Digitális szökőkút alapú transzport protokoll fejlesztése és elemzése

Solymos Szilárd
Távközlési és Médiainformatikai Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem
Budapest, Magyarország
solymos@tmit.bme.hu

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

***Kivonat*—Az Internet folyamatos fejlődése során megjelent számos különböző TCP (Transmission Control Protocol) verzió nem képes optimális megoldást nyújtani a jelenlegi folyamatosan változó, heterogén hálózati környezetek esetén. Erre a kérdésre egy olyan új koncepció adhat választ, amely a torlódásszabályozás helyett hatékony hibajavító kódolás alkalmazását javasolja. Az utóbbi években egy ilyen elven működő, DFCP-nek (Digital Fountain based Communication Protocol) elnevezett, szállítási rétegbeli protokollt fejlesztünk a tanszéken, amely Raptor kódolás alkalmazásával képes hatékony működést elérni. Ebben a cikkben egy olyan elemzést mutatunk be, amely kitér a protokoll működésére, fejlesztésére, és annak teljesítményét elterjedten használt TCP verziókkal hasonlítja össze szimulációk és teszthálózati mérések segítségével.**

Kulcsszavak: jövő Internetje, hibajavító kódolás, szökőkút kódolás, transzport protokoll

I. BEVEZETÉS

Jelenleg a TCP (Transmission Control Protocol) az egyik legszéleskörűbben használt szállítási rétegbeli protokoll, amely kapcsolat-orientált, és megbízható adatátvitelt valósít meg. A TCP torlódásszabályozást alkalmaz annak érdekében, hogy minél hatékonyabb működést érjen el a jelenlegi nagysebességű, nagy késleltetésű, vezeték nélküli és egyéb média esetén. A hatékonyság növelése céljából számos, gyakran igen eltérő teljesítményű, együttműködési problémákat okozó, egyre nagyobb komplexitású TCP változat jelent meg [1], [2], amelyek egy adott környezet esetén ugyan képesek jobb működést elérni, de nem képesek univerzális, optimális megoldást nyújtani napjaink hálózatainak kihívásaira.

A TCP esetén fellépő hatékonysági és együttműködési kérdések megoldása céljából jelenleg számos kutatás folyik és új koncepciók jelentek meg. Az egyik ilyen, a GENI (Global Environment for Network Innovations) által javasolt, elképzelés szerint egyáltalán ne alkalmazzunk torlódásszabályozást [3]. Az ötlet ígéretesnek tűnik, de eddig néhány, ezzel az elvvel kapcsolatos munkán kívül semmilyen realizációt, vagy további finomítást nem publikáltak, így a koncepció részletes elemzése és hozzá kapcsolódó releváns következtetések nem álltak rendelkezésre.

Raghavan és Snoeren tanulmányozta a torlódásszabályozás nélküli hálózat nyújtotta előnyöket, és bemutatott egy *torlódásmentesítőt*, mint egy lehetséges megoldás alapját [4]. Bonald és mások szintén megvizsgálták a torlódásszabályozás nélküli hálózat viselkedését [5]. Az ő eredményük megmutatta, hogy nem igaz az a hiedelem, amely szerint a torlódásszabályozás nélkülözése a hálózat összeomlásához vezetne. López és mások a játékelméleten keresztül vizsgálták meg egy szökőkút kódoláson alapuló protokoll teljesítményét [6]. Megmutatták, hogy elérhető egy olyan Nash-egyensúly, amelynél a hálózat teljesítménye hasonló ahhoz, mintha minden végpont TCP-t alkalmazna. Úgy tűnik, hogy a ráta nélküli kódok jól alkalmazhatóak hibajavító kódolásként. Például [7] bemutatja az élő videó közvetítés egy ráta nélküli kódoláson alapuló forgatókönyvét és tartalmazza a hozzá kapcsolódó kísérleti eredményeket is. Botos és mások egy olyan transzport protokollt mutattak be, amely nagy csomagvesztésű hálózaton alkalmaz ráta nélküli kódolást [8]. Kumar és mások [9] pedig egy vezeték nélküli hálózatokon ígéretes teljesítményt nyújtó transzport protokollt fejlesztettek ki, amely szökőkút kódoláson alapul.

