How to Transfer Flows Efficiently via the Internet?

Sándor Molnár[†*], Zoltán Móczár[†*], Balázs Sonkoly[†]
[†]Dept. of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary
[*]Inter-University Centre for Telecommunications and Informatics, Kassai út 26., 4028 Debrecen, Hungary
E-mail: {molnar, moczar, sonkoly}@tmit.bme.hu

***Abstract*—In this paper we present a performance evaluation study of different transport protocols to reveal their efficiency regarding the transfer of different flows. The current versions of *Transmission Control Protocol* (TCP) are compared to *Digital Fountain based Communication Protocol* (DFCP), which is our newly developed transport protocol where congestion control is not applied, but Raptor code based erasure coding scheme is used to recover lost packets. Our results demonstrate that both short-lived flows (web objects) and long-lived flows (DVD objects) can be transferred more efficiently by DFCP than by TCPs.**

I. Introduction

From the early days of the Internet when congestion collapse occurred [1] till today congestion control is used to regulate traffic and avoid situations where increasing network load results in a decrease in the useful work done by the network. This functionality is mostly performed by the *Transmission Control Protocol* (TCP), which was continuously developed and tuned over the previous decades. The development of TCP was unavoidable due to the emerging challenges of the next generation networks like high speed communication, communication over different media, etc. [2], [3]. TCP is a connection-oriented unicast transport protocol that offers reliable data transfer as well as flow and congestion control. Basically, TCP maintains a congestion window that controls the number of outstanding unacknowledged data packets in the network.

However, not all applications use TCP, and a broad range of different congestion control schemes has also been developed. On the other hand, it was desirable to define appropriate rate adaptation rules and mechanisms for non-TCP traffic as well that are compatible with the rate adaptation mechanism of TCP [4]. Such TCP-friendliness can be a useful concept, but we expect that future network architectures will allow or require different definitions of fairness.

The limitations of TCP and the need of up-to-date tuning of its underlying mechanisms may result in TCP versions, which cannot be optimal for all environments and are becoming more and more complex with emerging drawbacks. This was the reason to rethink the concept of this transport protocol and design it from scratch with a brave step towards omitting its congestion control functionality. The idea was first presented by GENI (Global Environment for Network Innovations), which advocated a Future Internet without congestion control [5] by suggesting efficient erasure coding schemes to recover lost packets. We did not find any realization or further refinement of this suggestion, thus we made our own design and implementation of this concept resulting in a transport protocol called *Digital Fountain based Communication Protocol* (DFCP) [10]. The idea related to DFCP is that end hosts can send their data at maximal rates while *fair schedulers* deployed in the network nodes are responsible for providing fairness among competing flows. We note that several implementations approximating the ideal fair scheduling, such as Deficit Round Robin (DRR) [6], are already available and can be configured easily. If a packet loss is detected (it is very likely since no congestion control is applied), efficient digital fountain based (rateless) codes [7] are used to recover lost packets. We have designed DFCP with Raptor codes [8] and implemented in Linux [9].

The proper evaluation of transport protocols is important and requires thorough investigations. The performance must be analyzed carefully, and it is crucial whether congestion control is applied or not and how efficiently it can work. Fairness properties are also of high importance including both inter- and intra-protocol fairness and TCP-friendliness. For practical point of view the deployment of these protocols is the key factor that must be studied as well. It is known that the performance of the implemented transport protocols (e.g. TCP versions) in the Internet differ from theory due to the interactions between TCP and middleboxes along the network path [11]. For example, Performance Enhancing Proxies (PEPs) break single TCP connections into two connections potentially changing the end-to-end behavior.

