Fountain Code or Congestion Control Based Transfer: Evaluation on Different Testing Platforms

Sándor Molnár, Zoltán Móczár, Balázs Sonkoly
High Speed Networks Laboratory, Department of Telecommunications and Media Informatics,
Budapest University of Technology and Economics, H–1117, Magyar tudósok krt. 2., Budapest, Hungary
E-mail: {molnar,moczar,sonkoly}@tmit.bme.hu

***Abstract*—One of the key questions for future networks is how to efficiently transfer data traffic generated by versatile applications in heterogeneous and fast changing environments. The lesson learned from the history of the Internet is that congestion control performed by TCP can be a solution. However, due to the limitations of TCP versions a novel paradigm applying fountain code based transfer seems to be very promising. In this paper we address this idea and introduce the prototype of a new fountain code based transport protocol called Digital Fountain based Communication Protocol (DFCP). The operation of DFCP is extensively investigated using various network topologies and settings, and validated on three different testing platforms including our laboratory testbed, the Emulab network emulation environment and the ns-2 network simulator. Moreover, our results of a comprehensive performance evaluation study comparing DFCP to widely used TCP versions are presented and discussed.**

I. Introduction

In the history of the Internet closed-loop congestion control was the successful paradigm to avoid congestion collapse and the related performance degradation due to the overload of network resources. Congestion control is performed by the Transmission Control Protocol (TCP), which transport more than 90% of Internet traffic. The success of TCP was not even questioned until the fast development of networks, mobile devices and user applications resulted in heterogeneous and complex environments with a huge amount of applications in the last decades. In order to fit these changes significant research was carried out to further develop TCP, and therefore, several different TCP versions has been proposed [1], [2], [3]. However, it turned out that it will be very difficult —if not impossible— to modify TCP to work efficiently as a universal transport protocol.

In the recent years an alternative paradigm was suggested, which motivate to rethink the concept of future transport protocols. The idea is that instead of applying congestion control we may handle congestion with efficient erasure coding techniques. An interesting idea was presented by GENI (Global Environment for Network Innovations) [4], recommending the omission of congestion control and promoting erasure coding schemes instead. Unfortunately, no realization or further refinement of the concept has been published so far with some exceptions that we will discuss in the followings.

Raghavan and Snoeren suggested a decongestion controller and investigated its properties in [5]. Bonald et al. studied the network behavior in the absence of congestion control [6]. We emphasize their astonishing result that operating a network without congestion control does not result in congestion collapse in several cases. López et al. presented a fountain based protocol using game theory [7]. They found that a Nash equilibrium can be obtained, and at this equilibrium, the performance of the network is close to the performance experienced when all hosts use TCP. Botos et al. proposed a transport protocol based on the modification of TCP. It is designed for high loss rate environment by utilizing rateless erasure codes [8].

We studied the possibility of applying fountain code [9] based transport protocol instead of congestion control based TCP. As far as we know fountain code based transport protocol has not been developed and implemented with a comprehensive performance evaluation study yet, hence, we made our prototype implementation and carried out a thorough analysis. Moreover, we compared this protocol to different TCP versions regarding their performance in several environments, topologies and settings on three testing platforms. Our results are summarized in this paper, which has the following contributions:

- we present our novel transport protocol called Digital Fountain based Communication Protocol (DFCP);
- we present the validation results of DFCP obtained on three different testing platforms (our laboratory testbed, the Emulab network emulation environment and the ns-2 network simulator);
- we present a comparative performance evaluation study of DFCP and TCP versions (TCP Cubic and TCP NewReno with SACK) on different network topologies (dumbbell and parking lot) using multiple platforms.

The paper is organized as follows. First, we introduce our new transport protocol together with its architecture concept in Section II. Then Section III gives the validation results of DFCP on three different testing platforms. The comparative performance evaluation study of DFCP and different TCP versions can be found in Section IV. Finally, Section V concludes the paper.

II. DFCP: Concept and Prototype Implementation

This section is devoted to briefly summarize the concept of our transport mechanism and to highlight the main implementation issues concerning our first prototype implementation called Digital Fountain based Communication Protocol (DFCP).

A. Overview

Our novel transport protocol is based on the idea outlined by GENI, which advocates a Future Internet without congestion control [4] by suggesting efficient erasure coding schemes to recover lost packets. Applying this approach yields hosts sending at "maximal" rate, thus the network can easily be driven to a state with heavily congested, fully utilized links. However, preliminary works have shown that under some realistic assumptions this behavior will not cause problems, furthermore, the mechanism could exhibit beneficial properties with several respects.

