Átjáró szerver választása a GMPLS PCE-architektúrában

ZAHEMSZKY ANDRÁS,
BME Távközlési és Telematikai Tanszék, za488@hszk.bme.hu
CSÁSZÁR ANDRÁS, TÓTH GÁBOR, TAKÁCS ATTILA
Ericsson Magyarország Kft.
{andras.csaszar, gabor.toth, attila.takacs}@ericsson.com

Lektorált

Kulcsszavak: GMPLS, átjárószerver, útvonalszámító eszköz

Ebben a cikkben bemutatjuk, hogy a GMPLS új útvonalválasztó architektúrája, az Útvonal Számító Eszközre (Path Computation Element – PCE) épülő modell hogyan alkalmazható átjáró szerverek választására. A választás célja egyrészt az, hogy elkerüljük a szerverek túlterheléséből adódó hívás blokkolást, másrészt az, hogy minimalizáljuk a transzport hálózat összterhelését, és ezekkel együttesen elérjük, hogy a felhasználók által érzékelt szolgáltatási minőség ne romoljon. Olyan algoritmusokat mutatunk be, amelyek csak egyik vagy mindkét szempontot (a szerverek terheltsége illetve a hálózati topológia) figyelembe veszik, amikor a felhasználó kérésére átjárót választanak. Hálózatszimuláció segítségével összehasonlítjuk a különböző algoritmusokat és megmutatjuk, hogy alacsony blokkolási arány, illetve megfelelően alacsony hálózati terhelés érhető el azokkal, amelyek mindkét szempontot figyelembe veszik a döntés során.

1. Bevezetés

A mai számítógép-hálózatokban gyakran osztunk szét valamilyen képességet több entitás között. Ilyenkor minden entitás ugyanolyan képességekkel rendelkezik, a felhasználó számára mindegyik pontosan ugyanazt a szolgáltatást tudja nyújtani. Az erőforrások ilyenforma szétosztásával a rendszer skálázhatóságát növeljük, mert elkerüljük, hogy az erőforrások a hálózat szűk keresztmetszetévé váljanak. Mivel több szerver is rendelkezésre áll, biztosítani kell egy választási mechanizmust, mely az adott igényhez egy erőforrást rendel a csoportból.

A választási algoritmusnak több célja is lehet. Egyrészt fontos a szerverek terhelését megfelelően szétosztani, annak érdekében, hogy egyik szerver se legyen még ideiglenesen sem túlterhelt. A túlterhelés ugyanis oda vezethet, hogy a szolgáltatás egy bizonyos ideig a felhasználók egy része számára nem lesz elérhető, illetve bizonyos esetekben a felhasználói adatforgalom is megszakad, így a felhasználók által érzett minőség csökken. Másrészt fontos szempont, hogy a választott szerver a felhasználótól rövid úton elérhető legyen. A rövid út a válaszidő, illetve a késleltetés csökkenését jelentheti a felhasználónak, illetve a torlódás kialakulásának valószínűségét is csökkenti, hiszen kevesebb kapcsolat terhelését növeli egy új adatfolyam. Nem mellékes mellékhatás az sem, hogy a hálózat kihasználtsága is javul.

Hálózati terheléselosztással több szituációban is találkozhatunk. Legkézenfekvőbb alkalmazása a Web [1]. Ekkor több Web szerver áll rendelkezésre, amelyek mind ugyanazt a tartalmat tárolják. Előfordulhat olyan eset, amikor egy weblap látogatottsága hirtelen megnő, s ilyenkor a terhelés megfelelő elosztása nagyon fontossá válik. Cél, hogy a szerverek terheltségét egyenletesre állítsuk be. Erre egy úgynevezett hálózat-alapú megoldási lehetőség (azaz amikor nem a felhasználó, hanem a hálózat választ) a DNS-szervereken alapuló megoldás. Ennek lényege, hogy a DNS-szerver az IP-cím feloldásakor mindig más és más szervert választhat. Azonban mivel lokálisan is tárolhatjuk a név-IP-cím összerendeléseket, illetve a közbülső DNS-szerverek is megteszik ezt, nem biztos, hogy a kérés minden esetben eljut a megfelelő helyre. A DNS-szerver tipikusan egy egyszerű algoritmus alapján választ szervert, például *körülfordulásos (Round-Robin) stratégiát* alkalmaz.