In order to evaluate the performance of different techniques like congestion control based TCPs, or methods without congestion control like DFCP, right metrics must be chosen. Besides the broadly investigated *throughput*, the *Flow Completion Time* (FCT) also serves as an important metric since most of the applications use flow transfers and users' main interest is to download their flows as fast as possible [12]. FCT is the time from when the first packet of a flow is sent until the last packet is received. Flows transmitted via the Internet have very complex characteristics [13] and the mechanisms of different transport protocols can handle them differently. For example, it is well-known that TCP enters the congestion avoidance phase after slow-start, which takes many round-trip times (RTT), and the majority of short-lived flows never leave slow-start so it is the mechanism that can determine how fast a short-lived flow is downloaded. For long-lived flows the additive increase of the congestion avoidance phase limits the fast finishing of these flows and the fact that TCP fills the bottleneck buffer making extra delay also contributes to the increase of FCT and it is far from being optimal. Therefore, it is of high interest how different transport protocols are able to cope with different flows in the Internet, which was the motivation of this research and this paper.

We present our performance evaluation study carried out in a testbed environment where recent TCP transport protocols

are compared to DFCP regarding their file transfer efficiency for various packet loss rates and round-trip times. We emphasize that our results not only include the comparison of these transport protocols, but as far as we know the first time to implement the concept of transport protocol without congestion control.

The paper is structured as follows. In Section II a brief overview of related work is given. A short description of the investigated TCP versions and DFCP is provided in Section III. In Section IV we present our testbed results of the performance analysis. Finally, Section V concludes the paper.

II. Related Work

Regarding transport protocols of TCP types a huge volume of literature exists since TCP and its different versions (HSTCP, CUBIC, FAST, Compound, Westwood, etc.) determined the mainstream of this research. For a comprehensive tutorial see [14]. The performance of these versions were also investigated and compared, for instance [3], [14]. In addition, there have been intensive research efforts on other protocols like eXplicit Control Protocol (XCP), Rate Control Protocol (RCP), etc. to overcome the limitations of TCP [12].

As far as developing transport protocol without congestion control no relevant research has been done since the presentation of the idea in GENI [5]. In a related work a decongestion controller was suggested by Raghavan and Snoeren [15]. Bonald et al. analyzed the network behavior in the absence of congestion control [16]. They found that the common belief, namely, operating a network without congestion control necessarily leads to congestion collapse is false. López et al. investigated a fountain based protocol using game theory [17]. They showed that a Nash equilibrium can be achieved, and at this equilibrium, the performance of the network is similar to the performance obtained when all hosts comply with TCP. Botos et al. presented a transport protocol based on the modification of TCP for high loss rate environment using rateless erasure codes [18].

Surprisingly, the issue of how to design a transport protocol, which minimizes FCT is addressed only in a few papers. It is known that for a single link the *Shortest Remaining Processing Time* (SRPT) scheduling discipline minimizes FCT [19]. The practical implementation of SRPT is limited in the Internet due to many problems, for example, it requires the flow size information for the end hosts, which is not available when a flow starts. There are many suggestions trying to approximate SRPT, and a practical approach is to consider Processor Sharing (PS) policy instead [12]. PS is able to approximate SRPT and it has the advantage that it does not require flow size information in advance. Most of the TCP versions can approximate PS behavior for long-lived flows, but they fail to achieve it for short-lived flows. This is because they can react over many round-trip times, which is too late since short-lived flows are processed already in the slow-start phase.

III. Overview of the Transport Protocols

A. TCP Versions

TCP is a connection-oriented transport protocol that provides reliable data transfer in end-to-end communication. It means that lost packets are retransmitted, and therefore, each sent packet will be delivered to the destination. The most important feature of TCP is its congestion control mechanism, which is used to avoid congestion collapse by determining the proper sending rate and to achieve high performance. TCP maintains a congestion window that controls the number of outstanding unacknowledged data packets in the network. Over the years, many versions of TCP have been developed in order to fit the ever-changing requirements of communication networks. In this paper we investigate two popular and widely used TCP variants in comparison to DFCP, namely TCP Cubic which is the default congestion control algorithm in the Linux kernel and TCP NewReno with SACK option.

The window growth function of Cubic is governed by a cubic function in terms of the elapsed time since the last loss event, which provides a good stability and scalability [20]. Furthermore, the real-time nature of the protocol keeps the window growth rate independent of RTT making the protocol TCP-friendly under both short and long RTT paths.