Fountain codes, also known as rateless erasure codes, are a class of erasure codes with the property that a potentially limitless sequence of encoded symbols can be generated from a given set of source symbols such that the original source symbols can ideally be recovered from any subset of the encoded symbols of size equal to or only slightly larger than the number of source symbols [9]. The first practical realization of universal digital fountain based codes were the Luby Transform (LT) codes [10], but they failed to provide low complexity encoding and decoding operations. Therefore, we propose the use of Raptor codes [11] to cope with packet losses as an efficient forward error correction mechanism, which is an extension of LT codes with linear time encoding and decoding complexity. The network architecture relying on *digital fountain based error correction* is shown in Figure 1 for a simple dumbbell topology. We have multiple senders communicating with the corresponding receivers by producing potentially infinite stream of encoded symbols from the original message of size k. Each received packet at the destination host increases the probability of successful decoding, and once any subset of size $\lceil (1+\varepsilon)k \rceil$ encoded symbols arrive to the receiver, decoding can be performed successfully with high probability (here $\epsilon > 0$ denotes the amount of redundancy added to the original message).[1]

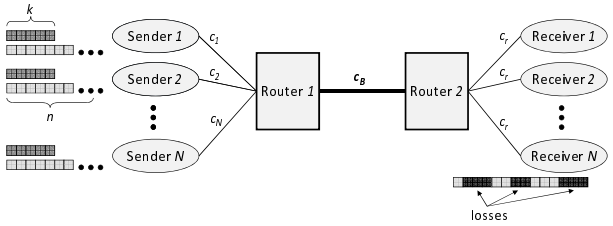


Fig. 1. The network architecture with N sender-receiver pairs

Another important issue that must be treated by this novel network architecture is fairness. More exactly, mechanisms have to be provided to solve the share allocation problem among competing traffic flows sending at different maximum transfer rates. As a solution, we suggest the use of *fair schedulers* in the network nodes. Several implementations approximating the ideal fair scheduling, such as Deficit Round Robin (DRR) [12], are available and can be configured easily.

B. The Operating Mechanism of DFCP

In this part we give only a brief summary about the operation of our transport protocol, but the interested reader can find the details in [13]. DFCP has been implemented in the Linux kernel version 2.6.26-2 and tested under the Debian Lenny distribution. It is a connection-oriented transport protocol providing reliable end-to-end communication. The connection establishment and termination processes are based on the traditional TCP mechanisms with an extension for negotiating all the parameters necessary for decoding between hosts. However, the data transfer phase is completely different since DFCP does not use any congestion control algorithm. Instead, it encodes the data using Raptor codes and sends the encoded data towards the receiver at maximal rate achieving excellent performance. Admittedly, coding needs some overhead, but it will be shown in the following sections that the approach has many advantages and can eliminate several drawbacks of TCP.

The encoding mechanism of the protocol works as follows. First, application layer data is segmented into blocks and stored in a kernel buffer until free space is available. Then the encoding mechanism is called for waiting blocks sequentially. As shown in Figure 2, the implemented Raptor coding scheme is comprised of two phases. Specifically, precoding is realized by LDPC (Low-Density Parity-Check) coding [14], which adds some redundant bytes to the original message symbols then LT coding [11] is invoked to generate a potentially infinite stream from the precoded symbols. Finally, the encoded block is sent after appending a specific protocol header to it including the parameters necessary for identifying and decoding. When a block is successfully decoded at the receiver side, an acknowledgement is sent back.

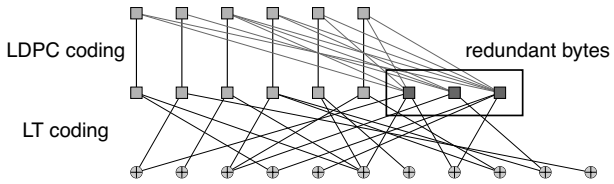


Fig. 2. Encoding phases of message blocks

Sending at maximal rate can flood not only the network but the receivers as well. In order to prevent buffer overflows at the receiver side, we introduce a simple sliding window based *flow control* mechanism. The window size at the DFCP sender gives the maximum number of unacknowledged blocks in the network and controls the burstiness of data transfer. Receiving an acknowledgement yields one unit shift of the sliding window, and the transmission of the next block is started. To guarantee the in-order delivery of message blocks, they are marked by continuously increasing unique identifiers in a dedicated field of the packet header. Thus, receiver can easily determine the block, which the received packet belongs to. When sufficient number of packets of a given block is received, decoding can be accomplished with high probability.[2]

III. Validation on Multiple Platforms

Performance evaluation of a transport protocol in practice requires using different tools to get a clear picture about its behavior and specific properties, and to draw right conclusions.

[1] In realistic scenarios, only slightly more packets are required than the original size of the message.

[2] If decoding is unsuccessful it is not a serious problem, because subsequent packets guarantee that decoding can be performed later.

Even so, most researchers choose only one way to investigate their proposed protocols, namely simulation or testbed measurements. Especially for novel protocols and algorithms it can be misleading due to the unique nature of such environments. On the one hand, the main risk of relying only on simulation results is the fact that simulation environments are far from realistic in most cases, thus many real-world factors can easily be neglected [15], [16]. On the other hand, performing only testbed measurements can also lead to the loss of generality, because special hardware components can affect the results. In addition, building a network testbed is a time-consuming process, and measurements are very difficult to repeat as well [17], [18].