A továbbiakban azt tételezzük fel, hogy egyáltalán nem alkalmazunk torlódásszabályozást. Ebben az esetben a hálózatban minden entitás maximális sebességgel küldheti az adatokat, az átvitel során fellépő, főleg börsztös csomagvesztések hatékony hibajavító kódolás segítségével állíthatóak helyre. Az igazságos működéshez a hálózatban igazságos ütemezést tételezünk fel, mint például a WFQ (Weighted Fair Queuing). Mivel ennek az új koncepciónak nem volt elérhető további, részletesebb leírása, implementációja, egy teljesen új protokollt terveztünk meg és valósítottunk meg, amely a DFCP (Digital Fountain based Communication Protocol) nevet kapta.

A cikk a következőképpen épül fel. A következő, II. részben a DFCP-t megalapozó koncepciót, a III. fejezetben a protokoll fejlesztését és megvalósításának alapjait ismertetjük. A IV. fejezet az elvégzett szimulációs és teszthálózati mérések eredményeit mutatja be és összehasonlítja a protokoll teljesítményét néhány elterjedten alkalmazott TCP variánssal. Végül az V. fejezet összefoglalja a leírtakat.

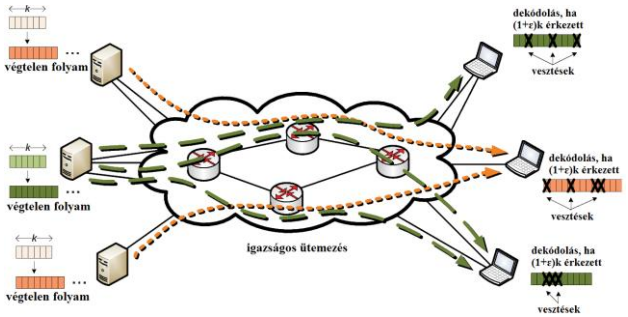
II. SZÖKŐKÚT KÓDOLÁSON ALAPULÓ ARCHITEKTÚRA

Napjainkban az Internet fő szabályozó mechanizmusa a torlódásszabályozás, amely minél hatékonyabban próbálja kihasználni a hálózati erőforrásokat. A különböző hálózati környezetek jelentette problémák megoldására számos

különböző TCP verzió jelent meg, amelyek az adott viszonyok között jobb teljesítményt tudnak ugyan nyújtani, de az is látható, hogy ezek a protokollok nem mindig képesek az igazságos együttműködésre és a különböző TCP változatok nagyon eltérő teljesítményt érnek el.

A jövő hálózataira vonatkozóan a GENI adott egy olyan javaslatot, amely szerint a torlódásszabályozás helyett alkalmazzunk hibajavító kódolást. Ekkor a hálózat minden végpontja maximális sebességgel küldhet adatokat, tehát amíg csak van rendelkezésre álló adat egy adott végpont esetén, addig olyan gyorsan történhet a küldés, amennyire csak lehetséges. Természetesen ha minden végpont maximális sebességgel küld adatokat, akkor nagymértékű, főleg csomós jellegű csomagvesztés keletkezhet az erőforrások túlterhelése miatt. A GENI javaslata alapján ezt a csomagvesztést hatékony hibajavító kódolás alkalmazásával ellensúlyozhatjuk. Az eddig ismertetett megoldásnak számos előnye van. Az egyik a hatékonysága, ugyanis ez a módszer minden hálózati erőforrást mindig teljesen kihasznál és azonnal felhasználja a rendelkezésre álló új kapacitásokat is a hálózatban. Újabb előnye az egyszerűsége, valamint az, hogy a csomagvesztések hatékony hibajavító kódolással történő helyreállítása következtében támogatja a kis méretű bufferek alkalmazását a routerek esetén. Végül fontos a módszer stabilitása is, mivel a maximális sebességű adatküldés alkalmazása sokkal előrejelezhetőbb forgalmat jelent a TCP-hez képest, amely esetén a küldési sebesség nagy mértékben ingadozhat, ami ezt megnehezíti. Az itt leírt előnyök különösen kedvezőek az optikai hálózatokra nézve, ahol csak kisebb méretű bufferek alkalmazására van jelenleg lehetőség [10].