In TCP Reno, a lost packet is detected and retransmitted when triple duplicate acknowledgements are received or a timeout event occurs at the sender. TCP Reno is effective to recover from a single packet loss, but it still suffers from performance problems when multiple packets are dropped from a window of data [21]. TCP NewReno is a slight modification of TCP Reno intended to improve its performance when a burst of packets is lost [22].

B. Digital Fountain based Communication Protocol

DFCP is also a connection-oriented, reliable transport protocol, which can be found in the transport layer of the TCP/IP stack. However, unlike TCP our newly developed DFCP protocol does not use any congestion control. Instead, it uses efficient erasure coding based on Raptor codes [8], which are an extension of LT codes [7] offering linear time encoding and decoding complexity. Basically, DFCP sends the encoded data towards the receiver at maximal rate making possible to carry out a very efficient operation. In this case, efficient means that available resources in the network can be fully and quickly utilized without experiencing performance degradation.

The operation of the protocol consists of three main steps: connection establishment, data transfer and connection termination. The connection establishment uses a three-way handshake procedure and it also performs the negotiation of all the parameters between the sender and receiver, which are necessary for decoding of the application data. In the data transfer phase application data are first divided into blocks and each of them is stored in a kernel buffer until free space is available. The waiting blocks are sequentially encoded by a Raptor coding procedure as shown in Figure 1.

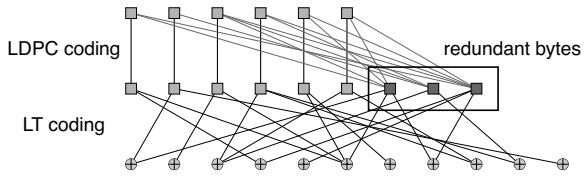


Fig. 1. Encoding phases of message blocks

The coding process has two main phases: an LDPC (Low-Density Parity-Check) based precoding phase [23] and an LT coding phase [8]. LDPC extends the original message symbols with some redundant bytes. LT coder receives the result of the LDPC coding phase as input and generates a potentially infinite stream of encoded bytes. In addition, a specific protocol header is also added to the encoded block and the block is immediately sent. The protocol header includes all necessary information for identifying and decoding the given message block.

As a flow control DFCP uses a sliding window mechanism. The window size gives the maximum number of unacknowledged blocks in the network. It can be set and it controls the burstiness of data transfer. When the destination host receives a block it sends an acknowledgement to the source host. If this acknowledgement is received by the sender the sliding window shifts by one unit and the next block is sent. To ensure in-order delivery DFCP assigns a continuously increasing unique identifier to each block in the protocol header, hence the receiver can recover the original order of blocks automatically. The received blocks can be decoded with high probability thanks to the very efficient Raptor coding scheme. When all data are successfully received by the destination host, the connection is released by the handshake procedure in the connection termination phase.

DFCP has been implemented in the Linux kernel version 2.6.26-2 and it has been tested under the Debian Lenny distribution. For further details of DFCP please see [9].

IV. Testbed Analysis

In this section we present and discuss our measurement results comparing DFCP and two TCP variants. Our purpose was twofold: (1) to investigate the transient behavior of the transport protocols, and (2) to reveal how the flow completion time depends on different loss and delay parameters of the network. The performance analysis was carried out in a testbed environment for different network topologies and test scenarios. In order to focus on the effects of these parameters and to avoid the impact of the Raptor codec implementation, its delay was excluded from the results.

TABLE I
Hardware components of 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 the network emulator

A. Testbed Environment

The measurement scenarios were composed of senders, receivers and a Dummynet network emulator. We were able to tune different network parameters by Dummynet such as packet loss probability, delay, queue length and bandwidth [24]. The main hardware specification of the test computers are given in Table I.

B. Transient Behavior

It is of high importance to investigate the transient behavior of different transport protocols since a huge number of applications download short-lived flows (e.g. web objects) that are performed mostly or fully in the transient phases of these protocols.

