Technical University Berlin

Telecommunication Networks Group

Performance Evaluation of an Improved Common Congestion Controller for WWW-based TCP Connections

Michael Savorić, Holger Karl

{savoric,karl}@ee.tu-berlin.de

Berlin, June 2004

TKN Technical Report TKN-04-007

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

Today’s standard Internet transport protocol implementations perform flow and congestion control separately for each data stream, in isolation from all other data streams. It is advantageous in terms of improving the overall performance of the data streams, i.e., the throughput and fairness, to reuse network information and—as an extension—to establish a common congestion control between some of the data streams of an end system.

In this technical report, we describe the design goals and explain the algorithms of a common congestion control approach for TCP connections called ”ensemble flow congestion management” (EFCM). In addition, we investigate the performance gain of the EFCM approach compared to the standard TCP congestion and flow control under different network conditions. Simulations with the EFCM approach show a considerable increase in throughput and fairness without increasing the aggressiveness of a set of TCP connections; the effect on background traffic is also negligible. The proposed EFCM controller algorithms are easy to implement and have a low additional complexity.

Keywords: TCP, Congestion Control, Flow Control, Network Information Reuse, Common Congestion Control

Notes

Due to a bug in the ns-2 implementation of the WWW traffic model, the absolute values of earlier published simulation results considering the EFCM controller [12] were incorrect. This technical report investigates the performance of the EFCM controller using the corrected ns-2 implementation of this WWW traffic model.

And in contrast to earlier performed EFCM simulations [12], we use in the current simulations TCP senders that set their initial congestion windows to the highest possible value allowed by the newest TCP standard [1].

Contents

Chapter 1

Introduction

In today's Internet, most of the data streams are using the TCP or UDP transport protocols. One of the most important responsibilities of a transport protocol are flow and congestion control. A TCP stream performs congestion control by slowly increasing its sending window, probing the network's capacity and trying to adapt to it in order not to overload the network; in a sense, a TCP stream collects information about the current network conditions. For a UDP stream, these tasks are left to the application, since UDP has no built-in congestion and flow control.

For both types of transport protocols, all data streams of an end system act separately and independently of each other. This means that, for example, TCP data streams do not share their information about the current network conditions and UDP data streams do not adjust their sending window to the current network conditions as detected by some TCP data streams. As a result, the overall performance of these data streams can be suboptimal, since the performance of each data stream is optimized only by using its locally available network information.

Exploiting such network information that can be present within an end system's protocol stack and sharing this information between multiple streams should improve overall performance. This information sharing can happen once at the start of a new data stream to initialize its flow and congestion control variables with more adequate values. This approach is called one-time network information reuse. As an extension, information can be shared continuously among several data streams during their whole lifetime in order to *jointly* control them; this second approach is called common congestion control. This technical report focuses on common congestion control.

Sharing network information between data streams can only happen among streams that use the same network path. Hence, a common congestion approach is only reasonable between data streams of an end system which have the same receiver or at least receivers in the same part of the network. These data streams form a (data stream) *ensemble*. The algorithms that determine which, how, and when network information among different data streams of an ensemble is shared form the actual controller of a common congestion control approach.

A common congestion controller's job can be divided into two main tasks: First, a common congestion controller has to manage the one-time network information exchange between *existing* (or *recently closed*) data streams of an ensemble and a *new* data stream joining this particular ensemble. This task is similar to the controller's job in existing pure network information reuse approaches like the ensemble or temporal TCP control block interdependence (TCBI) [15, 10, 14]. Second, a common congestion controller is responsible for the continuous network information exchange between *concurrent* data streams of an ensemble to reach a common congestion control for this ensemble.

These two tasks can be fulfilled in a number of different ways. The main ideas and methods of the four most relevant approaches have been described in reference [9]. One of these approaches, the ensemble TCP (E-TCP) [2], has been identified as a good basis for our new common congestion control approach called ”ensemble flow congestion management” (EFCM). The E-TCP approach provides a common congestion control among TCP connections of an end system in a way that an ensemble of n TCP connections is no more aggressive to the network than a single TCP connection. Network information is shared between a new TCP connection and existing or recently closed TCP connections and between concurrent TCP connections. In addition, for every ensemble the E-TCP approach uses a scheduler that determines which TCP connection of an ensemble sends the next segment. A rate-based pacing mechanism can be optionally used for TCP connections of an ensemble. The assumption that an entire ensemble should be at most as aggressive as a single connection appears very conservative; it is here that EFCM and E-TCP differ most (details are presented in Chapter 2). We claim that the EFCM approach results in considerable performance gains at a moderate increase in implementation complexity and only limited adverse effects on the network.

Nevertheless, making common congestion control a practical solution is still faced with a practical challenge: In a simple common congestion controller the IP address of the receiver or the IP subnet address of the receivers are used to determine if some TCP connections can form an ensemble and use a common congestion control or not. If some mechanisms like NAT or Mobile IP are used, the IP addresses of the receivers can no longer form the criterion to build an ensemble, since with these mechanisms the IP addresses of the receivers do not reflect the current location of the receivers. Therefore, it might be sometimes difficult or even impossible to find out which TCP connections of an end system can form an ensemble and can use a common congestion control. There exist some mechanisms based on measurements to find out which TCP connections share the same path or bottleneck link, for example [8]. But further research must be done on this topic to determine the constraints and possible solutions for using a common congestion control in the real Internet.

The remainder of this technical report is organized as follows: The design goals and algorithms of the EFCM approach are explained in Chapter 2. These algorithms were evaluated by simulations. The network topology and simulation scenarios are shown in Chapter 3, appropriate evaluation metrics are described in Chapter 4, and Chapter 5 presents and discusses the simulation results. Finally, Chapter 6 contains the conclusion of this technical report and gives an outlook on our future research activities. In Appendix A, a rough estimate of the additional time and space consumption of the EFCM controller compared to standard TCP is given. Appendix B contains both a description of the statistical evaluation method as well as a detailed exposition of the evaluation results. Since the EFCM controller is under development, Appendix C describes the additional algorithms of the latest version of the EFCM controller compared to the EFCM controller depicted in Chapter 2.

Chapter 2

Ensemble Flow Congestion Management (EFCM)

In Chapter 1, the general concept of sharing network information between some data streams has been presented, using the notion of a controller that manages the information exchange. A controller is an abstract entity which needs to be specified further to determine a concrete, implementable and testable functionality. One possibility of such a controller, the EFCM controller, is presented here.

2.1 The EFCM design constraints

The design constraints of the ensemble flow congestion management are:

- The control algorithms of the EFCM approach are confined to a sending end system, i.e., an end system where the senders of the data streams are located. This means that except for some code in the transport protocol no adaptations and additional changes neither in the network nor in the receiving end system(s) have to be done.
- The EFCM approach supports standard transport layer interfaces, i.e., sockets. Therefore, the EFCM is transparent for all applications and Internet services running on the end system.
- In contrast to the E-TCP approach, the algorithms of the EFCM controller ensure that an ensemble of n data streams must be no more aggressive to the network than n separate data streams of an end system.
- Another design constraint of the EFCM controller is a fair sharing of the available bandwidth among the data streams in an ensemble.

2.2 The EFCM controller

The EFCM controller investigated in this technical report is an improved version of the EFCM controller described in [11]. This improved version of the EFCM controller performs one-time network information reuse for a new connection as well as common congestion control between concurrently existing connections of an ensemble. It does not, however, reuse network information obtained from recently closed TCP connections (as, e.g., temporal TCBI does [15]). It also does not control UDP data streams with network information obtained from existing or recently closed TCP connections.

To avoid a bursty sending behavior of EFCM-controlled TCP connections, the EFCM controller uses a rate-based pacing mechanism for consecutive TCP segments. In addition, the EFCM controller is equipped with a joint ack clocking mechanism for every ensemble of TCP connections. This ensemble ack clocking (EAC) allows a fair partitioning of the sent but currently not acknowledged TCP segments, i.e., the on-the-fly TCP segments, between the TCP connections in an ensemble.

2.3 The EFCM jointly controlled TCP variables

TCP uses the following variables for congestion control: congestion window (cwnd), slow start threshold (ssthresh), round trip time (rtt), smoothed round trip time (srtt), and round trip time variance (rttvar). The first two TCP control variables restrict the load a single TCP connection can send into the network, the last three TCP control variables lead to adequate timeout timer values for TCP segments send from a single TCP connection.

The EFCM controller jointly controls the congestion window, the slow start threshold, the smoothed round trip time, and the round trip time variance of TCP connections in an ensemble. Hence, the EFCM controller restricts the load the TCP connections of an ensemble can send into the network and all TCP connections of an ensemble obtain the same adequate value for their timeout timer.

2.4 The EFCM control algorithms

In the next two paragraphs, we describe the proposed algorithms of the EFCM controller for both tasks, i.e., initializing new connections and updating concurrent connections, of a common congestion controller. Afterwards, the used rate-based pacing mechanism is described in another paragraph. These descriptions are made by considering the (aggregated) congestion window and slow start threshold in units of segments and the (aggregated) smoothed round trip time and round trip time variance in units of seconds.

2.4.1 The network information reuse of the EFCM controller for a new TCP connection

If useful network information is available for a new TCP connection, the new TCP connection will reuse this network information and will start with more adequate values for the load the network can cope with and the timeout timer value. The algorithms of the EFCM controller for the network information reuse between existing TCP connections of an ensemble and a new TCP connection of the same ensemble are described in the following list:

Congestion window: The EFCM controller computes the sum of all current congestion windows of the existing TCP connections of the ensemble plus the standard initial congestion window 3 (segments) (cf. [1]), representing the new TCP connection. This value is used to calculate a fair share, i.e., an arithmetic mean value, of the congestion window for all TCP connections in the ensemble. At the beginning of a new TCP connection, all TCP connections of the ensemble get this congestion window fair share as their new congestion window.

Slow start threshold: The EFCM controller computes the sum of all current slow start thresholds of the existing TCP connections of the ensemble plus the standard initial slow start threshold 44 (65535 byte divided by the maximum segment size (MSS) of 1460 byte), representing the new TCP connection. This value is used to calculate a fair share of slow start threshold for all TCP connections in the ensemble. At the beginning of a new TCP connection, all TCP connections of the ensemble are assigned this slow start threshold fair share as their new slow start threshold.

Smoothed round trip time: The EFCM controller uses the current value of an aggregated smoothed round trip time of the existing TCP connections of an ensemble as the initial smoothed round trip time of the new TCP connection. If the new TCP connection is the only stream in its ensemble then the initial smoothed round trip time of the new TCP connection is set to the standard value.

Round trip time variance: The EFCM controller uses the current value of an aggregated round trip time variance of the existing TCP connections of an ensemble as the initial round trip time variance of the new TCP connection. If the new TCP connection is the only stream in its ensemble then the initial round trip time variance of the new TCP connection is set to the standard value.

2.4.2 The common congestion control of the EFCM controller for concurrent TCP connections

Whenever a standard TCP implementation would change the value of one of the commonly controlled variables of a connection, EFCM uses this change to trigger updates to the these variables for all other connections within the same ensemble, according to the following rules:

Congestion window: After every change of the congestion window of one of the existing TCP connections in an ensemble, an aggregated congestion window for this ensemble is computed by adding all current congestion windows of the TCP connections in the ensemble. This value is used to calculate a fair share of congestion window. This congestion window fair share is the new congestion window of every TCP connection in an ensemble.

Slow start threshold: After every change of the slow start threshold of one of the existing TCP connections in an ensemble an aggregated slow start threshold for this ensemble is computed by adding all current slow start thresholds of the TCP connections in the ensemble. This value is used to calculate a fair share of slow start threshold. This value is the new slow start threshold of every TCP connection in an ensemble.

Smoothed round trip time: After every change of the smoothed round trip time of one of the n TCP connections in an ensemble an aggregated smoothed round trip time of this ensemble is updated by a weighted calculation of $(n-1)/n$ times the last value of the aggregated smoothed round trip time plus $1/n$ times the new smoothed round trip time. All TCP connections in an ensemble get this calculation result of the smoothed round trip time as their new smoothed round trip time.