Since DFCP is based on a novel paradigm, it is crucial to ensure that our performance evaluation results are reliable and the conclusions are valid. In order to fit these requirements we carried out a validation study on multiple platforms including our laboratory testbed, the Emulab network emulation environment [19] and the ns-2 network simulator [20]. In this section the network topologies and scenarios are presented, and the description of these platforms is given focusing on the settings and parameters used in the test scenarios. Finally, we show that DFCP performs in a similar way in these environments providing a strong evidence for the operability of our protocol.

A. Network Topologies and Scenarios

The performance of DFCP was evaluated on different network topologies including the simple dumbbell topology and the more complex parking lot topology frequently used in the literature for experiments [21], [22]. Measurements lasted for 1 minute in all scenarios, and the results were obtained by excluding the first 15 seconds in order to ignore the impact of transient behavior of the investigated transport protocols. In case of multiple flows they were started at the same time, and for scheduling discipline WFQ (Weighted Fair Queueing) was applied with equal weights [23].

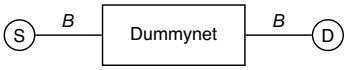


Fig. 3. Dumbbell topology with one source-destination pair

The test scenarios performed on the dumbbell topology can be seen in Figure 3 were intended to reveal the ability of DFCP to resist against varying delay and packet loss rate parameters of the network. In this case the data were carried by one flow from source to destination, and the bottleneck link (B) capacity was set to 1 Gbps.

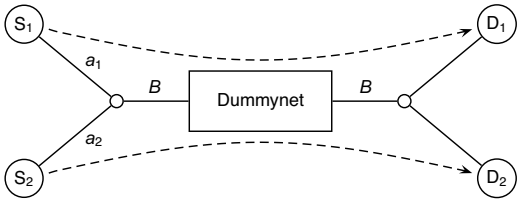


Fig. 4. Dumbbell topology with two source-destination pairs

To examine the fairness properties of DFCP we ran experiments on the topology of Figure 4. The main purpose was to observe how DFCP behaves in a situation when two flows compete for the available bandwidth determined by the bottleneck link. In this scenario both the access links (a_1, a_2) and the bottleneck link (B) had a capacity of 1 Gbps.

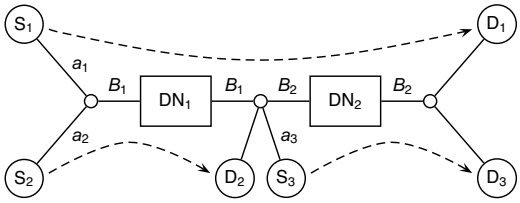


Fig. 5. Parking lot topology with three source-destination pairs

The topologies and test scenarios described above made possible to explore the fundamental features of DFCP. Beyond these experiments DFCP was studied in a more realistic environment too. Figure 5 depicts a parking lot topology with three sender and receiver nodes, which contains two bottleneck links. In a real network multiple bottlenecks are common, and therefore, it is indispensable to evaluate how a transport protocol performs in such conditions. In these tests the capacity was 1 Gbps for each access link (a_1, a_2, a_3), and the bottleneck link (B_1, B_2) capacities were set to different values as discussed in the following sections.

B. Test Environments

To validate the performance evaluation results the test scenarios were executed on the following three different platforms independently: (1) our laboratory testbed, (2) the Emulab network emulation environment and (3) the ns-2 network simulator. In this subsection we give a brief description about these platforms.

The laboratory testbed consisted of senders, receivers and a Dummynet network emulator, which was used for simulating various network parameters such as queue length, bandwidth, delay and packet loss probability [24]. Each test computer was equipped with the same hardware components according to Table I.

TABLE I
HARDWARE COMPONENTS OF OUR LABORATORY TEST COMPUTERS

Component	Type and parameters
Processor	Intel® Core™2 Duo E8400 @ 3 GHz
Memory	2 GB DDR2 RAM
Network adapter	TP-Link TG-3468 Gigabit PCI-E
Operating system	Debian Lenny with modified kernel

(a) Hardware components of senders and receivers

Component	Type and parameters
Processor	Intel® Core™ i3-530 @ 2.93 GHz
Memory	2 GB DDR2 RAM
Network adapter	TP-Link TG-3468 Gigabit PCI-E
Operating system	FreeBSD 8.2

(b) Hardware components of network emulators

The second platform we worked on, Emulab, is a network testbed giving researchers a wide range of environments in which to develop, debug and evaluate their systems [19]. The measurement setup was identical to the one used in

our laboratory testbed for each test scenario, but the test machines were equipped with different hardware components as summarized in Table II. The type of the sender and receiver nodes was *pc3000* according to the Emulab label system, and the network emulators were run on *d710* type nodes. Similar to the testbed measurements our modified kernel including the implementation of DFCP was loaded into the test computers.

