Performance Comparison of Router Assisted Congestion Control Protocols: XCP vs. RCP

Magnus Proebster, Michael Scharf, and Simon Hauger
Institute of Communication Networks and Computer Engineering
University of Stuttgart, Germany
E-mail: {magnus.proebster, michael.scharf, simon.hauger}@ikr.uni-stuttgart.de

ABSTRACT

In the current Internet, network overload is prevented by the congestion control of the Transmission Control Protocol (TCP). The traditional TCP congestion control is an end-to-end mechanism that suffers from some inherent shortcomings. A design alternative for the Future Internet is to use more feedback from the routers. Such router-assisted congestion control schemes can achieve a more efficient utilization of network resources and better fairness, even in environments with large bandwidth-delay products. Two promising proposals are the eXplicit Control Protocol (XCP) and the Rate Control Protocol (RCP).

This paper evaluates the performance of XCP and RCP and compares them with the existing TCP congestion control. In order to verify previous work, a new simulation tool has been developed independently of the existing ns-2 code basis. This simulator is used to study the basic behavior of the algorithms and to analyze several degrees of freedom in the protocol design. Furthermore, the performance of the different approaches is compared using realistic Internet traffic scenarios. The results show that indeed both XCP and RCP efficiently utilize the link capacity without requiring packet loss. Unlike XCP, RCP improves the reactivity of data transfers by reducing the flow completion time. These results confirm previously published results and show that in particular RCP has the potential to replace TCP congestion control in the Future Internet.

1. INTRODUCTION

The Transmission Control Protocol (TCP) is the standard transport protocol for reliable, elastic traffic in the Internet. One major TCP function is the congestion control [1]. TCP's traditional loss-based congestion control algorithms have inherent shortcomings, in particular if the bandwidth-delay product of the path is large. In general, for any end-to-end congestion control it is challenging to efficiently utilize the bandwidth, to minimize the delay, and to maintain fairness at the same time [2]. The root cause for this problem is that end-systems lack precise information about the current path characteristics.

New router-assisted congestion control schemes address this issue by using additional feedback from the routers along a path, which can monitor the actual link utilization more accurately and which can detect congestion faster. However, in order to convey this knowledge to the end-systems, additional signaling mechanisms are needed. This router assistance could be realized by TCP extensions such as the Explicit Congestion Notification (ECN) [3] or Quick-Start TCP [4] that use sporadic coarse-grained feedback from routers in certain situations. An alternative, radically new approach is to use fine-grained per-packet feedback, which allows to achieve both efficiency and fairness even in TCP-unfriendly environments. Recently, several approaches have been proposed for such a congestion control in the Future Internet. The two most elaborated mechanisms are the eXplicit Control Protocol (XCP) [5] and the Rate Control Protocol (RCP) [6].

Both XCP and RCP claim to significantly outperform the existing TCP congestion control. These statements are backed by simulation studies as well as some testbed experiments [5, 6]. However, most published studies rely on the implementation in the ns-2 simulator only. This paper tries to verify the published performance results using the newest protocol specifications (e. g., [7]). We have implemented both XCP and RCP from scratch in a new simulation tool, which also allows a comparison with state-of-the-art TCP stacks. With this simulator, we study these router-assisted congestion control protocols both in simple and complex scenarios and quantify the potential improvement compared to standard TCP. In addition to confirming previous studies, our simulation results also reveal new effects that have not been reported so far.

The rest of this paper is structured as follows: Section 2 gives an overview of router-assisted congestion control mechanisms and introduces XCP and RCP. In Section 3, our new simulation tool and our evaluation methodology is presented. Section 4 verifies the basic function of the different protocols and discusses several design and implementation alternatives. Then, in Section 5, the performance of TCP, XCP and RCP is compared for realistic Internet traffic scenarios. Finally, Section 6 concludes this paper.

2. ROUTER ASSISTED CONGESTION CONTROL

Router-assisted congestion control schemes have recently been discussed as an approach to replace the TCP conges-

QoSim 2009, March 6, 2009, Rome, Italy

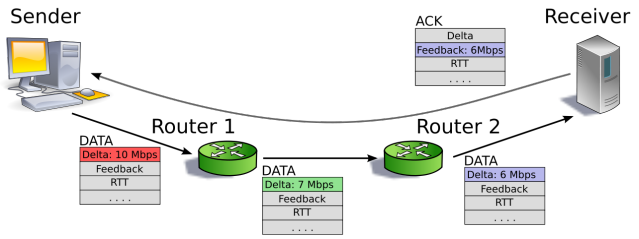

Figure 1: Principal function of an explicit congestion control with dedicated packet headers changed on the path

tion control that does not scale well for large distance, high capacity links. In the following, we give a short overview of the basic principle of these schemes. Afterwards, we detail on two of the most popular protocols, namely XCP [7] and RCP [6], which are investigated in this paper.