Round trip time variance: After every change of the round trip time variance of one of the n TCP connections in an ensemble an aggregated round trip time variance of this ensemble is updated by a weighted calculation of $(n-1)/n$ times the last value of the aggregated round trip time variance plus $1/n$ times the new round trip time variance. All TCP connections in an ensemble get this calculation result of the round trip variance as their new round trip time variance.

If one of the TCP connections in an ensemble is affected by a packet loss, all TCP connections of the ensemble will fairly reduce their congestion window and slow start threshold to the new calculated values. This maintains the congestion control of standard TCP if the packet loss is caused by congestion in the network.

If a TCP connection leaves the ensemble, i.e., the TCP connection has been closed, the current aggregated congestion window and the current aggregated slow start threshold are fairly shared among the remaining TCP connections in the ensemble.

Considering the congestion window and slow start threshold algorithms of the EFCM controller we have analytically shown in [13] that the aggressiveness of these algorithms is comparable to the aggressiveness of standard TCP.

2.4.3 The pacing mechanism of the EFCM controller

The pacing mechanism used in the EFCM controller is implemented by using a rate-based mechanism: Every TCP connection in an ensemble can send at most two TCP segments in a burst. The time Δt between two consecutive packet bursts of a TCP connection is calculated by using the aggregated smoothed round trip time and the aggregated congestion window of an ensemble:

$$\Delta t = \alpha_{\text{pacing}} \cdot \text{aggregated SRTT}/\text{aggregated CWND}$$

In the current version of the EFCM controller the factor α is set to the fixed value 2. Some remarks about other values for the factor α can be found in Appendix C.

Evidently, these EFCM computations impose some overhead in time and space. A rough estimate of the additional time and space complexity of the EFCM controller compared to standard TCP is given in Appendix A.

Chapter 3

Simulation Model

Evaluating the throughput and fairness gain of the EFCM approach is done by simulations. The network topology for these simulations is intended to reflect common client-server architectures in which the client end systems, i.e. the receivers, obtain information from applications running on the (optionally) EFCM-controlled server end system, i.e., the (EFCM) end system. In typical Internet setups, a server is connected to an ISP's backbone network with a high-bandwidth, reliable link. Typical client end systems are connected via networks with smaller bandwidths, e.g., DSL lines or low-speed LANs. These networks can be both reliable or, in case of modern wireless LAN installations, unreliable. The backbone network itself has a bandwidth-delay product varying over time depending on its current load. We chose our simulated network topology with regard to these properties of the Internet.

In addition to the network topology there are other factors that influence the performance of the EFCM approach, e.g., the number of TCP sender instances in the (EFCM) end system, the type of applications running on the (EFCM) end system, the round trip time between the (EFCM) end system and the receiver(s), the packet loss rate in the links between the (EFCM) end system and the receiver(s), the background traffic load in the path(s) from the (EFCM) end system to the receiver(s), and the maximum segment size (MSS) of TCP connections. For each of these factors a reasonable subset of values must be defined and for every combination of parameter settings standard TCP must be compared with the EFCM approach to reach a complete performance evaluation of the EFCM approach.

In this paper, we consider one specific network model with a reasonable choice of parameters. This model is described in detail in the following Section 3.1; the load models are contained in Section 3.2.

3.1 Simulated network topology

The performance evaluation of the EFCM approach uses a simulated network topology shown in Figure 3.1. The simulation network topology consists of several TCP senders (S1,..., S3 and BS1, BS2) and TCP receivers (R1,..., R3 and BR1, BR2), two routers, and one Ethernet-type LAN on the sender side. The sender LAN is characterized by a bit rate of 100 Mbps with and propagation delay of 0.5 μs. The routers are connected via links with a bit rate of 100 Mbps and a propagation delay of 30 ms. For each incoming link the routers have a queuing capacity of 20 IP packets. Together with the given load in this tiny simulation model some packet losses in the routers can be observed with

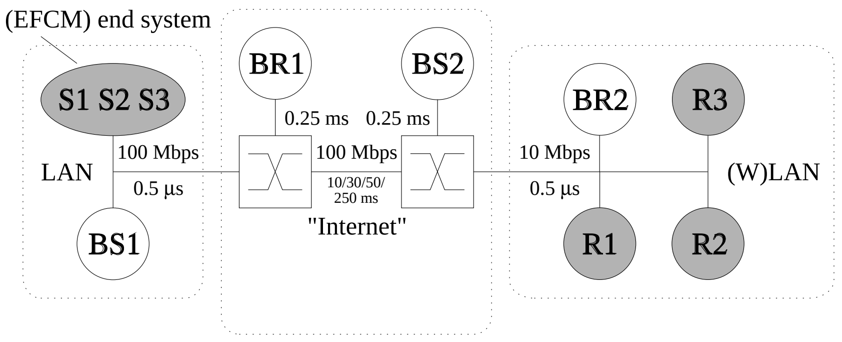


Figure 3.1: Structure of the simulated network topology

this relatively small queuing capacity.

The TCP senders S1, S2, and S3 are located in an end system which is optionally equipped with an EFCM controller. The other TCP senders BS1 and BS2 are located in different end systems and generate background traffic.

The end system of the background traffic TCP sender BS2 is connected to the network via links with a bit rate of 100 Mbps and a propagation delay of 0.25 ms. The end system of the background traffic TCP sender BS1 and the EFCM end system are connected to the network via the sender LAN. The end systems of the TCP receivers R1, R2, and R3 and the background traffic TCP receiver BR2 are connected to the network via the receiver LAN.

The receiver LAN consists of an Ethernet-type shared medium with an overall bit rate of 10 Mbps and a propagation delay of 0.5 μs. The packet loss rate in the receiver LAN is adjustable to investigate the influence of different packet loss probabilities in the last hop of a TCP connection on the overall throughput of the TCP connections. Hence, with this receiver LAN either a reliable (wired) Ethernet or an unreliable wireless LAN can be modeled. In the wired last hop scenario, no errors occur in the receiver LAN; in the wireless LAN case, packets can be lost with a fixed packet loss rate of 5 percent.

In summary, the chosen parameter values should represent typical Internet scenarios fairly well. It would be particularly interesting to consider the impact of various values for e.g. the round trip time, the maximum size of TCP segments, and the packet loss rate as important factors for the performance gain of EFCM. This technical report concentrates on varying the round trip time; other parameters are kept constant.

3.2 Traffic load models

Properly characterizing traffic loads for interactive Internet users is a difficult undertaking. We decided to test the performance of our EFCM controller by considering traffic generated by a WWW traffic model [7]. This traffic model is derived from real HTTP traces in corporate and educational environments and uses three abstraction levels: The session level, the page level, and the packet level. Here, a simplified version of this model is used which consists only of the first two levels. In every WWW session a log-normally distributed number of WWW pages with Pareto-distributed page sizes are sent. The time between the pages, i.e., the inter-connection or reading time, is gamma distributed. The load in the network can be easily adjusted by using the exponentially distributed session interarrival time with a different parameter. The distributions and parameters chosen for the stochastic variables of the simplified WWW traffic model are shown in Table 3.1.

Table 3.1: Distributions and parameters for the stochastic variables of the simplified WWW model

Stochastic variable	Distribution	Distribution parameter(s)
Inter-session time	Exponential	$\mu = 5.0$ s
Pages per session (pps)	Lognormal	$\mu = 25.807$ pps $\sigma = 78.752$ pps
Inter-page time (reading time)	Gamma	$\mu = 35.286$ s $\sigma = 147.390$ s
Page size	Pareto	$\alpha = 1.7584$ $\beta = 30458$ Bytes

The TCP senders of the (optionally) EFCM-equipped end system use the simplified WWW traffic model. The traffic of all background TCP senders is related to the WWW traffic model but uses modified inter-connection time and session interarrival time distributions, where TCP connections are immediately restarted once they have terminated. This is done to reach a higher load in the network with these few background traffic TCP senders.

The whole simulation model is implemented in ns-2 (version 2.1b9) [3]. For all standard TCP connections, the ns-2 implementation of a TCP Newreno [4] sender or receiver is used. The TCP connections of the (EFCM) end system are instances of a new TCP sender class (EFCM_TCP) derived from the ns-2 implementation of a TCP Newreno sender. This new TCP sender class provides additional network information reuse and common congestion control mechanisms between the TCP connections of an EFCM ensemble and some statistical performance evaluation methods.

Chapter 4

Evaluation Metric

In each simulated scenario and for all new TCP connections the mean throughput, the mean initial congestion window, the mean initial slow start threshold, the mean initial smoothed round trip time, and the mean initial round trip time variance are compared between the standard TCP and the EFCM controller. In addition, also a fairness index between concurrent TCP connections of an EFCM ensemble is used to compare standard TCP with the EFCM approach: If n TCP connections are established in parallel for a period of time and reach the mean throughputs $\bar{t}_i$, $1 \leq i \leq n$, in this period of time, then for these TCP connections a fairness index for this period of time can be computed as follows [5]:

$$I_f = \frac{\left(\sum\limits_{i=1}^{n} \bar{t}_i\right)^2}{n \cdot \sum\limits_{i=1}^{n} \bar{t}_i^2} \quad \text{with} \quad \frac{1}{n}\,\text{(very bad)} \leq I_f \leq 1\,\text{(excellent).}$$

A fairness index of $1/n$ denotes that one of the n concurrent TCP connection gets the entire available bandwidth while a fairness index of 1 means that all n concurrent TCP connections get the same portion of the available bandwidth.

Only those TCP connections are considered in the comparison shown in the following tables which either are controlled by the EFCM approach or are not controlled but could be controlled by the EFCM approach (as at least one concurrent TCP connection to the same LAN is already established and useful information about the network is available). These TCP connections are called EFCM-capable TCP connections. In general, the percentage of EFCM-capable TCP connections depends on the type of the EFCM end system. For example, the numerous TCP connections of a large WWW or proxy server have a higher probability of using the EFCM approach than the few TCP connections of an ordinary end system. In order not to reflect this dependency in the evaluation, the metric computations were restricted to the EFCM-capable TCP connections and are hence independent of the end system's type. The overall performance impact of the EFCM approach on the throughput of all TCP connections can then be approximated by using the share of the EFCM TCP connections of all TCP connections of the EFCM end system.

To compare standard TCP with the EFCM controller, two different mean throughput computations for the EFCM-capable TCP connections of the two application classes are used. If one of these TCP connections has sent s segments in duration d, then the two mean throughput calculations work as follows:

- Computation of the overall mean throughput ($\overline{T}_1$): The sum of sent segments of all n TCP connections is divided by the overall duration of these TCP connections, i.e.:

$$\overline{T}_1 = \frac{\sum\limits_{i=1}^{n} s_i}{\sum\limits_{i=1}^{n} d_i}$$

- Computation of the connection-oriented mean throughput ($\overline{T}_2$): For each of the n TCP connections a mean throughput t is calculated. All these mean throughput values are then used to compute the overall mean throughput of the TCP connections by a normal non-weighted arithmetic mean calculation independent of the number of segments sent by each of the TCP connections, i.e.:

$$\overline{T}_2 = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{s_i}{d_i} = \frac{1}{n} \cdot \sum_{i=1}^{n} t_i$$

With the former throughput calculation the overall throughput of the different TCP controllers can be evaluated. The latter throughput calculation gives a connection-oriented mean throughput which can be understood as the mean throughput a single TCP connection can expect if a particular TCP controller is used.

The throughputs for new EFCM-capable TCP connections entering an ensemble are measured for their whole lifetime d_i; for EFCM-capable concurrent TCP connections these measurements are only performed during their time of concurrency c_i (see Figure 4.1).