Az 1. ábra egy szökőkút kódoláson alapuló hálózati architektúrát ábrázol. A küldő folyamatok az adatot Raptor kódolással kódolják [11], majd maximális sebességgel küldik. Ezt a sebességet két dolog korlátozhatja, az egyik maga a küldő alkalmazás, a másik pedig a link kapacitása. A Raptor kódolás lehetővé teszi azt, hogy a beérkező kódolt folyam bármely $\lceil (1+\varepsilon)k \rceil$ méretű részéből nagy valószínűséggel visszaállíthassuk az eredeti k szimbólumot, bármely kis $\varepsilon > 0$ valós szám esetén. Ez a tulajdonság rendkívül hibatűrő adatátvitelt tesz lehetővé még akkor is, ha a csomagvesztés mértéke dinamikusan változik, vagy nagymértékben csomósodik. Ha ekkor a dekódolás sikertelen lenne, akkor a vevő megpróbálhat újabb kódolt szimbólumokat gyűjteni, és újra próbálkozhat a dekódolással.

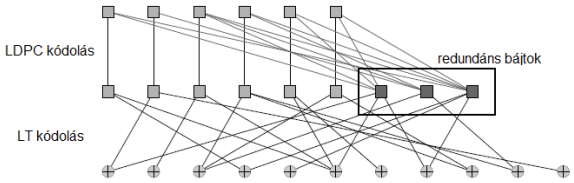


1. ábra Szökőkút kódoláson alapuló architektúra

A maximális sebességű adatküldés felveti az alkalmazott mechanizmus igazságosságának kérdését. A versengő folyamok hozzáférése különböző sebességű lehet. Ez egy szűk hálózati keresztmetszet esetén azt jelentheti, hogy a mohóbb folyamok kiéheztetik a kevésbé agresszív folyamokat. Ahhoz, hogy megoldjuk a maximális sebességgel küldő folyamok esetén ezt a problémát, igazságos ütemezést tételezünk fel a routerek esetén, mint például a WFQ, vagy a DRR (Deficit Round Robin) ütemező [12].

III. DIGITÁLIS SZÖKŐKÚT ALAPÚ TRANSZPORT PROTOKOLL

A Digitális szökőkút alapú transzport protokoll (Digital Fountain based Communication Protocol, DFCP) egy kapcsolat-orientált, megbízható működést biztosító transzport protokoll [13]. A TCP-vel szemben ez a protokoll egyáltalán nem alkalmaz torlódásszabályozást, hanem Raptor kódoláson alapuló hibajavító kódolás segítségével képes hatékony működést elérni. A DFCP implementációja a 2.6.26-2 verziójú Debian Linux kernelben történt C nyelven. A protokoll működése három fázisra bontható, amelyek a *kapcsolat-felépítés*, az *adatküldés* és a *kapcsolatbontás*. Ha a kapcsolat létrejött, akkor megkezdődhet az *adatátvitel*. Ebben a fázisban a küldő tetszőleges mennyiségű adatot küldhet, amelyet a kernel előre meghatározott, fix méretű blokkokra bont, majd a blokkokat egyenként Raptor kódolás alkalmazásával kódolja a 2. ábrán látható módon.



2. ábra A DFCP által alkalmazott Raptor kódolás

Az alkalmazott kódolás egy *előkódolásból* és egy *LT (Luby Transform) kódolásból* tevődik össze [11]. Az előkódolás gyakorlati realizációja LDPC (Low-Density Parity-Check) kódolással történik [14]. Az előkódolás során a blokkot alkotó bájtokhoz redundáns bájtokat rendelünk hozzá, amelyeket az eredeti blokk bájtjaiból származtatunk egy adott valószínűség-eloszlás szerint, XOR művelettel. Az előkódolás kimenetét LT kódolás segítségével elméletileg végtelen hosszú, a gyakorlatban korlátos, egy paraméter által szabályozható hosszúságú, kódolt bájtokból álló folyammá transzformáljuk. A kódolt bájtokat csomagokba, majd keretekbe szervezve a vevő felé továbbítjuk. A vevő a kapcsolat-felépítés során szinkronba kerül a küldővel, így nagy valószínűség szerint képes a kódolt bájtok dekódolására.