2.1 Overview

It is difficult to efficiently use large distance links with a high capacity by only taking into account lost packets (as in TCP) [1] or single header bits (as with ECN) [3]. On the one hand, packet loss is not a very reliable congestion indication as packets can get dropped for reasons other than congestion. Packet loss, or a single notification bit, only give binary and imprecise congestion indication. On the other hand, the absence of packet loss or ECN marks does not give any information whether an increase of the data rate is possible. These inherent shortcomings of today's end-to-end congestion control are addressed by recent research proposals like Quick-Start [4], XCP [5], RCP [6], or JetMax [8]. All these protocols use explicit, multi-byte congestion notifications from the routers on the path to improve the efficiency of the congestion control.

The basic idea of all these schemes is to place extra header fields between the IP header and the transport protocol header, which contain some form of rate request as well as other connection information from the sending end-system. These header fields are processed by all routers on the path. Depending on their egress load, the routers either accept the requested rate or decrease it, without keeping per-flow state information. The receiving end-system receives a rate that is accepted by all routers on the path and echoes this rate to the sender in another special header field. So, the sender can efficiently use the available bandwidth by always adjusting its sending rate according to the accepted rate, e. g. by resizing the congestion window in a window-based scheme.

In the following, we give a short overview of the protocol functions of XCP and RCP.

2.2 eXplicit Control Protocol (XCP)

XCP is one of the earliest and most well-known router-assisted congestion control approaches. A recent protocol specifications can be found in [7]. The objective of XCP is on the one hand to efficiently use the available link capacity, and on the other hand to distribute the available bandwith fairly between concurrent connections. These objectives are realized in a decoupled way on the routers by a so called *efficiency controller* (EC) and a *fairness controller* (FC), respectively. While the EC periodically computes an aggregate feedback, denoting the required change of the congestion window of all flows, the FC distributes this aggregate feedback fairly between packets, without keeping any per-flow state.

For its functioning, XCP requires a header in each packet which contains an *RTT* field, a *Throughput* field, a *Delta* field and a *Feedback* field. The *RTT* field contains the round-trip time (RTT) estimated by the end-system. The *Throughput* field denotes a measure for the actual injected data rate and contains the average inter-packet time. The *Delta*-field is initialised by the sending end-system with the desired change of the current sending rate, which is expressed on a per-packet basis and is calculated as follows:

$$\frac{desired\_rate - actual\_rate}{packets\_per\_rtt}$$

This field is decreased by the routers according to their outgoing load. The *Feedback*-field is used to echo the finally granted rate change back to the sending end-system.

The XCP routers measure the traversing traffic by keeping per-link statistics about the sum of outgoing traffic and the average RTT of all traversing flows. With these statistics the EC calculates the aggregate feedback in a periodic control interval of the order of the observed average RTT. At the beginning of each control interval, the aggregate feedback is calculated from the following equation:

$$F = \alpha\,(C - y(t)) - \beta\frac{q(t)}{d} \tag{1}$$

C is the capacity of the outgoing link, $y(t)$ the rate of its outgoing traffic, $q(t)$ the persistent queue during the previous control interval and d the average RTT. The first summand weighted by α shall adapt the traffic to the link capacity and the second summand weighted by β drains any standing queue.

The resulting aggregate feedback F can be positive or negative and is distributed among the traversing flows by the FC. Positive feedback is distributed equally among all flows and negative feedback is distributed in proportion to their current throughput. This implements the Additive Increase-Multiplicative Decrease (AIMD)-law which is taken from TCP and ensures the convergence to a fair allocation of bandwidth among all flows.

Another mechanism called *Traffic Shuffling* allows newly arriving flows to obtain bandwidth in a fully loaded system. If the aggregate feedback from equation 1 is approximately zero, 10% of the link capacity is redistributed during the next control interval.

2.3 Rate Control Protocol (RCP)

The design objective of RCP is to achieve very short flow completion times, i. e., that all flows are completed as fast as possible. In order to accomplish this goal RCP approximates the *processor sharing* scheduling algorithm when distributing the link capacity among all flows.

Unlike XCP, which signals rate changes, RCP always reports the target sending rate to the end-systems. Furthermore, the same rate is signalled to all flows bottlenecked at the respective link. This makes RCP inherently fair and simplifies the underlying algorithms. The proposed packet header only contains the *RTT* and *Feedback* fields as well as a *Rate* field with the allowed rate at the most congested link on the path, instead of the *Delta* field in XCP.