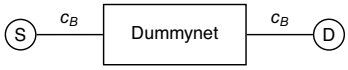
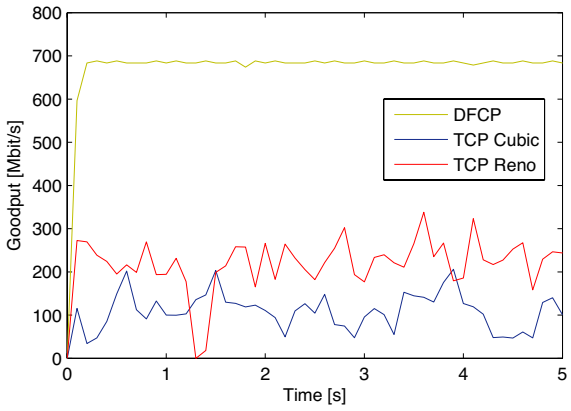
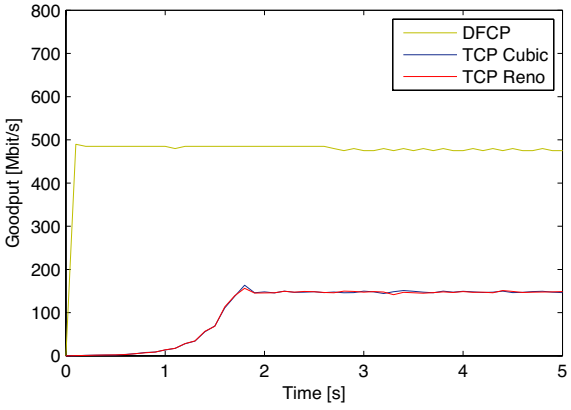


Fig. 2. Dumbbell topology with one source-destination pair

The experiments were performed on a simple dumbbell topology with one source and destination as shown in Figure 2. The measurement duration was 60 seconds for each test, and the flows were started separately. Regarding the network parameters only the packet loss rate and the round-trip time were varied. The buffer size was set to a high value in order to exclude it from the limiting factors, and the bottleneck link had a capacity $c_B = 1$ Gbps. In these scenarios we used the goodput (i.e. the number of useful bytes transferred per second) as the performance metric.



(a) Packet loss rate = 1%



(b) Round-trip time = 100 ms

Fig. 3. Transient behavior of the investigated transport protocols

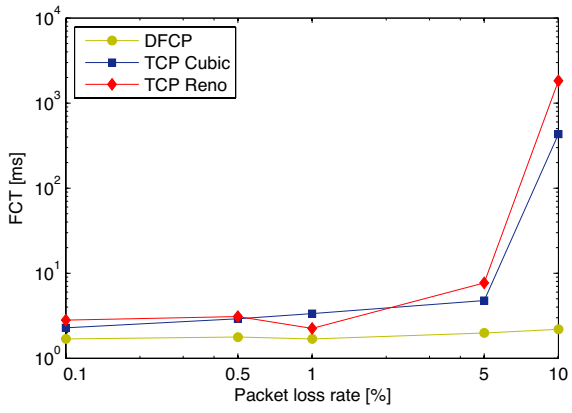
In Figure 3 the goodput is depicted for the first 5 seconds of the measurement simulating different network conditions. Figure 3a shows the case when the packet loss rate was fixed at 1%, and the redundancy parameter of DFCP was set to an optimal value. Optimal redundancy is the minimum coding overhead assuming a given loss rate that is necessary for successful data transmission and decoding at the receiver side. The figure clearly indicates that DFCP significantly outperforms both TCP versions in terms of goodput in a lossy environment, and unlike TCP the goodput of DFCP does not fluctuate over time. In other words, DFCP is much less sensitive to packet loss than TCP, which is an outstanding result since one of the most well-known drawbacks of TCP is that its performance degrades very quickly for increasing packet loss probability. Our analysis results (not presented here due to space limitations) also showed that, as we increase the packet loss rate, the difference between DFCP and TCPs becomes even more dramatic regarding the goodput. Figure 3b demonstrates the performance of the investigated protocols when the round-trip time was set to 100 ms. We can see that, while DFCP immediately achieves its full speed, the transfer rates of the TCP variants increase much more slowly, and the steady-state goodput is considerably lower compared to DFCP.

