Technical University Berlin

Telecommunication Networks Group

Improving Congestion Control in IP-based Networks Using Feedback from Routers

Michael Savorić

savoric@ee.tu-berlin.de

Berlin, July 2004

TKN Technical Report TKN-04-008

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

In the current Internet, TCP's congestion control is performed end-to-end possibly assisted by simple and limited congestion feedback mechanisms in routers, e.g., Explicit Congestion Notification (ECN) or Random Early Detection (RED). In order to control network traffic load more accurately by reducing or even preventing congestion in (parts of) the network and improving the overall performance of TCP flows, some more complex and more powerful feedback mechanisms in routers have been developed.

This technical report describes the most important of such Router Congestion Feedback (RCF) approaches based on network-information sharing (NIS) between routers and end systems in detail. In addition, the properties and functionalities of these approaches are compared to find potential candidates that can be used to improve congestion control in future IP-based networks.

Keywords: Distributed Congestion Management, Network-Information Sharing, Router Congestion Feedback, TCP, Congestion Control, Flow Control

Contents

Chapter 1

Introduction

TCP's current congestion control is based on an end-to-end collection and evaluation of network information on a per-connection basis. But even in today's Internet, the performance of such a congestion control might be far from optimality due to the insufficient or outdated information about the current network conditions collected in the TCP instances running in the end systems. This problem will be strengthen in future IP-based networks where the transparent integration of different wired and wireless access technologies with their specific bandwidth, delay, and error characteristics will play an important role. Therefore, it might be advantageous in terms of improving the overall performance of the Internet to assist TCP's end-to-end congestion control with more appropriate mechanisms based on network-information sharing (NIS). This chapter describes in general the functionalities and properties of a specific class of NIS approaches for TCP to improve the performance of TCP in the current Internet and also in future IP-based networks.

1.1 Fundamental Types of Network-Information Sharing for TCP

Two fundamental types of NIS approaches can be used to assist TCP's congestion control:

(1) network information can be shared between senders of TCP connections in an Internet end system,

(2) network information can be shared between Internet routers and end systems.

In general, approaches of both NIS types can be combined to develop a hybrid NIS approach in the Internet. The focus of this technical report is on the second type of NIS approaches.

These NIS approaches are able to improve the performance of those TCP connections which traverse NIS-capable routers in the network. Thus, dependent on the number and distribution of

RCF-capable (bottleneck) routers in the network only a few up to all TCP connections in the network or in parts of the network can benefit from such a NIS approach.

Chapter 2

Network-Information Sharing between Internet Routers and End Systems

In order to improve the overall performance of data streams in an IP-based network, network information can be shared between Internet routers and end systems, i.e., these approaches share network information which is distributed over the whole network. All of these NIS approaches are able to implicitly or explicitly send network information from the routers back to the senders (cf. Figure 2.1).

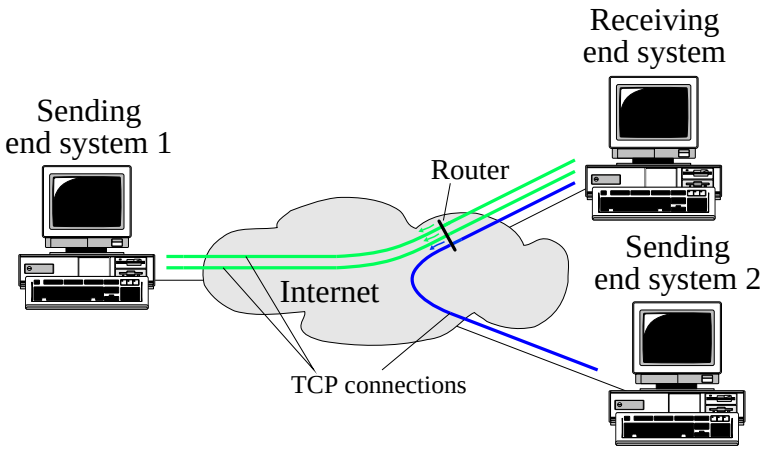

Figure 2.1: Network-information sharing between Internet routers and end systems

2.1 Router Congestion Feedback

This network information can include information about the current load in the routers, the load that can be currently accepted by the routers, and/or information about the current congestion status of the routers. Thus, some of these NIS approaches can be finer specified as router load feedback (RLF) or router congestion feedback (RCF) approaches. In the following, the abbreviation RCF is used for both variants of NIS approaches between routers and end systems as a synonym.

Two of such RCF approaches have been already standardized: Random Early Detection (RED) [6, 2] and Explicit Congestion Notification (ECN) [20]. But these two mechanisms provide only a implicit or explicit single-bit congestion information (no congestion / congestion) as a router feedback for the TCP senders. Therefore, with these two NIS approaches a TCP sender is not able to adequately react on the current load conditions in the routers providing these simple RCF mechanisms. And since these two router feedback mechanisms are only designed for that TCP senders could faster react on (impending) congestion in a router, their performance gain is limited to this case. Since ECN and RED are not able to signal to the TCP senders that additional bandwidth is available in the network, the TCP senders are not able to faster increase their sending window as they can do with the standard TCP congestion controller.

There exist other RCF approaches that are able to finer control the sending window of TCP connections in the case of (impending) congestion and also in the case of additional available bandwidth in parts of the network. The most important of these RCF approaches will be later described in detail. But in the following, the main tasks and design principles of such more sophisticated RCF approaches are generally described. In addition, an example is given that shows the potential benefit these RCF approaches can have compared to standard TCP.

2.1.1 Main Tasks

Three main tasks have to be performed by a RCF approach:

(1) **Calculation of feedback information:** Each router is equipped with a RCF controller that is able to measure the current load in the router and calculates feedback information according to this load information.

(2) **Transport of feedback information:** The feedback information calculated by the RCF controller in a router has to be transferred back to the senders.

(3) **Reaction on feedback information:** The senders receive the feedback information and adjust their sending window according to this feedback information.

2.1.2 Design Principles and Aspects

This section considers in detail the main design principles and aspects related to RCF approaches that share network information between Internet routers and end systems.

2.1.2.1 Direction of Network-Information Transfer

Some RCF approaches, e.g., RED and ECN, share network information only in one direction from the routers to the senders. Therefore, these RCF approaches are classified as one-way RCF approaches.

Other RCF approaches are based on a bi-directional network-information sharing between routers and senders. In the first direction, the TCP senders send network information, e.g., information about their current congestion-control status, to the routers. With these information and together with internal router status information, the routers are able to generate more appropriate congestion-control feedback information for each sender on a per-flow basis. This can be done without storing per-flow information inside the routers, a criterion that is necessary to develop a scalable RCF approach. In the second direction, this congestion-control feedback information is sent back from the routers to the senders. Such RCF approaches are classified as two-way RCF approaches.

2.1.2.2 Mechanisms to Transfer Network Information

If network information should be shared between end systems and routers it is important to consider how this can be done. The following list describes three possible mechanisms to implement such RCF approaches in the Internet:

(1) Network information can be transferred by using existing transport-protocol mechanisms of TCP. This is typically done in one-way RCF approaches where the congestion-control feedback information from routers is sent back to the TCP senders. These approaches have the advantage that they are transparent for the TCP instances in the end systems. But in general their expected performance gain is rather limited compared to the other two mechanisms.

(2) Network information can be also transferred encapsulated in additional protocol messages. These new protocol messages can be transferred piggy-back in standard protocol messages, e.g., as a new TCP header option, or in separate new PDUs. In addition, also the semantic of existing protocol messages can be changed to provide new RCF mechanisms. In all cases, this approach requires that the standard TCP algorithms in the end systems have to be (slightly) changed to support the new RCF functionalities. With this mechanism it is possible to provide both one-way and two-way RCF approaches with a much higher expected performance gain compared to RCF approaches based on mechanism (1).

(3) The most complex mechanism to implement a one-way or two-way RCF approach is to develop a complete new transport protocol, possibly based on TCP. It can be expected that the performance gain of such a RCF approach is at least as high as that of RCF approaches based on mechanism (2). But this advantage in terms of improving the performance is combined with huge problems regarding the deployability of such an approach in the current Internet.

2.1.2.3 Feedback Information

RCF approaches can implicitly or explicitly share network information between Internet routers and end systems. In addition, this network information can be shared once, i.e., at the setup of a new data stream, or continuous during the whole lifetime of a data stream.

Feedback information is used to control the load that senders can put into the network. Some RCF approaches are designed only to restrict this load while other RCF approaches have also the capability to increase this load much faster than it is possible with standard congestion-control mechanisms used in the current Internet.