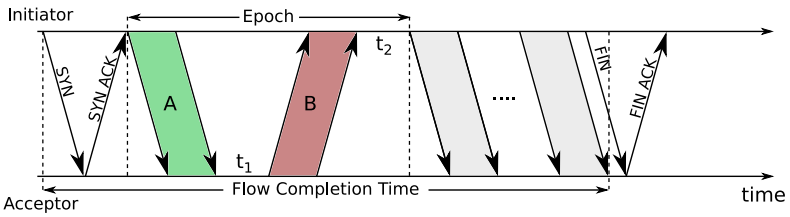

Figure 2: Visualization of the ATBT-model with one or several epochs; definition of the Flow-Completion-Time as the period from the initial handshake to the reception of the last packet

In the routers, the common maximum allowable rate is updated periodically with an interval corresponding to the average RTT of traversing flows. The calculation is derived from the following iterative equation:

$$R(t) = R(t-d) + \frac{\left(\alpha\left(C - y(t)\right) - \beta\frac{q(t)}{d}\right)}{\hat{N}(t)} \qquad (2)$$

where $R(t)$ is the common feedback rate, d is the average RTT and C is the capacity of the outgoing link. $y(t)$ is the aggregate outgoing traffic which was measured during the last control interval, $q(t)$ is the current queue occupation, and $\hat{N}(t)$ is the estimated number of active flows.

In RCP, the number of flows is estimated as $\hat{N}(t) = \frac{C}{R(t-d)}$, assuming that the traversing flows occupy the link with the signalled data rate. This estimation does not need to represent the exact number of flows but rather an effective number of flows in order to ensure the max-min fairness of the protocol. This is explained in more detail in [6]. With this estimation the feedback rate is obtained as follows:

$$R(t) = R(t-d)\left(1 + \frac{\left(\alpha\left(C - y(t)\right) - \beta\frac{q(t)}{d}\right)}{C}\right) \qquad (3)$$

2.4 Comparison

All router-assisted congestion control schemes need additional packet processing in the end-systems and also in all routers. These changes result in deployment challenges. However, there are proof-of-concept implementations that show that router-assisted congestion control can indeed be implemented at high link speeds [9, 10, 11]. The results show that XCP needs very complex processing steps in routers, whereas Quick-Start and RCP have lower processing requirements. Furthermore, in particular the performance of XCP capable routers might be limited due to synchronization issues [10]. RCP does not have this drawback.

Furthermore, several simulation studies have been performed by the authors of the respective protocols [5, 6]. In general, these simulations are based on the ns-2 simulator and its TCP implementation.

3. SIMULATION METHODOLOGY

We developed an independent simulation environment in order to verify the published results and to compare XCP and RCP in scenarios that have not been studied so far. In the following sections, we briefly present our network and traffic models.

3.1 Simulation Tool

All the functions related to congestion control in the XCP and RCP protocols were implemented in a network simulator that uses the IKR simulation library [12]. It is a C++ class library with a compact and clear design. It offers several network components which can be enhanced by the required functionality and connected arbitrarily through the underlying message-port concept. Simulations are conducted on an event-based calendar and, in our case, the results are extracted by post-processing from recorded traces. Differences between ns-2 and the simulation library are elaborated in [13].

Our TCP implementation is based on the network simulation cradle (NSC) [14] which has been adapted accordingly [15]. The NSC allows us to simulate a whole Linux stack including recent congestion control like CUBIC [16] as well as legacy mechanisms like the Reno algorithms. For our simulations, we use the Linux kernel version 2.6.18 with its default configuration. This means that Selective Acknowledgement (SACK) and window scaling are activated. By setting the sending and receiving buffer space to a large value (8 MiB), we effectively switch off the flow control because we are only interested in congestion control.

XCP and RCP are implemented in combination with an idealized transport protocol and its own signalling state machine. It implements connection setup with a three-way-handshake and a simplified two-way connection release. This abstract modeling ensures that congestion control is studied without interactions of other transport control mechanisms.

For XCP, we developed two implementations. One with the ordinary window-based behaviour derived from TCP and another rate-based variant which controls the inter-packet time to adjust the signalled rate change. The different behaviour of these implementations is discussed in Chapter 4. Apart from that, both implementations follow the specifications in [7].

Our RCP implementation realizes the algorithms described in [6]. It differs from the ns-2 implementation of RCP. There, so-called REF packets are used to signal the allowed sending rate with a constant delay, when the time interval between two consecutive packets becomes large due to a low sending rate. To the best of our knowledge, this feature is not mentioned in any description of RCP. However, such REF packets unnecessarily increase the traffic volume and don't scale with an increasing number of flows. Instead, we define a minimal rate the routers may specify in order to avoid the starvation of flows. Furthermore, we use the ACK packets of a receiving end-system to update the allowed sending rate on the reverse path and return this information with the next data packet from the sender. So the receiver knows the

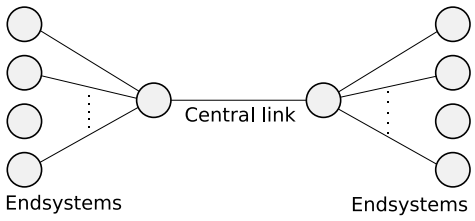

Figure 3: Simulated dumbbell topology with a central bottleneck link

sending rate it may use when it enters the sending mode. At the same time, the RTT measurement on the receiver side can be taken from the sender side as it is a symmetric property.