TABLE II
HARDWARE COMPONENTS OF THE EMULAB TEST COMPUTERS

Component	Type and parameters
Processor	Intel® Xeon® processors @ 3 GHz
Memory	2 GB DDR2 RAM
Network adapter	Intel Gigabit PCI-E
Operating system	Debian Lenny with modified kernel

(a) Hardware components of senders and receivers

Component	Type and parameters
Processor	Intel® Xeon® E5530 @ 2.40 GHz
Memory	12 GB DDR2 RAM
Network adapter	Broadcom NetXtreme II 5709 Gigabit PCI-E
Operating system	FreeBSD 8.3

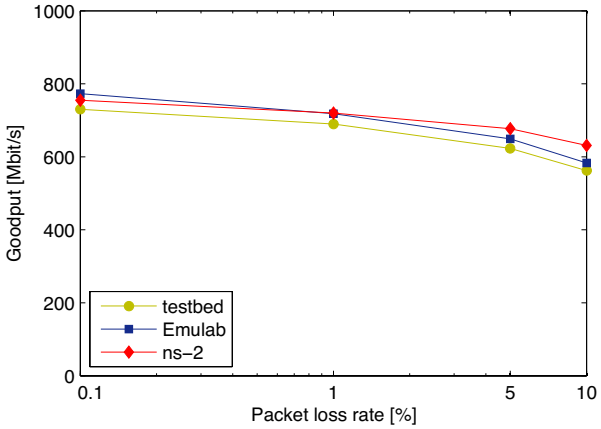
(b) Hardware components of network emulators

The third tool was the ns-2 network simulator to validate DFCP, which is widely used by researchers to try out and evaluate their new methods [20]. Since the first prototype of DFCP has been implemented in the Linux kernel, we had to find a way to simulate our protocol directly through the network stack of Linux. In fact, there are some tools available for this purpose, but only few of them can provide reasonable accuracy and efficiency, as well as support a wide range of operating systems and kernel versions [25]. Focusing on these requirements we chose Network Simulation Cradle (NSC), which is a framework for wrapping kernel code into simulators allowing the simulation of real-world behavior at little extra cost [26]. NSC supports the simulation of the network stacks of many operating systems such as FreeBSD, OpenBSD, lwIP and Linux. This tool has been validated by comparing situations using a test network with the same situations in the simulator, and it has been shown that NSC is able to produce extremely accurate results. Moreover, it has been ported to several network simulators including both ns-2 and ns-3. Although, NSC is an excellent tool for simulating different TCP versions and new TCP-like transport protocols, we had to carry out a challenging work to get NSC able to handle DFCP, which is based on a novel paradigm and it is significantly different compared to the principles applied by TCP.

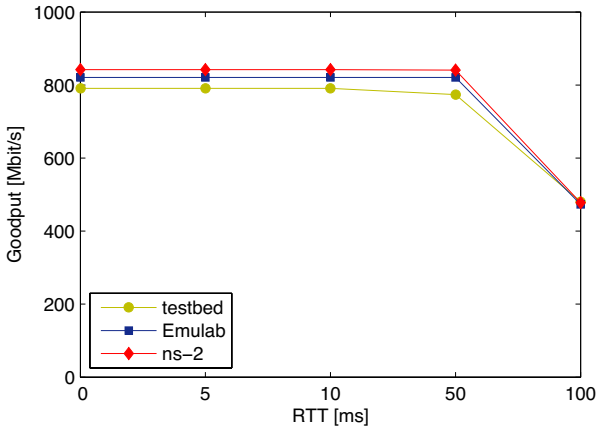
C. Validation Results

In this part we present the validation results performed on the three different platforms discussed previously. The performance of DFCP was measured in terms of goodput, which is a well-known and widely used performance metric. It gives the number of useful data bytes successfully transferred per second.

Figure 6 illustrates the main features of DFCP introducing its high resistance to varying network conditions such as packet loss rate and round-trip time (RTT). These measurements were carried out on a dumbbell topology with one source-destination pair (see Figure 3). Figure 6a shows the impact of packet loss rate on the goodput performance of DFCP. It is important to investigate this aspect since TCP is very sensitive to packet loss resulting in a quick performance degradation for increasing loss rate. The figure clearly indicates that DFCP can operate efficiently even in high loss rate environments using optimal redundancy. Optimal redundancy is the minimum coding overhead assuming a given loss rate that is necessary for successful data transmission and decoding at the receiver side. Figure 6b shows how the goodput performance of DFCP affected by the round-trip time parameter of the network. One can see that it achieves outstanding performance in a network with high delay, because goodput drops slowly as the round-trip time increases. Since the curves depicted in Figure 6 have very similar characteristics for the three platforms, we can state that these advantageous features of DFCP are validated.