Hálózati terheléselosztásról beszélhetünk akkor is, amikor egy ad hoc hálózatban a forgalmazni kívánó csomópontok számára átjárókat rendelünk az Internet felé. Az erre vonatkozó kutatások arra jutottak, hogy nem elég csupán a közelségi információkat figyelembe venni, hanem a szerverek terhelésére vonatkozó információk is szükségesek a jó döntéshez [2].

A terhelés minél egyenletesebb elosztására használható az IPv6-ban megtalálható *anycast* címzés is. Ekkor több szervernek is ugyanaz az anycast címe, ami azt fejezi ki, hogy egy csoportba tartoznak. A címzés lényege, hogy az anycast címre küldött csomagot a hálózat legalább egyik tagja meg fogja kapni, de azt a hálózat dönti el, hogy pontosan melyik. A megoldások java része a kérést a legközelebbi szerverhez irányítja, egy másik megoldás viszont a terhelésekre vonatkozó információkat használja fel az aktív útválasztó technológiát használva, azonban itt a közelségi információk figyelembevétele hiányzik [3].

Cikkünkben bemutatjuk, hogy a *GMPLS (Generalized Multiprotocol Label Switching)* PCE-alapú, új útvonalszámítási modellje hogyan használható hálózati terheléselosztásra és ezzel a szolgáltatás minőségének javítására. Az általunk vizsgált hálózatban a külső hálózatot átjáró szerverek egy csoportján keresztül érhetjük el. A felhasználói adatforgalom megkezdésekor a PCE fog átjárót rendelni a végberendezéshez. Sajnos

a szerverek túlterhelésének kivédése és mindig a legközelebbi szerver választása két, egymásnak erősen ellentmondó szempont. Egyszerű választási algoritmusokat javaslunk, amelyek mindkét szempontot figyelembe veszik a döntéskor, és ezeket összehasonlítjuk azokkal, amelyek csak egy szempont alapján döntenek.

2. A GMPLS és a PCE-architektúra

A GMPLS protokollcsalád egy széles körben használt, általános kontrollmegoldás [4]. Az MPLS általánosításaként azon alapszik, hogy tartalmazza az alagutazás (tunneling) koncepcióját: a felhasználói adatforgalom megkezdése előtt a forrás és a nyelő között egy úgynevezett *címkekapcsolt útvonalat (Label Switched Path – LSP)* hozunk létre. Az MPLS-ben a felhasználók csomagjait egy plusz fejrészben található címkék azonosítják, és a hálózaton belül a csomagok továbbítása a (címke, bemeneti interfész) pár alapján történik. A GMPLS az MPLS-sel szemben abban igazán új, hogy már nemcsak csomagkapcsolt hálózat felett képes működni, hanem más, például hullámhosszkapcsolt hálózatok felett is, és segítségével különböző, egymástól lényegesen eltérő hálózati technikákat lehet egységesen vezérelni. A GMPLS-ben már általánosított címkéről beszélünk, ami egy virtuális címke, amit például egyes időrések, hullámhosszok azonosíthatnak, és természetesen a korábbi MPLS-címke is GMPLS-címke.

A GMPLS protokollcsalád három protokollt tartalmaz. A jelzési protokoll feladata címkekapcsolt útvonalak létrehozása, módosítása, illetve törlése felhasználói igény felmerülése esetén. Elterjedt jelzési protokoll az *RSVP-TE (Resource Reservation Protocol – Traffic Engineering)*. A routing protokoll feladata útvonalválasztási információk terjesztése a GMPLS-hálózatban. A GMPLS gyakran használt routing protokollja az *OSPF-TE (Open Shortest Path First – Traffic Engineering)*. Az OSPF-TE opaque *LSA (Link State Advertisement)* [5] üzenete teszi lehetővé, hogy különböző forgalom menedzsmenthez szükséges információkat is elterjeszthessünk a hálózatban (például sávszélesség vagy képesség-információk). Opaque LSA használatával más alkalmazások is elterjeszthetik információikat a hálózatban. Ezeken kívül a GMPLS még tartalmaz egy menedzsment protokollt is (ez tipikusan az *LMP – Link Management Protocol)*, aminek feladata vezérlő csatornák létrehozása szomszédos vezérlő entitások között, hibák észlelése és lokalizálása.