The topology used in the simulations is illustrated in Figure 3. It is a simple dumbbell topology where the central link represents the common bottleneck of all flows. The delay and bandwidth of each link can be configured individually to create e.g. scenarios with a common RTT or a dedicated RTT distribution. This topology is a standard test case for congestion control [17].

A flow is modelled by a modified version of the representation used for TMix tool [18] which we call the ATBT model shown in Figure 2. A flow is defined by a start time denoting the sending time of the first SYN-packet followed by one or several so-called epochs which consist of a data amount A in one direction followed by a "think time" or "processing time" t_1, a data amount B in the reverse direction, and a second think time t_2. Here, we define the flow completion time (FCT) to be the difference between the time of reception of the last data packet at its destination, and the moment when the SYN packet is sent (see Figure 2).

3.2 Traffic Model

We model the workload either by stochastic distribution functions or by replaying measured traffic traces. In the former case, flows are automatically generated with a negative-exponentially distributed inter-arrival time and a Pareto distribution for the flow length. The Pareto distribution reproduces the typical characteristic of Internet traffic with a heavy-tailed distribution. For simplicity, the synthetically generated traffic in our simulation is unidirectional.

The traffic generated from traces reflects a more realistic situation. As proposed in [17], we use the bidirectional traffic traces from [19] and the specified distribution of RTTs for this scenario. The concurrent flows were removed from the files and all other flow-vectors converted to be compatible with the developed simulator.

3.3 Performance Metrics

We investigate two different kinds of scenarios: First, scenarios with few flows are used to investigate the protocol behaviour in detail, in particular, when a flow enters or leaves the network. Second, we consider scenarios with many flows that model typical Internet traffic.

In the scenarios with few flows, we measure the convergence time to reach the equilibrium point, i. e., the time for a new flow to reach its fair share of the bandwidth, or the time to reallocate spare bandwidth, when a flow leaves. Furthermore, we study the queue occupancy and the throughput. The throughput is determined with an appropriate sampling

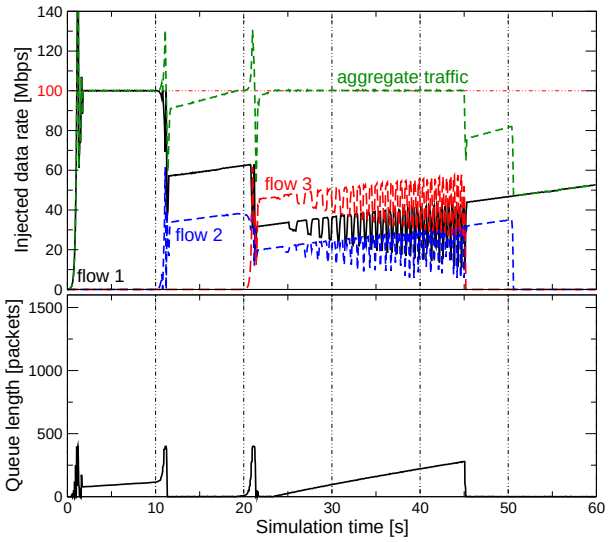

Figure 4: Behaviour of TCP with Reno; flow arrivals at $t = \{0, 10, 20\}$ s; droptail buffer of 400 packets

rate. For most scenarios, the sampling rate was chosen to be the same as the RTT.

In the scenarios with many flows, statistical properties like the flow completion time (FCT), the average number of active flows, the link occupation or the mean queue length are of interest. In particular, we analyze the FCT as a function of the flow length.

4. PRINCIPAL FUNCTION VALIDATION

In the following, we investigate the transient behaviour of the different protocols after the arrival or departure of flows.

4.1 TCP

Figure 4 shows the principal behaviour of the TCP Reno congestion control. Here, the central link has a capacity of 100 Mbps and a delay of 50 ms. The access links in the topology of Figure 3 are delay-less, i. e., the round-trip time (RTT) is 100 ms for all flows. The central link has a droptail queue with a size of 400 packets. At $t = 0$ s the first flow, which has an unlimited length, enters the network. At $t = 10$ s and $t = 20$ s, two further flows with limited length start. These flows leave the network after some time. In Figure 4, the injected data rate of the sending end-systems is plotted as a function of the simulation time. Also, the respective evolution of the queue occupation is illustrated.

The result illustrates the inherent problems of the standard TCP congestion control. There is no deterministic convergence to fairness, i. e., the three concurrent flows do not reach the same bandwidth. Furthermore, after flows 2 and 3 leave the network, the free bandwidth is reallocated very slowly. The queue is filled when a new flow arrives. The resulting packet loss is detected at the end-systems and the rate is reduced to avoid link congestion. This process is rather slow, and it takes the whole system rather long to utilize the whole bandwidth.