(a) The impact of packet loss rate on the performance



(b) The impact of round-trip time on the performance

Fig. 6. The main features of DFCP

Figure 7 presents how two DFCP flows having different round-trip times share the available bandwidth (see Figure 4), which is a common situation in real networks often referred to as RTT fairness problem in the context of transport protocols. It is an important property since traditional TCPs are unfair in the sense that the flow with lower RTT receives more bandwidth than the flow having higher RTT. In this scenario flow 1 had a fixed RTT of 10 ms, and the delay of flow 2 was increased

from 10 to 100 ms. The figure shows that DFCP behaves in a fair way on different platforms since both flows get an equal share of the bottleneck bandwidth independently of their RTTs.

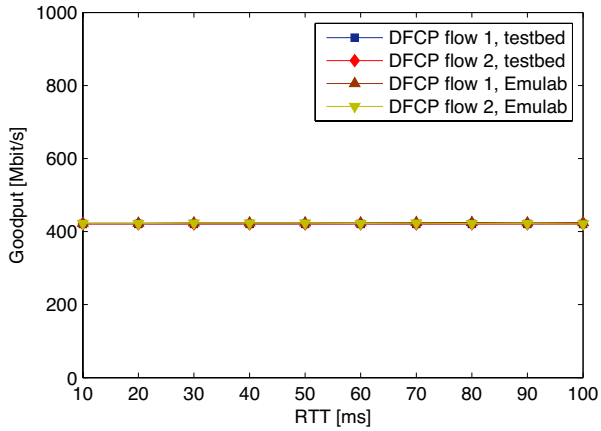


Fig. 7. Bandwidth sharing between two competing DFCP flows with different RTTs

The results of Figure 8 were obtained on the parking lot topology illustrated in Figure 5. The scenario was designed to study the behavior of DFCP in a multiple bottleneck environment. The capacity of the first bottleneck link (B_1) was set to 1 Gbps, and the second bottleneck link (B_2) had a capacity of 500 Mbps. The figure depicts the goodput of the three DFCP flows as the function of the RTT experienced on B_2 while the RTT of B_1 is fixed at 10 ms. We can observe that flow 1 and flow 3 receive an equal portion of the bandwidth available on B_2. Since the rate of flow 1 is limited by the capacity of B_2, flow 2 gets more bandwidth than flow 1 utilizing the available bandwidth of B_1. Therefore, each bottleneck link becomes fully utilized and is shared fairly by DFCP flows.

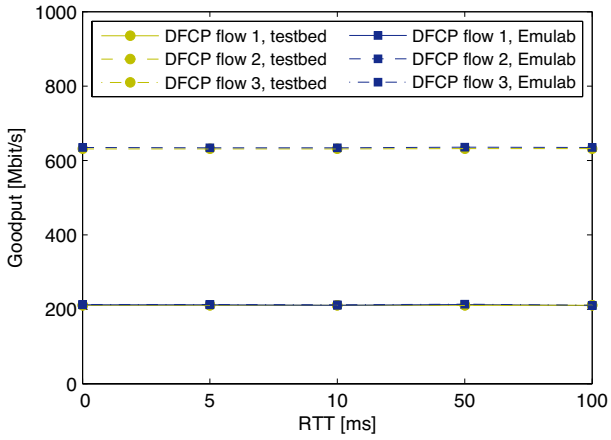
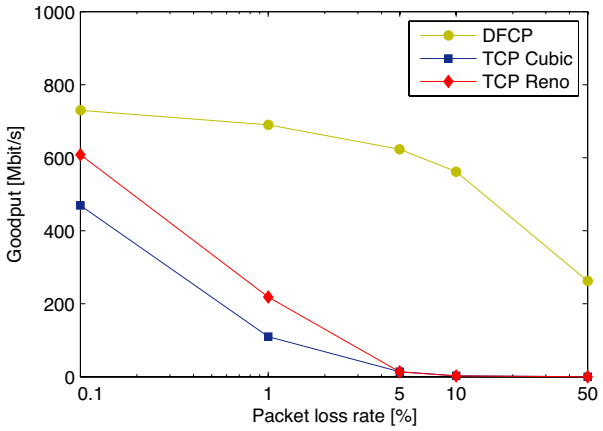


Fig. 8. The performance of DFCP in a network with multiple bottlenecks

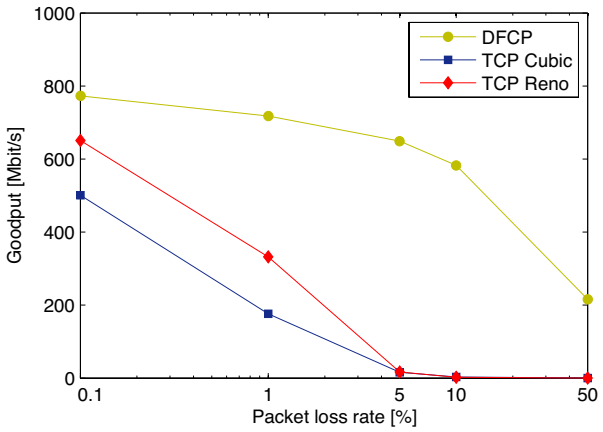
IV. Comparative Performance Evaluation

In this section we present a comprehensive performance analysis study by comparing DFCP to different TCP versions, namely TCP Cubic which is the default congestion control algorithm in the Linux kernel and TCP NewReno with SACK option. Measurements were carried out in two real networks including our laboratory testbed and the Emulab network emulation environment.