LSP kiépítése előtt a jelzési folyamatot megkezdő csomópont általában a jelzési üzenetbe ágyazza a célcsomópont felé haladó út csomópontjainak listáját, azaz forrás-vezérelt útvonalválasztásról beszélünk. Gyakran előfordul azonban olyan eset, amikor a teljes útvonal nem áll helyben rendelkezésre, tipikusan olyan esetben, amikor a cél csomópont a hálózat egy másik tartományában helyezkedik el, hiszen a csomópont csak a saját tartományáról rendelkezik teljes képpel. Ilyenkor egy lehetőség, hogy a forrás csomópont egy távoli entitáshoz fordul, amely képes – szükség szerint további entitásokkal együttműködve – útvonalat számolni a forrás és a célcsomópont között. A távoli entitás neve PCE (Path Computation Element). Egy tartományú esetben is van értelme PCE használatának: a csomópont feltételezheti, hogy a PCE további szempontokat is figyelembe tud venni a döntéskor, és így valamilyen értelemben jobb utat tud számolni.

A PCE definíciója nagyon általános: a PCE egy olyan entitás, amely képes egy, a hálózatot reprezentáló gráfon útvonalak kiszámítására, és eközben képes a megfelelő kényszerek figyelembevételére is [6]. Azt a csomópontot, amely útvonalszámítási kéréssel fordul a PCE-hez, *PCC-nek (Path Computation Client)* nevezzük. A PCC, illetve a PCE közötti kommunikáció egy egyszerű, kliens-szerver protokoll segítségével történik: ideális esetben a PCC útvonalszámítási kérelmét tartalmazó üzenetére, mely a célcsomópont címét és esetleges kényszereket tartalmaz, a PCE olyan üzenettel válaszol, mely tartalmazza a részleges, vagy teljes útvonallistát a célcsomópont felé.

A PCE-k számát, elhelyezkedését illetve együttműködését illetően a PCE-architektúrának több modellje létezik. Elhelyezkedés szempontjából a PCE lehet az útválasztó egy komponense, ilyenkor a PCC, illetve a PCE ugyanabban a hálózati csomópontban helyezkedik el. Másrészt a PCE lehet egy távoli hálózati komponens. Ez a megkülönböztetés nem éles, hiszen előfordulhat, hogy egy csomópont egy másik csomópont számára PCE-ként funkcionál, azonban bizonyos más helyzetekben kliensként fog működni.

Nem szükséges azonban, hogy egy útvonalat egyetlen PCE számoljon ki. Ilyenkor több PCE-általi útvonalszámításról beszélünk. Itt is két alesetet különböztetünk meg aszerint, hogy van-e PCE-PCE kommunikáció, vagy nincs.

A modellek elosztottság szempontjából is csoportosíthatók. A centralizált modellben egy tartományon belül az útvonalszámítási felelősség egy csomóponttá, míg az elosztott modellben egy tartomány több, egyszerre működő PCE-t is tartalmaz.

Vizsgálatainkban a centralizált modellre szorítkozunk, és a PCE minden csomópont számára távoli hálózati komponens. Az általunk használt modell az *1. ábrán* látható.

3. Átjáró-választási algoritmusok

Ez a fejezet a PCE által használt egyszerű átjáró-választási algoritmusokat mutatja be. Míg az átjáró-választási stratégia minden esetben más és más, az útvonalválasztás mechanizmusa mindig ugyanaz: a PCE a PCC és az átjáró közötti legkisebb lépésszámú utat fogja választani. A bemutatott algoritmusok két csoportba oszthatók. Az egyik csoportba azok az algoritmusok tartoznak, amelyek egy fázisban választanak átjárót, míg a másikba azok, amelyek két körben döntenek – azaz egymás után két szempontot is igyekeznek figyelembe venni.