2.1.2.4 Complexity

Another important design aspect of a RCF approach is its complexity. This includes the time and space consumption of the RCF controller located in the router(s) and of the possibly adapted congestion controller located in the end systems. In addition, also the protocol overhead that is necessary to transfer the shared network information between the end systems and the router(s) is of interest.

The following Table 2.1 summarizes the design principles and aspects of network-information sharing approaches between Internet routers and end systems.

Table 2.1: Design principles and aspects of network-information sharing approaches between Internet routers and end systems

Design principle / aspect	Possibilities
Network-information sharing direction	one-way, two-way
Transfer of network information	transparent, non-transparent
Feedback information	implicit / explicit, one-time / continuous
Complexity	time and space in routers / end systems, protocol overhead

2.2 Potential of Network-Information Sharing between Internet Routers and End Systems

The following Figure 2.2 shows a snapshot of a simple simulation scenario in which two TCP connections traverse a bottleneck router whose RCF-capabilities are either switched off or switched on.

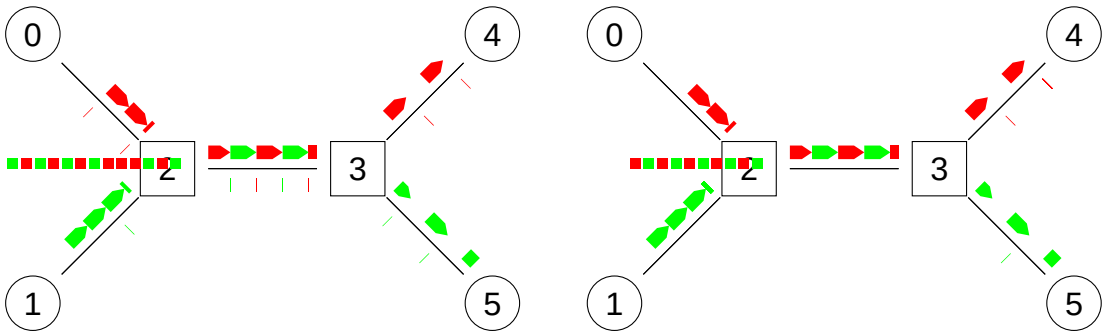

Figure 2.2: Potential of network-information sharing between Internet routers and end systems (simulation with non-RCF-capable routers on the left side, simulation with RCF-capable routers on the right side)

The senders of these two TCP connections are located in the end systems 0 and 1, their receivers are located in the end systems 4 and 5. The segments and acknowledgments of both TCP connections traverse the routers 2 and 3 in the network. Since all links in the network scenario have equal properties regarding the bandwidth (10 Mbps) and propagation delay (5 ms), router 2 is the bottleneck router in this network scenario. Its current queue length is also pictured in the figure. Both TCP connections start sending their segments into the network at the same time. If the RCF-capabilities in the bottleneck router are switched off, packet losses due to congestion in this router are likely to occur. This is the outcome of the uncontrolled grow of the queue in the bottleneck router up to its maximum queueing capacity of 19 packets. Since these packet losses are not equally shared among the two TCP connections over the simulated time, they obtain largely different mean throughputs of 102.00 and 496.00 segments per second during the simulated time. Thus, fairness between similar TCP connections cannot be guaranteed by the standard TCP congestion control as it is used in the current Internet.

But if the RCF-capabilities in the bottleneck router are enabled, the two TCP senders receive information about the load in this router that they can adjust their current sending window on this load information. Then, the queue length of the bottleneck router is accurately controlled that packet losses are very rare events. As a result, both TCP connections obtain comparable mean throughputs of 244.00 and 276.00 segments per second during the simulated time. It is obvious that these two TCP

connections are much more fairly treated by the more sophisticated TCP congestion control based on network-information sharing between Internet routers and end systems compared to the standard TCP congestion control.

2.3 Overview

The algorithms, mechanisms, and properties of the most important NIS approaches based on router congestion feedback (RCF) are described in the following Chapters 3 to 9. The existing RCF approaches considered in these chapters are: Explicit Window Adaptation (EWA) [11, 12], Explicit Control Protocol (XCP) [15, 14], Core-Stateless Fair Queueing (CSFQ) [27], Core-Stateless Fair Bandwidth Allocation for TCP (FBA-TCP) [13], and TCP Quick-Start (QS-TCP) [9].

EWA, CSFQ, and FBA-TCP can be classified as continuous one-way RCF approaches for TCP. While EWA and FBA-TCP use explicit congestion feedback from routers, CSFQ is based on an implicit network-information sharing using packet losses. XCP is a continuous two-way RCF approach based on a new transport protocol with a changed congestion control compared to TCP. Finally, QS-TCP is a one-time two-way RCF approach. It is developed to enable TCP senders to start with larger initial congestion windows according to the feedback information obtained from QS-TCP-capable routers.

In addition to these existing RCF approaches, two new continuous one-way RCF approaches are considered in detail. The first new approach, Fuzzy Explicit Window Adaptation (FEWA), is based on EWA. It modifies parts of the feedback-calculation algorithms of EWA in the routers. The second new RCF approach, Enhanced TCP (ETCP), uses the feedback information from FEWA-capable routers for an adapted congestion control in the TCP senders.

The properties and functionalities of all these RCF approaches are summarized and compared in Chapter 10. In addition, the applicability of these RCF approaches in the current Internet environment is investigated and compared in Chapter 11. Based on these comparisons, among the considered RCF approaches some potential candidates for an improved congestion control in the current Internet and in future IP-based networks are selected.

Chapter 3

Explicit Window Adaptation (EWA)

The EWA approach [11, 12] has been developed to explicitly inform the senders of TCP connections about the currently available bandwidth over a bottleneck link in a transparent way by using TCP's built-in flow-control mechanism. Thus, the source codes of a TCP sender and a TCP receiver are kept unchanged. In this section, the basic algorithm of EWA is described, it is shown how EWA can be deployed in the current Internet environment, some shortcomings of EWA are explained, and some possible improvements of EWA are depicted.

3.1 EWA Algorithm

After every measurement interval i with a fixed duration dependent on the bandwidths in the links the EWA-capable router is connected with, e.g., 10 ms, a router with EWA-capabilities measures its current queue length Q_i and computes the current mean queue length $\overline{Q}_i$. Q_i, $\overline{Q}_i$, and the second-last computed mean queue length $\overline{Q}_{i-1}$ are then used to calculate a new sending window for each TCP connection traversing this router:

$$\text{sending window} = \max\{\text{MSS}, \alpha \cdot \log_2(B - Q_i) \cdot \text{MSS}\} \tag{3.1}$$

where B is the maximum queue length in the router (i.e., at the same time at most $B + 1$ packets can be stored and forwarded in the router), MSS is the maximum segment size of all TCP connections traversing the router, and α is a dynamically determined factor whose calculation is later explained. B and Q_i are expressed in number of packets and MSS is expressed in number of bytes. The logarithmic expression in Equation (3.1) is introduced to reflect that TCP connections which are in slow start are able to send twice as much segments in their next round trip time (RTT) interval than now. As a result, the queue length of the router can exponentially grow in the near future. In addition, Equation (3.1) ensures that every TCP connection is always allowed to send at least one TCP segment with a MSS.

The alterable factor α in Equation (3.1) is introduced to better utilize the link if only a few TCP connections are transferring segments over the router. α is updated every 10 milliseconds as follows:

$$\alpha = f(\alpha, \overline{Q}_i) = \begin{cases} \alpha + w_{\text{up}} & \text{if } \overline{Q}_i < \text{threshold}_{\text{low}} \\ \alpha \cdot w_{\text{down}} & \text{if } \overline{Q}_i > \text{threshold}_{\text{high}} \end{cases} \tag{3.2}$$

with

$$\overline{Q}_i = \frac{127}{128} \cdot \overline{Q}_{i-1} + \frac{1}{128} \cdot Q_i \tag{3.3}$$

The initial value for the utilization factor α is set to 1, the parameters w_{up} to additively increase and w_{down} to multiplicatively decrease α are set to $1/8$ and $31/32$, and the low and high thresholds for the mean queue length are set to 20 % and 60 % of the maximum queue length B.

The calculated sending window is transferred to each TCP sender by modifying the advertised receiver window in the TCP acknowledgments. It is only *reduced* by the EWA-capable router if necessary, but never increased to maintain TCP's end-to-end flow control:

$$\text{advertised receiver window} = \min\{\text{sending window}, \text{advertised receiver window}\} \tag{3.4}$$

With this explicit congestion-feedback information, the TCP senders are able to react more adequately to the current load in the router than it is possible with other mechanisms, e.g., ECN or RED.

3.2 Applicability of EWA

To deploy EWA in a network it is not necessary to equip all routers in the network with EWA-capabilities. It is sufficient to provide only the bottleneck router(s) with EWA.

