XCP vs. CUBIC with Quick-Start: Observations on Implicit vs. Explicit Feedback for Congestion Control

Michael Welzl
University of Oslo
Email: michawe@ifi.uio.no

Armin Abfalterer
University of Innsbruck
Email: a.abfalterer@gmail.com

Stein Gjessing
University of Oslo
Email: steing@ifi.uio.no

***Abstract*—Congestion control in transport protocols can be performed implicit when packets are lost, or explicit with feedback from the network. In this paper we compare some aspects of implicit and explicit congestion control using a variant of CUBIC with Quick-Start, called CUBIQ. We compare CUBIQ with regular TCP, XCP and CUBIC, and find that CUBIQ performs as well as XCP in most cases, and much better than XCP when XCP is hampered by small buffers in large BDP-networks. These results indicate that the feedback provided by Quick-Start may be enough for a sender when it needs to find the right transmission rate, also in the initial phase of a communication.**

I. Introduction and related work

Finding transport protocols that are fair and at the same time utilize as much of the available bandwidth as possible is an important challenge, in particular for connections with high bandwidth delay products (BDP). In order to find the best transmission rate, a sender can either rely on implicit feedback from the network or on explicit notifications from the routers.

In an effort to improve Implicit Congestion Control (ICC) algorithms, most approaches have been directed towards the Congestion Avoidance implementation of Standard TCP, whilst only few proposals have been concerned with the problem of Slow Start at the beginning of a TCP connection. For instance, HighSpeed TCP (HS-TCP) [1] addresses the long recovery time of Standard TCP upon packet loss and ScalableTCP [2] modifies the window function of TCP so that the recovery time after congestion is proportional to the RTT. The enhancement of TCP Westwood [3] reduces the recovery time after retransmission time-outs by setting the size of the congestion window accurately to the estimated value of the bandwidth.

Among the ICC proposals, CUBIC stands out from the other TCP modifications due to its good performance and wide deployment [4]. The approach of CUBIC is to replace the linear growth function of TCP by a cubic function that ensures high network utilization due to fast recovery time upon packet loss. Its algorithm is less aggressive than some other high-speed TCP variants (e.g., HS-TCP and Scalable TCP), and shows more backward compatibility by leaving room for Standard TCP flows [5].

The pursuit to perform stricter and more accurate congestion control has led to the emergence of several Explicit Congestion Control (ECC) algorithms. Source Quench (specified in [6]) and its successor Explicit Congestion Notification (ECN) [7] describe the first simple steps towards the active cooperation of routers by signaling network congestion to TCP senders. More recent proposals, such as MAXNET [8], Rate Control Protocol (RCP) [9] and Binary Marking Congestion Control (BMCC) [10], solely determine the appropriate sending rate based on explicit feedback from routers.

The eXplicit Control Protocol (XCP) [11] could be seen as the explicit counterpart to CUBIC, as it performs well and is one of the most referenced and well known ECC protocols. What seems to be very successful in XCP is the concept of decoupling utilization control from fairness control. For this reason, XCP routers maintain two control algorithms that are executed periodically and that constantly generate congestion window feedback for competing flows. The efficiency controller is responsible for specifying maximum use of the outgoing link whereas the fairness controller is responsible for fair distribution of throughput to flows sharing the link.

The literature exhibits few evaluations that draw a comparison between implicit and explicit congestion control mechanisms. Evaluation work in [10] [12] [13] [14] [15] [16] [17][18] compare a number of high-speed protocols, but mostly includes protocols with similar approaches. So the question whether implicit or explicit feedback is more effective in problematic environments presents an open field of research.

The aim of this paper is to shed some light on performance differences between implicit and explicit congestion control. Intuitively one should think that it is in the initial phase of a communication that the sender knows the least about the best sending rate, and in general where algorithms based on explicit information should have the greatest advantage. In the sequel, we will report from our experiments, that are constructed, among other things, to answer if this is indeed the case.

