, mivel ez ellentmond az egyik fő tervezési szempontnak, a skálázhatóságnak. Vannak viszont olyan alkalmazások is, amelyek igénylik ezt az üzenettípust. Ide tartozik a Komondor is. Amikor egy egyed egy adott támadóról megfelelő számú jelentést gyűjtött, egy broadcast típusú, szórt üzenetet kell útjára indítson a hálózaton. Másik gyakori alkalmazás a tetszőleges típusú keresések megvalósítása az átfedőkben; a DHT-nek ugyanis ez nem alapszolgáltatása (például fájlcserélő esetén csak pontos fájlnévre tud keresni, részlegesre nem).

4. ábra
A sikeres kikeresések aránya a Kademlia átfedőben az alkalmazott becslés alapján számítva



A strukturált átfedők beépített topológiája, szervezettsége lehetőséget biztosít az ilyen üzenetek gyors és hatékony küldésére. Mindenképpen célszerű a meglévő topológiát használni erre a célra. Ennek egyik oka, hogy a meglévő topológián általában logaritmikus lépésben elérhető bármely egyed, vagyis a broadcast üzenet is logaritmikus időben el fog jutni minden egyedhez. Másik oka pedig, hogy az üzenet küldése közben nem szükséges új kapcsolatokat kialakítani, vagy kikereséseket indítani. Gyakorlatilag a topológia egy *implicit többesadás faként* használható (implicit multicast tree).

A Komondor is egy olyan alkalmazás, ahol fontos a minél gyorsabban történő üzenetszórás. Általában egyszerűen megoldható, hogy csomag újraküldés segítségével megbízható kommunikációt építsünk egy megbízhatatlan közlésrétegre. A csomagvesztés érzékeléséhez azonban időre van szükség. Méréseink szerint csomagvesztés nélkül ez az algoritmus néhány másodpercen belül képes biztosítani az üzenetszórást; egy csomagvesztés érzékeléséhez önmagában is szükség van ennyi időre. Ha nem próbáljuk meg újraküldeni a csomagokat, akkor a szimuláció segítségével megkaphatjuk azt a legrövidebb időt, amennyi alatt az algoritmus képes elvégezni az üzenetszórást. A replikáció támogatásával ez sokkal rövidebb lehet, mint a csomagvesztés észleléséhez szükséges idő. Az újraküldés nélküli szimulációval megkapjuk azt az arányt is, ahány százalékában az eseteknek képes garantálni a legrövidebb idő betartását.

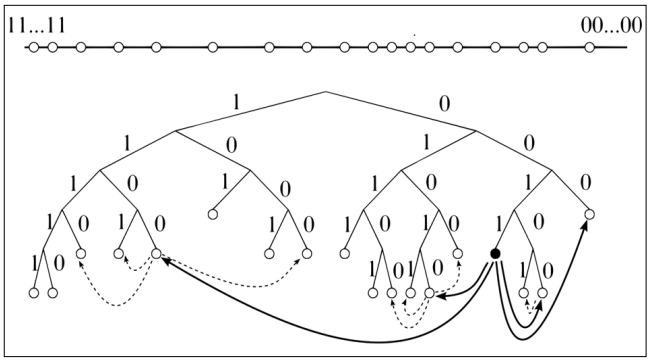
A szórt üzenet küldésére a Kademlia átfedőben háromféle megoldást dolgoztunk ki.

4.3.1. Szórt üzenetek elárasztással

Az első, legegyszerűbb megoldás esetén minden egyed az összes általa ismert egyednek továbbítja az üzenetet. Mivel ilyenkor egy adott üzenetet egy-egy egyed többször is megkaphat, az üzenetek azonosítókkal kell ellátni. Az ismert adatcsomagokat az egyedek eldobják, nem továbbítják és nem is dolgozzák fel többször. Ez a megoldás egyszerű, de igen nagy forgalmat generál különösen, ha a *k*-vödrök nagyok. Gyakorlati haszna nincsen, leginkább egy referenciaként használható; egy ilyen üzenetszórást szimulálva egy adott Kademlia átfedőn belül ugyanis megkaphatjuk, hogy mekkora az üzenetszóráshoz szükséges legkisebb idő. Ha az üzenet az összes lehetséges úton közlekedik, akkor a legrövidebb utat is bejárja.