4.2 XCP

Figure 5 presents the same scenario for XCP. The only difference is, that we now have unlimited buffer space to

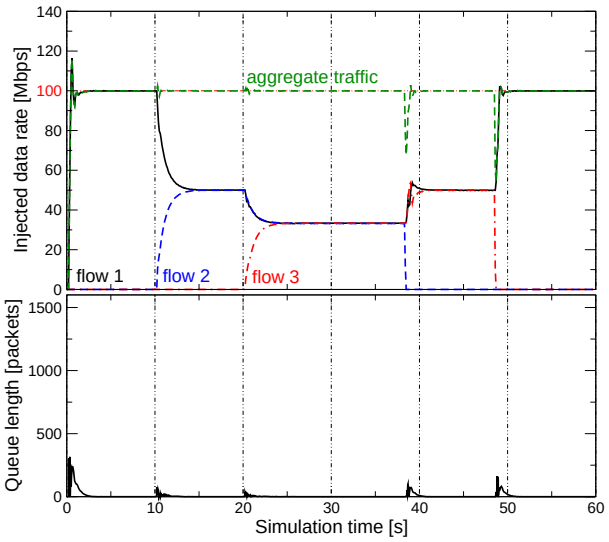

Figure 5: XCP behaviour with the window-based approach; $\alpha = 0.4$, $\beta = 0.226$

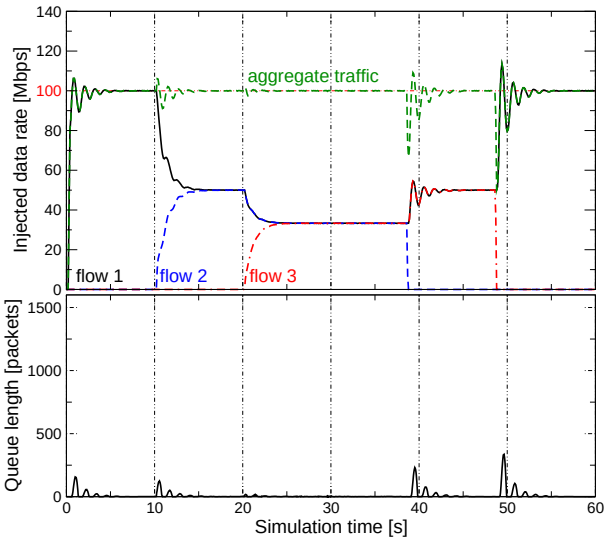

Figure 6: The rate-based XCP variant shows oscillating properties; $\alpha = 0.4$, $\beta = 0.226$

prevent eventual packet losses, because XCP falls back to TCP congestion control when losses occur. Parameters are chosen to be $\alpha = 0.4$ and $\beta = 0.226$ as proposed in the specifications [7]. This scenario uses the ordinary window-based approach as described in the XCP-specifications. XCP converges rather slowly to a fair allocation when the link is fully loaded. The transition takes place in the order of 50 RTTs. In the steady state, however, perfect fairness is achieved, i. e., all flows send with the same data rate. The slow convergence can be ascribed to the traffic shuffling mechanism which reassigns only 10% of the total capacity. This allocation is not given exclusively to the flow with a lower sending rate. Instead it is distributed anti-proportional to the current sending rates. Thus the convergence flattens more and more as the flows get closer to the equilibrium point. In contrast, when a flow leaves the network and free bandwidth becomes available, it is allocated very quickly to the remaining flows.

The queue occupation remains small during the whole scenario. There are short peaks in the queue which are caused by the bursty behaviour of a window protocol. When a new flow arrives, it receives a large feedback value in just one packet, which is used at once. Instantaneously, the link capacity is exceeded, but averaged over one RTT, there is no persistent queue. However, queueing occurs when a flow leaves the network. This is mainly due to microscopic synchronisation effects in this scenario, which are caused by the same RTT of all flows.

In order to eliminate the effects of a window-based protocol, we also study a rate-based implementation of XCP. One would expect that adjusting the rate eliminates the peaks in the queue and the synchronisation effects and thus makes the protocol behaviour smoother. However, the results for rate-based XCP, which are shown in Figure 6, reveal a detrimental behavior: The throughput oscillates significantly, and the queue is also utilized to a larger extent. Now, there are peaks in the queue occupancy on a longer time scale of the order of many RTTs.

The oscillation is caused by an unwanted prolongation of the XCP control loop. In the window-based variant, feedback becomes effective instantaneously, whereas in the rate-based case it is distributed equally over a whole RTT. As a consequence, the delay between the feedback calculation and the point in time when the rate change becomes effective at the routers is increased. This leads to a stronger overshooting of the aggregate traffic. In particlular, when a flow leaves the network there is a significant difference between available bandwidth and actual throughput, which causes significant oscillation. The throughput oscillation leads to intermediate packet buffering and a slightly lower overall link utilization.

The observed behaviour can be eliminated by setting the protocol parameter α to a smaller value of 0.2. However, a smaller α also increases the convergence time of XCP. These results indicate that there are no advantages of implementing XCP with a rate-based throttling mechanism.