EWA requires that the segments and acknowledgments of the considered TCP connections pass through the same (bottleneck) routers. If this necessary condition of symmetrical routing can be guaranteed for all (bottleneck) routers in the path from a TCP sender to its TCP receiver, such an approach can be used for an improved end-to-end congestion control based on a flow control between the (bottleneck) routers and the TCP senders.

In addition, if the TCP window scale option [8] is used it must be implemented by performing the continuous window-scale negotiation between a TCP sender and its TCP receiver. Otherwise, the EWA-capable router is not able to correctly evaluate and set the window-field in TCP acknowledgments, since an EWA-capable router cannot store the possibly negotiated window scale factor of every TCP connection traversing it (no per-flow information are available at the router).

3.3 Shortcomings of EWA

In my opinion, the basic EWA algorithm described in [11, 12] does not consider the in general most realistic case that the queue of a router with EWA-capabilities is below the lower threshold for a

longer period of time. In this case, the utilization factor α perpetually increases without an upper bound. If, after a while, the formerly lowly loaded router is highly loaded due to a large burst of incoming packets, it takes dozens or hundreds of measurement intervals to decrease the utilization factor α to an appropriate value. In the meantime, the EWA algorithm is not able to prevent the router from congestion and packets get lost. Thus, the history of the load of the router is too heavily weighted in the calculation of the utilization factor α of the EWA algorithm. This drawback of the EWA algorithm can be softened but not completely prevented if an upper bound for the utilization factor α is introduced, if the current queue length Q_i is much higher weighted in Equation (3.3), or if the parameter w_{down} is decreased to a value $\ll 1$. But it can be only prevented if a modified calculation of the utilization factor α with a much faster reaction on the current load in the router is used (cf. Section 4).

3.4 Improvements of EWA

The congestion feedback function (3.1) of EWA is one example function to calculate the sending window in a router. Other feedback functions are conceivable. These new feedback functions should adopt the well-chosen logarithmic expression from the EWA feedback function, but can adapt or replace the calculation of the utilization factor α. The FEWA algorithm described in Section 4 is an example approach for the latter case.

EWA and related approaches can be easily adapted so that symmetrical routing is no longer a necessary condition for its usage. This can be done by introducing a new TCP header option that carries the calculated sending window of the routers in the path from the TCP sender to the TCP receiver. The TCP receiver puts the minimum of this sending window and its currently supported receiver window as the new advertised receiver window in a normal TCP acknowledgment and sends it to the TCP sender. For this variant of EWA the TCP in the receiving end systems have to be slightly changed. Although this new implementation of EWA increases the delay of the EWA control loop between the routers and the TCP senders, it is not certain that this will decrease the overall performance of EWA. A positive effect of this new implementation on the performance of EWA is that the sending window of each EWA-capable router is calculated with a higher probability during the period of time the router has to handle segments from those TCP connections which get the feedback information. This is due to the inherent bursty sending behavior of the window-based congestion control of TCP (and in contrast to rate-based congestion-control mechanisms used in other networks, e.g., in ATM networks [26, 4, 10, 21, 7]). As a result, with the new EWA implementation the correlation between the current sending window and the obtained feedback might be increased from a single TCP sender's point of view. Whether the negative or positive effect of this new EWA implementation dominates the overall performance of EWA cannot be answered a priori. This has to be carefully investigated.

Chapter 4

Fuzzy Explicit Window Adaptation (FEWA)

In this section, a new EWA-related approach, called FEWA, is described whose queue-operating point is changed and whose calculation of the utilization factor α is based on a fuzzy controller.

4.1 FEWA Algorithm

The FEWA algorithm is similar to the EWA algorithm with two exceptions:

(1) The sending window (cf. Equation (3.1)) is calculated as follows:

$$\text{sending window} = \max\{\text{MSS}, \text{round}\,(\alpha \cdot \log_2(B - Q_i)) \cdot \text{MSS}\} \tag{4.1}$$

(2) And the utilization factor α (cf. Equation (3.2)) is calculated by a fuzzy controller only dependent on the current and the last but one measured queue length in a router:

$$\alpha = f(Q_i, Q_{i-1}) \tag{4.2}$$

Similar to EWA, the time of a control interval i of FEWA is set to 10 ms. But this value has to be adapted dependent on the bandwidths in the links the FEWA-capable router is connected with.

This fuzzy controller is related to the fuzzy controllers used in the Fuzzy Explicit Rate Marking Adaptation (FERMA) [23, 25] and in the more conservative FERMA-Modification (FERMAM) [24]. These approaches are located in Asynchronous Transfer Mode (ATM) switches and calculate congestion feedback which is used for the congestion control of Available Bit Rate (ABR) connections in ATM networks [10, 21, 7]. All linguistic rules and the membership functions of FERMAM are reused for FEWA and only a few parameters of the FERMAM fuzzy controller are adapted for FEWA.

4.1.1 General Fuzzy Controller

The most important part of a fuzzy control system is the fuzzy logic controller (FLC) [17, 18, 19]. With the FLC the human train of thoughts can be adapted in a simple way by using linguistic variables with their set of linguistic values and a small number of linguistic rules or relational expressions. One of the linguistic rules, linguistic rule R5, in the nonlinear FEWA fuzzy congestion controller (FCC) is for example (see Appendix A.1):

If the queue (length) is short and
the rate of change is increasing slowly, (R5)
then the utilization factor should be high.

"queue (length)", *"rate of change"*, and *"utilization factor"* are called linguistic variables; *"short"*, *"increasing slowly"*, and *"high"* are examples of their valid linguistic values. *"queue (length)"* and *"rate of change"* are the measured input variables of the linguistic rule while *"utilization factor"* is the output or control variable set by this linguistic rule.

The design of a FLC is split into two parts: First, the linguistic rules are set (surface structure), and second, the membership functions of the linguistic variables are determined (deep structure), quite often by a more intuitive and pragmatic choice.

Depending on the input parameter x the membership function m_V of a linguistic variable V denotes the weights $w_{v_k} \in [0,1]$ of the valid linguistic values v_k, $1 \leq k \leq n_V$:

$$m_V(x) = (w_{v_1}, \ldots, w_{v_{n_V}}) \in [0,1]^{n_V} \tag{4.3}$$

Here, each membership function m_V is represented by a composition of n_V membership functions of the fuzzy sets F_{v_k} of its corresponding linguistic values v_k:

$$\begin{aligned} m_V(x) &= m_{v_1}(x) \otimes \ldots \otimes m_{v_{n_V}}(x) \\ &= (w_{v_1}, \ldots, w_{v_{n_V}}) \end{aligned} \tag{4.4}$$

with the tensor operator $\otimes$. Let $\oplus$ denote the s-norm operator $\max$. For an FLC with a single output variable Y with the membership function $m_Y(y) = m_{y_1}(y) \otimes \ldots \otimes m_{y_{n_Y}}(y)$ its weighted membership function

$$\begin{aligned} m_Y^*(y) &= w_{y_1} \cdot m_{y_1}(y) \oplus \ldots \oplus \\ &\quad\; w_{y_{n_Y}} \cdot m_{y_{n_Y}}(y) \end{aligned} \tag{4.5}$$

represents the outcome of the L related fuzzy rules $v^{(l,1)} \times \ldots \times v^{(l,n)} \rightarrow y^{(l)}$, $1 \leq l \leq L$, with n linguistic values $v^{(l,1)}, \ldots, v^{(l,n)}$ of n linguistic variables $V^{(1)}, \ldots, V^{(n)}$ and their weights $w^{(l,1)}, \ldots, w^{(l,n)}$, dependent on the input-value vector $(x_1, \ldots, x_n)$ of the FLC, i.e., for $1 \leq l \leq L$, $1 \leq j \leq n$ and exactly one k, $1 \leq k \leq n_{V^{(j)}}$, it holds:

$$\begin{aligned} v^{(l,j)} &= v_k^{(j)} \\ w^{(l,j)} &= w_{v_k^{(j)}} = m_{v_k^{(j)}}(x_j) \end{aligned} \tag{4.6}$$

If none of the linguistic values $v^{(l,j)}$ of a linguistic variable $V^{(j)}$ are used in a linguistic rule l, i.e., the linguistic rule l is independent of $V^{(j)}$, then $w^{(l,j)}$ is set to 1.

Applying the often used norms $\min$ and $\max$ the weights w_{y_k}, $1 \leq k \leq n_Y$, of Y can be expressed as:

$$w_{y_k} = \max_{y^{(l)}=y_k} \{\min\{w^{(l,1)}, \ldots, w^{(l,n)}\}\} \tag{4.7}$$

The weights of all linguistic values of the linguistic variables are used to determine the required fuzzy value by a weighted analysis of the linguistic rules with a fuzzy inference engine. Due to computational simplicity the membership function of a linguistic variable is often triangular or trapezoidal shaped (cf. Figures 4.1, 4.2 and Appendix A.2).