4.3.2. Szórt üzenetek küldése a topológia kihasználásával

A második megoldásban az egyes részfákhoz felelősöket jelölünk ki, akik az adott részfán belül tovább szórják az üzenetet *(5. ábra)*. Az ábrán a 00110 című, fekete ponttal jelölt egyed indítja a szórt üzenetet azáltal, hogy elküldi minden vödréből egy-egy szabadon választott egyednek, a folytonos nyíl szerint. Ezek az 11000, a 01010, a 00100 és a 00000. Az üzenetet fogadó egyedek a saját részfájukon belül (amelyek rendre az 1****, 01***, 000** és 0010* részfák) felelősek az üzenet tovább szórásáért, mégpedig a szaggatott nyilak szerint. Az üzenet szórása így logaritmikus időn belül lezajlik.

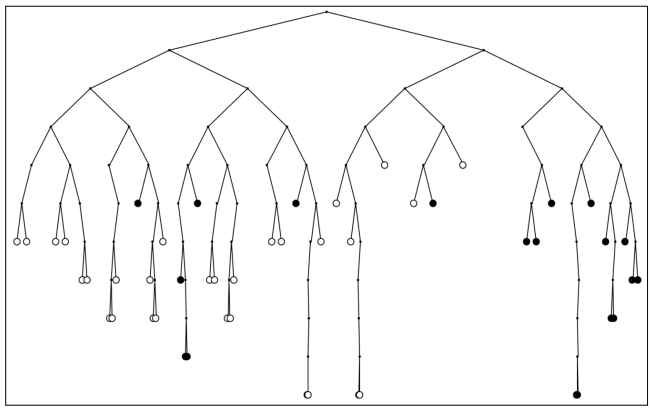


5. ábra Az üzenetszórás lépései a Kademlia átfedőben

Az üzenetek továbbküldéséhez az egyedeknek tudniuk kell, hogy ők melyik részfáért felelősek. Ezért minden broadcast üzenet mellé szükséges még egy kis egész számot is csatolni, amely a bitek számát jelöli; hogy hány kezdő bitben kell megegyeznie a következő címzetteknek a fogadó egyed címével. Mivel az egyes részfákhoz tartozó egyedeket minden esetben tartalmazzák a *k*-vödrök, az üzenet szórásához kiegészítő útválasztási információra nincsen szükség. Az üzenet továbbítását a megadott és annál kisebb részfákba végzi el minden egyed:

```
broadcast (üzenet szövege, magasság)
  ciklus i=magasságtól a címbitek számáig
      ha az  i. vödör nem üres, akkor
             i. vödörbôl egyed választása
                                véletlenszerûen
             üzenet küldése az egyednek,
                tartalma: üzenet szövege, i+1
      feltétel vége
  ciklus vége
```

Az algoritmus igen takarékos, minden egyed csak egyszer kapja meg az üzenetet. Az üzenetek száma exponenciálisan növekszik, vagyis az üzenetküldés logaritmikus időben lezajlik. Problémák csomagvesztés esetén lépnek fel, ugyanis egy-egy eltűnt csomag esetén nem egyetlen egyed, hanem részfák maradnak ki az üzenetszórásból. Az üzenetek tulajdonképp részfáknak szólnak: az eredeti feladó elküldi a másik fél részfának az üzenetet, illetve felel a saját fél fájáért. Elküldi a saját fáján belül az egyik negyednek, és maga felel a másik negyedért. Elküldi egy nyolcadnak, és maga felel a másik nyolcadért stb. Viszont minden ilyen részfának csak egy felelőse van. A *6. ábra* egy szimuláció eredményét mutatja. Fehér körök jelzik azokat az egyedeket, akik megkapták az üzenetet, a feketék pedig a kimaradókat. Láthatóan az átfedőben találhatók teljesen fekete részfák is.