4.3 RCP

In Figure 7, the transient behaviour of RCP can be seen. Again, we have an unlimited queue size because the reaction of RCP on packet losses is not specified yet. α and β are chosen like in the XCP scenario to make them comparable. As to be expected, all active flows have the same sending rate all the time. When a new flow arrives, it is signalled the same data rate as existing flows. This means that if there is only one active flow when a new flow starts, the initial aggregate traffic is two times the link capacity. Of course, it is reduced quickly to match the link capacity. Nevertheless, a significant amount of buffering is needed during a short period. In the used simulation configuration, the queue length exceeds the Bandwidth-Delay-Product (BDP) by about 70%.

This effect makes RCP vulnerable to flash crowd effects, which is discussed in detail in [20]. It can be mitigated by choosing a bigger β and smaller α, but there is a trade-off between queue length and convergence speed. However, if the arrival rate of new flows does not increase drastically, RCP remains stable although some packet losses might occur. Apart from that, RCP is inherently fair and quickly allocates free capacity.

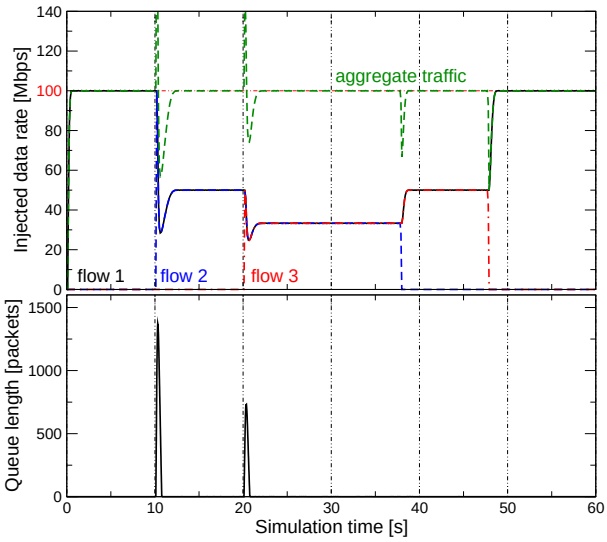

Figure 7: RCP behaviour for the same scenario with $\alpha = 0.4$, $\beta = 0.226$

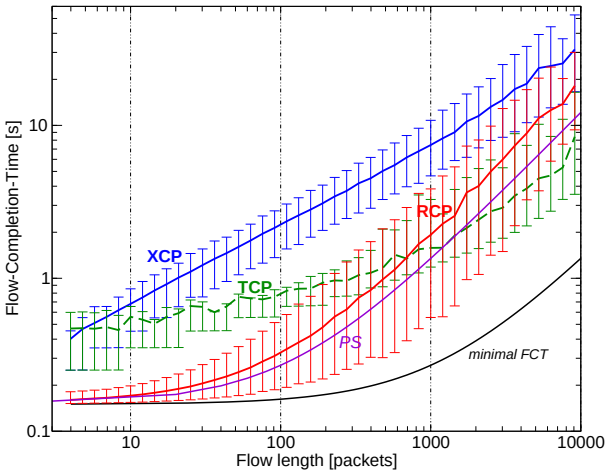

Figure 8: FCT study comparing TCP, XCP and RCP; for comparison, the analytically determined minimal FCT and PS curves are also shown

5. SPEEDUP COMPARISON

To evaluate the expectable speedup of an explicit congestion control protocol in a realistic network environment, we simulate scenarios with many synthetically created flows as described in Section 3.3. In a second scenario, traffic is replayed from traces.

Our main performance metric is the flow completion time. Of course, other metrics, such as link utilization, are also important characteristics of a congestion control algorithm, in particular from the viewpoint of network operators. Still, the end-user is mainly interested in a fast completion of data transfers, in particular when using interactive applications. The speedup of data transfers can be best quantified by the FCT [21].

5.1 Synthetic Flow Size Distribution

In the following, the central link has a capacity of 100 Mbps and the RTT is 100 ms for all flows. For the TCP simulation, we use CUBIC with an RED-Queue limited to 1,000 packets with a threshold value of 250 packets and a maximum drop probability of 0.1. XCP and RCP operate on an unlimited queue, as motivated before. For XCP we take the window-based variant with parameters $\alpha = 0.4$ and $\beta = 0.226$. RCP parameters are set to $\alpha = 0.1$ and $\beta = 1.0$, to reduce queue occupation and FCT (these parameters are also chosen in some scenarios in [6]).

The mean flow length is 25 packets, whereas the size of a packet equals 1500 bytes, corresponding to the maximum transmission unit (MTU) in most networks. Flow lengths are distributed according to a truncated Pareto distribution with shape 1.2 and cut-off at 40,000 packets. The mean inter-arrival time was chosen to be 3.33 ms. These parameters result in an overall link utilization of approximately 90% for the central link. This is a relatively high value, but a highly loaded link is the key challenge for any congestion control.