One of the main beneficial properties of DFCP can be seen in Figure 9. It demonstrates that DFCP is much more resistant to packet loss than TCP Cubic and Reno if optimal redundancy is used. The difference in goodput is already considerable for packet loss of 0.1%, but for increasing loss rate DFCP highly outperforms both TCP variants. For example, for packet loss of 1% the ratio between the goodput obtained by DFCP and TCP Reno is about 3, and this ratio is 6 for TCP Cubic. When the loss rate attains 10%, DFCP gets more than 250 times faster compared to TCPs, and it works efficiently even in the case of extremely high loss (50%) in contrast to TCPs, which are unable to operate under these network conditions. Note that the performance characteristics of the investigated transport protocols seem to be very similar in our laboratory testbed and in the Emulab environment.



(a) Testbed

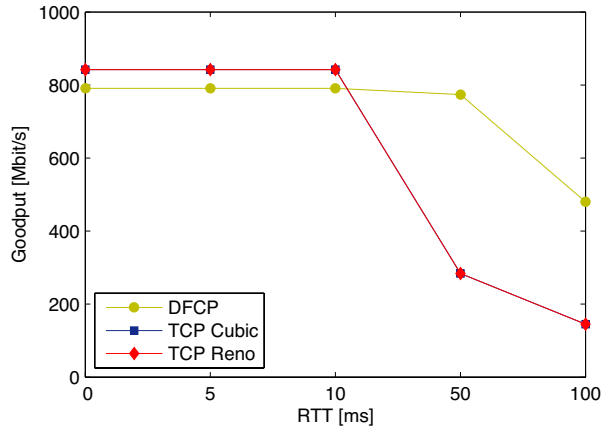


(b) Emulab

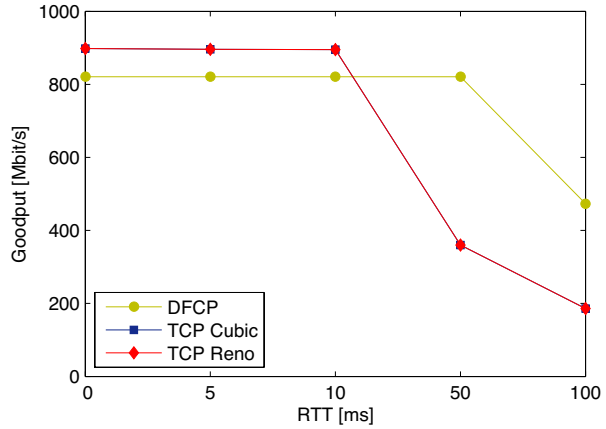
Fig. 9. The performance of DFCP and TCPs in a lossy environment

Figure 10 shows the performance comparison results of DFCP and TCPs for varying round-trip time. The figure illustrates that in the RTT interval 0–10 ms TCP versions perform better than DFCP in terms of goodput, but the difference is negligible and it is due to the coding overhead. Nevertheless, for delay values greater than 10 ms DFCP achieves significantly higher transfer rate compared to TCP Cubic and Reno. Since the typical value of round-trip time in a real network

exceeds 10 ms, DFCP can function more efficiently than TCP in such conditions [27].