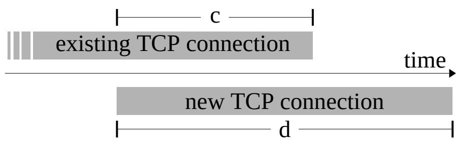


Figure 4.1: Time periods considered for throughput calculations

Chapter 5

Simulation Results

5.1 Overview

The same simulations were performed without and with the EFCM controller. The last hop was either reliable or had a packet loss rate (PLR) of 5 %. The simulation results for the two TCP variants, the different minimum round trip times, and the two last hop scenarios are shown in detail in the next subsections and summarized and discussed at the end of this chapter.

In the following Tables 5.1, 5.7, 5.9, and 5.15, TCP 1, TCP 2, and TCP 3 are WWW TCP connections of the (EFCM) end system. The simulation results of each simulation scenario shown in the tables are averages over 12 independent simulation runs, each of these runs is performed for a large simulated time of 250000 seconds.

Both stated mean throughput metrics ($\overline{T}_1$, $\overline{T}_2$) of the EFCM-capable TCP connections are measured in TCP segments per second; in every TCP segment, the payload length is set to 1460 bytes. For the new TCP connections entering an ensemble also the mean initial congestion window ($\overline{\text{cwnd}}$), the mean initial slow start threshold ($\overline{\text{ssthresh}}$), the mean initial smoothed round trip time ($\overline{\text{srtt}}$), and the mean initial round trip time variance ($\overline{\text{rttvar}}$) are shown. For the concurrent TCP connections of an ensemble both mean throughput metrics ($\overline{T}_1$, $\overline{T}_2$) and the mean fairness index ($\overline{I}_f$) are shown.

The last column Δ in the following tables denotes whether the simulation results are statistically significantly different or not for a given confidence level. A '+' or '-' denotes that the EFCM controller or standard TCP is significantly better. A '=' means that with the simulation results no significant difference between the EFCM and the standard TCP controller can be concluded. The details of the statistical evaluation of the simulation results can be found in Appendix B of this technical report.

For standard TCP connections the mean initial smoothed round trip time and the mean initial round trip time variance are not applicable (N/A) for the computation of the initial timeout timer, i.e., the initial timeout timer is set to the fixed standard value.

For all simulations, both controllers (standard TCP/no EFCM, EFCM) are investigated for a simulated time of 250000 s in each simulation run. This means that, for example, in the case with a minimum round trip time of 100 ms approximately 21000 TCP connections in the reliable last hop scenario and approximately 20500 TCP connections in the unreliable last hop scenario starting at the (EFCM) end system can be observed at an average during the simulated time. Only some of them, i.e., those TCP connections which have concurrent TCP connections, are controlled or could be controlled by the new network information reuse and common congestion control mechanisms provided by the EFCM controller. In the simulation model and with the chosen traffic load model the percentage of concurrent TCP connections is relatively low. An average computation over all simulations with a minimum round trip time of 100 ms shows that approximately 8.5 % of the new TCP connections in the reliable last hop scenarios and approximately 13.9 % of the new TCP connections in the unreliable last hop scenarios are controlled or could be controlled by the EFCM. This amount of EFCM-controllable TCP connections increases with the the minimum round trip time between the TCP senders and the TCP receivers to values up to 18.0 % for the reliable last hop and 31.1 % for the unreliable last hop in the case with a minimum round trip time of 500 ms.

In all simulations with the EFCM controller and with the observed occurrence of having concurrent TCP connections the mean throughput of the background TCP connections is not negatively affected by these EFCM-controlled TCP connections. Starting from these simulation results, we can guess that EFCM-controlled TCP connections are no more aggressive to the network than standard TCP connections. But in [13] we have even analytically shown that the aggressiveness of the algorithms of the EFCM controller regarding the congestion window and the slow start threshold is comparable to the aggressiveness of standard TCP.

5.2 Simulation 1: TCP Newreno, RTT $\geq$ 20 ms

5.2.1 Reliable last hop

Table 5.1 shows the simulation results for new TCP connections entering an ensemble. Compared to standard TCP, for new WWW TCP connections the overall mean throughput is slightly decreased by 4 % by the EFCM controller. But the connection-oriented mean throughput is increased by approximately 8 % if the EFCM controller is used.

Table 5.1: Simulation results of simulation 1, scenario 1 (new TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

		no EFCM	EFCM	Δ
$\overline{T}_1$ of new TCP connections [segments/second]	TCP 1	42.25	41.48	$=$
	TCP 2	42.30	39.88	$-$
	TCP 3	42.12	40.37	$-$
$\overline{T}_2$ of a new TCP connection [segments/second]	TCP 1	46.61	50.68	$+$
	TCP 2	46.74	49.95	$+$
	TCP 3	46.89	50.22	$+$
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	3.00	34.52	
	TCP 2	3.00	32.94	
	TCP 3	3.00	35.01	
$\overline{\text{ssthresh}}$ of a new TCP connection [segments]	TCP 1	44.00	70.35	
	TCP 2	44.00	68.56	
	TCP 3	44.00	71.24	
$\overline{\text{srtt}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.121	
	TCP 2	N/A	0.119	
	TCP 3	N/A	0.122	
$\overline{\text{rttvar}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.079	
	TCP 2	N/A	0.077	
	TCP 3	N/A	0.080	

Table 5.2 shows the simulation results for concurrent TCP connections of an ensemble. Compared to standard TCP, the EFCM controller reaches a slight performance degradation of approximately 3 % for the overall mean throughput and a performance gain of approximately 8 % for the connection-oriented mean throughput. The fairness between concurrent TCP connections is remarkably improved if the EFCM controller is used.

Table 5.2: Simulation results of simulation 1, scenario 1 (concurrent TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, $\overline{I}_f$ is the mean fairness index, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

	no EFCM	EFCM	Δ
$\overline{T}_1$ of concurrent TCP connections [segments/second]	39.97	38.86	$-$
$\overline{T}_2$ of a concurrent TCP connection [segments/second]	38.52	41.39	$+$
$\overline{I}_f$ of concurrent TCP connections []	0.7916	0.8316	$+$

5.2.2 Unreliable last hop

Table 5.3 shows the simulation results for new TCP connections entering an ensemble. Compared to standard TCP, the EFCM controller achieves a huge performance gain of approximately 68 % for the overall mean throughput but a slight performance degradation of approximately 1 % for the connection-oriented mean throughput.

Table 5.3: Simulation results of simulation 1, scenario 2 (new TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

		no EFCM	EFCM	Δ
$\overline{T}_1$ of new TCP connections [segments/second]	TCP 1	23.56	41.37	$+$
	TCP 2	24.20	40.60	$+$
	TCP 3	24.99	40.20	$+$
$\overline{T}_2$ of a new TCP connection [segments/second]	TCP 1	69.12	69.35	$=$
	TCP 2	69.69	67.98	$-$
	TCP 3	69.78	68.33	$-$
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	3.00	12.33	
	TCP 2	3.00	12.07	
	TCP 3	3.00	12.07	
$\overline{\text{ssthresh}}$ of a new TCP connection [segments]	TCP 1	44.00	40.64	
	TCP 2	44.00	40.60	
	TCP 3	44.00	40.44	
$\overline{\text{srtt}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.046	
	TCP 2	N/A	0.045	
	TCP 3	N/A	0.045	
$\overline{\text{rttvar}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.033	
	TCP 2	N/A	0.032	
	TCP 3	N/A	0.032	

Table 5.4 shows the simulation results for concurrent TCP connections of an ensemble. Compared to standard TCP, the overall mean throughput is largely increased by approximately 25 % by the EFCM controller. For the connection-oriented mean throughput a slight performance degradation of approximately 4 % can be observed if the EFCM controller is used. But the fairness is largely increased by the EFCM controller.

Table 5.4: Simulation results of simulation 1, scenario 2 (concurrent TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, $\overline{I}_f$ is the mean fairness index, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

	no EFCM	EFCM	Δ
$\overline{T}_1$ of concurrent TCP connections [segments/second]	35.48	44.49	$+$
$\overline{T}_2$ of a concurrent TCP connection [segments/second]	65.65	63.23	$-$
$\overline{I}_f$ of concurrent TCP connections []	0.8261	0.8636	$+$

5.3 Simulation 2: TCP Newreno, RTT $\geq$ 60 ms

5.3.1 Reliable last hop

Table 5.5 shows the simulation results for new TCP connections entering an ensemble. Compared to standard TCP, the EFCM controller slightly increases the overall mean throughput of new WWW TCP connections by approximately 3 % (not significant). But the connection-oriented mean throughput of these TCP connections is largely increased by approximately 16 % if the EFCM controller is used.

Table 5.5: Simulation results of simulation 2, scenario 1 (new TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

		no EFCM	EFCM	Δ
$\overline{T}_1$ of new TCP connections [segments/second]	TCP 1	33.82	34.95	$=$
	TCP 2	33.98	34.93	$=$
	TCP 3	33.35	34.15	$=$
$\overline{T}_2$ of a new TCP connection [segments/second]	TCP 1	37.12	43.35	$+$
	TCP 2	37.49	43.46	$+$
	TCP 3	37.42	43.18	$+$
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	3.00	34.56	
	TCP 2	3.00	34.09	
	TCP 3	3.00	34.96	
$\overline{\text{ssthresh}}$ of a new TCP connection [segments]	TCP 1	44.00	70.74	
	TCP 2	44.00	70.08	
	TCP 3	44.00	71.22	
$\overline{\text{srtt}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.161	
	TCP 2	N/A	0.159	
	TCP 3	N/A	0.159	
$\overline{\text{rttvar}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.083	
	TCP 2	N/A	0.081	
	TCP 3	N/A	0.080	

Table 5.6 shows the simulation results for concurrent TCP connections of an ensemble. For these connections, standard TCP and the EFCM controller reach nearly the same overall mean throughput. But the connection-oriented mean throughput is increased by approximately 10 % if the EFCM controller is used. Also the fairness between concurrent TCP connections is remarkably improved if the EFCM controller is performed in the end system.

Table 5.6: Simulation results of simulation 2, scenario 1 (concurrent TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, $\overline{I}_f$ is the mean fairness index, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

	no EFCM	EFCM	Δ
$\overline{T}_1$ of concurrent TCP connections [segments/second]	31.93	31.86	$=$
$\overline{T}_2$ of a concurrent TCP connection [segments/second]	31.12	34.17	$+$
$\overline{I}_f$ of concurrent TCP connections []	0.7943	0.8324	$+$

5.3.2 Unreliable last hop

Table 5.7 shows the simulation results for new TCP connections entering an ensemble. Compared to standard TCP, new WWW TCP connections benefit from the EFCM controller with a huge gain of approximately 74 % for the overall mean throughput and a gain of approximately 16 % for the connection-oriented mean throughput.

Table 5.7: Simulation results of simulation 2, scenario 2 (new TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

		no EFCM	EFCM	Δ
$\overline{T}_1$ of new TCP connections [segments/second]	TCP 1	19.67	34.76	+
	TCP 2	20.71	34.52	+
	TCP 3	19.75	35.16	+
$\overline{T}_2$ of a new TCP connection [segments/second]	TCP 1	43.10	49.99	+
	TCP 2	43.36	49.81	+
	TCP 3	43.09	50.71	+
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	3.00	11.87	
	TCP 2	3.00	12.05	
	TCP 3	3.00	11.98	
$\overline{\text{ssthresh}}$ of a new TCP connection [segments]	TCP 1	44.00	40.08	
	TCP 2	44.00	40.11	
	TCP 3	44.00	40.34	
$\overline{\text{srtt}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.075	
	TCP 2	N/A	0.076	
	TCP 3	N/A	0.075	
$\overline{\text{rttvar}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.030	
	TCP 2	N/A	0.030	
	TCP 3	N/A	0.031	

Table 5.8 shows the simulation results for concurrent TCP connections of an ensemble. Compared to standard TCP, the EFCM controller achieves a large gain for the overall mean throughput of approximately 33 % and a gain for the connection-oriented throughput of approximately 13 % for concurrent TCP connections. For these TCP connections also a considerable mean fairness improvement of the EFCM controller can be observed.