6. ábra
Az implicit fás üzenetszórás hibái a Kademlia átfedőben

Könnyen előfordulhat az is, hogy egy olyan üzenet veszik el, amelyet egy sok csomópontot (magas részfát) kezelő egyednek szántak. Általánosságban elmondható, hogy az üzenetet meg nem kapó egyedek száma, a csomagvesztés arányától *függetlenül* akár az összes egyed felénél is több lehet szerencsétlen esetben. Bár a hálózat decentralizált, ez az algoritmus nem követi a decentralizálás filozófiáját, ugyanis az egyes üzenetek fontossága eltérő. A fontosságuk itt attól függ, hogy milyen magas részfáért felelős egyednek küldik azokat. A kiinduló egyednél akár a legmagasabb fa is lehet.

4.3.3. Szórt üzenetek küldése a topológia kihasználásával, replikációval

A fenti probléma kivédésére használható a harmadik, javított módszer, amely tulajdonképpen az első kettő ötvözése. Az algoritmus lényege megegyezik a második módszernél bemutatottal; minden egyre kisebb részfából kijelölünk egy-egy egyedet, hogy azon belül végezze el az üzenet további szórását. A különbség az, hogy nem egy, hanem több egyednek is elküldjük az üzenetet, ezzel próbálva meg kivédeni a csomagvesztések hatását. Így hatványozottan csökken annak az esélye, hogy egy adott részfa kimarad az üzenetszórásból. Mivel ebben az esetben is lehetséges többszörös kézbesítés, az üzeneteket nem csak a részfa magasságával, hanem egy kvázi-egyedi azonosítóval is el kell látni. A replikáció kétszerestől a *k*-vödrök méretéig terjedhet. A replikáció nélküli eset az előző algoritmust adja vissza.

4.4. Az üzenetszórási algoritmusok összehasonlítása

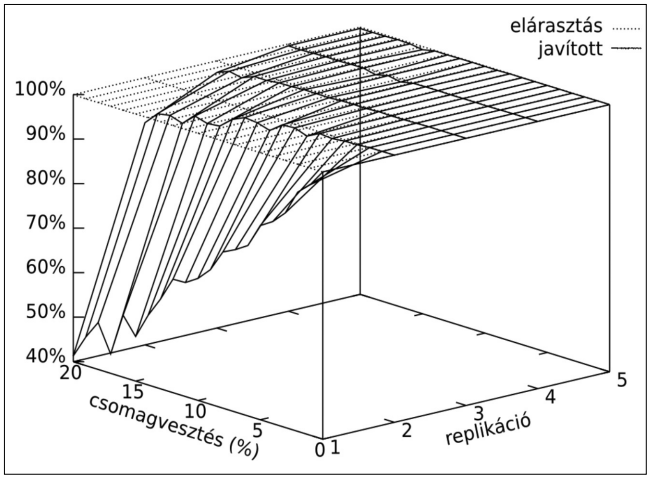
A fent ismertetett algoritmusok tesztelésére és összehasonlítására készítettünk egy szimulátor programot. A program a szimuláció során a következő adatokat jegyzi fel:

– küldött üzenetek száma,
– az üzenetek átlagos száma egyedenként,

– szórt üzenetet megkapó egyedek száma, aránya,
– az időpont,
  amikor az összes egyedhez eljutott a szórt üzenet,
– az üzenet késleltetések
  minimum, átlagos és maximum értéke.

Az üzenetek számát tekintve az elárasztással történő szórás adja a legrosszabb eredményt. Az egyedenkénti üzenetszám az egyedszámmal és a replikáció mértékének növelésével is gyorsan növekszik. Az implicit fás megoldás értelemszerűen konstans egy üzenet/egyedet ad. A harmadik, javított algoritmus üzenetszáma a replikáció növelésével gyorsan nő, az egyedszám növelésével viszont kevésbé változik, 100 egyednél 7, 1000 egyednél is csak 9 körül adódik, ötszörös replikáció esetén.