To conduct our experiments we need good protocols that are representative for each of the two directions. However, new protocols are developed all the time, so this is a moving target and not an easy choice. Although RCP and BMCC have been reported to perform better than XCP [18][10], we will, because XCP is probably the most well known explicit feedback protocol, use the XCP protocol as the representative for ECC protocols in this paper. We chose CUBIC as the representative for ICC protocols. These two algorithms have been chosen since each of them represents a strong competitor in its direction, and thus, they allow a good estimation of what today's implicit and explicit feedback mechanisms are able to accomplish. Since we are particularly interested in the initial phase of a communication, and fear that regular CUBIC will get feedback from the network too slowly, we have extended CUBIC with Quick-Start, and called the resulting protocol CUBIQ (more details are given in the next section). We will also run our experiments with a version of regular TCP that uses Quick-Start.

The rest of the paper is organized as follws: In section II we present our simulation environment, and in section III the results are presented and discussed. Finally, in section IV we conclude.

II. The Experimental setup

The addition of Quick-Start to CUBIC has been researched in [19]. It has been shown that in CUBIC, an initial large congestion window does not affect the aggregate performance of long-lived flows. This implies that the most efficient version of CUBIC with Quick-Start can be defined based on its performance in the initial phase. Neither Quick-Start with the adaption of a Slow Start threshold, nor Quick-Start with Limited Slow Start performs better than CUBIC with regular Quick-Start. Hence, in this paper we use CUBIC with regular Quick-Start, and call this combination CUBIQ. In addition, the request rate of 100% is the best choice to quickly exploit under-utilized network paths [19].

In this paper we compare CUBIQ to CUBIC and XCP in three different scenarios. Several more experiments are reported in [19]. The chosen scenarios adhere closely to the proposals in "Towards a Common TCP Evaluation Suite" [20]. For the sake of completeness, most of our experimental results also show performance compared to "regular TCP". The TCP variant used for this purpose is TCP NewReno in combination with Quick-Start and the adaptation of a Slow Start threshold (denoted TCP ADA), which is the best way of using Quick-Start for NewReno [19]. All experiments are run using ns-2.

Our experiments are based on a single bottleneck "dumb-bell" topology with one central network link connecting two routers, and several edge links connected to each of these routers. Source and destination will be in each end of this topology. As link delays and link capacities differ in most of the scenarios, these values are explicitly given for each experiment. However, when nothing else is stated, the edge links have the same propagation latency as the central link. Hence, if e.g. the propagation delay for the central link is 10 ms, the end to end propagation delay is 30 ms and the propagation RTT is 60 ms. In order for the central link to always be the bottleneck, and without loss of generality, edge links have ten times the capacity of the central link. Traffic sources are long-lived FTP flows, mostly starting at simulation time 0 and lasting until the end. To ensure that flows reach a consistent behavior, simulation times are very long (600 seconds if nothing else is stated). Furthermore, to

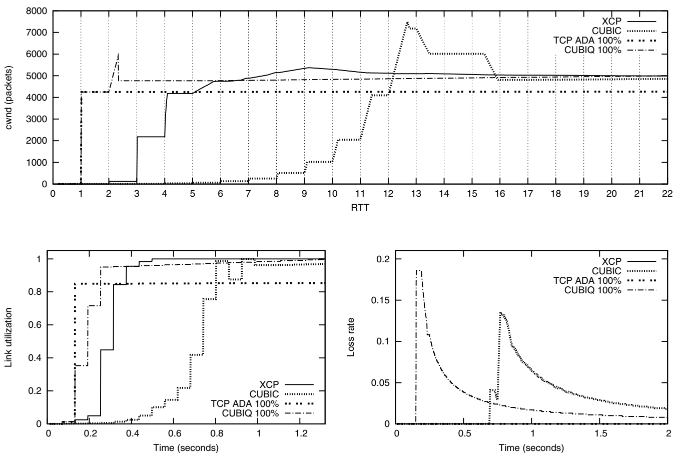

Fig. 1. Performance of all protocols under test in the Startup phase of the Single Flow scenario. 1 Gbps bottleneck capacity and 60 ms RTT. Top: Congestion window growth per RTT. Bottom: Link utilization and loss rate. Notice that there are only two visible curves in the loss rate graph, since XCP and TCP ADA have no loss.

avoid global network synchronization among competing flows, most of the scenarios consist of cross-traffic and reverse-path traffic that are generated according to Poisson processes. In Quick-Start, the request rates of flows are given as percentages of the bottleneck link capacity for the sake of convenience. To avoid misunderstandings of presented results, the reader should know that if a flow requests full sending speed (rate request is 100%), the maximum achievable sending rate is 85% of the bottleneck link capacity, because routers reduce requests that exceed the threshold in order to reserve resources for succeeding flows. More discussion of parameter settings can be found in [19].