Table 5.8: Simulation results of simulation 2, scenario 2 (concurrent TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, $\overline{I}_f$ is the mean fairness index, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

	no EFCM	EFCM	Δ
$\overline{T}_1$ of concurrent TCP connections [segments/second]	27.32	36.38	$+$
$\overline{T}_2$ of a concurrent TCP connection [segments/second]	39.65	44.77	$+$
$\overline{I}_f$ of concurrent TCP connections []	0.8342	0.8692	$+$

5.4 Simulation 3: TCP Newreno, RTT $\geq$ 100 ms

5.4.1 Reliable last hop

Table 5.9 shows the simulation results for new TCP connections entering an ensemble. Compared to standard TCP, new WWW TCP connections benefit from the EFCM controller with a gain of approximately 14 % for the overall mean throughput and a large gain of approximately 31 % for the connection-oriented mean throughput.

Table 5.9: Simulation results of simulation 3, scenario 1 (new TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

		no EFCM	EFCM	Δ
$\overline{T}_1$ of new TCP connections [segments/second]	TCP 1	30.45	34.29	+
	TCP 2	30.52	34.95	+
	TCP 3	30.13	34.24	+
$\overline{T}_2$ of a new TCP connection [segments/second]	TCP 1	31.11	40.87	+
	TCP 2	31.20	41.10	+
	TCP 3	31.16	40.73	+
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	3.00	39.29	
	TCP 2	3.00	39.15	
	TCP 3	3.00	38.46	
$\overline{\text{ssthresh}}$ of a new TCP connection [segments]	TCP 1	44.00	76.37	
	TCP 2	44.00	76.09	
	TCP 3	44.00	75.39	
$\overline{\text{srtt}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.199	
	TCP 2	N/A	0.197	
	TCP 3	N/A	0.196	
$\overline{\text{rttvar}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.090	
	TCP 2	N/A	0.087	
	TCP 3	N/A	0.088	

Table 5.10 shows the simulation results for concurrent TCP connections of an ensemble. Compared to standard TCP, the EFCM controller achieves a gain for the overall mean throughput of approximately 9 % and a gain for the connection-oriented throughput of approximately 19 % for concurrent TCP connections. For these TCP connections also a remarkable mean fairness improvement of the EFCM controller can be observed.

Table 5.10: Simulation results of simulation 3, scenario 1 (concurrent TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, $\overline{I}_f$ is the mean fairness index, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

	no EFCM	EFCM	Δ
$\overline{T}_1$ of concurrent TCP connections [segments/second]	27.51	30.07	$+$
$\overline{T}_2$ of a concurrent TCP connection [segments/second]	25.97	30.93	$+$
$\overline{I}_f$ of concurrent TCP connections []	0.7907	0.8174	$+$

5.4.2 Unreliable last hop

Table 5.11 shows the simulation results for new TCP connections entering an ensemble. Compared to standard TCP, new WWW TCP connections benefit from the EFCM controller with a huge gain of approximately 76 % for the overall mean throughput and a large gain of approximately 30 % for the connection-oriented mean throughput.

Table 5.11: Simulation results of simulation 3, scenario 2 (new TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

		no EFCM	EFCM	Δ
$\overline{T}_1$ of new TCP connections [segments/second]	TCP 1	16.94	29.89	+
	TCP 2	16.89	29.96	+
	TCP 3	16.96	29.76	+
$\overline{T}_2$ of a new TCP connection [segments/second]	TCP 1	30.38	39.45	+
	TCP 2	30.18	39.13	+
	TCP 3	30.41	39.29	+
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	3.00	12.03	
	TCP 2	3.00	12.01	
	TCP 3	3.00	12.03	
$\overline{\text{ssthresh}}$ of a new TCP connection [segments]	TCP 1	44.00	40.26	
	TCP 2	44.00	40.26	
	TCP 3	44.00	40.26	
$\overline{\text{srtt}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.111	
	TCP 2	N/A	0.111	
	TCP 3	N/A	0.112	
$\overline{\text{rttvar}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.034	
	TCP 2	N/A	0.034	
	TCP 3	N/A	0.034	

Table 5.12 shows the simulation results for concurrent TCP connections of an ensemble. Compared to standard TCP, the EFCM controller achieves a large gain for the overall mean throughput of approximately 40 % and a large gain for the connection-oriented throughput of approximately 23 % for concurrent TCP connections. For these TCP connections also a considerable mean fairness improvement of the EFCM controller can be observed.

Table 5.12: Simulation results of simulation 3, scenario 2 (concurrent TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, $\overline{I}_f$ is the mean fairness index, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

	no EFCM	EFCM	Δ
$\overline{T}_1$ of concurrent TCP connections [segments/second]	21.48	29.90	$+$
$\overline{T}_2$ of a concurrent TCP connection [segments/second]	27.48	33.89	$+$
$\overline{I}_f$ of concurrent TCP connections []	0.8402	0.8743	$+$

5.5 Simulation 4: TCP Newreno, RTT $\geq$ 500 ms

5.5.1 Reliable last hop

Table 5.13 shows the simulation results for new TCP connections entering an ensemble. Compared to standard TCP, new WWW TCP connections benefit from the EFCM controller with a large gain of approximately 33 % for the overall mean throughput and a huge gain of approximately 62 % for the connection-oriented mean throughput.

Table 5.13: Simulation results of simulation 4, scenario 1 (new TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

		no EFCM	EFCM	Δ
$\overline{T}_1$ of new TCP connections [segments/second]	TCP 1	13.16	16.98	+
	TCP 2	12.97	17.34	+
	TCP 3	13.10	16.92	+
$\overline{T}_2$ of a new TCP connection [segments/second]	TCP 1	12.28	19.74	+
	TCP 2	12.14	19.81	+
	TCP 3	12.22	19.63	+
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	3.00	50.24	
	TCP 2	3.00	48.91	
	TCP 3	3.00	49.48	
$\overline{\text{ssthresh}}$ of a new TCP connection [segments]	TCP 1	44.00	88.57	
	TCP 2	44.00	87.00	
	TCP 3	44.00	87.88	
$\overline{\text{srtt}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.564	
	TCP 2	N/A	0.559	
	TCP 3	N/A	0.559	
$\overline{\text{rttvar}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.149	
	TCP 2	N/A	0.144	
	TCP 3	N/A	0.144	

Table 5.14 shows the simulation results for concurrent TCP connections of an ensemble. Compared to standard TCP, the EFCM controller achieves a large gain for the overall mean throughput of approximately 21 % and a large gain for the connection-oriented throughput of approximately 37 % for concurrent TCP connections. For these TCP connections also a remarkable mean fairness improvement of the EFCM controller can be observed.

Table 5.14: Simulation results of simulation 4, scenario 1 (concurrent TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, $\overline{I}_f$ is the mean fairness index, Δ denotes the statistical significance of the simulation results (’+’: EFCM controller is significantly better, ’=’: no significant difference between standard TCP and EFCM controller, ’-’: standard TCP is significantly better)

	no EFCM	EFCM	Δ
$\overline{T}_1$ of concurrent TCP connections [segments/second]	11.65	14.06	$+$
$\overline{T}_2$ of a concurrent TCP connection [segments/second]	10.45	14.35	$+$
$\overline{I}_f$ of concurrent TCP connections []	0.7757	0.8097	$+$

5.5.2 Unreliable last hop

Table 5.15 shows the simulation results for new TCP connections entering an ensemble. Compared to standard TCP, new WWW TCP connections benefit from the EFCM controller with a huge gain of approximately 72 % for the overall mean throughput and a huge gain of approximately 51 % for the connection-oriented mean throughput.

Table 5.15: Simulation results of simulation 4, scenario 2 (new TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

		no EFCM	EFCM	Δ
$\overline{T}_1$ of new TCP connections [segments/second]	TCP 1	6.30	10.87	+
	TCP 2	6.31	10.82	+
	TCP 3	6.29	10.77	+
$\overline{T}_2$ of a new TCP connection [segments/second]	TCP 1	7.78	11.82	+
	TCP 2	7.80	11.77	+
	TCP 3	7.79	11.78	+
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	3.00	12.56	
	TCP 2	3.00	12.56	
	TCP 3	3.00	12.61	
$\overline{\text{ssthresh}}$ of a new TCP connection [segments]	TCP 1	44.00	40.51	
	TCP 2	44.00	40.48	
	TCP 3	44.00	40.62	
$\overline{\text{srtt}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.501	
	TCP 2	N/A	0.502	
	TCP 3	N/A	0.503	
$\overline{\text{rttvar}}$ of a new TCP connection [seconds]	TCP 1	N/A	0.093	
	TCP 2	N/A	0.093	
	TCP 3	N/A	0.094	

Table 5.16 shows the simulation results for concurrent TCP connections of an ensemble. Compared to standard TCP, the EFCM controller achieves a large gain for the overall mean throughput of approximately 45 % and a large gain for the connection-oriented throughput of approximately 41 % for concurrent TCP connections. For these TCP connections also a considerable mean fairness improvement of the EFCM controller can be observed.

Table 5.16: Simulation results of simulation 4, scenario 2 (concurrent TCP connections) — $\overline{T}_1$ is the overall mean throughput, $\overline{T}_2$ is the connection-oriented mean throughput, $\overline{I}_f$ is the mean fairness index, Δ denotes the statistical significance of the simulation results ('+': EFCM controller is significantly better, '=': no significant difference between standard TCP and EFCM controller, '-': standard TCP is significantly better)

	no EFCM	EFCM	Δ
$\overline{T}_1$ of concurrent TCP connections [segments/second]	6.25	9.05	$+$
$\overline{T}_2$ of a concurrent TCP connection [segments/second]	6.38	9.00	$+$
$\overline{I}_f$ of concurrent TCP connections []	0.8571	0.8904	$+$

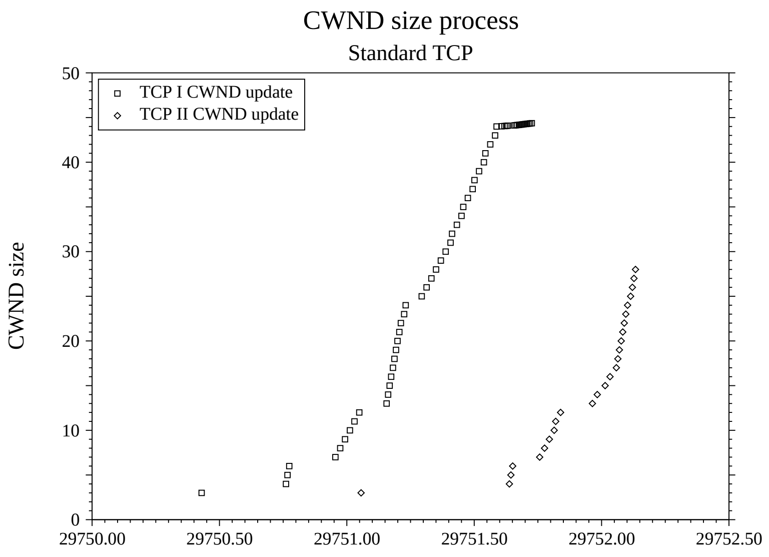
5.6 Summary and discussion

Evidently, the benefits and disadvantages of a a joint congestion control depend to a large degree on the system scenario. To summarize:

- Reliable last hop scenario: In all considered simulation scenarios the connection-oriented mean throughput of new EFCM-controlled WWW TCP connections is increased compared to new standard TCP connections. Thus, a new WWW TCP connection can expect a mean increase in its throughput if the end system is equipped with EFCM. If the overall mean throughput of new WWW TCP connections is considered, the EFCM controller reaches a slightly lower and a slightly larger performance in the first two simulation scenarios with minimum round trip times of 20 ms or 60 ms, respectively. But in the simulation scenarios with minimum round trip times above 60 ms, the EFCM controller is able to largely improve the overall mean throughput of new WWW TCP connections. A similar result can be observed for concurrent TCP connections. EFCM is able to considerably improve the connection-oriented throughput of these TCP connections in all simulation scenarios. But for simulation scenarios with lower minimum round trip times, EFCM reaches no significant gain of the overall mean throughput. The reason for this behavior of EFCM is in the pacing algorithm which seems to be too conservative for lower round trip times. In Appendix C, some possible adaptations of the EFCM pacing algorithm are described that allow a finer control of TCP connections in cases with a lower round trip time. In all simulation scenarios, a large fairness gain for concurrent TCP connections can be observed if the EFCM controller is used.