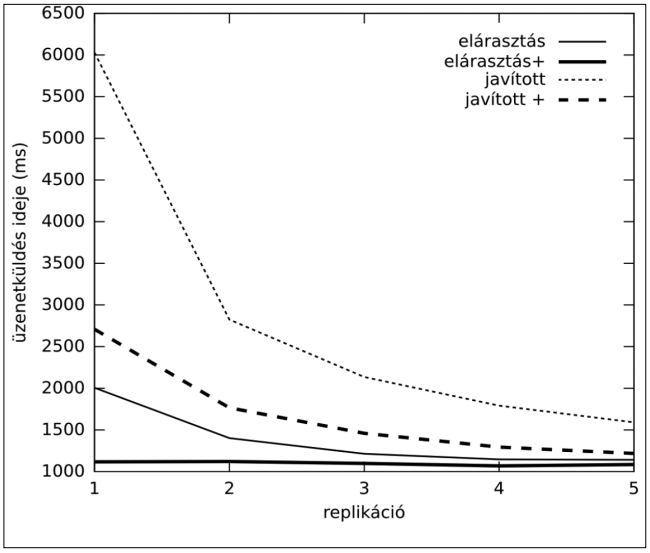
Az üzenetszórások sikerességét egy 200 egyedet számláló átfedő szimulációjával vizsgáltuk. A csomagvesztés 0% és 20% között, a replikáció 1 és 5 között változott. Az elárasztás minden esetben szinte tökéletes eredményre vezetett; az igen kis hibaarány a *7. ábrán* a vonalvastagságba olvad. Ez betudható az elküldött üzenetek az előző mérésben tapasztalt igen nagy számának. A javított algoritmus megbízhatósága természetesen k=1 esetén az implicit fás eredményt adja vissza, ezért az utóbbit nem is ábrázoltuk külön. Viszont k=2 használatával már átlagosan 90% körüli eredményt mutat még a szokatlanul magas, 20%-os csomagvesztés esetén is, k=3-ra pedig 97% adódik.



7. ábra
Az üzenetszórás sikeressége különböző algoritmusok esetén

Az üzenetszóráshoz szükséges időt elsősorban a k-vödrökben tárolt elérhetőségek felé a fizikai hálózat késleltetése határozza meg. Ha az eredeti Kademlia ajánlással szemben nem a régóta ismert egyedeket tartalmazzák a k-vödrök, hanem olyanokat, akik felé a hálózati kapcsolat gyors, a kikeresések és az üzenetszórások ideje is jelentősen lecsökken. A késleltetéseket az egyedek legegyszerűbben PING üzenetekkel mérhetik, de néhány adat ismeretében becsülető is [6]. A program által szimulált esetben 2,5-szeres a gyorsulás; ez az arány nyilvánvalóan függ a mérhető késleltetések eloszlásától.

A leggyorsabbnak természetesen az elárasztás bizonyul *(8. ábra)*, lévén a mindenfelé elküldött üzenetek a legrövidebb útvonalat is bejárják. A replikáció a nem válogatott sebességű kapcsolatok esetén gyorsít az üzenetszóráson, a rendezett esetben természetesen nem. Az implicit fás megoldás merevsége miatt a leglassabb (ezt nem ábrázoltuk, mert megegyezik a javított algoritmus k=1-es esetével). A javított algoritmus az előbbi kettő között teljesít; replikáció esetén gyorsabb lehet, mint a merev implicit fás megoldás válogatott kapcsolatokkal. Ez is annak tudható be, hogy az üzenetek több lehetséges útvonalat bejárva hamarabb eljuthatnak a távoli egyedekhez. Az ábrán „+” jellel jelöltük azt a szimulációt, amelyben az egyedek kiválasztották a gyors hálózati kapcsolatokat.