We observed as a general conclusion from all of our testbed analysis that for increasing packet loss rate and round-trip time, the difference in the goodput performance of DFCP and TCP variants also gets larger showing that DFCP is much more insensitive to varying network conditions.

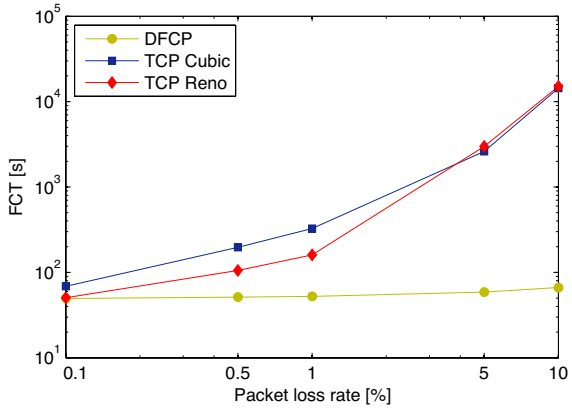
C. Flow Completion Time

As we mentioned in Section I, flow completion time is one of the most important performance metrics from the user's point of view because of the fact that users want to download web pages, softwares, movies and many other contents as fast as possible. Accordingly, we investigated two different categories: (1) web object (150 kB, the mean size is about 100–200 kB [25]) and (2) DVD (4.7 GB), which represent short and long data transfers, respectively.

Figure 4 illustrates how the flow completion time depends on packet loss rate. The flow completion times longer than 60 seconds were calculated by using the steady-state goodput for each figure of this subsection. One can see that in both cases DFCP provides the fastest download indicating its potential in case of web traffic as well as heavy data transfers, however, the benefit is more significant in the latter case. By transferring a typical web object, the most considerable performance gain can be experienced for high packet loss rates (see Figure 4a). However, if we transfer a full DVD the advantage of DFCP is pronounced in the whole range of packet loss rate (see Figure 4b). Moreover, with optimal redundancy parameters, DFCP becomes almost insensitive to packet loss in these practically relevant scenarios.

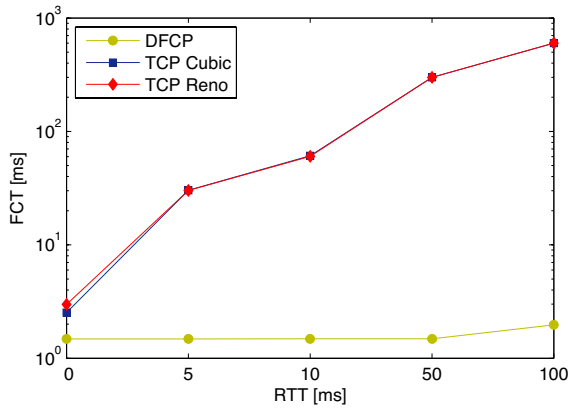


(a) Web object

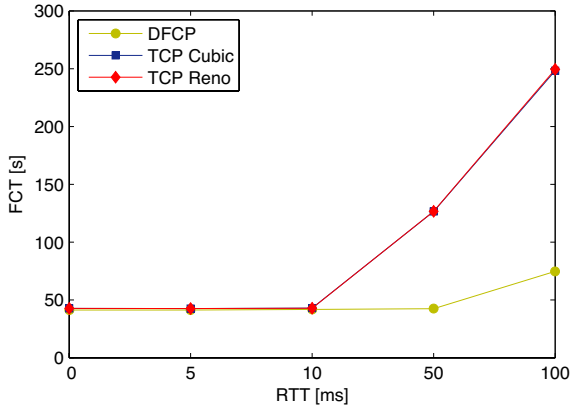


(b) DVD

Fig. 4. Flow completion time for different packet loss rates



(a) Web object



(b) DVD

Fig. 5. Flow completion time for different round-trip times

Investigating the impact of round-trip time we can also find significant differences in the performance of DFCP and TCPs as shown in Figure 5. Specifically, in case of a web object there are several orders of magnitude between the download time of DFCP and TCP for increasing round-trip time (see Figure 5a). Considering the category of DVD it can be stated that, for low RTT values, the difference in download time is negligible, however, for high RTT values it gets more and more significant (see Figure 5b).