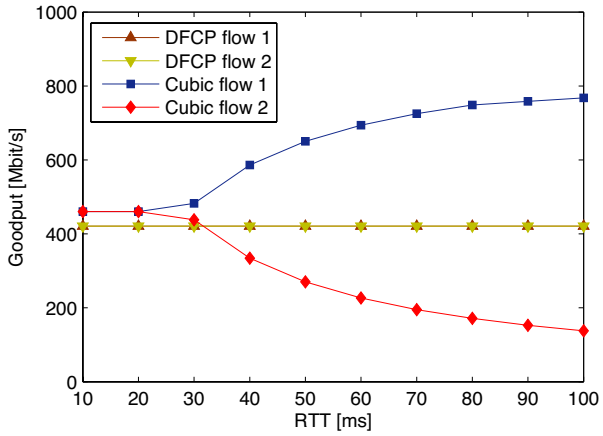
(a) Testbed



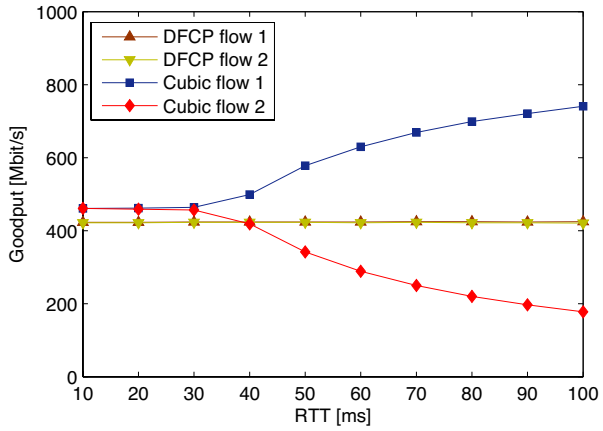
(b) Emulab

Fig. 10. The performance of DFCP and TCPs for varying RTT

Additionally, it is essential to reveal and investigate how a transport protocol shares the available bandwidth of a bottleneck link among competing flows often called as fairness property. As widely known, base TCP cannot provide an equal portion of the bottleneck bandwidth for competing flows with different round-trip times [28] due to its AIMD mechanism [29]. Figure 11 depicts the goodput for two competing DFCP and TCP Cubic flows. The delay of flow 1 was fixed at 10 ms, and for flow 2 we varied the delay parameter between 10 and 100 ms. Since the results for TCP Reno were quite the same as in case of TCP Cubic, only the latter was plotted. The figure shows that the bottleneck link capacity is equally shared by the two TCP flows for RTT values less than 20 ms regarding our testbed measurements (see Figure 11a). However, for RTTs greater than 20 ms the goodput of flow 2 starts to decrease, and as a result, flow 1 with lower RTT can gain access to a greater portion of the available bandwidth, indicating the unfairness behavior of TCP. In contrast, DFCP flows achieve perfect fairness as they share the bottleneck capacity equally and they are much less sensitive to the round-trip time compared to TCP. We note that the difference can be observed in the goodput of DFCP and TCP flows for RTT values less than 20 ms is due to the coding overhead. Comparing the results obtained on different platforms, we experienced that the behavior of DFCP in Emulab (see Figure 11b) is as same as in our laboratory testbed (see Figure 11a) while TCP Cubic achieves a slightly better fairness in Emulab.



(a) Testbed

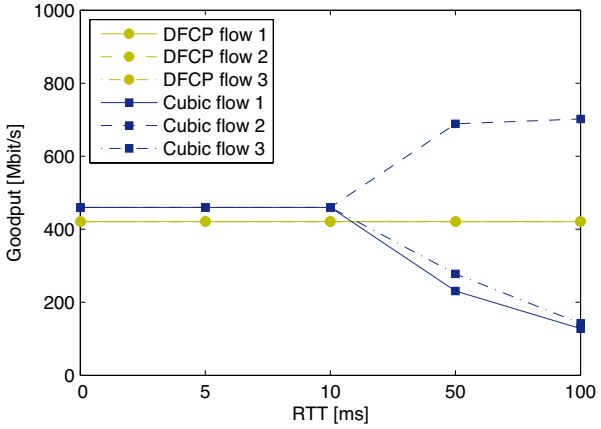


(b) Emulab

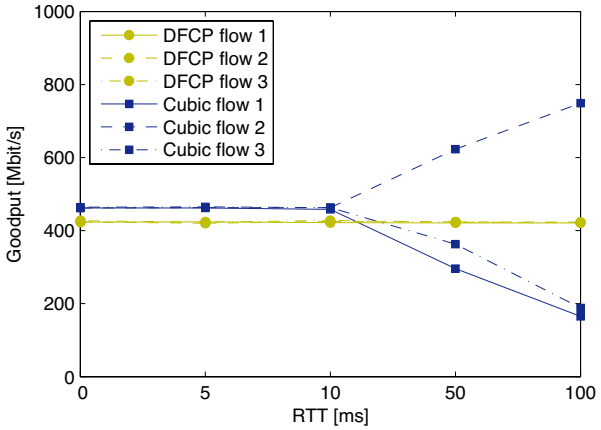
Fig. 11. The performance of DFCP and TCP in case of two competing flows

Figure 12 presents the performance comparison of DFCP and TCP Cubic carried out on the parking lot topology illustrated in Figure 5 by starting three concurrent flows. In this test scenario the capacity was set to 1 Gbps for both bottleneck links denoted by B_1 and B_2. The round-trip time was fixed at 10 ms on B_1, but it was increased on B_2 from 0 to 100 ms. Looking at the figure we can make the following observations. Until the round-trip time experienced on B_2 attains 10 ms, both DFCP and TCP Cubic share the bottleneck bandwidth of B_1 and B_2 in a fair way. However, for higher delay values TCP Cubic gradually becomes unfair due to the fact pointed out in this section, namely, TCP is sensitive to round-trip time. As the goodput obtained by flow 1 and flow 3 drops for increasing RTT (since they go through B_2), flow 2 with lower RTT receives more and more bandwidth. Accordingly, TCP Cubic does not provide fairness between flow 1 and flow 2 having different RTTs. Moreover, in this case the available capacity of B_2 is also shared unequally, and hence, flow 1 and flow 3 achieve different goodput performance. As we mentioned earlier it is an undesirable behavior, and the results show that DFCP can solve this issue by providing perfect

fairness for each flow independently of their RTTs thanks to its robustness to varying network conditions.