The resulting flow completion times are presented in Figure 8. The simulation duration was 3700 s, neglecting samples during the first 100 s in order to avoid effects of the ramp-up phase. Because the flow length ranges over several orders of magnitude, we depict them on a logarithmic scale. To get a sufficient amount of samples, the values of similar flow lengths are aggregated. The range of each aggregation is chosen to get points of equal distance on a logarithmic scale. We calculate the average as well as 5%- and 95%-quantiles of each aggregation. Due to the characteristics of the Pareto distribution, the number of samples per point decreases for large flow lengths.

In this scenario, RCP has the shortest flow completion time. It is much smaller than the FCT of TCP that is mainly dominated by the slow-start. XCP performs even worse than TCP. This can be explained by the slow allocation of bandwidth to new flows in a situation with high link utilization. Due to the XCP algorithms, new flows may even starve when there is temporary overload and there is few positive feedback. This effect, which has already been identified in [6], increases the flow completion time of short flows. As a side effect of the larger FCT, XCP accumulates many flows in the network (Little's law). When there are more active flows in the system, the bandwidth of an individual flow is also smaller. Simulation results that are not shown here indicate that in a system with lower load XCP is faster than TCP, because there is in average more free bandwidth that can be assigned to new flows.

These results confirm that RCP is indeed better suited for such situations, since RCP provides a fast start for short flows. Its performance comes closest to the minimum possible FCT. With the chosen parameters, RCP approximates processor sharing (PS). An interesting effect is that TCP has a shorter FCT for very long flows. But this comes at the cost of larger delays for short flows which are limited by slow-start and results in a longer average FCT.

5.2 Trace-driven Evaluation

In order to study the three protocols in a realistic traffic scenario, we simulated our network model with real traffic traces from [19]. This means that unlike the previous case we now have bidirectional traffic with very different burst sizes. The load was controlled by adapting the arrival times with a negative exponential distribution with a mean value of

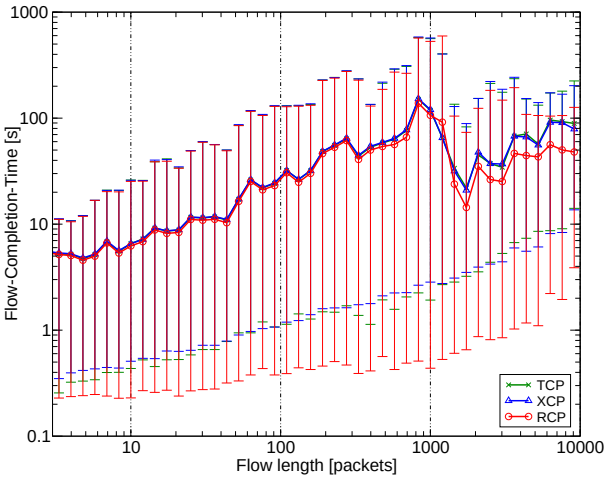

Figure 9: FCT for the simulation of bidirectional real internet traffic traces

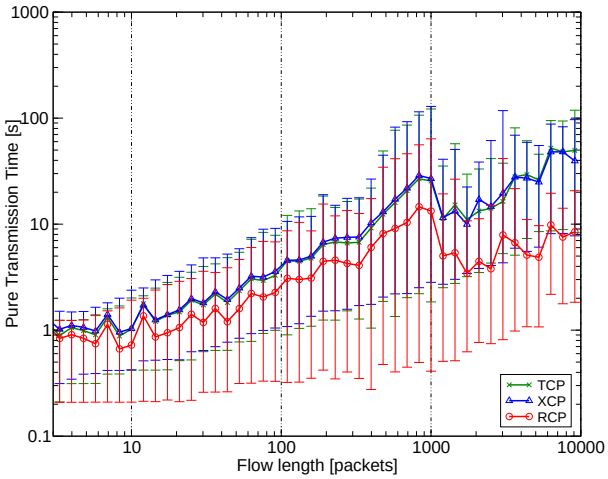

Figure 10: Pure transmission times for the traffic replay scenario; think times are stripped from the FCT

12 ms. This gives us a load of about 2% in forward direction (upstream) and a load of about 17% in backward direction (downstream). The simulation time was 1100s.

The results of this scenario can be seen in Figures 9 and 10. They are depicted in two different ways. The curves in Figure 9 contain the overall FCT which is largely dominated by the think or processing times contained in a flow. This also explains the big difference between the 5%- and 95%-quantiles. In Figure 10, the think times have been subtracted from the FCT in order to get the absolute transmission times.

The average FCT including think times in Figure 9 does not significantly depend on the congestion control mechanism. Only for large flow sizes one can observe a slight advantage for RCP. There are also some differences in the 5%-quantiles, which correspond to flows with small think times. In this case, RCP greatly reduces the FCT in comparison to TCP and XCP, which perform almost equal.