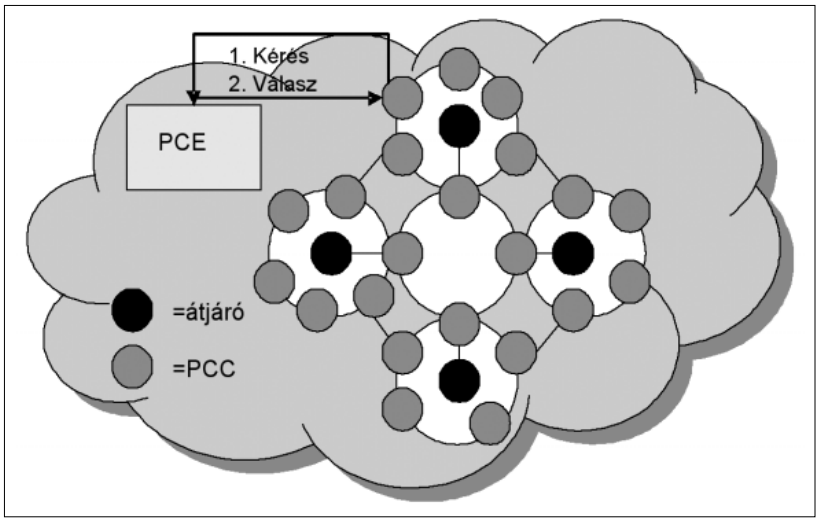
Az egyik legegyszerűbb algoritmus a UNIFORM. Ekkor az átjárók között egyenletes eloszlással véletlen választunk, és az egymást követő választások függetlenek egymástól.

A ROUND-ROBIN algoritmus egyszerű körbefordulásos stratégia alapján rendel átjárót a felhasználóhoz.

A következő két algoritmus már figyelembe vesz az átjárók elhelyezkedésére, illetve terhelésére vonatkozó információkat. Így a NEAREST algoritmus mindig a legközelebbi átjárót fogja a PCC-hez rendelni. Ez lehetséges, hiszen a tartomány teljes topológiáját a topológiaváltozások utáni rövid tranziensek kivételével pontosan ismeri. Amennyiben több átjáró is pontosan ugyananyi lépésre lenne a PCC-től, akkor a döntő logika egy előre definiált, önkényes sorrendet figyelembe véve oldja fel a holtversenyt. Így vegyük észre, hogy ezen algoritmus, amennyiben nem történik topológia változás, az egyes PCC-k számára minden esetben ugyanazt az átjárót választja.

A GREEDY algoritmus olyan értelemben mohó, hogy mindig a legkisebb terhelésű átjáróhoz irányítja a felhasználót. Azonban itt, nem úgy, mint a topológia-információk figyelembevételekor, már nem garantált, hogy a PCE pontos képpel rendelkezik a hálózatról. A PCE által a szerverek terheltségéről látott képet két dolog befolyásolja döntően: egyrészt az átjárók által opaque LSA-kba ágyazott és rendszeresen küldött terheltségi információk, másrészt az a tény, hogy a PCE a saját döntéseit képes nyomon követni, így az átjárók állapotában történő változásokat bizonyos mértékben figyelembe tudja venni, azonban a felhasználói forgalom befejezéséről nem értesül. Emiatt fontos tényező, hogy az átjárók milyen gyakran frissítik a terhelési információkat. Amennyiben az átjárók gyakran küldenek üzenetet állapotukról, akkor a PCE-k helyes képpel fognak rendelkezni a hálózatról, azonban ennek ára sok vezérlő üzenet, ami a skálázhatóság rovására megy. Ezzel ellentétben, hosszú frissítési periódus esetén a PCE-k által látott kép elavulttá válhat, ekkor viszont sokkal kevesebb vezérlő üzenetre van szükség.

1. ábra A PCE architektúra és egy példa



A következő két algoritmus két szempontot is figyelembe vesz. A LOADNEAR(d) az első körben a *d* darab legkevésbé terhelt átjárót engedi át a rostán, majd ezek közül a PCC-hez legközelebbit választja. Észrevehető, hogy az algoritmus d=1 paraméter esetén éppen úgy működik, mint a GREEDY, míg ha az első körben egyetlen átjárót sem szűrünk ki (d=∞), a NEAREST algoritmushoz jutunk.

A NEARLOAD(d) ettől csak a szempontok figyelembevételének sorrendjében különbözik: először a topológia-információk alapján engedünk át *d* szervert a szűrőn, majd ezek közül a legkisebb terhelésűt választjuk. Természetesen a PCE által látott kép helyességére való fenti megfontolás ennél a két algoritmusnál is érvényes.

4. Szimulációs eredmények ismertetése

A szimulációkhoz az ns-2 hálózati szimulátor alkalmazást [7] bővítettük ki a szükséges funkciókkal.