  Figures 5.1 and 5.2 show some typical processes of the congestion window and the sequence number for either two standard TCP Newreno connections (with a fairness index of $I_f = 0.6098$) or two EFCM-controlled TCP Newreno connections ($I_f = 0.9996$) of one ensemble over a reliable last hop and a minimum round trip time of 100 ms. In Figure 5.1, the concurrent TCP connections are separately controlled and obtain very different congestion windows. In Figure 5.2, the concurrent TCP connections are jointly controlled and obtain the same congestion windows. It can be seen that the concurrent TCP connections have no bursty sending behavior, since the current EFCM controller has a built-in pacing mechanism. During the slow start phase, some TCP segments are sent sporadically and not in burst of two TCP segments as it is done in standard TCP. This is the outcome of the ensemble ack clocking mechanism used in the EFCM controller.

- Unreliable last hop scenario: In all considered simulation scenarios and for both new WWW TCP connections and concurrent TCP connections, the EFCM controller achieves a large gain in the overall mean throughput compared to standard TCP. Therefore, a common congestion control for TCP connections can partly compensate for the negative influence of packet losses in the last hop of the network on the throughput of single TCP connections. The fairness gain of the EFCM controller for concurrent TCP connections is comparable to the results obtained in the simulations with a reliable last hop. These unexpected fairness results can be explained as follows: If two or more TCP connections of one ensemble are established in parallel during one measurement period for the fairness index, they have equal values for their jointly controlled TCP variables. This is ensured by the EFCM controller. But the EFCM controller can not prevent that these TCP connections observe different segment loss patterns during this



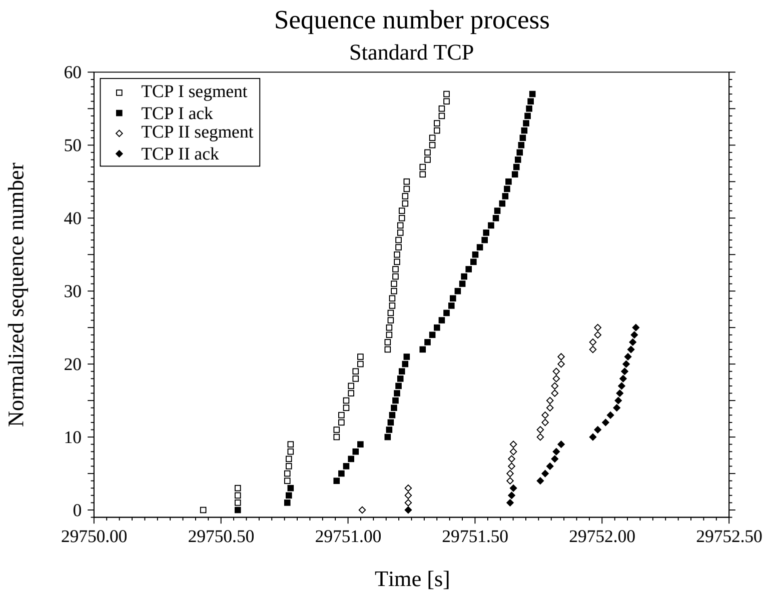
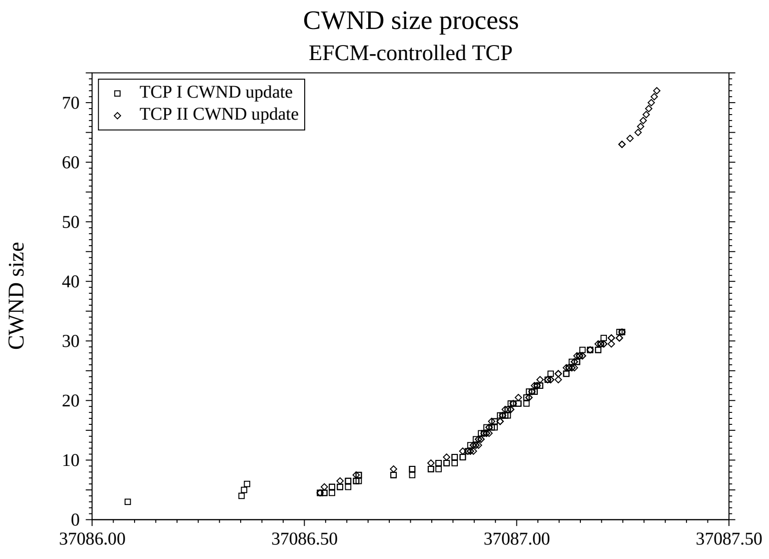


Figure 5.1: Standard TCP connections over a reliable last hop — □ and ◇ represent the first and second TCP connection of an ensemble, respectively

measurement period. Therefore, some TCP senders have to wait for acknowledgments of outstanding segments while other TCP senders can send their (new) segments. As a result, the concurrent TCP connections of an ensemble can have (very) different throughputs in a measurement period. Hence, the fairness gain obtained only by the basic EFCM algorithms (cf. Section 2.4.2) can be lower. But the ensemble ack clocking mechanism of the EFCM controller is able to eliminate this negative effect, since TCP connections which are waiting for TCP acknowledgments are allowed to send new TCP segments according to their interim increased congestion window.



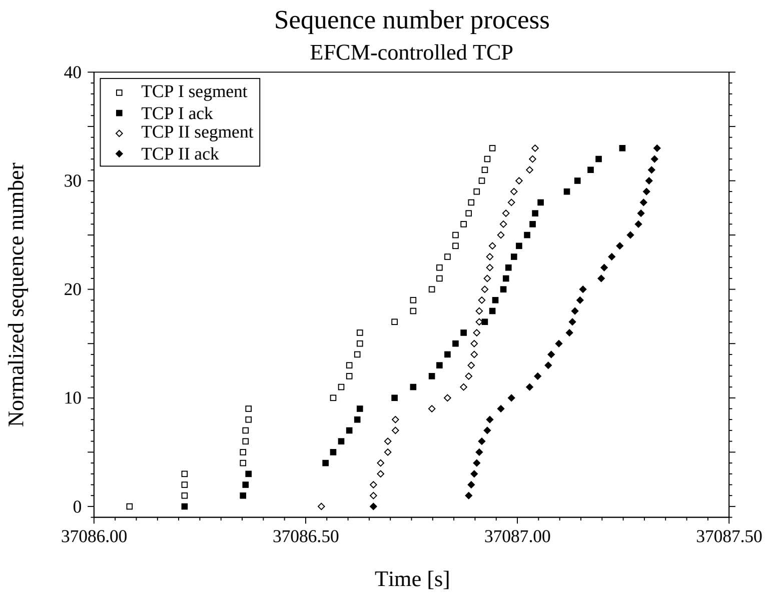
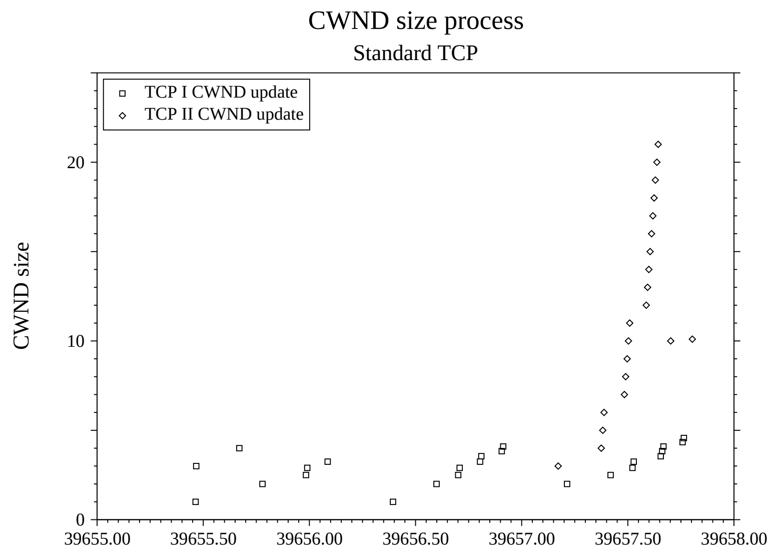


Figure 5.2: EFCM-controlled TCP connections over a reliable last hop — □ and ◇ represent the first and second TCP connection of an ensemble, respectively

For new WWW TCP connections, the gain for the overall mean throughput is nearly constant if the round trip time increases (the values are in the range of 68 % to 76 %). But the gain for the connnection-oriented mean throughput increases with the round trip time. Also concurrent TCP connections reach a larger gain (or a gain at all) for both throughput metrics if the round trip time increases.

Figures 5.3 and 5.4 show some typical processes of the congestion window and the sequence number for either two standard TCP Newreno connections ($I_f = 0.8112$) or two EFCM-controlled TCP Newreno connections ($I_f = 0.9977$) over an unreliable last hop with a mini-



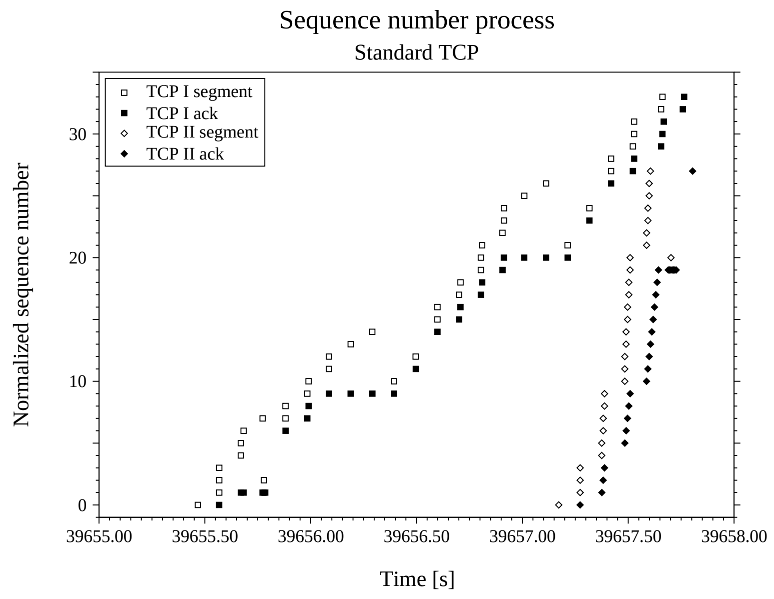


Figure 5.3: Standard TCP connections over an unreliable last hop — □ and ◇ represent the first and second TCP connection of an ensemble, respectively

mum round trip time of 100 ms. In Figure 5.3, the concurrent TCP connections are separately controlled and use different congestion windows. In Figure 5.4, the concurrent TCP connections are jointly controlled and use the same congestion windows. Due to packet losses some concurrent TCP connections have to wait for a longer period of time to send some new segments. But the negative influence of packet losses on the throughput of TCP connections is noticeably absorbed by a common congestion control.