Unless otherwise specified, the RED queuing discipline is used. Furthermore, the default queue buffer sizes are set differently for XCP and the other protocols under test. The buffer size of XCP queues is set to the BDP related to the end-to-end round-trip delay of the network path, whereas regular TCP and CUBIC (and CUBIQ) operate with buffer sizes set to the BDP related to single link delays. The reason for these different buffer sizes is to maintain the simulation settings of related work.

III. The experiments

A. *Single Flow, Startup*

We first show the performance of a single flow scenario to provide an overall picture. The capacity of the bottleneck link is set to 1 Gbps and the RTT is 60 ms.

Figure 1 shows the strength of Quick-Start in comparison to Slow Start. The upper plot shows that TCP ADA and CUBIQ get a congestion window of 4000 packets due to their Quick-Start requests of 100% of the link capacity. As seen in the lower left plot, for both these two protocols, this results in high link utilizations after only two to three RTTs (less than 0.2 seconds). Also XCP achieves a large congestion window

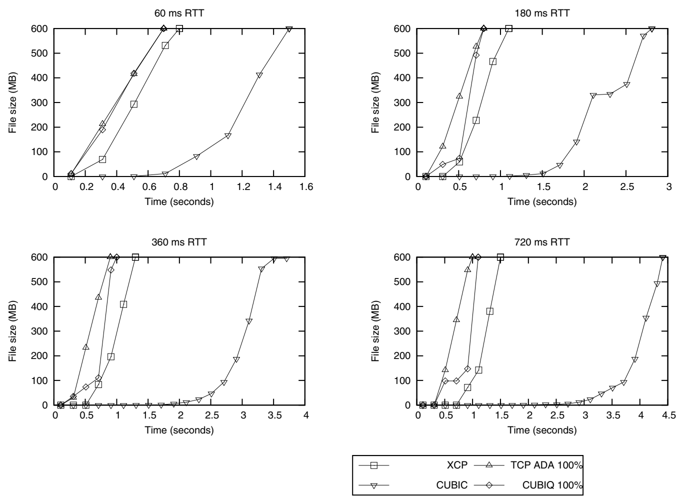

Fig. 2. Performance of all protocols under test in the Single Flow, Large file (600 MB) scenario. Four plots with different RTTs. Each plot shows number of Mega bytes transfered as a function of time. Queue type is DropTail.

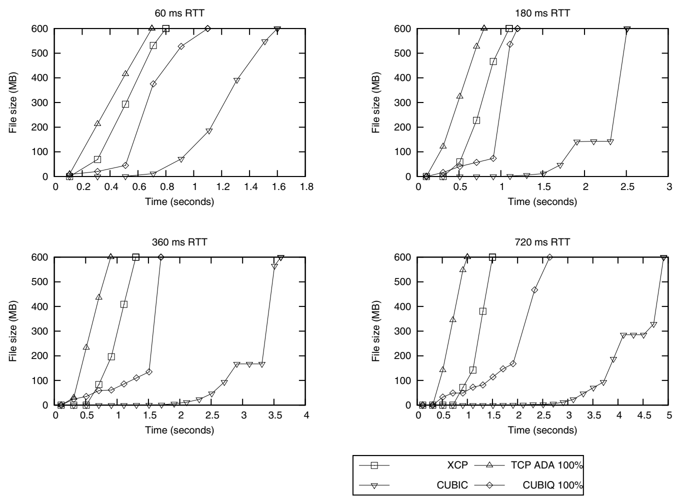

Fig. 3. Performance of all protocols under test in the Single Flow, Large file (600 MB) scenario. Four plots with different RTTs. Each plot shows number of Mega bytes transfered as a function of time. Queue type is RED.

after 4 RTTs and, thus, it is barely slower in the initial phase. In contrast to this, CUBIC takes more than 11 RTTs to get to the same size and consequently takes much longer to fully utilize the network (more than 0.8 seconds). In addition, the lower right plot shows that XCP and TCP ADA have no loss due to network probing. The loss rates of CUBIC and CUBIQ are almost identical even though they are time-displaced due to the Quick-Start request.