4.1.2 FEWA Fuzzy Controller

At the beginning of every control interval i with a fixed duration, e.g., 10 ms, FEWA measures the current queue length Q_i and its current growth rate $G_i = Q_i - Q_{i-1}$ or its current fractional growth rate $\Delta G_i = G_i/B$, respectively. The membership function of FEWA regarding the queue length relative to the chosen target queue length QT is shown in Figure 4.1.

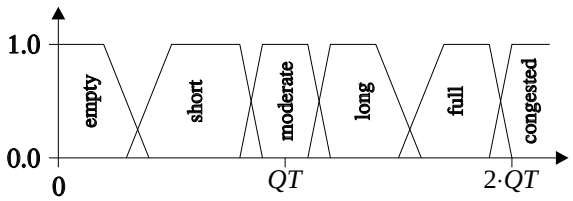

Figure 4.1: Membership function of the queue length Q of FEWA

The input range of the membership function of the fractional growth rate ΔG (see Figure 4.2) has been changed to substantial lower values because of the chosen target queue length $QT = \lfloor 0.25 \cdot B \rfloor$ at each router with FEWA-capabilities in the network.

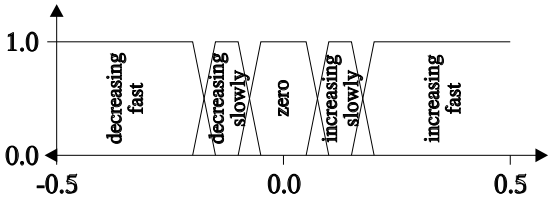

Figure 4.2: Membership function of the fractional queue growth rate ΔG of FEWA

Before every control interval i the weights of all linguistic values are set to 0. Then the weights for w_{Q_k}, $1 \leq k \leq 6$, for the six linguistic values (“empty”, “short”, “moderate”, “long”, “full”, and

"congested") of the current queue length Q_i and the weights w_{G_k}, $1 \leq k \leq 5$, for the five linguistic values ("decreasing fast", "decreasing slowly", "zero", "increasing slowly", "increasing fast") of the current fractional queue growth rate ΔG_i are determined. These weights are then used to calculate the weights w_{α_k}, $1 \leq k \leq 6$, for the linguistic values "very very little", "very little", "little", "medium", "high", and "very high" of the utilization factor α (cf. Appendix A.1). For the linguistic rule (R5), for instance, the last step of this computation can be expressed as (cf. Equation (4.7)):

$$w_{\alpha_5} = \max\{w_{\alpha_5}, \min\{w_{Q_2}, w_{\Delta G_4}\}\} \tag{4.8}$$

There may be other linguistic rules which have an influence on w_{α_5}. For FERMA, FERMAM, and for the fuzzy controller of FEWA, a different formula has been chosen to calculate the weights of the linguistic values of α. The linguistic rule (R5), for instance, can then be computationally expressed as:

$$w_{\alpha_5} = w_{\alpha_5} + w_{Q_2} \cdot w_{\Delta G_4} \tag{4.9}$$

There may be other linguistic rules which have an influence on w_{α_5}, too. By the weights w_{α_k}, $1 \leq k \leq 6$, the membership function m_α of α is converted to m^*_α (cf. Equation (4.5)). The weighted linguistic values of α are combined by a special fuzzy calculation, called Center-of-Gravity (COG), to the new utilization factor α (cf. [19]):

$$\alpha = \frac{\int y \cdot m^*_\alpha(y)\, dy}{\int m^*_\alpha(y)\, dy} \tag{4.10}$$

Analog to FERMA and FERMAM, the utilization factor α is determined by a simplified calculation where only the weights of the linguistic values of α are considered. Then, Equation (4.10) can be transformed into

$$\alpha = \frac{\sum_{k=1}^{6} w_{\alpha_k} \cdot \alpha_k}{\sum_{k=1}^{6} w_{\alpha_k}} \tag{4.11}$$

for a simplified computation with $\alpha_1 = 1.00$, $\alpha_2 = 2.00$, $\alpha_3 = 4.00$, $\alpha_4 = 6.00$, $\alpha_5 = 9.00$, and $\alpha_6 = 15.00$ in the current version of the FEWA fuzzy controller. The values for the α_k, $1 \leq k \leq 6$, are chosen to set the utilization factor α to a low value of 1 ($= \alpha_1$) if the queue is congested. If the queue is empty, the utilization factor is set to a high value of 15 ($= \alpha_6$). If the queue is neither congested nor empty, the utilization factor α is set to a value between 1 and 15 according to the linguistic rules of the FEWA FLC (cf. Appendix A.1).

Compared to the FLC of FERMAM, only these a_i's are adapted for the FLC of FEWA. Similar to FERMAM, these values are selected to achieve a moderate queue length Q in the neighborhood of the target queue length QT. But due to the more complex congestion control in IP-based networks compared to ATM networks and the less available information in IP routers compared to ATM

switches, e.g., the number of TCP flows and their current congestion-control mode (slow start, congestion avoidance) are unknown, the target queue length in IP routers cannot be met as well as it can be done in ATM switches (cf. [25, 24]). Here, the precise values for the α's have been chosen both by general considerations and by evaluating various test simulations.

Figure 4.3 shows the current FEWA fuzzy controller's control surface for the utilization factor α and for the sending window with $B = 99$ packets and $QT = 24$ packets, respectively. The fuzzy controller of FEWA is developed in such a manner that the current queue length Q_i has the main influence on the calculation of the utilization factor α or of the sending window, respectively; the current (fractional) growth rate $(\Delta)G_i$ is used to refine this calculation (in Figure 4.3, Q_i has the dominant influence). For the FEWA algorithm the described membership functions of the linguistic variables (see Appendix A.2) and the values of the utilization factor are the result of a more intuitive and pragmatic choice and not of an analytic approach (that this can work is one of the main advantages of fuzzy-logic controllers compared to other controller types). Nevertheless, this selection provides promising results.

It should be noted that due to the differences in the congestion-feedback functions of FERMAM and FEWA, in the FEWA fuzzy controller the values for the parameters α_i cannot be independently chosen from the maximum queue length B in a router. Therefore, these fuzzy controller parameters have to be carefully determined for each maximum queue length B in FEWA-capable routers in a network. For other maximum queue lengths B than 99, the α_i's should be selected that the new control surface for the sending window patterns the control surface for the sending window shown in Figure 4.3. A rule of thumb for this α_i-selection can be found in Appendix B.

4.2 Practicability of FEWA

In a real IP router there is no need for a complex and computational intensive fuzzy inference engine in the FLC. After the linguistic rules have been found and the linguistic values are tuned by a simulator, the control surface is known and can be stored as a lookup table for selected sampling points requiring only a few kilobytes of read-only memory (ROM) in a FEWA-capable router. In combination with a simple interpolation algorithm FEWA can be implemented in such a way with a very fast response time.