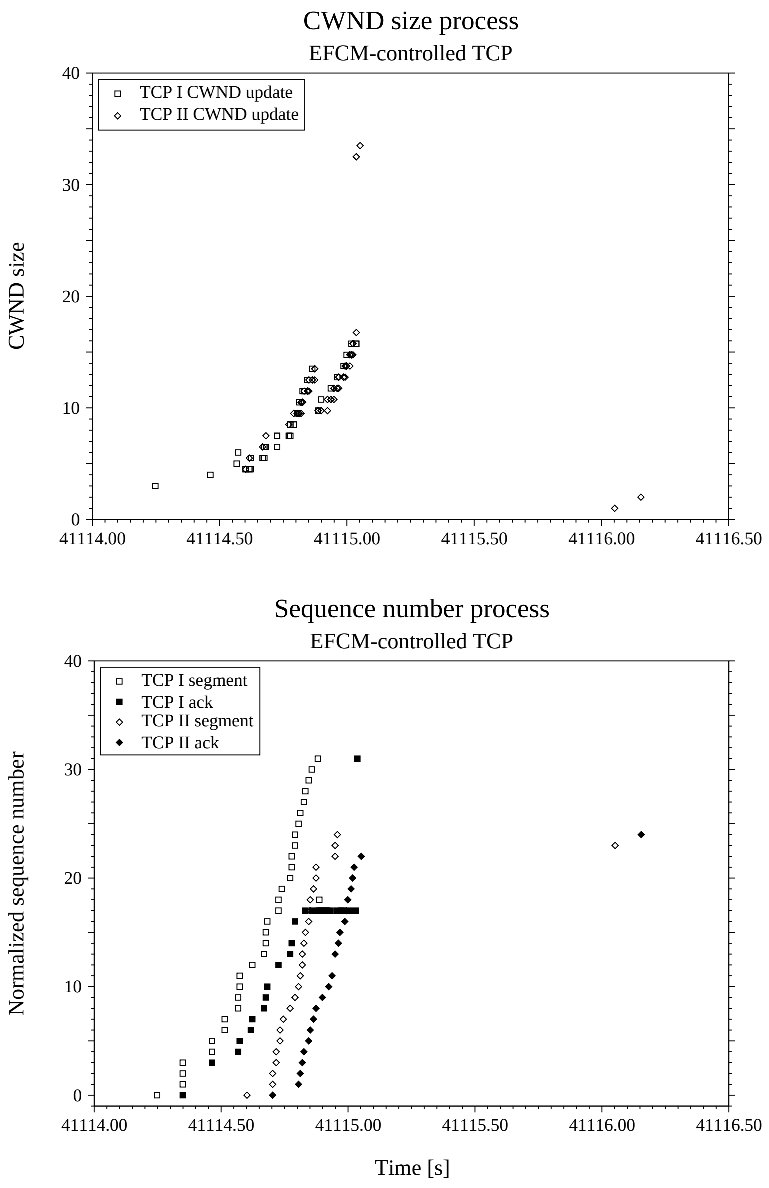


Figure 5.4: EFCM-controlled TCP connections over an unreliable last hop — □ and ◇ represent the first and second TCP connection of an ensemble, respectively

In the unreliable last hop simulations the EFCM controller significantly improves both throughput metrics compared to standard TCP. In these simulations the negative influence of packet losses on the throughput of TCP connections can be compensated for by sharing the decreased congestion window and slow start threshold of the TCP connection affected by packet losses among all ensemble members.

Figures 5.5 and 5.6 show the histograms of the fairness index for concurrent standard or EFCM-controlled TCP connections for both last hop scenarios in the simulations with a minimum round trip time of 100 ms. Especially for the reliable last hop scenario the positive influence of the EFCM controller on the fairness of concurrent TCP connections is obvious.

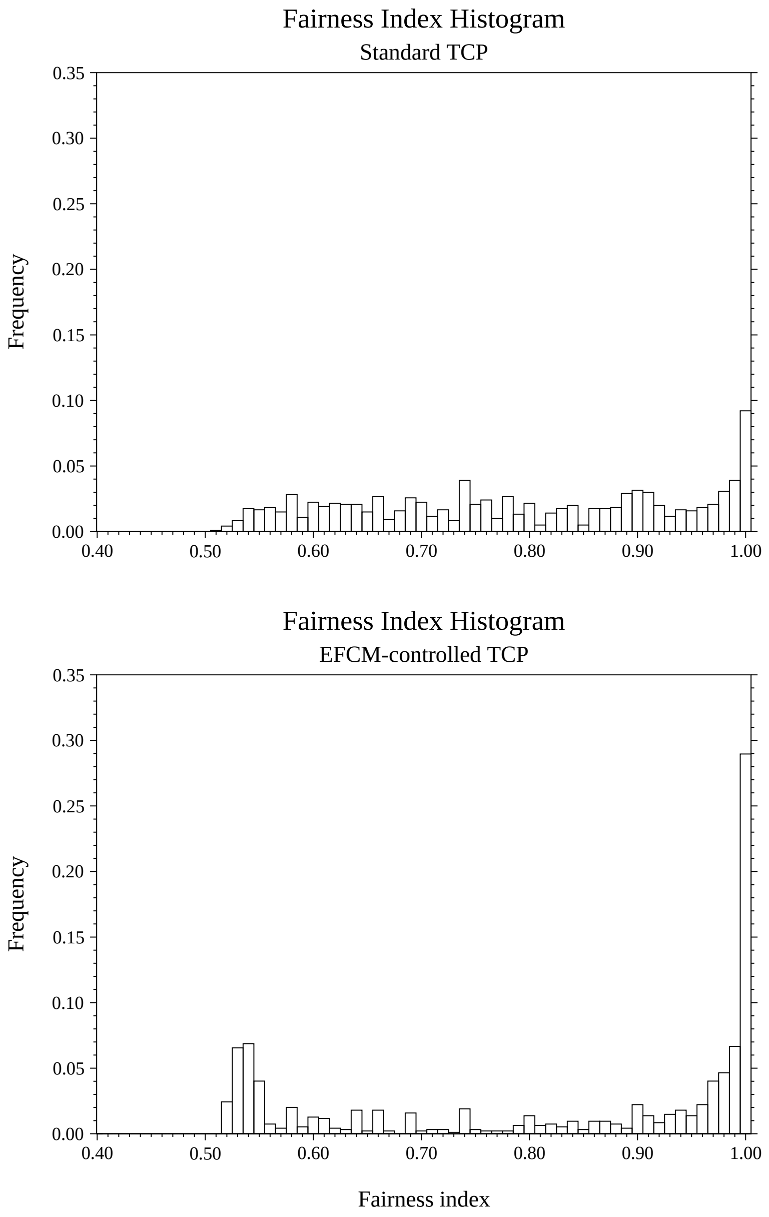


Figure 5.5: Fairness index histogram for concurrent standard or EFCM-controlled TCP connections over a reliable last hop

Figures 5.7 and 5.8 show the histograms of the throughput for concurrent standard or EFCM-controlled TCP connections for both last hop scenarios in the simulations with a minimum round trip time of 100 ms.

It is remarkable that in the first two simulations with an unreliable last hop the connection-oriented mean throughput (not the overall mean throughput!) of some TCP connections which are controlled or could be controlled by the EFCM approach is higher than in the simulation scenarios with a reliable

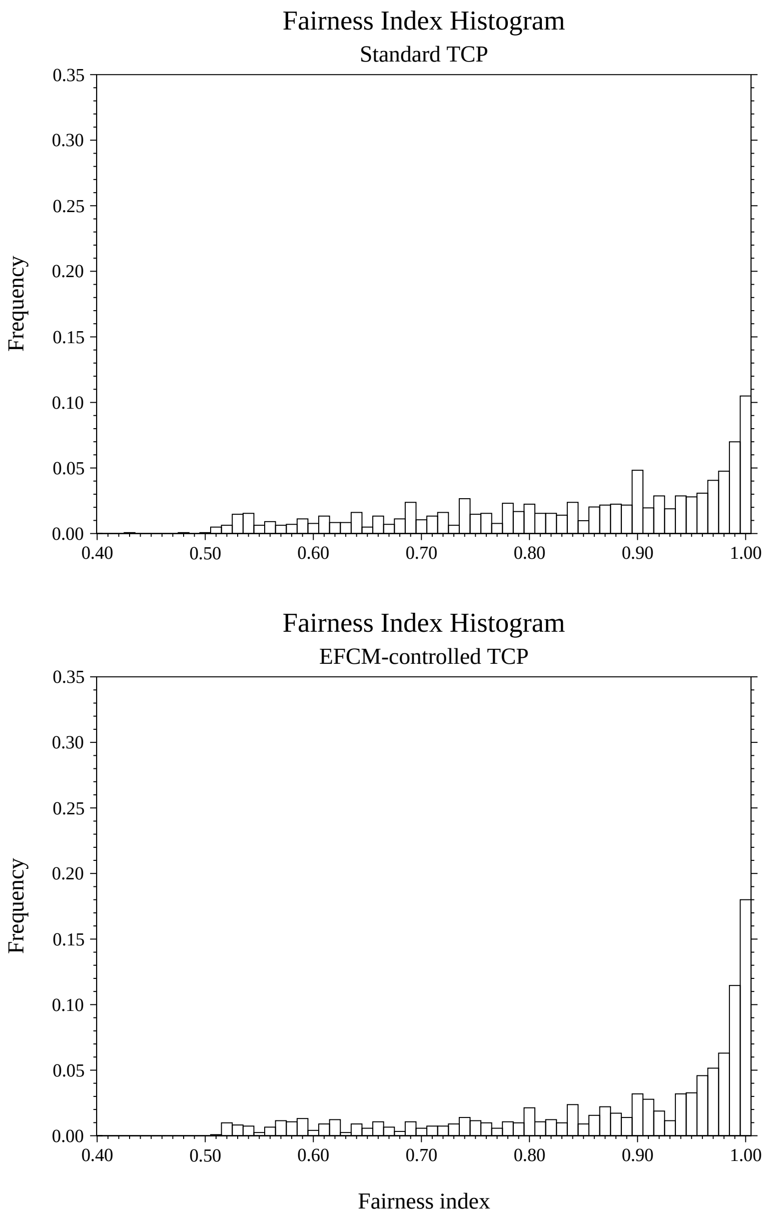


Figure 5.6: Fairness index histogram for concurrent standard or EFCM-controlled TCP connections over an unreliable last hop

last hop. But this—at first astonishing—result can be easily explained: The background traffic TCP connections with receivers in the receiver LAN are often affected by packet losses in the unreliable last hop. Due to the TCP congestion control algorithms these background traffic TCP connections reach a smaller overall sending window, allocate less bandwidth, and produce a substantially lower load in the receiver LAN. The other TCP connections with receivers in the receiver LAN are also affected by packet losses in the unreliable last hop. But from a single TCP connection's point of view, the lower load in the shared medium of the receiver LAN can sometimes compensate for and