Minden csomag tartalmaz egy olyan *sorszámot*, amely meghatározza, hogy az adott csomag mely blokkhoz tartozik. Először is, ha a csomagok nem megfelelő sorrendben érkeznek meg a vevőhöz, a sorszám alapján a blokkot megtudhatja. Másodszor, a blokkokon belül nem szükséges a csomagok sorrendjének visszaállítása, mert az alkalmazott Raptor kódolás tulajdonságai biztosítják a protokoll számára, hogy csak a kódolt bájtok mennyisége számítson, így a sorrendjük nem lényeges.

A DFCP által alkalmazott *folyamvezérlés* szerepe, hogy megóvja az esetleg lassabb vevőt az elárasztástól. A DFCP

esetén egy csúszóablakos mechanizmus került megvalósításra, ahol az ablak mérete meghatározza azt, hogy hány blokk kerülhet elküldésre, mielőtt nyugtára kezd várakozni a küldő oldal. A vevő oldal minden blokk sikeres vételéről nyugtát küld, amelyek megérkezése esetén a küldő további blokkokat küldhet.

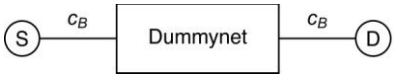
IV. Mérési eredmények

A DFCP teljesítményének elemzése céljából összehasonlító jellegű méréseket végeztünk az eddig elkészült megvalósítással szimulációk és teszthálózati mérések által [15]. Ennek során két elterjedt TCP verzióval hasonlítottuk össze a DFCP-t. Az egyik TCP verzió a TCP NewReno volt, amely a hagyományos TCP hatékonyságát növelő mechanizmusokat tartalmaz, mint például a SACK (Selective Acknowledgment) [16]. A másik a TCP CUBIC volt, amely jelenleg az alapértelmezett TCP variáns a Linux kernelekben. A mérések során hálózati topológiákat hoztunk létre, majd különböző egyenletes eloszlás szerinti csomagvesztés és késleltetés értékeket emuláltunk a hálózatban. Megmértük az említett három protokoll *goodput* értékét, amely az egy másodperc alatt továbbított hasznos bitek mennyiségét jelenti, és ezáltal összehasonlítottuk a protokollok által elért teljesítményt.

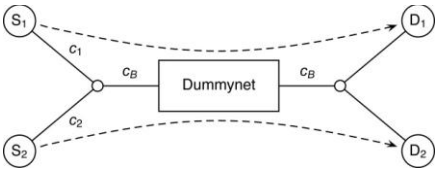
A. Mérési környezet

A teszthálózati mérések során a hálózati paraméterek emulációjához a *Dummynetet* [17], a szimulációk esetén pedig az *ns-2* programot használtuk fel [18]. Az elvégzett mérések között volt olyan, amikor egyszerre csak egyetlen folyam volt a hálózaton, illetve megvizsgáltuk azt is, mi történik versengő folyamok esetén, amikor két, ugyanazt a protokollt alkalmazó folyam van egyszerre jelen a hálózaton.

Az egy folyamos mérésekhez a 3. ábrán látható topológiát, a versengő folyamokhoz a 4. ábrán látható topológiát használtuk. A bottleneck link kapacitása C_B = 1 Gbps, a mérések időtartama 60 másodperc volt.