The results in Figure 10 for the transmission times without think times are very clear: RCP definitively outperforms TCP and XCP. For large flow sizes, the transmission times are reduced by almost an order of magnitude. Also, the 5%-quantiles and 95%-quantiles are significantly smaller. On the other side, XCP offers no performance gain in comparison to TCP. Thus, there is no justification for the high additional complexity of XCP in this case.

The scenario shows the great potential of RCP to reduce the FCT with real Internet traffic and designates it as a candidate for Future Internet congestion control.

6. CONCLUSIONS

Router-assisted congestion control is a promising solution for the Future Internet. The new protocols XCP and RCP can utilize links more efficiently and fairer than state-of-the-art TCP mechanisms, since they do not only rely on packet losses to signal congestion. However, they require some limited additional processing in routers. This paper studies whether these new congestion control mechanisms indeed reveal the claimed performance benefits. We verify the basic protocol functions with a new and independent simulation tool. Basic experiments show that some implementation design choices can have a significant impact on the operation of these protocols. Furthermore, we test the protocols in realistic traffic scenarios. The results of our simulations indicate that RCP outperforms XCP. In particular, the flow completion time is significantly reduced, and RCP has also a lower implementation complexity. As a consequence, RCP is certainly a candidate for Future Internet congestion control. Still, our performance comparison is not complete so far. Further work is needed to compare XCP and RCP to other congestion control schemes, such as the Quick-Start TCP extension.

7. REFERENCES

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581 (Proposed Standard), April 1999.

[2] M. Welzl, D. Papadimitriou, M. Scharf, and B. Briscoe. Open research issues in Internet congestion control. IRTF Internet Draft, work in progress, 2008.

[3] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), September 2001.

[4] S. Floyd, M. Allman, A. Jain, and P. Sarolahti. Quick-Start for TCP and IP. RFC 4782 (Experimental), January 2007.

[5] D. Katabi. *Decoupling Congestion Control and Bandwidth Allocation Policy with Application to High Bandwidth-Delay Product Networks*. PhD thesis, MIT, Dept. of Electrical Engineering and Computer Science, March 2003.

[6] N. Dukkipati. *Rate Control Protocol (RCP): Congestion Control to Make Flows Complete Quickly*. PhD thesis, Stanford University, Dept. of Electrical Engineering, October 2007.

[7] A. Falk, Y. Pryadkin, and D. Katabi. Specification for the Explicit Control Protocol (XCP). Internet Draft (Work in progress), July 2007.

[8] Y. Zhang, D. Leonard, and D. Loguinov. Jetmax: Scalable max-min congestion control for high-speed

heterogeneous networks. *Computer Networks*, 52(6):1193–1219, April 2008.

[9] M. Scharf and H. Strotbek. Performance evaluation of Quick-Start TCP with a Linux kernel implementation. In *Proc. IFIP Networking 2008, Springer LNCS 4982*, pages 703–714, May 2008.

[10] S. Hauger, M. Scharf, J. Kögel, and C. Suriyajan. Quick-Start and XCP on a network processor: Implementation issues and performance evaluation. In *Proc. IEEE High Performance Switching and Routing (HPSR), Shanghai, China*, May 2008.

[11] N. Dukkipati, G. Gibb, N. McKeown, and J. Zhu. Building a RCP (rate control protocol) test network. In *Proc. IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 91–98, 2007.

[12] Institute of Communication Networks and Computer Engineering. IKR Simulation Library. http://www.ikr.uni-stuttgart.de/Content/IKRSimLib/, 2008.

[13] F. Querzola. ns-2/IKRSimLib comparison. http://www.ikr.uni-stuttgart.de/INDSimLib/Resources/IR47_TCP_Querzola.pdf, 2004.

[14] Network Simulation Cradle. http://research.wand.net.nz/software/nsc.php.

[15] C. Zeeh. Integration of the Linux-TCP/IP Protocol Stack into an Event-Driven Simulation Environment. Diploma thesis, IKR, University of Stuttgart, November 2006.

[16] I. Rhee, L. Xu, and S. Ha. CUBIC for Fast Long-Distance Networks. Technical report, August 2008.

[17] L. Andrew and S. Floyd. Common TCP Evaluation Suite. Internet Draft (Work in progress), July 2008.

[18] M. C. Weigle, P. Adurthi, F. H. Campos, K. Jeffay, and F. D. Smith. Tmix: a tool for generating realistic TCP application workloads in ns-2. *SIGCOMM Comput. Commun. Rev.*, 36(3):65–76, 2006.

[19] WAN in Lab - Traffic Traces. http://wil.cs.caltech.edu/suite/TrafficTraces.php.

[20] F. Abrantes, J. T. Araújo, and M. Ricardo. Flash Crowd Effect in RCP. *Proc. PFLDnet*, March 2008.

[21] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *SIGCOMM Comput. Commun. Rev.*, 36(1):59–62, 2006.