B. Single Flow, Large file

The topology and simulation setup of the Single Flow scenario is re-used with a 10 Gbps bottleneck link, to measure the time required to transfer a 600 MB file. The simulation is run for all protocols with two different queue types, RED and DropTail, in order to evaluate the impact of different queue disciplines on Quick-Start. In addition to the round-trip-time of 60 ms, higher network delays are also used in order to study the impact of different propagation delays (RTTs are 60, 180, 360 and 720 ms) on the performance of each protocol. These simulations do not include cross-traffic in order to ensure ideal network conditions and to focus on the maximal capability of each protocol.

Figures 2 and 3 illustrate the performance of each protocol with DropTail and RED queues. It is well known that DropTail queues start to drop packets only when queues are completely full, whereas RED queues randomly drop packets to avoid global synchronization and unfair queue occupation among competing flows. All flows using CUBIC or CUBIQ get a slight profit from the slow reaction caused by the Drop Tail queues. The reason is that early dropped packets cause a momentary decrease in the sending rate. It is notable that TCP ADA does not show differences with the queue types, as network probing after Quick-Start is disabled. It is also notable that all protocols have delayed completion times as the network propagation delay is increased, however, most noticeable for CUBIC. The results also show that CUBIQ significantly outperforms its counterpart CUBIC using Slow Start independent of the propagation delay and the queue discipline used. Using DropTail queues, CUBIQ even performs better than XCP. TCP ADA shows best overall performance as its Quick-Start request and the following threshold adaptation creates no packet losses.

C. More Flows

The Single Flow scenario of the previous experiments is enhanced with a second parallel and competing flow to determine how quickly existing flows make room for new ones. The first flow transmits an “infinitely” large file, and the second flow starts later, at least so late that the first one has finished its Slow Start phase (if used). The simulation time is 600 seconds. The ingress and egress links are different for the two flows, only the central bottleneck link with a 1 Gbps capacity and 10 ms one way propagation delay is shared. The length of the ingress and egress links are varied (but for one flow the ingress and the egress links have the same length), so that the existing and newly arriving flow have RTTs of respectively (80 ms, 80 ms), (30 ms, 120 ms) and (120 ms, 30 ms). Throughout the experiments there is a 10% reverse traffic stream. The metric of these experiments is the time until $1460 * 10^n$ bytes of the second flow are received, for $n = 1, 2, ..., 7$.

On the left side, Figures 4, 5 and 6 show the completion times of the second flow for each pair of RTTs and for each protocol under test. Notice that the x-axis is the n just defined. On the right side, the figures show the loss rate of the new flow at its beginning.

In Figure 4, where RTTs are (80 ms, 80 ms), all protocols achieve $n = 7$, that corresponds to a file size of 13.6 GB. For $n \leq 5$, the difference in completion times are in the range of a few seconds, as already demonstrated previously in the

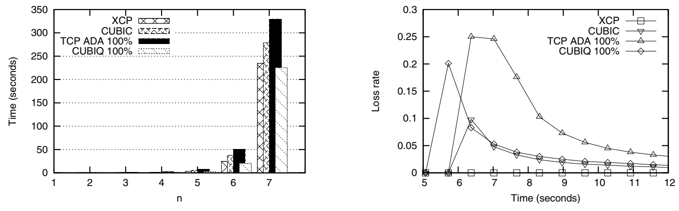

Fig. 4. Convergence times and loss rate of the newly arrived flow for each protocol in the More Flows scenario. Both flows have 80 ms RTT. See the description of the More Flows scenario for a detailed understanding of the two plots.