8. ábra Az üzenetszóráshoz szükséges idő

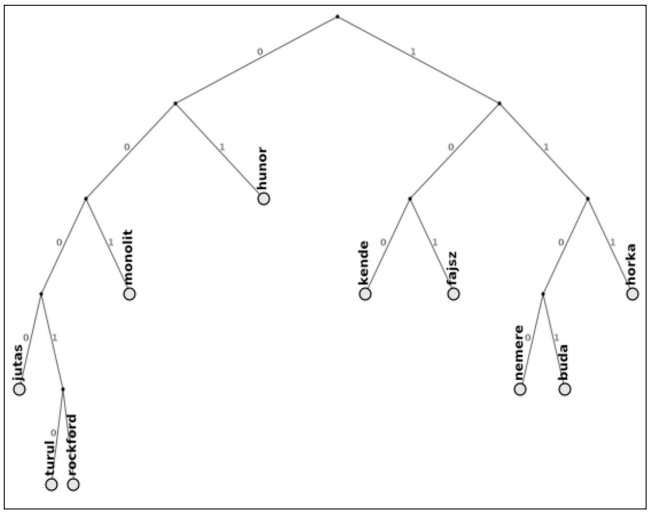
A mérés 100 üzenetszórás átlagolt idejeit mutatja. A leggyorsabb kapcsolat késleltetése a mérésekben 15 ms körüli, az átlagos késleltetés 0,5 s, a legnagyobb pedig 1,3 s körül volt.

5. Összefoglalás

A cikkben bemutatott DHT alapú biztonsági alkalmazás az eddigi tapasztalatok alapján alkalmas az egyes résztvevők védelmének erősítésére. A P2P kommunikációs modellt alkalmazó strukturált átfedővel történő megvalósítás miatt az érzékelés elosztottan történik, mégis kis terhelés többletet jelent az egyedek és a hálózat számára. A működéshez használt két alapvető szolgáltatás, az üzenetküldés és az üzenetszórás megbízhatósága is tetszőleges mértékben növelhető a replikáció segítségével, amelynek mértéke a szerzők által kidolgozott módszerek segítségével előre meghatározható.

A *9. ábra* egy kisebb, működő Komondor hálózatot mutat, a bemutatott bináris fa szerinti elrendezésben. A hónapokon keresztül tartó működés alatt a rendszer több betörési kísérletet is érzékelt, illetve akadályozott meg, miközben az átfedő kellően stabilnak bizonyult. A több

egyed által hasznosítható adatok jelentős része SSH, illetve HTTP alapú támadásokról szóló jelentések voltak. Az érzékeléshez használt Snort program sok olyan eseményt is rögzített, amelyek megosztása nem bizonyult hasznosnak; főként a vírusok aktivitása, azok ugyanis nem célzottan, kitartóan támadnak. Az ezek elleni védekezésre inkább a PROMIS rendszer használható [12].



9. ábra A Komondor működő átfedője

További kutatásaink témája éppen a megosztandó adatok vizsgálata. El kell különíteni az érzékelt támadások közül azokat, amelyekkel érdemes egy elosztott rendszerben is foglalkozni. Különös tekintettel arra az esetre, amikor az egyes Komondor példányok más típusú operációs rendszereket és alkalmazásokat védenek. A heterogenitás növelheti a biztonságot, könnyebb érzékelni a támadást, ha a rendszer ellenáll egy adott típusú támadásnak. Az adatok újrahasznosíthatóságát azonban nehezíti; a védelmet mindig az adott környezethez kell igazítani.

A későbbi kutatások egy másik iránya a rosszakaratú beépülő egyedek elleni védelem kialakítása lehet. Könnyen elképzelhető ugyanis, hogy egy ilyen egyed hamis tapasztalatok megosztásával szolgáltatás megtagadást indít jogosult felhasználók ellen. Ilyen problémára sajnos bármelyik elosztott érzékelő rendszer esetén számítani kell.