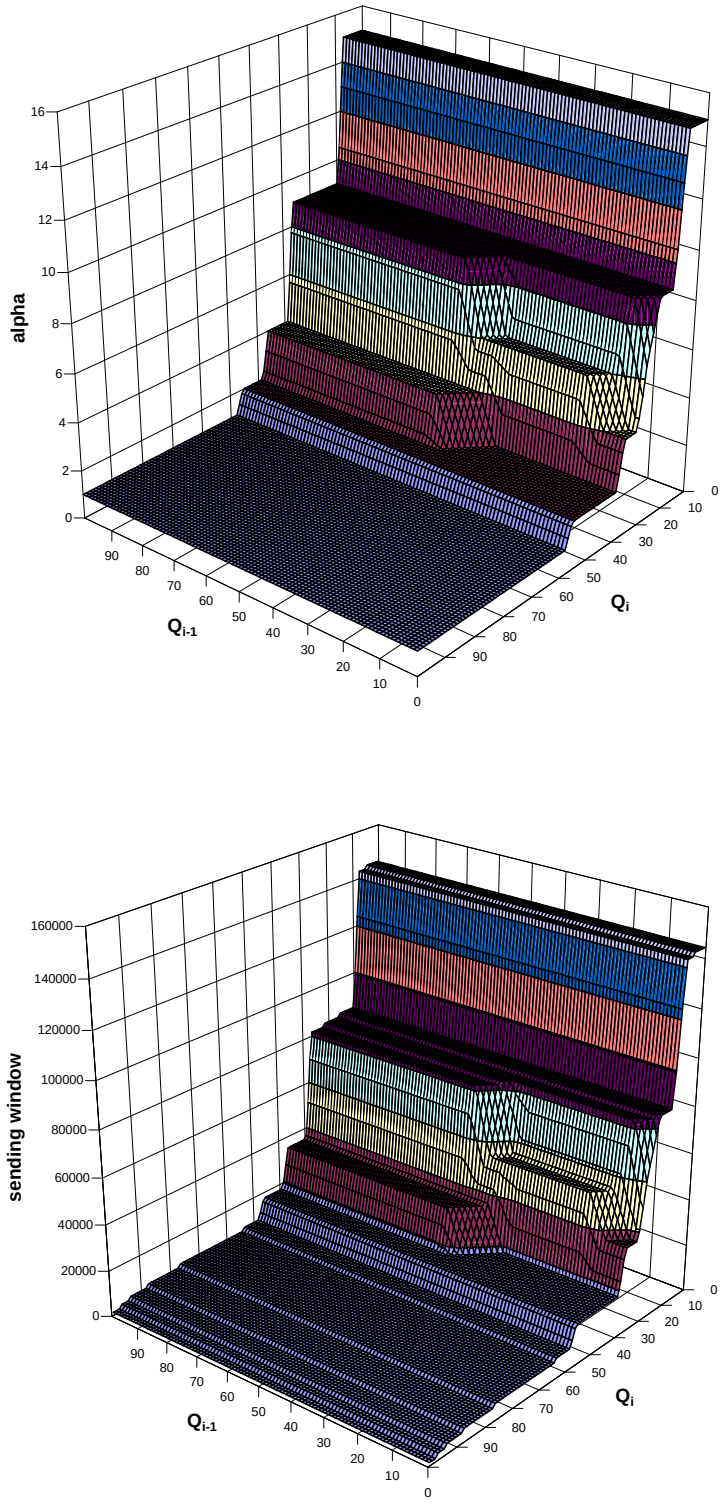

Figure 4.3: Control surfaces of the FEWA fuzzy controller for the utilization factor α and of the FEWA algorithm for the sending window

Chapter 5

Enhanced TCP (ETCP)

EWA and the related FEWA approach are developed to improve TCP's end-to-end congestion control by using TCP's built-in flow control mechanism. The same mechanism is used by Core-Stateless Fair Bandwidth Allocation for TCP (FBA-TCP, cf. Section 8). If a standard TCP sender is assisted by (F)EWA or FBA-TCP it limits its sending window to the minimum of its current congestion window and its current advertised receiver window. This advertised receiver window has been possibly decreased by a (F)EWA- or FBA-TCP-capable router if necessary due to (impending) congestion in the router. Therefore, a standard TCP sender in combination with one of these approaches can faster react on (impending) congestion in this router. But feedback information about additional available bandwidth in such a router can often not be used by the sender to increase its sending window as fast as it decreases its sending window in the case of (impending) congestion, particularly, if the TCP sender is in the slow start phase with a small congestion window or in the congestion avoidance phase.

5.1 Basic ETCP Mechanism and Possible Variants

If all routers—or at least the bottleneck routers—in a network are equipped with (F)EWA- or FBA-TCP-capabilities the TCP senders receive accurate and useful information about the current network load. Then, the overall performance of (F)EWA and FBA-TCP can be significantly increased if the congestion-control algorithm in a standard TCP sender is slightly changed by adapting the semantic of the advertised receiver window stored in every TCP acknowledgment: The current advertised receiver window of a TCP sender is not just an upper bound for its sending window or congestion window (cf. [1]), respectively, it is used to calculate the new congestion window more appropriately. This new TCP congestion-control mechanism based on a finer evaluation of the advertised receiver window stored in TCP acknowledgments is called Enhanced TCP (ETCP).

After the reception of an acknowledgment, the TCP congestion-control mechanisms slow start and

congestion avoidance are not used any longer in ETCP to increase the congestion window. This is the task of the new ETCP congestion-control mechanisms interpreting the current advertised receiver window. But after a segment loss, basic TCP congestion-control mechanisms as well as fast retransmit / fast recovery are used as fallback procedures also in ETCP. Thus, for a single ETCP connection its history is a less attached value to the congestion control than it is for a single TCP connection.

In the following, three ETCP variants are considered which differ in the calculation of the new congestion window:

- **ETCP 1:**
  The new congestion window of an ETCP sender is set to the current advertised receiver window:

$$\mathrm{CWND} = \mathrm{ARWND} \tag{5.1}$$

- **ETCP 2:**
  A TCP sender performing slow start doubles its congestion window after it has received CWND acknowledgments. This property of TCP is imitated by ETCP. Thus, the new congestion window of an ETCP sender converges to its doubled value after the ETCP sender has received CWND acknowledgments:

$$\mathrm{CWND} = \min\left\{\mathrm{CWND} \cdot 2^{1/\mathrm{CWND}}, \mathrm{ARWND}\right\} \tag{5.2}$$

- **ETCP 3:**
  Making an ETCP sender more aggressive—if allowed by the network—than in its variant 2, another strategy can be used. Here, the new congestion window of an ETCP sender converges to the current advertised receiver window after the ETCP sender has received CWND acknowledgments:

$$\mathrm{CWND} = \begin{cases} \mathrm{ARWND} & \text{if } \mathrm{CWND} \geq \mathrm{ARWND} \\ \mathrm{CWND} \cdot (\mathrm{ARWND}/\mathrm{CWND})^{1/\mathrm{CWND}} & \text{else} \end{cases} \tag{5.3}$$

  A variation of this ETCP variant is the following:

$$\mathrm{CWND} = \begin{cases} \mathrm{ARWND} & \text{if } \mathrm{CWND} \geq \mathrm{ARWND} \\ \mathrm{CWND} \cdot (\mathrm{ARWND}/\mathrm{CWND})^{1/\mathrm{IW}} & \text{if } \mathrm{ARWND} > \mathrm{IW} > \mathrm{CWND} \\ \mathrm{CWND} \cdot (\mathrm{ARWND}/\mathrm{CWND})^{1/\mathrm{CWND}} & \text{else} \end{cases} \tag{5.4}$$

  This restricts the aggressiveness of an ETCP sender to its initial aggressiveness in cases of small current congestion windows. As a result, the ETCP congestion control is more robust if packet losses in the network can occur that are not caused by congestion in (RCF-capable) routers. But such a variation of the ETCP variant 3 is not considered in this technical report.

After the first acknowledgment (the SYN/ACK) of a connection has been received by the ETCP sender, the ETCP sender knows the first ARWND that reflects the current load conditions in the network path. This first ARWND is used in (1) to set the first valid CWND to this value. The ETCP variants 2 and 3 use this first ARWND together with the initial CWND to calculate the first valid CWND according to the explained formulas. The following Figure 5.1 shows the schematic CWND process of the different ETCP variants compared to standard TCP for a constant example-ARWND of 64 segments and dependent on the number of consecutively received acknowledgments with the SYN/ACK included.

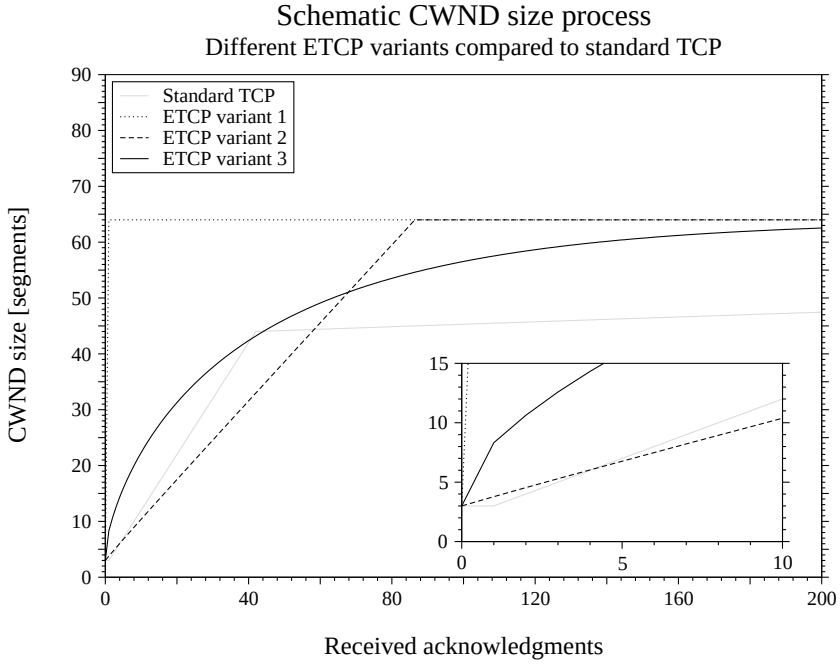

Figure 5.1: Schematic CWND process of the different ETCP variants compared to standard TCP

5.2 Pacing Algorithm

In order to avoid a bursty sending behavior of an ETCP sender, a pacing mechanism is implemented in the ETCP controller. This pacing mechanism works as follows: An ETCP sender can send at most two TCP segments in a burst. The time Δt between two consecutive segment bursts of an ETCP sender is calculated by using the smoothed round trip time and the congestion window of the ETCP sender:

$$\Delta t = \alpha \cdot \text{SRTT}(T_k)/\text{CWND}(T_k)$$

In the current version of an ETCP sender the pacing factor α is set to the fixed value 2. If for a single ETCP sender a pacing timer with an expiration time Δt is started, it is not updated by interim changed values of its smoothed round trip time and its congestion window. Thus, in some cases where its congestion window tends to increase often during the pacing time Δt, e.g., this is more likely to happen for lower round trip times, the pacing algorithm might be too conservative in sending new TCP segments compared to standard TCP. In future versions of ETCP, this pacing mechanism could be improved if the pacing factor α is adapted dependent on the current smoothed round trip time, e.g., $\alpha = f(\text{SRTT}(T_k))$ using a monotonic increasing function f with a range starting from 1 for lower smoothed round trip times and growing to 2 for larger smoothed round trip times.