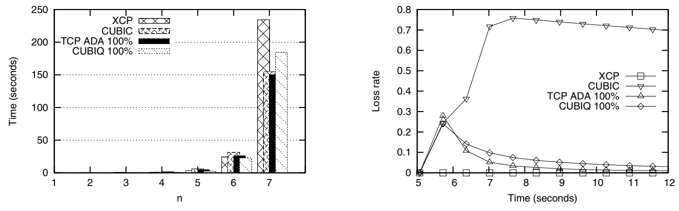

Fig. 6. Convergence times and loss rate of the newly arrived flow for each protocol in the More Flows scenario. First flow has RTT of 120 ms and second has RTT of 30 ms. See the description of the More Flows scenario for a detailed understanding of the two plots.

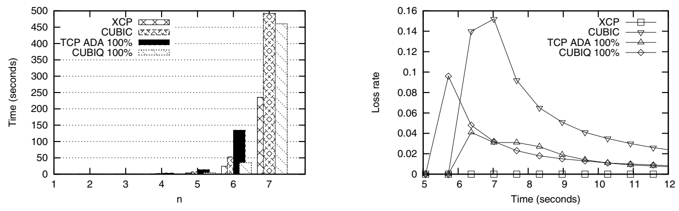

Fig. 5. Convergence times and loss rate of the newly arrived flow for each protocol in the More Flows scenario. First flow has RTT of 30 ms and second has RTT of 120 ms. For n=7, TCP ADA does not complete within 600 seconds. See the description of the More Flows scenario for a more detailed understanding of the two plots.

experiment in Section III-B. For $n \geq 6$, the differences are easily observable, revealing that the TCP ADA flows need the most time to complete. The measurements show that regular TCP reluctantly makes room for the newly arrived flow leading to high packet loss and unequal sending rates. In fact, TCP ADA shares the bottleneck link with averaged sending rates of (502 Mbps, 375 Mbps) among the flows whereas CUBIQ and XCP achieve quite similar sending rates for the two flows (about 500 Mbps each). CUBIQ performs better than its counterpart CUBIC, although the protocol causes more drops at the beginning. However, the flow is able to benefit from the approved congestion window as it completes faster.

In Figure 5, with RTTs of (30 ms, 120 ms), TCP ADA does not complete for $n = 7$. Furthermore, the difference between XCP and CUBIC (and CUBIQ) is significant. Both CUBIC variants show a drastic performance drop after the first quarter of the simulation time. The reason is a series of packet drops that occasionally leads to a minimum sending rate of 10 Mbps for flow 2. In this period (not shown in the figure), flow 1 dominates the network and does not make room for flow 2 for a long time. Flow 2 recovers after 400 seconds and achieves half of the bandwidth. The average sending rate for both flows is (706 Mbps, 292 Mbps). As in the previous experiment, XCP flows share the network capacity equally and show no packet loss.

In Figure 6, with RTTs of (120 ms, 30 ms), CUBIC and TCP ADA both show fast convergence times for flow 2 but with significant suppression of flow 1. The average sending rates of both flows give information about the intensity of the suppression: (110 Mbps, 862 Mbps) for TCP ADA and (265 Mbps, 709 Mbps) for CUBIC. Again, the two XCP flows show uniform distribution and the same convergence times as they did with previous RTTs. CUBIQ derives a great benefit from Quick-Start as the sending rate distribution is (413 Mbps, 583 Mbps). The reason for the difference in performance is the aggression of Slow Start. With CUBIC, the Slow Start of flow 2 causes high network perturbation, that makes flow 1 totally drop its sending rate. The number of drops of flow 1 with CUBIC in the first 10 seconds is about 600,000 packets, whereas flow 1 with CUBIQ causes only about 10,000 packets to be dropped.

In summary it can be said that competing XCP flows equally share the bottleneck link capacity irrespective of the RTT. XCP operates without packet loss even in experiments where a flow with small RTTs joins the network. The CUBIC variants show good convergence with the same RTTs for both flows but they show a weakness if an existing flow with a small RTT dominates the network. The enhancement of CUBIC with Quick-Start shows less packet loss due to moderate network perturbation of newly arrived flows. Especially, joining flows with small RTTs (that tends to control the network) are prevented from overshooting the network capacity.