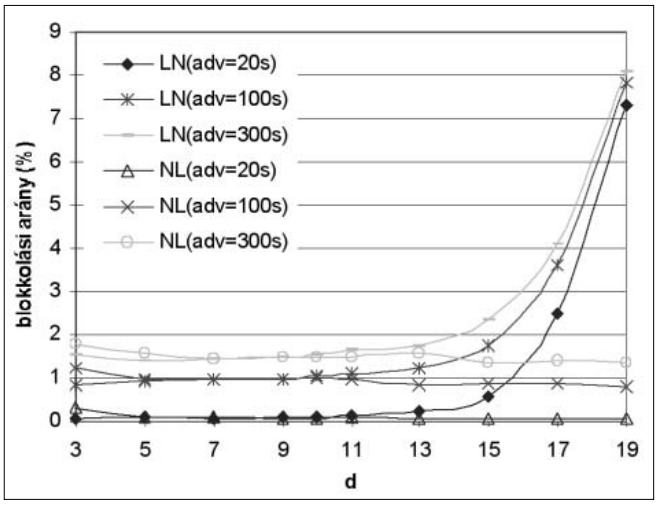
A vizsgálatokat többszintű, összekapcsolt gyűrűkből álló topológián végeztük. Az első ábrán az egyszerűség kedvéért csak egy kétszintű hálózatot tüntettünk fel, azonban a munka során háromszintű hálózattal dolgoztunk, amely összesen 84 csomópontot tartalmazott. A csomópontok egyben PCC-k is, melyekhez a felhasználók kapcsolódnak. A hálózatban egy központi PCE-t helyeztünk el, átjáró szerverek pedig az első és a második hierarchiaszint gyűrűihez csatlakoznak, azaz számuk összesen 20.

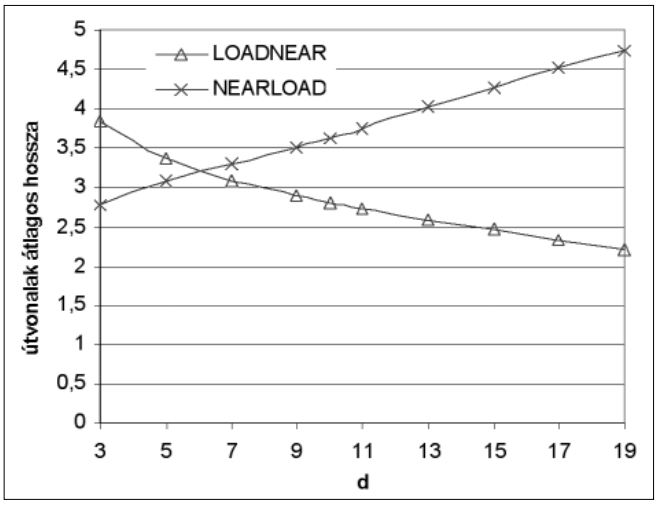
A felhasználói igények érkezési folyamata minden PCC esetében Poisson-folyamat, azonban a köztes idők eloszlása minden PCC-re más és más, választása egy meghatározott intervallumból történik egyenletes eloszlás alapján. Így lesznek olyan csomópontok, amelyek gyakran fordulnak útvonal-választási kérelemmel a PCE-hez, illetve olyanok is, amelyek ritkábban. A hívástartási idő minden felhasználó esetén exponenciális eloszlású, 120 másodperces várható értékkel.

Az alkalmazott modellben az átjárók kapacitása véges. Ez azt jelenti, hogy az általuk egyszerre kiszolgálható felhasználók száma korlátozott. Az egyes átjáró szerverek kapacitását úgy állítottuk be, hogy ideális elosztás esetén az átlagosan felajánlott forgalom 120%-át képesek legyenek elvezetni.

A szimulációk során a blokkolási arányt és az utak átlagos hosszát mértük. Blokkolás akkor léphet fel a rendszerben, ha a PCE olyan átjárót rendel a felhasználóhoz, ami éppen tele van. Az utak átlagos hosszával pedig a transzporthálózat terheltségére, a késleltetésekre és a torlódás valószínűségére következtethetünk.

Először megvizsgáltuk az általunk javasolt két új algoritmus (LOADNEAR, illetve NEARLOAD) viselkedését a *d* para-





2. és 3. ábra Blokkolási arány és az útvonalak átlagos hossza a d *paraméter függvényében*

méter, azaz az első fázisban kiválasztott átjárók számának függvényében. Az eredmények a *2. és a 3. ábrán* láthatók.