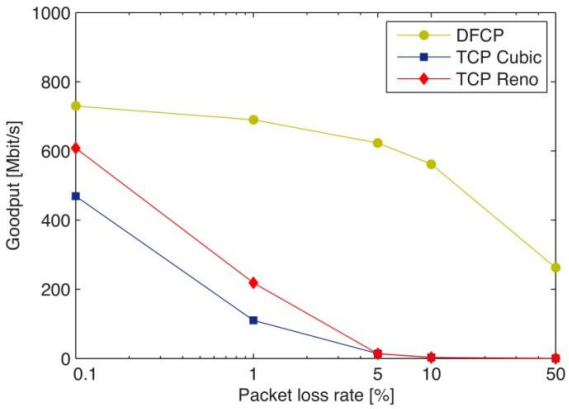
3. ábra Egy folyamos mérésekhez használt topológia



4. ábra Két folyamos mérésekhez használt topológia

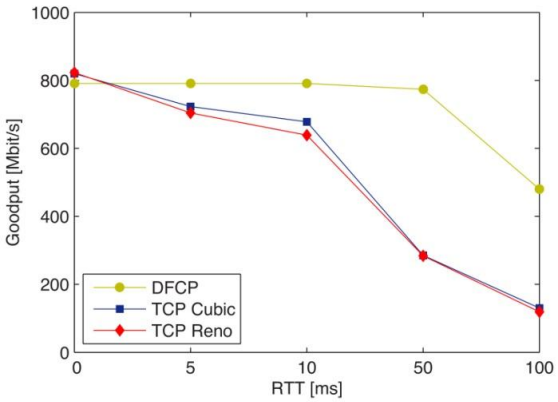
B. Az elvégzett mérések

Először az egy folyammal elvégzett mérések eredményeit mutatjuk be. Az 5. ábrán azt láthatjuk, hogy változó, egyre növekvő csomagvesztés esetén hogyan változik az elért *goodput* érték az egyes vizsgált protokollok esetén. Itt minden csomagvesztés értéknél egy optimális redundancia értéket állítottunk be a DFCP esetén, amely azt jelenti, hogy minden blokkból azt a rögzített, minimális számú csomagot küldte ki a protokoll, amennyi éppen elegendő a sikeres dekódoláshoz. Ezt az optimális redundancia értéket előzetesen, külön-külön minden csomagvesztés értékhez meg kellett keresni, és ezen értékek használatával történtek a mérések. Azt a következtetést lehet levonni, hogy a DFCP esetén nagyobb csomagvesztésnél sem csökken le szignifikáns mértékben a *goodput*, így sokkal kevésbé érzékeny a protokoll a csomagvesztésre, szemben a TCP-vel, amelynél jelentős csökkenés figyelhető meg már kisebb csomagvesztés értékeknél is. Azt is láthatjuk, hogy a DFCP még igen magas csomagvesztésnél is működőképes marad.



5. ábra Az elért *goodput* a csomagvesztés függvényében

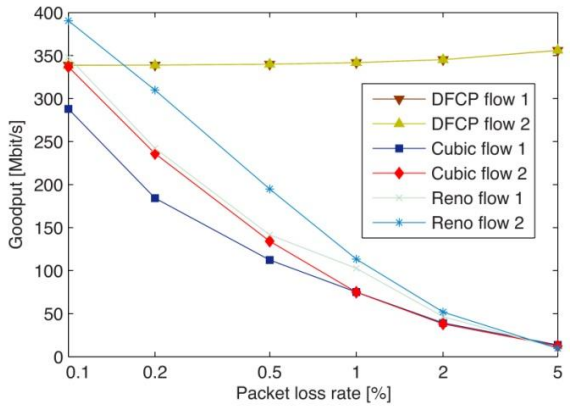
A 6. ábrán a késleltetés volt a vizsgált paraméter. Megfigyelhető, hogy a növekvő késleltetés ellenére a DFCP teljesítménye közel állandó, szemben a TCP-vel, amelynek teljesítménye megint fokozatosan lecsökken, ahogyan nő a késleltetés. Kezdetben a DFCP esetén a kódolási overhead miatt a TCP kicsit jobban teljesít, de utána látható, hogy csak az 50 ms értéknél kezd csökkenni a DFCP teljesítménye, amely annak tudható be, hogy a DFCP ablakmérete ebben az esetben korlátozta az adatküldés sebességét.