D. Impact of the Queue Size

This scenario investigates the impact of the queue sizes. Again, the bottleneck link capacity is 1 Gbps and there are two parallel flows, both starting at simulation time 0. The RTT of both flows varies from 60 ms to 480 ms increasing by a factor of 2. Only the plots for the 60 ms case are shown in this paper. In order to study the impact of different queue sizes, the buffer length of the bottleneck queue is varied from 20% to 5% of the BDP in steps of 5%.

The top four graphs in Figure 7 show the oscillating behaviour of XCP if the buffer size is less than or equal to 20% of the BDP and the flow has a RTT of 60 ms. It can be seen that the throughput gets worse as the buffer size is decreased. The reason for the poor performance is the assignment of a large congestion window to both flows by the efficiency controller. As both flows receive the permission to increase their sending rate, the bottleneck buffer overflows, forcing the controller to reset the window size to the initial value of 1. When congestion is over, the same procedure recommences by re-assigning a

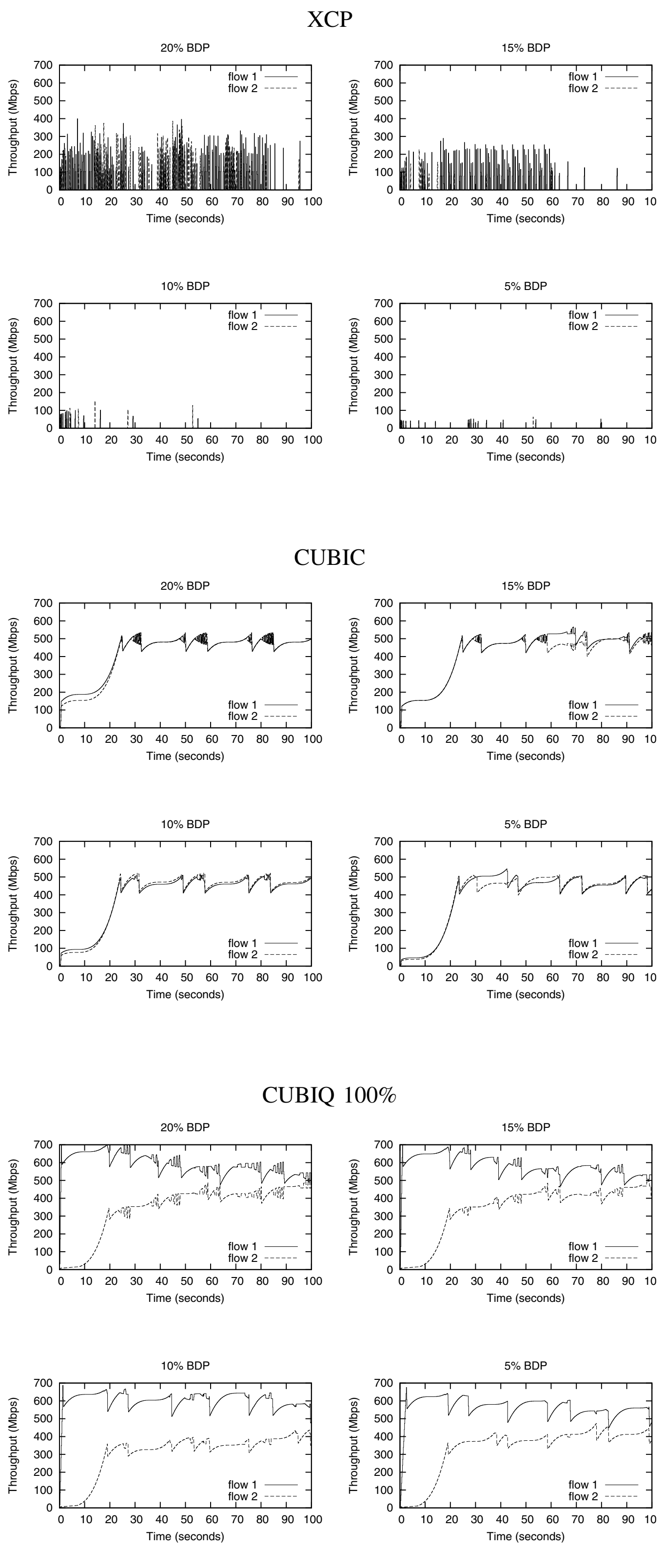

Fig. 7. Throughput of two flows, both starting at time 0 and with 60 ms RTTs for the XCP (top), CUBIC (middle) and CUBIQ (bottom) protocols. Different small buffer sizes of 20%, 15%, 10% and 5% of the BDP.