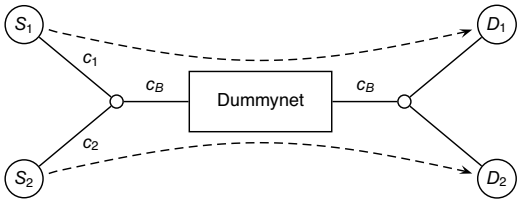
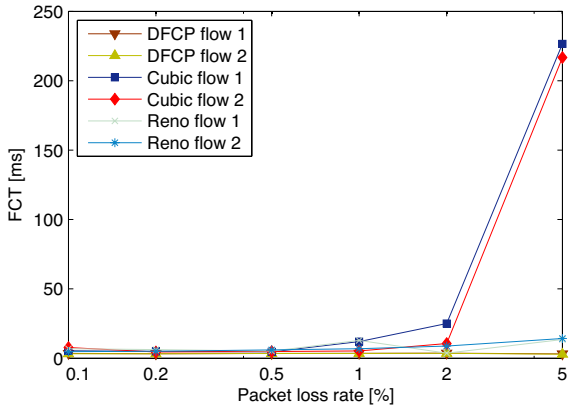
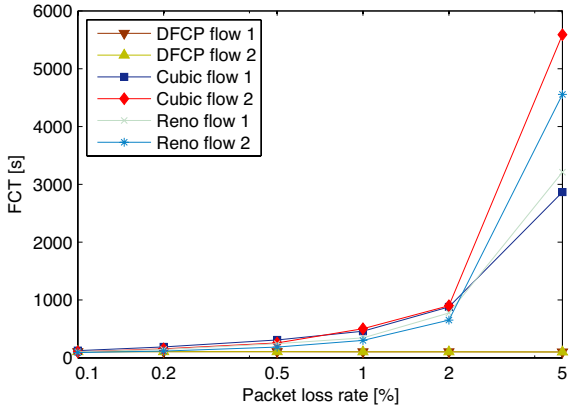


Fig. 6. Dumbbell topology with two source-destination pairs

We also performed experiments with two competing flows of the same type to study the fairness properties of the transport protocols. The second measurement setup can be seen in Figure 6 where all parameters were set similarly as described at the first dumbbell topology complemented by the condition $c_1 = c_2 = 1$ Gbps. The flows were started together and we used WFQ (Weighted Fair Queueing) as the scheduling method with equal weights (i.e. 50-50%) [26].