A szerzőkről

CZIRKOS ZOLTÁN a Budapesti Műszaki és Gazdaságtudományi Egyetem doktorandusz hallgatója. Fő érdeklődési köre a betörésvédelem és a peer-to-peer kommunikáció. 2005-ben részt vett a Tudományos Diákköri Konferencián a „P2P alapú biztonsági szoftver fejlesztése" című munkájával, amellyel második díjat nyert. Több szakmai cikket jelentetett meg és társszerzőként könyvfejezetek írásában is részt vett az elosztott betörésvédelem témakörében.

HOSSZÚ GÁBOR a műszaki tudomány kandidátusa, docens a Budapesti Műszaki és Gazdaságtudományi Egyetem Elektronikus Eszközök Tanszékén. Szakterületei az internetes médiakommunikáció, a többesadás, az alkalmazási szintű hálózatok, a hálózat-alapú betörésvédelem, valamint a karakterkódolás kérdései. 2001-ben jelent meg az „Internetes médiakommunikáció", 2005-ben pedig „Az internetes kommunikáció informatikai alapjai" című könyve. Kutatási eredményeit több mint száz publikációban jelentette meg.

Irodalom

[1] Czirkos Z.:
P2P alapú biztonsági szoftver fejlesztése,
TDK dolgozat, BME Tudományos Diákköri Konf.
(II. helyezés), Budapest, 2005. november 11.

[2] Gnutella honlap:
http://www.gnutella.org/
(Letöltés ideje: 2008. május 26.)

[3] Snort – the de facto standard for intrusion detection/prevention,
http://www.snort.org/
(Letöltés ideje: 2008. május 26.)

[4] OSSEC –
Open Source Host-based Intrusion Detection System,
http://www.ossec.net/
(Letöltés ideje: 2008. május 26.)

[5] P. Maymounkov, D. Mazieres:
Kademlia: A Peer-to-peer Information System Based on the XOR Metric.
In Proc. of IPTPS02, Cambridge, USA, March 2002.
http://www.cs.rice.edu/Conferences/IPTPS02/

[6] F. Dabek, R. Cox, F. Kaashoek, R. Morris:
Vivaldi: A Decentralized Network Coordinate System.
In Proc. of the ACM SIGCOMM'04 Conference,
Portland, OR, August 2004.

[7] Hosszú G.:
Az internetes kommunikáció informatikai alapjai,
Novella Kiadó, Budapest, 2005.

[8] D. Karger, E. Lehman, F. T. Leighton, M. Levine,
D. Lewin, R. Panigrahy:
Consistent hashing and random trees:
Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web.
In Proc. of the 29th Annual ACM Symposium on Theory of Computing, pp.654–663, May 1997.

[9] I. Stoica, R. Morris, D. Karger,
M. F. Kaashoek, H. Balakrishnan:
Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications.
Technical Report TR-819, MIT, March 2001.

[10] S. Rhea, D. Geels, T. Roscoe, J. Kubiatowicz:
Handling Churn in a DHT.
In Proc. of USENIX Technical Conf., June 2004.

[11] Hosszú G., Czirkos Z.:
„Network-Based Intrusion Detection" chapter in book,
Encycl. of Internet Technologies and Applications.
Editors: Mário Freire and Manuela Pereira,
Information Science Reference, Hershey, USA, 2007,
ISBN: 978-1-59140-993-9, pp.353–359.

[12] Vasileios Vlachos, Diomidis Spinellis:
A Proactive Malware Identification System based on the Computer Hygiene Principles.
Information Management and Computer Security,
15(4):295–312, 2007.

[13] Feng Zhou, Li Zhuang, Ben Y. Zhao, Ling Huang,
Anthony D. Joseph, John Kubiatowicz:
Approximate Object Location and Spam Filtering on Peer-to-peer Systems, In ACM Middleware 2003.