large congestion window. The fluctuation in the congestion window causes both flows to perform badly; for instance, if the buffer size is fixed to 20% of the BDP, the flows achieve an average sending rate of about 40 Mbps each. In contrast, CUBIC and CUBIQ still perform well with small queues. In the same experiment the CUBIC flows achieve average sending rates of around 415 Mbps, whereas the CUBIQ flows achieve about 460 Mbps. The difference between CUBIC and CUBIQ in the initial phase is due to Quick-Start, as the first flow gets its requested rate of 100% whereas the request of the second flow is rejected. Nevertheless, we found that due to Quick-Start the link utilization in the first 20% of the simulation is increased, whereas CUBIC shows very long start-up times.

With increasing propagation delay (120 ms, 240 ms and 480 ms, not shown here but in [19]), a significant performance drop is observed for all protocols. CUBIC's and CUBIQ's delay in link utilization increases. In addition, the network utilization for both CUBIC protocols declines. For instance, in the experiments with the lowest queue size and highest propagation delay (5%, 480 ms) both protocols utilize only half of the network capacity. However, the situation becomes worst for XCP as the sending rate of both flows seems to converges to zero with increasing propagation delay. In summary it can be said that XCP shows heavy performance losses in association with low buffer sizes. The results are unacceptable sending rates. CUBIC and CUBIQ also show difficulties, but with respect to the harsh network conditions both achieve acceptable results.

IV. Summary and Conclusions

In this paper, we have shown a number of experiments that reflect the problems of Standard TCP in today's high-speed networks with connections with high Bandwidth Delay Products. We have compared Implicit Congestion Control and Explicit Congestion Control algorithms. The ICC algorithms used are CUBIC, CUBIC with Quick-Start (called CUBIQ) and standard TCP NewReno with Quick-Start and the adaption of a Slow Start threshold (called TCP ADA). As a representative for the ECC algorithms, XCP has been used.

Our experiments show that the properties of XCP are fast convergence, low loss, high link utilization and high fairness irrespective of network delays and capacities. XCP performs well both with DropTail and RED queues. However, XCP makes demands to the network in order to work properly. With small bottleneck queue sizes (beneath 20% of the BDP) the advantages of XCP start to decrease. Although the protocol operates without losses under favorable conditions, low buffer sizes causes XCP to drop packets. When the RTT is high (e.g. 480 ms) and the buffer size is low (e.g. 5% of the BDP), the performance of XCP may deteriorate to unacceptably low transmission rates.

CUBIC also performs well in large BDP networks, in terms of high network utilization, proper convergence times and increased fairness compared to Standard TCP. The protocol works well under a wide range of network conditions and shows a solid performance with low buffer sizes. In an environment with more competing flows having high differences in RTTs, CUBIC, however, exhibits unfair behaviour since flows with low delays tend to dominate the network and hesitantly make room for other transfers. The congestion window increase of CUBIC at the beginning of a connection by the use of Slow Start causes network perturbations and sluggish utilization of bandwidth.

In comparison, CUBIQ, the combination of CUBIC and Quick-Start, appears to be a promising approach, as the early congestion window growth ensures high sending rates within a few RTTs and less perturbation by fixing the plateau more appropriately. Unsurprisingly, the results also show that XCP as well as CUBIC (and CUBIQ) outperform Standard TCP in high bandwidth and large delay networks, especially with flows that last longer than Slow Start.

The main conclusion of this paper is that it may be the case that, with the help of Quick-Start, congestion control protocols based on implicit feedback from the network can perform comparable to congestion control protocols based on explicit feedback from the network, also when short files are transmitted in high BDP connections. We even saw that the chosen ECC protocol, XCP, does not perform well in networks with small buffers. As stated in the introduction, there are ECC protocols that are reported to perform better than XCP, and it must be left to future work to compare these protocols to ICC protocols, in particular in networks with small buffers. It must also be left to future work to see how realistic it is to implement CUBIQ (CUBIC with Quick-Start) in today‘s Internet.