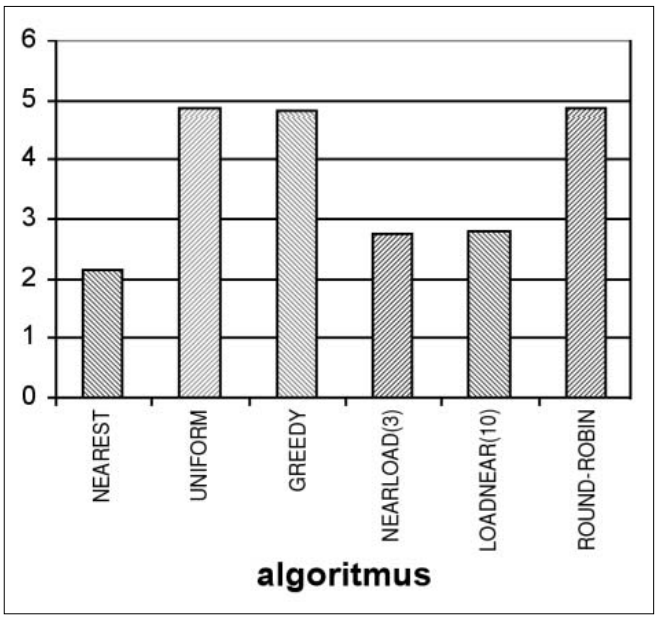
A periodikus frissítés gyakorisága az utak átlagos hosszára nem volt hatással, ezért ezt nem is ábrázoltuk. A *d* paraméter növelésével a LOADNEAR esetében csökkent, míg a NEARLOAD esetében nőtt a választott útvonalak hossza. Az ábrákról leolvasható, hogy a NEARLOAD algoritmus már *d*=3 esetén is alacsonyan tartja a blokkolást. A LOADNEAR algoritmus ezzel szemben, amíg a *d* paraméter értéke nem éri el az átjárók számának felét, jól teljesít, később viszont a blokkolási arány egyre rohamosabb ütemben növekszik.

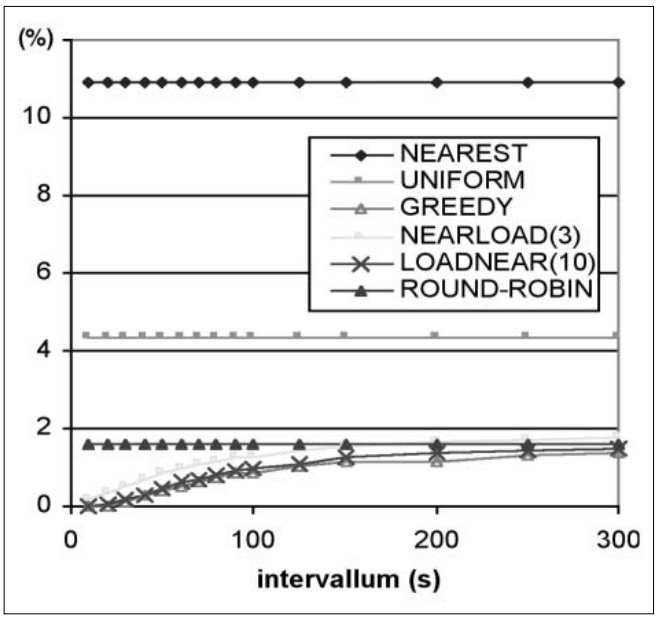
Ezen meggondolásokat megfontolva arra juthatunk, hogy ezen a topológián a NEARLOAD *d*=3, míg a LOADNEAR *d*=10 esetén produkálja a legjobb eredményeket, mindkét szempontot figyelembe véve. A további ábrákon ezért már csak ezen paraméterértékek melletti eredményeket tüntettük fel.

A *4. ábrán* láthatóak a különböző algoritmusok által produkált blokkolási arányok az információterjesztés gyakoriságának függvényében. Látható, hogy a legrosszabb eredményeket a NEAREST algoritmus produkálta, azonban az *5. ábra* mutatja, hogy sikeresen minimalizálta az utak hosszát. Véletlen algoritmus használatával (UNIFORM) a blokkolási arány csökkenthető, azonban ennek ára az utak megnövekedett hossza. A ROUND-ROBIN algoritmussal az utak hosszát tekintve javulást nem tudunk elérni, azonban a visszautasított hálózati kapcsolatok száma csökken a UNIFORM-hoz képest. Ennek oka abban keresendő, hogy ez az algoritmus mindig új átjárót választ, míg a UNIFORM esetében lehetséges egy átjáró többszöri egymás utáni választása is.