(a) Testbed



(b) Emulab

Fig. 12. The behavior of DFCP and TCP in a more complex network

V. Conclusion

In this paper we investigated two alternative paradigms as possible data transport mechanisms in future networks: the congestion control based transfer (TCP) and a fountain code based transfer. For the sake of thorough analysis we chose recent TCP versions (TCP Cubic and TCP NewReno with SACK) and we designed and implemented a new fountain code based transport protocol called Digital Fountain based Communication Protocol (DFCP). In order to draw solid conclusions we studied the operation of DFCP using various network topologies (dumbbell and parking lot) on three testing platforms including our laboratory testbed, the Emulab network emulation environment and the ns-2 network simulator. We also carried out a comparative performance evaluation study of TCP and DFCP on these platforms. We showed that the goodput performance of DFCP is significantly better than in case of the investigated TCP versions in a wide range of packet loss rates and round-trip times. The results suggest that DFCP is a promising approach, and the possible application areas cover the high latency and high loss rate network environments.

Acknowledgment

The research was supported by the OTKA-KTIA grant CNK77802. The authors would like to thank Szilárd Solymos for the prototype implementation, Tamás Csicsics and Gábor Zóber for the measurements carried out in the laboratory testbed and in the Emulab environment, respectively.

References

[1] A. Afanasyev, N. Tilley, P. Reiher, L. Kleinrock, "Host-to-Host Congestion Control for TCP", *IEEE Communications Surveys and Tutorials*, vol. 12, no. 3, pp. 304–342, 2010.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] L. López, A. Fernández, V. Cholvi, "A Game Theoretic Comparison of TCP and Digital Fountain Based Protocols", *Computer Networks, Elsevier*, vol. 51, no. 12, pp. 3413–3426, 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. J. C. MacKay, "Fountain Codes", *IEE Proceedings – Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.

[10] M. Luby, "LT Codes", *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pp. 271–280, Vancouver, BC, Canada, 2002.

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

[12] M. Shreedhar, G. Varghese, "Efficient Fair Queuing Using Deficit Round-Robin", *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, 1996.

[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

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

[15] K. Pawlikowski, H.-D. Joshua Jeong, J.-S. Ruth Lee, "On Credibility of Simulation Studies of Telecommunication Networks", *IEEE Communications Magazine*, vol. 40, no. 1, pp. 132–139, 2002.

[16] M. Bateman, S. Bhatti, "TCP Testing: How Well Does ns2 Match Reality?", *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications*, pp. 276–284, Perth, Australia, 2010.

[17] S. Floyd, E. Kohler, "Tools for the Evaluation of Simulation and Testbed Scenarios", *Technical Report, IETF*, 2006.

[18] M. P. Fernandez, S. Wahle, T. Magedanz, "A New Approach to NGN Evaluation Integrating Simulation and Testbed Methodology", *Proceedings of the 11th International Conference on Networks*, pp. 22–27, Saint-Gilles, Réunion Island, France, 2012.

[19] Emulab Network Emulation Testbed, http://www.emulab.net/

[20] ns-2 Network Simulator, http://www.isi.edu/nsnam/ns/

[21] D. X. Wei, P. Cao, S. H. Low, "Time for a TCP Benchmark Suite?", *Technical Report, California Institute of Technology*, Pasadena, CA, USA, 2005.

[22] H. Shimonishi, M. Y. Sanadidi, T. Murase, "Assessing Interactions among Legacy and High-Speed TCPs", *Proceedings of the 5th International Workshop on Protocols for Fast Long-Distance Networks*, pp. 91–96, Marina del Rey, CA, USA, 2007.

[23] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–1396, 1995.

[24] Dummynet Network Emulator, http://info.iet.unipi.it/∼luigi/dummynet/

[25] M. Lacage, "Experimentation Tools for Networking Research", *Ph.D. Thesis* (available at http://cutebugs.net/files/thesis.pdf), 2010.

[26] Network Simulation Cradle, http://www.wand.net.nz/∼stj2/nsc/

[27] S. Kaune, K. Pussep, C. Leng, A. Kovacevic, G. Tyson, R. Steinmetz, "Modelling the Internet Delay Space Based on Geographical Locations", *Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pp. 301–310, Weimar, Germany, 2009.

[28] T. V. Lakshman, U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss", *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, pp. 336–350, 1997.

[29] D.-M. Chiu, R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Journal of Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, 1989.