6. ábra Az elért *goodput* a késleltetés függvényében

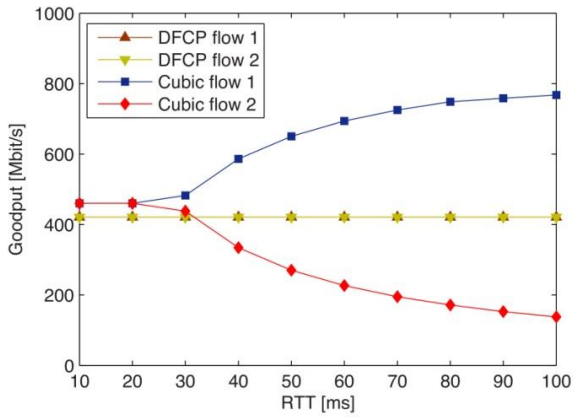
A következő két esetben több folyam volt jelen egyszerre a hálózaton, a mért jellemző itt szintén a *goodput* a csomagvesztés vagy a késleltetés függvényében. A két folyam elindítása egy időben történt, a hálózatban WFQ ütemezőt alkalmaztunk azonos súlyozással. A 7. ábrán egy olyan esetet láthatunk, amikor a korábban említett két TCP verzióval vetjük össze a DFCP teljesítményét. Minden vizsgált protokoll esetén két folyam volt egyszerre jelen a hálózaton. A vizsgált TCP verziók nem voltak képesek igazságosan megosztozni a sávszélességen, amíg a két DFCP folyam teljesítménye végig közel azonos volt, ez amiatt történt, mert a DFCP esetén

állandó redundancia értéket állítottunk be, amely az 5%-os csomagvesztéshez tartozó optimális redundancia érték volt, így a DFCP esetén az overhead mindig azonos volt.



7. ábra A *goodput* a csomagvesztés függvényében, két folyam

A 8. ábrán a változó késleltetés hatásait elemeztük a TCP CUBIC és DFCP esetén. Az első folyam mindig 10 ms késleltetésű, míg a második folyam késleltetése folyamatosan változott 10 ms és 100 ms között. Látható, hogy a TCP folyamok 20 ms-ig képesek igazságosan megosztani a sávszélességet, de ennél nagyobb késleltetéseknél a fix, 10 ms késleltetésű folyam átveszi az irányítást a bottleneck link felett. Ez az úgynevezett *RTT unfairness* probléma a TCP esetén [19]. A DFCP esetén a két folyam végig képes volt igazságosan megosztozni a rendelkezésre álló sávszélességen, mivel a DFCP sokkal kevésbé érzékeny az eltérő késleltetés értékekre.



8. ábra A *goodput* a késleltetés függvényében, két folyam

V. Összefoglalás

A cikkben egy olyan új koncepciót mutattunk be a jövő Internetje számára, amely hibajavító kódoláson alapul és képes hatékony, igazságosság szempontjából is kedvező működést elérni. Megvalósítottunk egy transzport protokollt, amely ezt az elvet alkalmazza és különböző hálózati paraméterek emulációjával megvizsgáltuk a működését, viselkedését, mind egy folyamos, mind több folyamos körülmények között. A kezdeti mérések eredményei alapján ígéretesnek tűnik az elképzelés, amely szerint a torlódásszabályozást elhagyhatjuk, mivel számos esetben a DFCP teljesítménye jobb a jelenleg elterjedten használt TCP verzióknál. A jövőben a protokollt folyamatosan fejlesztjük és további méréseket végzünk többféle topológián (pl. parking lot), illetve többféle ütemezőt tartalmazó architektúrákon (pl. WFQ, DRR és Droptail), összevetve a teljesítményét a jelenlegi protokollokkal annak érdekében, hogy megtaláljuk az esetleges gyenge pontokat mind a TCP, mind a DFCP jelenlegi megvalósítása esetén, így minél jobb megoldást adva a jövő Internetje számára.