Az intelligensebb algoritmusok (GREEDY, NEARLOAD, LOADNEAR) teljesítményére kihat az információfrissítés gyakorisága. Az eredményekből azt tapasztaltuk, hogy az információküldés ritkításával egyre rosszabb

4. és 5. ábra Az útvonalak átlagos hossza és a blokkolási arány különböző információterjesztési intervallumok esetén





eredményt érnek el, illetve mindhárom teljesítménye a ROUND-ROBIN algoritmus eredményeihez konvergál. Ez azért van így, mert a PCE figyelembe veszi saját korábbi döntéseit, azonban a felhasználók távozásáról nem értesül. Így, ha két frissítés között sok idő telik el, először kiegyenlíti a szerverek terheltségét az általa látott hálózati képben, majd gyakorlatilag körülfordulásos módon választ átjárót a következő frissítés vételéig.

Azt kaptuk, hogy a LOADNEAR, illetve a NEARLOAD megfelelő paraméterértékek melletti alkalmazásával – a blokkolási ráta kis növekedése árán – jelentősen csökkenteni tudjuk az átjárókhoz vezető útvonalak átlagos hosszát. A legjobb blokkolási ráta a GREEDY alkalmazásával érhető el, de mivel ez nem veszi figyelembe a topológiát a döntéskor, hosszú útvonalakat fog létesíteni a hálózatban.

5. Összefoglalás

Ha egy bizonyos igénytípust hálózati szerverek egy csoportja képes kiszolgálni, akkor egy konkrét igény kiszolgálását elvégző szerver kiválasztása egy megoldandó feladat. Mivel a hálózati terhelés alacsonyan tartása, illetve a szerverek terheltségének megfelelő elosztása két, egymásnak ellentmondó szempont, így lehetetlen olyan algoritmust találni, amely mindkét szempont szerint optimális eredményt ad. Szerencsére egyszerű algoritmusok használatával, melyek mindkét szempontot figyelembe veszik, jó kompromisszumot lehet elérni. Összességében a paraméterek megfelelő választása mellett a NEARLOAD, illetve a LOADNEAR megfelelő eredményt nyújt.

Irodalom

[1] Cardellini, V., Colajanni, M., Yu, P.S., „Dynamic load balancing on Web-server systems”, Internet Computing, IEEE, Vol. 3, No.3, pp.28–39., May/June 1999.

[2] J. Shin, H. Lee, J. Na, A. Park, „Load Balancing among Internet Gateways in Ad Hoc Networks”, in Proc. of Vehicular Technology Conference, 2005. Vol. 3, pp.1677–1680.

[3] Miura, H., Yamamoto, M., Nishimura, K., Ikeda, H., „Server Load Balancing with Network Support: Active Anycast”. in H. Yasuda (ed.), Proc. of 2nd International Working Conference on Active Networks (IWAN 2000), Springer LNCS 1942, p.371., Tokyo, October 2000.

[4] Mannie E. (ed.), „Generalized Multi-Protocol Label Switching (GMPLS) Architecture”, RFC3945, October 2004.

[5] Coltun, R., „The OSPF Opaque LSA Option”, RFC2370, July 1998.

[6] Farrel, A., Vasseur J.-P., Ash J., „A Path Computation Element (PCE)-Based Architecture”, RFC4655, August 2006.

[7] Ns-2 honlap, http://www.isi.edu/nsnam/ns

Helyreigazítás

Legutóbbi, 2007/3-as számunk 19-23. oldalán, **Csurgai-Horváth László és Bitó János: „Fading időtartam-modellezés műholdas földi mozgó rádiócsatornán”** című cikkébe sajnálatos módon több, értelemzavaró hiba került, melyek a következők:

- Az *1. táblázat* helyére egy rokon témájú cikk hasonló tartalmú táblázata került. Az ide tartozó táblázat helyesen:

Műhold neve	MARECS (d=39150 km)
Eleváció	24°
Frekvencia	1.54 GHz
Mintavételezési frekvencia	300.5 Hz
Csatorna ID	14
Környezet	Autópálya
Mérési időtartam	81.2 perc
Jármű sebessége	60 km/h

- A *(3) és (4) képletekben* az eredeti, görög szumma helyett hibás karakter jelent meg.

- A *2. táblázat* felirata pedig helyesen: „Paraméterek a (7-8) egyenletekhez”

A szerzőktől és olvasóinktól egyaránt elnézést kérünk!