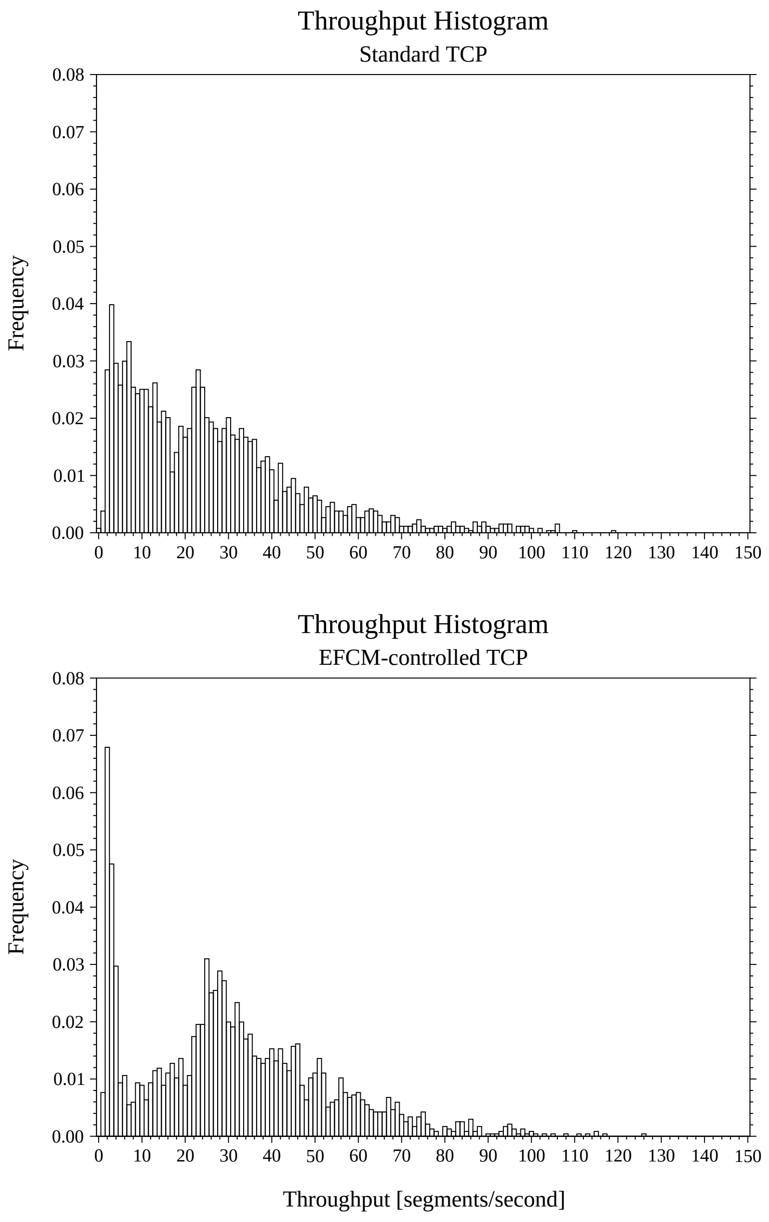


Figure 5.7: Throughput histogram for concurrent standard or EFCM-controlled TCP connections over a reliable last hop

even overcompensate for the in general negative influence of packet losses in an unreliable last hop on the mean throughput of a TCP connection. For example, some of the WWW TCP connections are not affected at all by packet losses during their whole lifetime. These TCP connections can highly benefit from the lower load in the receiver LAN and reach a higher connection-oriented mean throughput. Therefore, the observed higher connection-oriented mean throughput in the unreliable last hop scenario is only based on the lower load in the receiver LAN.

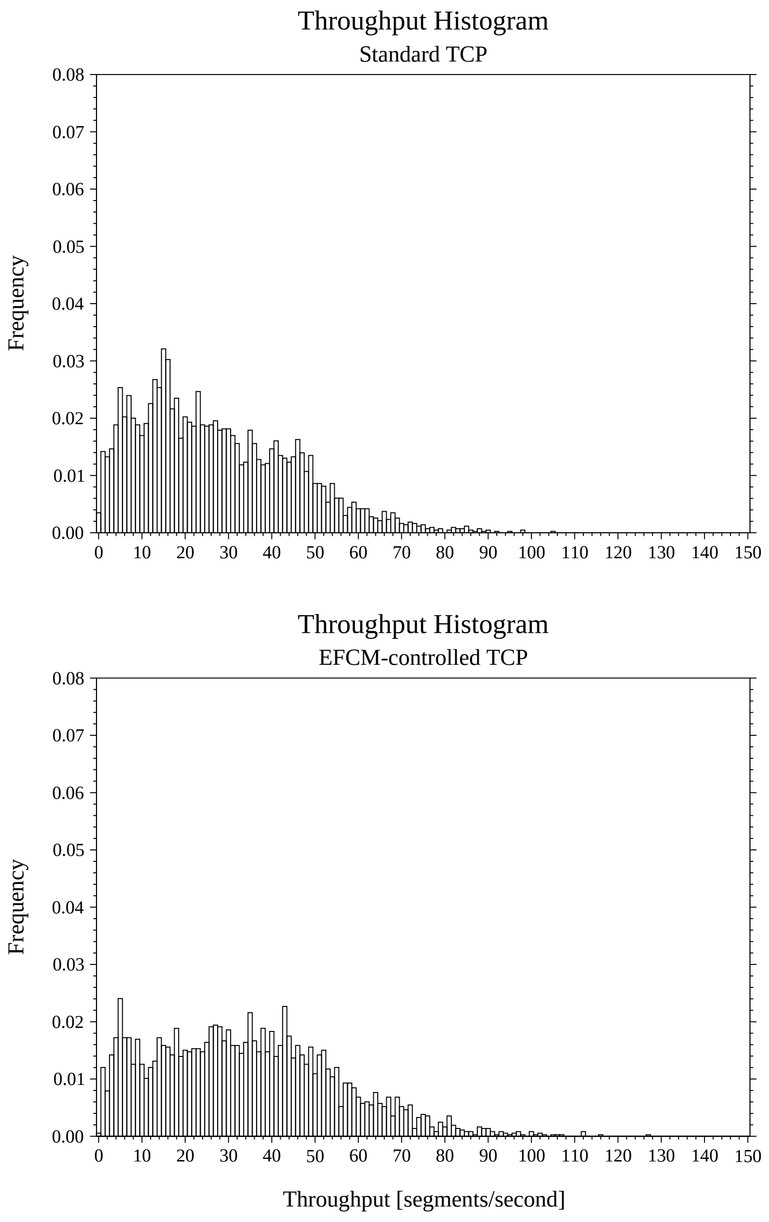


Figure 5.8: Throughput histogram for concurrent standard or EFCM-controlled TCP connections over an unreliable last hop

In Table 5.17 the performance of the EFCM controller derived from the simulation results is summarized.

A ’++’ or ’+’ denotes that the EFCM controller achieves a large gain or a gain (with respect to the performance metric shown in this row) compared to the standard TCP controller; a ’=’ denotes that no significant difference between the standard TCP and the EFCM controller can be observed; and a ’-’ denotes that standard TCP achieves a slight gain compared to the EFCM controller.

Table 5.17: Performance of the EFCM controller — $\overline{T}_{1,\mathrm{new,www}}$ is the overall mean throughput for new WWW TCP connections, $\overline{T}_{2,\mathrm{new,www}}$ is the connection-oriented mean throughput for new WWW TCP connections, $\overline{T}_{1,\mathrm{concurrent}}$ is the overall mean throughput for concurrent TCP connections, $\overline{T}_{2,\mathrm{concurrent}}$ is the connection-oriented mean throughput for concurrent TCP connections, and $\overline{I}_{f,\mathrm{concurrent}}$ is the mean fairness index for concurrent TCP connections (* = results differ for each of the considered minimum round trip times 20 ms, 60 ms, 100 ms, and 500 ms)

	Reliable last hop	Unreliable last hop
$\overline{T}_{1,\mathrm{new,www}}$	-,=,++,++*	++
$\overline{T}_{2,\mathrm{new,www}}$	+,+,+,++*	-,+,+,++*
$\overline{T}_{1,\mathrm{concurrent}}$	-,=,+,+*	+,++,++,++*
$\overline{T}_{2,\mathrm{concurrent}}$	+,+,+,++*	-,+,+,++*
$\overline{I}_{f,\mathrm{concurrent}}$	+	+

Chapter 6

Conclusion and Outlook

The simulation results show that the common congestion control approach EFCM improves the performance of standard TCP. If the EFCM controller is used in a scenario with a reliable last hop, a remarkably improved fairness can be observed between concurrent TCP connections of an ensemble. In addition, also the throughput of concurrent TCP connections is improved in most cases. In the unreliable last hop scenario, the EFCM approach significantly improves the throughput and fairness of concurrent and new TCP connections of an ensemble in nearly all cases. Particularly for such unreliable last hop scenarios the usage of the EFCM approach is highly recommended.

These benefits are achieved despite the fact that the EFCM controller investigated here uses relatively simple algorithms. In future investigations of the EFCM approach, some more complex algorithms will be considered, in the given simulation model with the here described parameter setting as well as in extended simulation models with the same and other parameter settings. For example, if one TCP connection of an ensemble does not use its whole fair share of the aggregated sending window in a given time interval then the other TCP connections of the ensemble should increase their sending window to reach the allowed aggregated sending window. The intermittently silent TCP connection should get a credit for this unused sending window to be allowed to have a sending window higher than the fair share in the future. This new controller mechanism can be implemented by using a utilization factor for each TCP connection in an ensemble. More generally, allowing the controller to adapt sending windows within an ensemble enables new form of application support. As an example, the transport layer could support interactive applications by assigning them a greater share of an ensemble's total bandwidth budget and reducing this share again when the application has no data to send (compensating other applications for their backlog).

Another important extension of the current EFCM controller is that the scope of an ensemble can be extended by including information from recently closed connections, assuming that the network conditions are slowly varying at best; the speed with which such information becomes inaccurate is an important parameter for such an undertaking. In a similar vein, the EFCM approach could also be used for handover TCP connections, i.e., TCP connections whose mobile receivers perform a handover. In this case, the handover must be non-transparent for the transport layer of the EFCM end system.

The current EFCM controller uses the IP address of the receiving end systems to determine whether a TCP connection belongs to a given ensemble or not. As explained in Section 1, this decision criterion based on IP addresses might not work in the real Internet, e.g., if mechanisms like Mobile IP or NAT are used. Therefore, further research must be done to investigate other and more appropriate decision criteria for the buildup of TCP ensembles in an EFCM controller.

The mid-term objective will be the extension of the EFCM approach to jointly control TCP and UDP data streams, i.e., to reach a TCP-friendly behavior of UDP data streams in an EFCM end system. In addition, it is planned to implement a framework for a common congestion controller into the linux kernel to evaluate the performance of the EFCM controller by measurements in the Internet environment.

Appendix A

Complexity of the current EFCM controller

The complexity of a controller can be expressed using the O-calculus [6] that gives an asymptotic upper bound of a complexity metric, e.g., the time or space consumption of an algorithm or a controller. Here, the time and space complexity of the EFCM controller is compared with standard TCP by using an implementation of the EFCM controller which is optimized for time consumption.

Let v be the number of jointly controlled TCP variables of the EFCM controller. For v_{fs} of these TCP control variables a fair share calculation, e.g., for the congestion window or the slow start threshold, and for the remaining v_{w} TCP control variables a weighted calculation, e.g., for the smoothed round trip time and the round trip time variance, is used. Let n be the number of concurrent TCP connections which form an ensemble and are jointly controlled by the EFCM controller. For every change of one of the jointly controlled TCP variables of the EFCM controller, the additional time consumption of the EFCM controller compared with standard TCP can be estimated as:

$$\begin{aligned}
\text{\# Additions } (+,-) &= O(1) \\
\text{\# Multiplications } (\cdot,/) &= O(1) \\
\text{\# Assignments} &= O(n)
\end{aligned}$$

The considered implementation of the EFCM controller has an additional space consumption compared to standard TCP whose upper bound is in $O(n \cdot v_{\mathrm{fs}} + v_{\mathrm{w}})$ per ensemble. If an EFCM controller has to manage e ensembles at the same time the upper bound of the additional space consumption is in $O(e + \sum_{i=1}^{e} n_i \cdot v_{\mathrm{fs},i} + v_{\mathrm{w},i})$ where the additional e in the equation represents the space consumption of the search list containing the IP addresses of the receiving end systems of existing TCP connections that is used to determine whether a new TCP connection can join an existing ensemble or not. The overall additional time and space complexity of the EFCM controller compared to standard TCP is low. And this additional complexity, except for the search list, is only needed if the EFCM controller has to manage TCP connections in an ensemble.

Appendix B

Statistical Evaluation

B.1 Statistical evaluation method

For the simulated scenarios with either a reliable or an unreliable last hop the results of the standard TCP (no EFCM) controller are statistically compared with the results of the EFCM controller.

The statistical evaluation method used for this comparison is called the t-test for unpaired observations of two alternatives and is described in detail in [5]. The main idea of this method is to compute a confidence interval for the difference of the mean values of both alternatives for a given confidence level. Then the decision criterion is:

- If the confidence interval includes zero, then the two alternatives can not be distinguished.
- If the confidence interval is above/below zero, then the first/second alternative is the better one.

Tests with confidence intervals give not only a yes-no answer like other hypothesis tests, they also give an answer to the question how precise the decision is. A narrow confidence interval indicates that the precision of the decision is high whereas a wide confidence interval indicates that the precision of the decision is rather low.