Thus, the pacing mechanism is mainly used to avoid a bursty sending behavior of an ETCP sender in cases where the CWND has been largely increased. In addition, such a pacing mechanism makes the congestion control of an ETCP sender more resistant to different round trip times.

Remarks: In some specific cases, e.g., after the reception of a cumulative TCP acknowledgment, a standard TCP sender is allowed to send more than two or its initial number of TCP segments in a burst. Even in these cases the pacing algorithm is performed in ETCP senders. The reason for this conservative decision is that it is hardly possible to distinguish between all cases that can lead to such a bursty sending behavior of a TCP sender in a real end system without largely increasing the complexity of the ETCP controller.

5.3 Results in a Test Scenario

In the following Figure 5.2, the CWND process of a single ETCP connection is compared with the CWND process of a single standard TCP connection using an example-network path with a round trip time of 100 ms traversing a single router. In this specific network scenario the differently controlled ETCP connections reach throughputs of 132.42, 117.38, and 125.42 segments per second and outperform the standard TCP connection which reaches a throughput of 112.25 segments per second. The reason why even ETCP variant 2 reaches a slightly higher throughput than standard TCP is that an ETCP sender increases its initial CWND after it has received the first valid ARWND stored in the SYN/ACK while a standard TCP sender increases its initial CWND after it has received the acknowledgment of its first sent data segment. Thus, the number of outstanding segments of an ETCP sender following the second ETCP algorithm is slightly larger than the number of outstanding segments of a standard TCP sender—at least at the beginning of a connection (cf. enlarged part of Figure 5.1).

A very important difference between the first ETCP and the other two ETCP variants is that the ETCP variants 2 and 3 keep the self-clocking property of standard TCP, only some modifications are done how the CWND is increased by the ETCP sender after an acknowledgment has been received. As a result, even in combination with the pacing mechanism the ETCP variant 1 might be too ag-

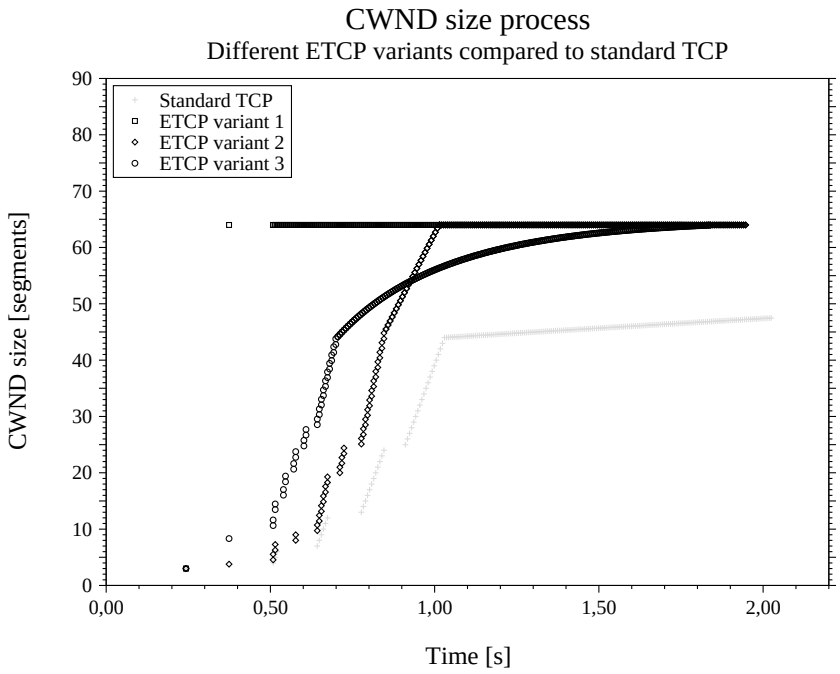

Figure 5.2: CWND process of the different ETCP variants compared to standard TCP

gressive to the network in load scenarios with concurrent traffic. This has been evaluated by test simulations. In addition, the ETCP variant 3 reaches the best performance in these test simulations. Therefore, it is recommended to use the ETCP variant 3.

If the bottleneck router is not equipped with (F)EWA or FBA-TCP capabilities, the ETCP senders do not receive adequate information about the current load in the network. In this worst-case scenario, an ETCP sender following the first ETCP variant is allowed to send too many segments that congestion in the network is likely to occur. The other ETCP variants, and particularly the second one, are more robust in this case.

Since EWA and FEWA are developed to work with standard TCP, the use of ETCP might have an influence on the load observed in (F)EWA-capable routers. Thus, if ETCP is used in end systems either the (F)EWA control algorithms, the pacing algorithm in the end systems, or the congestion-window calculations used in the different ETCP variants have to be revised to avoid too largely loaded or even congested routers in the network.

Currently, ETCP is only a promising idea how congestion control can be improved in future IP-based networks by using slightly adapted existing TCP congestion- or flow-control mechanisms. Further research has to be done to investigate how ETCP influences the traffic characteristics in the network and how a distributed congestion management can be advantageously used together with ETCP. In addition, due to the changed semantic of the advertised receiver window in ETCP, the

original end-to-end flow control of TCP is confined or even eliminated. Which effects this can have in existing or future TCP-based applications has to be carefully investigated.

Chapter 6

Explicit Control Protocol (XCP)

The XCP [15, 14] is a new transport protocol related to TCP. Unlike TCP, XCP provides explicit congestion feedback from XCP-capable routers to XCP senders. Therefore, XCP senders are able to more adequately control their sending window to reach an efficient, fair, scalable and stable congestion control in the whole network.

The feedback congestion-control algorithm in an XCP-capable router is divided in two parts: an efficiency and a fairness control algorithm. With this approach, efficiency of and fairness among XCP connections in a router can be separately managed. In this section, the protocol mechanisms and the feedback congestion-control algorithms of XCP are described in detail.

6.1 XCP Congestion Feedback Transfer

Each data packet of an XCP connection carries a congestion header (CH) (see Figure 6.1). The first

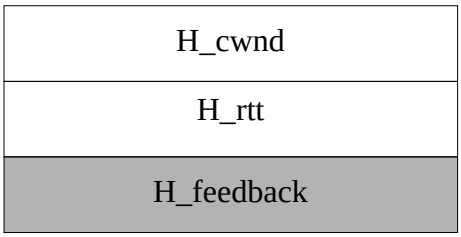

Figure 6.1: Congestion header in an XCP data packet/acknowledgment

two values, H_cwnd and H_rtt, are set by an XCP sender to its current congestion window and its current RTT estimate and are kept unchanged during the transport. The third value H_feedback is used for the congestion feedback of the routers. It is initialized by an XCP sender to the desired increase of its current congestion window and can be modified by a router according to the first two

values in the congestion header and the efficiency and fairness control algorithms performed in the router. In more detail: If the XCP sender has a desired sending rate r, the initial value for H_feedback can be computed as follows:

$$\text{H\_feedback} = (r \cdot \text{rtt} - \text{cwnd})/\text{number of packets in congestion window} \tag{6.1}$$

In the first packet of an XCP connections, H_feedback is initialized to 0, since the XCP sender has no valid estimation of the current RTT in the network path.

The XCP receiver copies the congestion header of an arriving data packet to an acknowledgment and sends the acknowledgment including the congestion header back to the XCP sender. After an acknowledgment arrival, the XCP sender adapts its new congestion window according to the router feedback stored in the congestion header:

$$\text{cwnd} = \max\{\text{cwnd} + \text{H\_feedback}, s\} \tag{6.2}$$

where s is the packet size.