References

[1] S. Floyd, “HighSpeed TCP for Large Congestion Windows,” RFC 3649 (Experimental), Dec. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3649.txt

[2] T. Kelly, “Scalable tcp: improving performance in highspeed wide area networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 83–91, 2003.

[3] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, “TCP Westwood: Bandwidth estimation for enhanced transport over wireless links,” in *MobiCom ’01: Proceedings of the 7th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2001, pp. 287–297.

[4] S. Ha, I. Rhee, and L. Xu, “CUBIC: a new TCP-friendly high-speed tcp variant,” *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.

[5] E. He, P. V.-B. Primet, M. Welzl, M. Goutelle, Y. Gu, S. Hegde, R. Kettimuthu, J. Leigh, C. Xiong, and M. M. Yousaf, “A survey of transport protocols other than standard TCP,” Open Grid Forum, Tech. Rep. OGF 55, November 2005.

[6] J. Nagle, “Congestion control in IP/TCP internetworks,” RFC 896, 1984. [Online]. Available: http://www.ietf.org/rfc/rfc896.txt

[7] K. Ramakrishnan and S. Floyd, “A Proposal to add Explicit Congestion Notification (ECN) to IP,” RFC 2481 (Experimental), 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2481.txt

[8] B. P. Wydrowski, L. L. H. Andrew, and I. M. Y. Mareels, “MaxNet: Faster flow control convergence,” in *Proceedings Networking 2004, 2004, Springer Lecture Notes in Computer Science LNCS*, 2004.

[9] I. Dukkipati, M. Kobayashi, R. Zhang-shen, and N. Mckeown, “Processor sharing flows in the Internet,” in *IWQoS*, 2005.

[10] I. Qazi, T. Znati, and L. Andrew, “Congestion Control using Efficient Explicit Feedback,” in *INFOCOM 2009*, 2009.

[11] D. Katabi, M. Handley, and C. Rohrs, “Congestion control for high bandwidth-delay product networks,” in *SIGCOMM ’02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2002, pp. 89–102.

[12] K. Tokuda, G. Hasegawa, and M. Murata, “Performance Analysis of HighSpeed TCP and its Improvement for High Throughput and Fairness against TCP Reno Connections,” in *High-Speed Networking Workshop (HSN)*, 2003.

[13] M. C. Weigle, P. Sharma, and I. Jesse Freeman, “Performance of Competing High-Speed TCP Flows,” in *Proceedings of NETWORKING*, Coimbra, Portugal, 2006, pp. 476–487.

[14] Y.-T. Li, D. Leith, and R. N. Shorten, “Experimental evaluation of TCP protocols for high-speed networks,” *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, 2007.

[15] D. J. Leith, L. L. H. Andrew, T. Quetchenbach, R. N. Shorten, and K. Lavi, “Experimental Evaluation of Delay/Loss-based TCP Congestion Control Algorithms,” in *Protocols for Fast, Long Distance Networks (PFLDnet)*, 2008.

[16] Y. Gu, X. Hong, and R. L. Grossman, “Experiences in Design and Implementation of a High Performance Transport Protocol,” in *Proceedings of the ACM/IEEE SC2004 Conference Supercomputing*, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1392952

[17] H. Bullot, R. L. Cottrell, and R. Hughes-Jones, “Evaluation of Advanced TCP Stacks on Fast Long- Distance Production Networks,” in *Protocols for Fast, Long Distance Networks (PFLDnet)*, 2004.

[18] M. Proebster, M. Scharf, and S. Hauger, “Performance comparison of router assisted congestion control protocols: XCP vs. RCP,” in *Simutools ’09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 1–8.

[19] A. Abfalterer, “A Critical Evaluation of Explicit Feedback for Congestion Control, Master Thesis, University of Innsbruck, 2009.” [Online]. Available: http://heim.ifi.uio.no/michawe/teaching/dipls/abfalterer-thesis.pdf

[20] L. Andrew, C. Marcondes, S. Floyd, L. Dunn, R. Guillier, W. Gang, L. Eggert, S. Ha, and I. Rhee, “Towards a Common TCP Evaluation Suite,” in *PFLDnet 2008*, 2008.