Irodalomjegyzék

[1] Y.-T. Li, D. Leith, R. N. Shorten, "Experimental Evaluation of TCP Protocols for High-Speed Networks", IEEE/ACM Transactions on Networking, vol. 15, no. 5, pp. 1109–1122, 2007.

[2] S. Molnár, B. Sonkoly, T. A. Trinh, "A Comprehensive TCP Fairness Analysis in High Speed Networks", *Computer Communications, Elsevier*, vol. 32, no. 13–14, pp. 1460–1484, 2009.

[3] D. Clark, S. Shenker, A. Falk, "GENI Research Plan (Version 4.5)", April 23, 2007.

[4] B. Raghavan, A. C. Snoeren, "Decongestion Control", *Proceedings of the 5th ACM Workshop on Hot Topics in Networks*, pp. 61–66, Irvine, CA, USA, 2006.

[5] T. Bonald, M. Feuillet, A. Proutiere, "Is the 'Law of the Jungle' Sustainable for the Internet?", *Proceedings of the 28th IEEE Conference on Computer Communications*, pp. 28–36, Rio de Janeiro, Brazil, 2009.

[6] L. Lopez, A. Fernandez, V. Cholvi, "A Game Theoretic Comparison of TCP and Digital Fountain Based Protocols", *Computer Networks, Elsevier*, vol. 51, no. 12, pp. 3413–3426, 2007.

[7] Shakeel Ahmad, Raouf Hamzaoui, and Marwan Al-akaidi. Robust Live Unicast Video Streaming with Rateless Codes. In Proceedings of 16th International Workshop on Packet Video, PV 2007, pages 78–84, Lausanne, Switzerland, November 2007.

[8] A. Botos, Z. A. Polgar, V. Bota, "Analysis of a Transport Protocol Based on Rateless Erasure Correcting Codes", *Proceedings of the 2010 IEEE* International Conference on Intelligent Computer Communication and *Processing*, vol. 1, pp. 465–471, Cluj-Napoca, Romania, 2010.

[9] D. Kumar, T. Chahed, E. Altman, "Analysis of a Fountain Codes Based Transport in an 802.11 WLAN Cell", *Proceedings of the 21st* International Teletraffic Congress, pp. 1–8, Paris, France, 2009.

[10] N. Beheshti, Y. Ganjali, R. Rajaduray, D. Blumenthal, N. McKeown, Buffer sizing in all-optical packet switches, in: Optical Fiber Communication Conference, OFC, Anaheim, CA, OThF8, 2006.

[11] A. Shokrollahi, "Raptor Codes", IEEE Transactions on Information *Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[12] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin." in Proc. ACM SIGCOMM, 1995, pp. 231–242.

[13] S. Molnár, Z. Móczár, B. Sonkoly, Sz. Solymos, T. Csicsics, "Design and Performance Evaluation of the Digital Fountain based Communication Protocol", *Technical Report*, 2012. http://hsnlab.tmit.bme.hu/~molnar/files/DFCPTechReport.pdf , accessed April 5, 2014

[14] A. Shokrollahi, "LDPC Codes: An Introduction", Technical Report, Digital Fountain Inc., 2003.

[15] S. Molnár, Z. Móczár, A. Temesváry, B. Sonkoly, Sz. Solymos, T. Csicsics, "Data Transfer Paradigms for Future Networks: Fountain Coding or Congestion Control?", Proceedings of the IFIP Networking 2013 Conference, pp. 1–9, New York, NY, USA, 2013.

[16] T. Henderson, S. Floyd and A. Gurtov, The Newreno Modification to TCP's Fast Recovery Algorithm, RFC 6582, April 2012, 1-16

[17] Dummynet Network Emulator, http://info.iet.unipi.it/~luigi/dummynet/ , accessed April 5, 2014

[18] ns-2 Network Simulator, http://www.isi.edu/nsnam/ns/ , accessed April 5, 2014

[19] E. Gavaletz and J. Kaur, "Decomposing RTT-Unfairness in Transport Protocols," IEEE Workshop on Local and Metropolitan Area Networks, pp 1-6, May 2010.