This t-test for unpaired observations of two alternatives is used for the statistical evaluation of the simulation results for both the overall mean throughput ($\overline{T}_1$) and the connection-oriented mean throughput ($\overline{T}_2$) of new or concurrent TCP connections controlled by the standard TCP or the EFCM controller and the mean fairness index ($\overline{I}_f$) of concurrent TCP connections.

The values for this statistical evaluation are produced by five independent simulation runs for every last hop scenario in the simulation model.

B.2 Statistical evaluation results

In the following Tables B.1 to B.8 the statistical evaluation of the simulation results are shown. For each confidence interval also the confidence level (0.90, 0.95 or 0.99) is depicted. If the simulation results for the standard TCP and the EFCM controller are not significantly different even for the confidence level 0.90, then the confidence interval for the confidence level 0.90 is stated.

Table B.1: Statistical evaluation of the simulation results of simulation 1, scenario 1

		no EFCM ↔ EFCM
$\overline{T}_1$ of new TCP connections [$1-\alpha$: conf()]	TCP 1	$0.90 : (-\ 0.73, +\ 2.28)$
	TCP 2	$0.95 : (+\ 0.45, +\ 4.39)$
	TCP 3	$0.95 : (+\ 0.01, +\ 3.49)$
$\overline{T}_2$ of a new TCP connection [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 5.77, -\ 2.37)$
	TCP 2	$0.99 : (-\ 4.64, -\ 1.77)$
	TCP 3	$0.99 : (-\ 4.95, -\ 1.70)$
$\overline{T}_1$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.95 : (+\ 0.25, +\ 1.99)$
$\overline{T}_2$ of a concurrent TCP connection [$1-\alpha$: conf()]		$0.99 : (-\ 3.79, -\ 1.94)$
$\overline{I}_f$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 0.05, -\ 0.03)$

Table B.2: Statistical evaluation of the simulation results of simulation 1, scenario 2

		no EFCM ↔ EFCM
$\overline{T}_1$ of new TCP connections [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 19.76, -\ 15.84)$
	TCP 2	$0.99 : (-\ 18.90, -\ 13.90)$
	TCP 3	$0.99 : (-\ 17.37, -\ 13.04)$
$\overline{T}_2$ of a new TCP connection [$1-\alpha$: conf()]	TCP 1	$0.90 : (-\ 1.49, +\ 1.03)$
	TCP 2	$0.90 : (+\ 0.27, +\ 3.15)$
	TCP 3	$0.90 : (+\ 0.17, +\ 2.73)$
$\overline{T}_1$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 10.35, -\ 7.67)$
$\overline{T}_2$ of a concurrent TCP connection [$1-\alpha$: conf()]		$0.99 : (+\ 1.01, +\ 3.82)$
$\overline{I}_f$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 0.05, -\ 0.03)$

Table B.3: Statistical evaluation of the simulation results of simulation 2, scenario 1

		no EFCM ↔ EFCM
$\overline{T}_1$ of new TCP connections [$1-\alpha$: conf()]	TCP 1	$0.90 : (-\ 2.31, +\ 0.05)$
	TCP 2	$0.90 : (-\ 2.10, +\ 0.20)$
	TCP 3	$0.90 : (-\ 1.76, +\ 0.16)$
$\overline{T}_2$ of a new TCP connection [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 7.18, -\ 5.28)$
	TCP 2	$0.99 : (-\ 6.89, -\ 5.06)$
	TCP 3	$0.99 : (-\ 6.63, -\ 4.90)$
$\overline{T}_1$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.90 : (-\ 0.61, +\ 0.76)$
$\overline{T}_2$ of a concurrent TCP connection [$1-\alpha$: conf()]		$0.99 : (-\ 3.72, -\ 2.39)$
$\overline{I}_f$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 0.05, -\ 0.03)$

Table B.4: Statistical evaluation of the simulation results of simulation 2, scenario 2

		no EFCM ↔ EFCM
$\overline{T}_1$ of new TCP connections [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 16.59, -\ 13.58)$
	TCP 2	$0.99 : (-\ 14.84, -\ 12.77)$
	TCP 3	$0.99 : (-\ 17.54, -\ 13.29)$
$\overline{T}_2$ of a new TCP connection [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 7.70, -\ 6.08)$
	TCP 2	$0.99 : (-\ 7.34, -\ 5.56)$
	TCP 3	$0.99 : (-\ 8.94, -\ 6.30)$
$\overline{T}_1$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 9.83, -\ 8.31)$
$\overline{T}_2$ of a concurrent TCP connection [$1-\alpha$: conf()]		$0.99 : (-\ 5.82, -\ 4.42)$
$\overline{I}_f$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 0.04, -\ 0.03)$

Table B.5: Statistical evaluation of the simulation results of simulation 3, scenario 1

		no EFCM ↔ EFCM
$\overline{T}_1$ of new TCP connections [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 5.07, -\ 2.60)$
	TCP 2	$0.99 : (-\ 5.78, -\ 3.06)$
	TCP 3	$0.99 : (-\ 5.78, -\ 2.43)$
$\overline{T}_2$ of a new TCP connection [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 10.47, -\ 9.05)$
	TCP 2	$0.99 : (-\ 10.48, -\ 9.32)$
	TCP 3	$0.99 : (-\ 10.31, -\ 8.83)$
$\overline{T}_1$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 3.30, -\ 1.81)$
$\overline{T}_2$ of a concurrent TCP connection [$1-\alpha$: conf()]		$0.99 : (-\ 5.45, -\ 4.47)$
$\overline{I}_f$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 0.03, -\ 0.02)$

Table B.6: Statistical evaluation of the simulation results of simulation 3, scenario 2

		no EFCM ↔ EFCM
$\overline{T}_1$ of new TCP connections [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 13.68, -\ 12.21)$
	TCP 2	$0.99 : (-\ 14.20, -\ 11.95)$
	TCP 3	$0.99 : (-\ 13.66, -\ 11.93)$
$\overline{T}_2$ of a new TCP connection [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 9.81, -\ 8.34)$
	TCP 2	$0.99 : (-\ 9.53, -\ 8.38)$
	TCP 3	$0.99 : (-\ 9.50, -\ 8.27)$
$\overline{T}_1$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 8.96, -\ 7.89)$
$\overline{T}_2$ of a concurrent TCP connection [$1-\alpha$: conf()]		$0.99 : (-\ 6.85, -\ 5.96)$
$\overline{I}_f$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 0.04, -\ 0.03)$

Table B.7: Statistical evaluation of the simulation results of simulation 4, scenario 1

		no EFCM ↔ EFCM
$\overline{T}_1$ of new TCP connections [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 4.28, -\ 3.35)$
	TCP 2	$0.99 : (-\ 5.04, -\ 3.71)$
	TCP 3	$0.99 : (-\ 4.40, -\ 3.23)$
$\overline{T}_2$ of a new TCP connection [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 7.71, -\ 7.22)$
	TCP 2	$0.99 : (-\ 8.05, -\ 7.30)$
	TCP 3	$0.99 : (-\ 7.70, -\ 7.13)$
$\overline{T}_1$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 2.63, -\ 2.18)$
$\overline{T}_2$ of a concurrent TCP connection [$1-\alpha$: conf()]		$0.99 : (-\ 4.10, -\ 3.70)$
$\overline{I}_f$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 0.04, -\ 0.03)$

Table B.8: Statistical evaluation of the simulation results of simulation 4, scenario 2

		no EFCM ↔ EFCM
$\overline{T}_1$ of new TCP connections [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 4.77, -\ 4.44)$
	TCP 2	$0.99 : (-\ 4.70, -\ 4.39)$
	TCP 3	$0.99 : (-\ 4.70, -\ 4.30)$
$\overline{T}_2$ of a new TCP connection [$1-\alpha$: conf()]	TCP 1	$0.99 : (-\ 4.19, -\ 3.96)$
	TCP 2	$0.99 : (-\ 4.11, -\ 3.87)$
	TCP 3	$0.99 : (-\ 4.16, -\ 3.83)$
$\overline{T}_1$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 2.88, -\ 2.71)$
$\overline{T}_2$ of a concurrent TCP connection [$1-\alpha$: conf()]		$0.99 : (-\ 2.70, -\ 2.54)$
$\overline{I}_f$ of concurrent TCP connections [$1-\alpha$: conf()]		$0.99 : (-\ 0.04, -\ 0.03)$

B.3 Summary of the statistical evaluation

The statistical evaluation of the simulation results show for the throughput metrics rather wide confidence intervals. For new WWW TCP connections the EFCM controller reaches significantly higher connection-oriented mean throughputs in all considered simulation scenarios and significant higher overall throughputs in all considered simulation scenarios with a minimum round trip time larger than 60 ms. A similar result can be observed for both throughput metrics for concurrent TCP connections.

In all simulation scenarios the fairness index of concurrent EFCM-controlled TCP connections is significantly higher than the fairness index of concurrent standard TCP connections.

Appendix C

Current developments of the EFCM controller

The pacing algorithm of the current EFCM controller is evaluated with different values for the factor α. The preliminary result of these simulations is that the factor α should be dynamically adapted to the calculated smoothed round trip time of the ensemble. For small minimum round trip times less than 60 ms the factor α should be set to a small value, e.g., $\alpha = 1$. For a larger minimum round trip time the factor α should be set to a larger value, e.g., $\alpha = 2$. It might be possible to derive a function for the factor α that provides a better adaptation of the pacing algorithm of an ensemble on the current calculated smoothed round trip time, i.e.,

$$\alpha = \alpha(\text{SRTT}(t))$$

This function should be a monotonic increasing function in the range from 1 to 2.

Acknowledges

The work presented in this technical report has been partially supported by a research contract with Ericsson Research.

Bibliography

[1] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's initial window. RFC 3390, October 2002.

[2] L. Eggert, T. Henderson, and J. Touch. Effects of ensemble TCP. *ACM SIGCOMM Computer Communication Review*, 30(1):15–29, 2000.

[3] K. Fall and K. Varadhan. *The ns Manual*. VINT Project, http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf, 2001.

[4] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. RFC 2582, April 1999.

[5] R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley & Sons, 1991.

[6] K. Mehlhorn. *Data Structures and Efficient Algorithms*. EATCS Monographs. Springer, 1984.

[7] A. Reyes-Lecuona, E. Gonzáles-Parada, E. Casilari, and J. Casasola. A page-oriented WWW traffic model for wireless system simulations. In *Proceedings ITC 16*, pages 1271–1280, 1999.

[8] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM Transactions on Networking*, 10(3):381–395, June 2002.

[9] M. Savorić. Identifying and evaluating the potential of reusing network information from different flows. Technical report, TKN-01-019, http://www-tkn.ee.tu-berlin.de/publications/papers/ccc_tr.pdf, 2001.

[10] M. Savorić. The TCP control block interdependence in fixed networks — some performance results. In *Proceedings QOFIS 2001*, LNCS 2156, pages 261–272, 2001.

[11] M. Savorić and H. Karl. Performance evaluation of a common congestion controller for TCP connections. Technical report, TKN-02-005, http://www-tkn.ee.tu-berlin.de/publications/papers/efcm_tr.pdf, 2002.

[12] M. Savorić and H. Karl. Performance evaluation of an improved common congestion controller for TCP connections — new simulation results. Technical report, TKN, http://www-tkn.ee.tu-berlin.de/publications/papers/efcm_tr_3.pdf, 2003.

[13] M. Savorić and H. Karl. Validation of some ensemble flow congestion management control algorithms. Technical report, TKN, http://www-tkn.ee.tu-berlin.de/publications/papers/efcm_av_tr.pdf, 2003.

[14] M. Savorić, H. Karl, and A. Wolisz. The TCP control block interdependence in fixed networks — new performance results. *Computer Communications*, 26(4):366–375, February 2003.

[15] J. Touch. TCP control block interdependence. RFC 2140, 1997.