The basic XCP sender does not use a pacing mechanism. But in order to avoid a bursty sending behavior of an XCP sender after a large increase of the congestion window, such a pacing mechanism should be additionally implemented in an XCP sender.

6.2 XCP Feedback Congestion Controller

As already mentioned, the feedback congestion controller in an XCP-capable router is divided in an efficiency controller (EC) and a fairness controller (FC). The efficiency controller's task is to maximize the link utilization and to minimize the packet loss rate and the persistent queue of a link. The EC deals only with aggregated traffic of a link and does not consider any fairness issues between flows of that aggregated traffic. This is the task of the fairness controller. Using the current per-link congestion feedback information computed by the EC the FC calculates the current per-packet congestion feedback information for each flow. This per-packet congestion information is stored in the H_feedback field of the congestion header in every packet and carried back to each XCP sender.

For each link a router maintains a control timer that is set to the most recent estimate of the average RTT seen by the XCP senders on that link. After every timeout of each link control timer the EC and the FC are used to calculate current values of the feedback congestion control for XCP flows traversing this link.

In the following, the mathematical expressions of the EC and FC calculations are explained. More information about the background of the XCP control algorithms and an example implementation based on a pseudo code can be found in [15, 14].

6.2.1 XCP Efficiency Controller (EC)

d is the average RTT estimated for a link, S is the spare bandwidth of a link defined as the difference between the input traffic rate for that link and the link capacity. And Q is the persistent queue size of a link in bytes which is the queue size that does not drain in a round trip propagation delay. Q is computed as the minimum of all queue sizes seen by any arrived packet in the last propagation delay. Then the aggregated congestion feedback ϕ in bytes of this link can be computed as

$$\phi = \alpha \cdot d \cdot S - \beta \cdot Q \tag{6.3}$$

with two constants $\alpha = 0.4$ and $\beta = 0.226$ (cf. [15, 14]). The aggregated congestion feedback ϕ should be proportional to S, since if the link is underutilized ($S > 0$) or congested ($S < 0$) a positive or a negative feedback should be sent to the XCP senders. But ϕ should be also proportional to $-Q$ to drain the queue. For example, if the input traffic rate matches the link capacity, i.e., $S = 0$, the aggregated congestion feedback ϕ must be set to a negative value to drain the queue. Equation (6.3) ensures that the aggregated congestion feedback ϕ of a link is proportional to S and $-Q$.

6.2.2 XCP Fairness Controller (FC)

The fairness controller is based on the additive-increase-multiplicative-decrease (AIMD) principle known from TCP, i.e., the per-packet congestion feedback follows this policy:

- If $\phi > 0$, the throughput increase of all flows is the same.
- If $\phi < 0$, the throughput decrease of a flow is proportional to its current throughput.

This policy ensures a continuous fairness convergence if ϕ is not equal to zero. But if the efficiency is approximately optimal, i.e., $\phi \approx 0$, this convergence can be stalled. In order to prevent this, the concept of bandwidth shuffling is used where in every control interval a small amount of traffic h with

$$h = \max\{0, \gamma \cdot y - |\phi|\}, \tag{6.4}$$

with a constant $\gamma = 0.1$ and the input traffic y in an average RTT, is redistributed according to the AIMD principle over all flows.

For a flow i the per-packet congestion feedback H_feedback$_i$ can be written as a linear combination of a positive congestion feedback p_i and a negative congestion feedback n_i:

$$\text{H\_feedback}_i = p_i - n_i \tag{6.5}$$

In a single control interval, p_i and n_i are computed by using the values

$$\xi_p = \frac{h + \max\{\phi, o\}}{d \cdot \sum \frac{\text{H\_rtt}_i \cdot s_i}{\text{H\_cwnd}_i}} \tag{6.6}$$

and

$$\xi_n = \frac{h + \max\{-\phi, o\}}{d \cdot \sum s_i} \tag{6.7}$$

where the sum in each denominator is considered over all packets in a control interval. These two formulas are later explained. Then p_i and n_i can be computed as follows:

$$p_i = \xi_p \cdot \frac{\text{H\_rtt}_i^2 \cdot s_i}{\text{H\_cwnd}_i} \tag{6.8}$$

$$n_i = \xi_n \cdot \text{H\_rtt}_i \cdot s_i \tag{6.9}$$

Equation (6.8) can be explained as follows (my own considerations): Due to the policy of the fairness controller, each XCP flow should get the same amount of additional throughput ΔT if the aggregated congestion feedback is above zero, i.e., $\Delta T_i = \Delta T_j = \Delta T$ for any two XCP flows i and j. For each XCP flow i this additional throughput is equal to the additional increase in the congestion window divided by the round trip time of flow i: $\Delta T = \Delta\text{H\_cwnd}_i / \text{H\_rtt}_i$, i.e., $\Delta\text{H\_cwnd}_i = \Delta T \cdot \text{H\_rtt}_i$. This additional increase in the congestion window must be shared over all packets the router sees from the XCP flow i during the control interval d to achieve a per-packet congestion feedback for XCP flow i. During a control interval d $\frac{\text{H\_cwnd}_i}{s_i} \cdot \frac{d}{\text{H\_rtt}_i}$ packets from flow i traverse the router. Therefore, each of these packets should carry a positive congestion feedback of

$$p_i = \frac{\Delta T}{d} \cdot \frac{\text{H\_rtt}_i^2 \cdot s_i}{\text{H\_cwnd}_i} \tag{6.10}$$

In a control interval additional $h + \max\{\phi, 0\}$ bytes should be allocated by all XCP flows. Thus, the total increase in the aggregated traffic rate is

$$\frac{h + \max\{\phi, 0\}}{d} = \sum \frac{p_i}{\text{H\_rtt}_i} \tag{6.11}$$

where the sum is considered over all packets during a control interval. If Equation (6.10) is inserted in Equation (6.11) it follows that (cf. Equation (6.6))

$$\Delta T = \frac{h + \max\{\phi, o\}}{\sum \frac{\text{H\_rtt}_i \cdot s_i}{\text{H\_cwnd}_i}} = d \cdot \xi_p \tag{6.12}$$

Equation (6.9) can be explained as follows (my own considerations): Due to the policy of the fairness controller, each XCP flow i should reduce its future throughput by a throughput-reduction factor ΔR that is proportional to its current throughput if the aggregated congestion feedback ϕ is below zero, i.e., $\Delta R_i = \Delta R_j = \Delta R$ for any two XCP flows i and j. For each XCP flow i this throughput-reduction factor ΔR is equal to the decrease in the congestion window divided by the current congestion window of flow i: $\Delta R = \Delta\text{H\_cwnd}_i / \text{H\_cwnd}_i$, i.e., $\Delta\text{H\_cwnd}_i = \Delta R \cdot \text{H\_cwnd}_i$. This decrease in the congestion window must be shared over all $\frac{\text{H\_cwnd}_i}{s_i} \cdot \frac{d}{\text{H\_rtt}_i}$ packets from flow i that traverse the

router during the control interval. Therefore, each of these packets has to carry a negative congestion feedback of

$$n_i = \frac{\Delta R}{d} \cdot \text{H\_rtt}_i \cdot s_i \tag{6.13}$$

In a control interval $h + \max\{-\phi, 0\}$ bytes must be deallocated by all XCP flows. Thus, the total decrease in the aggregated traffic rate is

$$\frac{h + \max\{-\phi, 0\}}{d} = \sum \frac{n_i}{\text{H\_rtt}_i} \tag{6.14}$$

where the sum is considered over all packets in a control interval. Insert Equation (6.13) in Equation (6.14) it follows that (cf. Equation (6.7))

$$\Delta R = \frac{h + \max\{-\phi, o\}}{\sum s_i} = d \cdot \xi_n \tag{6.15}$$

Equations (6.8) and (6.9) maintain the self-clocking property of TCP for XCP. The difference of XCP to TCP is that an XCP sender is able to increase its congestion window by other values than 1 (or $1/\text{CWND}$) as a TCP sender does in slow start (or congestion avoidance) after the reception of an acknowledgment. And in contrast to TCP, an XCP sender is able to decrease its congestion window even after reception of an acknowledgment. Hence, an XCP sender can earlier and more accurately react on lower available bandwidth in a network than a TCP sender can do. In general, if XCP is used packet losses are rare events. But if packet loss events occur, a fallback procedure is used in which an XCP sender acts similar to a TCP sender.

6.3 Practicability of XCP

Implementing XCP in end systems is relative simple. Only slight changes in the source codes of TCP senders and TCP receivers must be done to make them XCP-capable. Equipping routers with XCP-capabilities is more costly but still simple. But the complexity of XCP in a router is comparatively high: For the most efficient implementation of the XCP algorithms (cf. appendix of [15, 14]) a couple of additions and 3 multiplications per packet are needed. Nevertheless, XCP is a promising candidate for improving congestion control in future IP-based networks.