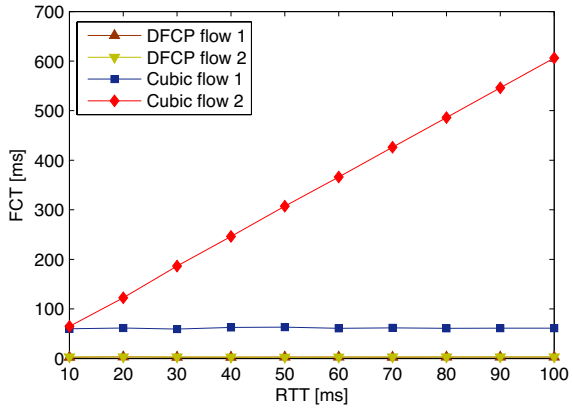
(a) Web object



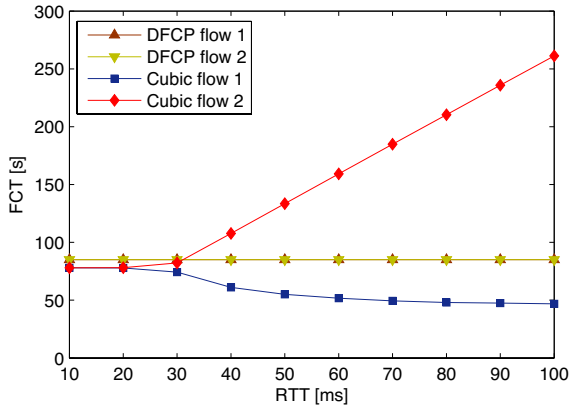
(b) DVD

Fig. 7. Flow completion time for two competing flows with equal loss rate

In the followings we highlight two practically interesting cases. On the one hand, the case when the packet loss rate is equal for each flow, and on the other hand, the case when the round-trip time has different values. The former is depicted in Figure 7 where the redundancy parameter of DFCP was adjusted to packet loss of 5%. We can observe that the difference between the flow completion times of the two flows of the same type for a web object is quite small. Therefore, each transport protocol behaves in a quasi-fair way, but perfect fairness is only achieved by DFCP (see Figure 7a). It is also important to note that the download time of DFCP is independent of the packet loss rate. Considering the category of DVD we can conclude that TCP variants become unfair at high loss rates in case of long data transfers (see Figure 7b).



(a) Web object



(b) DVD

Fig. 8. Flow completion time for two competing flows with the one having a fixed RTT of 10 ms and the other one having an RTT varied between 10 and 100 ms

Figure 8 shows the flow completion time for two competing DFCP and TCP Cubic flows where the first flow has a fixed RTT of 10 ms and the delay of the second flow is varied between 10 and 100 ms. We observed that the results for TCP Reno were quite the same as in case of TCP Cubic, hence only the latter was depicted. Looking at Figure 8a one can see that in case of a web object DFCP produces excellent results. It does not only provide 20 times faster download than TCP even in the worst case, but also achieves perfect fairness, thus both DFCP flows have nearly the same download time. If we transfer a full DVD, the two TCP flows behaves

in a fair way, but only for RTT values less than 20 ms (see Figure 8b). In contrast, DFCP flows attain equal download time in the whole range since DFCP protocol is insensitive to high RTTs compared to TCP. We note that the difference in the flow completion times of DFCP and TCP flows for RTT values less than 20 ms is due to the coding overhead of DFCP.

V. Conclusion

The important issue of efficiently transferring flows via the Internet regarding different transport protocols is addressed in this paper. A performance comparison study including recent TCP versions and also our newly designed and implemented DFCP protocol is given. The analysis focuses on the transient behavior and the flow completion time of short- and long-lived flow transfers. The results demonstrate the outstanding performance of DFCP compared to TCPs in all scenarios. The analysis also highlights the drawbacks of the currently used TCP versions and motivate research to develop transport protocols based on different concepts for Future Internet. Our future research will include the further development and evaluation of such concepts, especially the DFCP protocol.

Acknowledgment

This work was supported by the High Speed Networks Laboratory (HSNLab) at Budapest University of Technology and Economics, the OTKA-KTIA grant CNK77802 and the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 research project. The project has been received support from the European Union, co-financed by the European Social Fund. The authors would like to thank Szilárd Solymos for the prototype implementation.

References

[1] S. Floyd, K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.

[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] J. Widmer, R. Denda, M. Mauve, "A Survey on TCP-Friendly Congestion Control", *IEEE Network*, vol. 15, no. 3, pp. 28–37, 2001.

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

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

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

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

[9] 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

[10] 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*, New York, USA, 2013.

[11] A. Medina, M. Allman, S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet", *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 37–52, 2005.

[12] N. Dukkipati, N. McKeown, "Why Flow-Completion Time is the Right Metric for Congestion Control", *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 59–62, 2006.

[13] S. Molnár, Z. Móczár, "Three-dimensional Characterization of Internet Flows", *Proceedings of the 2011 IEEE International Conference on Communications*, pp. 1–6, Kyoto, Japan, 2011.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] L. Schrage, "A Proof of the Optimality of the Shortest Remaining Processing Time Discipline", *Operations Research*, vol. 16, no. 3, pp. 687–690, 1968.

[20] I. Rhee, L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", *Proceedings of the 3rd International Workshop on Protocols for Fast Long-Distance Networks*, pp. 1–6, Lyon, France, 2005.

[21] K. Fall, S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 3, pp. 5–21, 1996.

[22] S. Floyd, T. Henderson, A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", *RFC 3782, IETF*, 2004.

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

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

[25] HTTP Archive, http://www.httparchive.org/interesting.php

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