XCP can be smoothly and incrementally deployed in current IP-based networks. Two cases have to be distinguished for that:

(1) some routers and receivers are not XCP-capable, and

(2) a mix of XCP and non-XCP connections coexist in the network.

In the first case, the XCP sender must check that all routers in the path and the receiver are XCP-capable. This can be done with existing TCP and IP mechanisms. If they are not XCP-capable, the

XCP sender cannot use the XCP protocol and has to switch to a conventional transport protocol, e.g., TCP. In the second case, an XCP-capable router should be able to fairly handle both types of traffic, i.e., XCP flows should behave TCP-friendly. To reach this, an XCP-capable router has to distinguish between XCP and non-XCP traffic and queues them separately. Then the packets in both queues are processed such that XCP flows from the one queue reach the same average throughput than non-XCP flows from the other queue. This can be reached with a weighted fair queueing mechanism with dynamically adapted weights according to a TCP-Friendly Rate Control (TFRC) [5] approach (cf. [15, 14]).

Chapter 7

Core-Stateless Fair Queueing (CSFQ)

The basic idea of Core-Stateless Fair Queueing [27] is to partition the network in islands of routers and to distinguish between the edge and the core of an island (cf. Figure 7.1). Each edge router

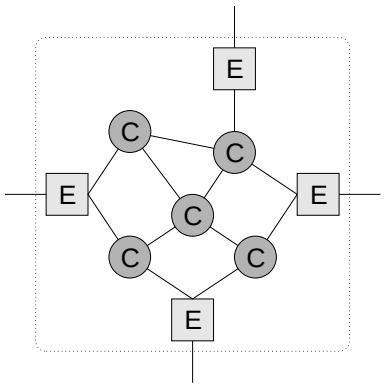

Figure 7.1: Island of edge (E) and core (C) routers with CSFQ-capabilities

estimates a per-flow rate for every incoming flow of packets that pass through the edge router into the CSFQ-capable network. For this calculation, the edge routers need to store per-flow states. The estimated per-flow rates are used to label the packets by inserting the estimated per-flow rate into each packet header of a flow. The core routers are equipped with FIFO queues and do not have to store any per-flow states. This property of the core routers is called core-stateless. The core routers perform a probabilistic dropping algorithm for arriving packets that is based on own measurements of the aggregated traffic and includes the information stored in the labels of the packets. The main goal of this dropping algorithm is to reach a fair share of bandwidth allocation between the flows passing a core router.

In the following Sections 7.1 and 7.2, the rate-estimation algorithm performed in an edge router and the packet-dropping algorithm performed in a core router are described. And in Section 7.3, some

possible solutions how CSFQ-information can be carried in packets between edge and core routers are depicted. More details about CSFQ, e.g., pseudocodes for the algorithms used in edge and core routers or extensions to the here described basic CSFQ mechanism, can be found in [27].

7.1 CSFQ Estimation of Per-Flow Rates in an Edge Router

In an edge router the estimated flow rate $\widehat{r}_i$ of a flow i is updated every time a new packet of this flow arrives. This flow-rate estimation is done by using an estimation based on exponential averages. Let $t_i^{(k)}$ and $l_i^{(k)}$ be the arrival time and length of the k-th packet of flow i. Then $\widehat{r}_i$ is updated as follows:

$$\widehat{r}_i^{\text{new}} = (1 - e^{-T_i^{(k)}/K}) \cdot \frac{l_i^{(k)}}{T_i^{(k)}} + e^{-T_i^{(k)}/K} \cdot \widehat{r}_i^{\text{old}} \tag{7.1}$$

with $T_i^{(k)} = t_i^{(k)} - t_i^{(k-1)}$ and a constant value K. A rule of thumb for K (and the later used constants K_α and K_c) is that K should be set to a value that is approximately twice the maximum queueing delay. A packet of flow i is labeled with the latest update of $\widehat{r}_i$:

$$\text{label}_i = \widehat{r}_i \tag{7.2}$$

In addition to the per-flow rate estimation, the packet-dropping algorithm of CSFQ core routers described in the following Section 7.2 is also performed in an edge router.

7.2 CSFQ Packet-Dropping Algorithm in a Core Router

In a fluid-flow model the total aggregated arrival rate of n flows in a single link of a core router is

$$A = \sum_{i=1}^{n} r_i \tag{7.3}$$

Each of these flows should use a fair share rate α that the output link speed C of the core router is utilized:

$$C = \sum_{i=1}^{n} \min\{r_i, \alpha\} \tag{7.4}$$

Since the flows are packetized, A and α have to be estimated by $\widehat{A}$ and $\widehat{\alpha}$. Then the accepted aggregated traffic rate

$$F(\widehat{\alpha}) = \sum_{i=1}^{n} \min\{\widehat{r}_i, \widehat{\alpha}\} \tag{7.5}$$

of a core router can be also estimated.

The aggregated arrival rate of a core router is estimated based on exponential averages by

$$\widehat{A}^{\text{new}} = (1 - e^{-T/K_\alpha}) \cdot \frac{l}{T} + e^{-T/K_\alpha} \cdot \widehat{A}^{\text{old}} \tag{7.6}$$

with the inter-arrival time T between the current and the previous packet and a constant value K_α. An analogous formula is used for the estimated aggregated traffic rate $\widehat{F}$ accepted by a router.

If $\widehat{A} \geq C$ during the whole interval K_c, a link is assumed to be congested. If $\widehat{A} \leq C$ during the whole interval K_c, a link is assumed to be not congested.

A new value for the estimated fair share rate $\widehat{\alpha}$ is only computed after an interval in which the link has been classified as congested or not congested. If the link has been congested then $\widehat{\alpha}$ is updated as follows:

$$\widehat{\alpha}^{\text{new}} = \frac{C}{\widehat{F}} \cdot \widehat{\alpha}^{\text{old}} \tag{7.7}$$

If the link has been not congested then $\widehat{\alpha}^{\text{new}}$ is set to the largest rate of any active flow, i.e., to the largest label seen in a packet, during the interval K_c. In addition, two simple heuristics are used to limit the fluctuations between consecutive $\widehat{\alpha}$-calculations: (1) After every queue overflow, $\widehat{\alpha}$ is decreased by a small fixed fraction, e.g., 0.01, and (2) in order to avoid overcorrection, the decrease of $\widehat{\alpha}$ is limited by 0.25 of its previous value.

$\widehat{\alpha}$ is then used to calculate the probabilities for dropping packets of the flows. For flow i each incoming bit is dropped in a core router with the probability:

$$P_i = \max\left\{0, 1 - \frac{\widehat{\alpha}}{\widehat{r}_i}\right\} \tag{7.8}$$

If packets of flow i are dropped, the new rate r_i of flow i is changed to approximately $\widehat{\alpha}$, since the arrival-rate of flow i at the next core or edge router is approximately $\widehat{\alpha}$. Therefore, the packets of flow i should be re-labeled by

$$\text{label}_i^{new} = \min\{\widehat{\alpha}, \text{label}_i^{old}\} \tag{7.9}$$

Therefore, after a packet has past the edge router on the receiver-side of its flow the label of the packet contains the current flow rate the CSFQ-capable network part is able to provide. This information could be transferred to the receivers and sent back to the senders to finer tune the load the senders of the flows put into the network. But CSFQ does not provide such an explicit congestion-feedback mechanism. Only the packet losses in the edge and core routers of the CSFQ-capable network part are used to implicitly inform the senders about (impending) congestion in this part of the network. An extension of CSFQ with an additional explicit congestion-feedback mechanism is described in Section 8.

7.3 Carrying CSFQ-Information between Edge and Core Routers

The CSFQ-label can be carried inside the type-of-service (TOS) field in a normal IP header. With a 4m(antisse)4e(xponent) floating-point representation flow rates between 1 kbps and 65 Mbps can be stored in the TOS field with an accuracy of at least 6.25 % (cf. [27]). It is also possible to define an IP option in IPv4 or to introduce a hop-by-hop extension header in IPv6 to label packets according to the CSFQ mechanism.

Chapter 8

Core-Stateless Fair Bandwidth Allocation for TCP (FBA-TCP)

FBA-TCP [13] uses the CSFQ mechanisms described in the last section to improve the congestion control of TCP connections. FBA-TCP works as follows: In edge routers of a network island (cf. Figure 8.1) FBA-TCP uses the same algorithms than CSFQ to estimate the rate of a flow and to label the packets of a flow with this estimated flow rate. In each edge and core router of a network island a

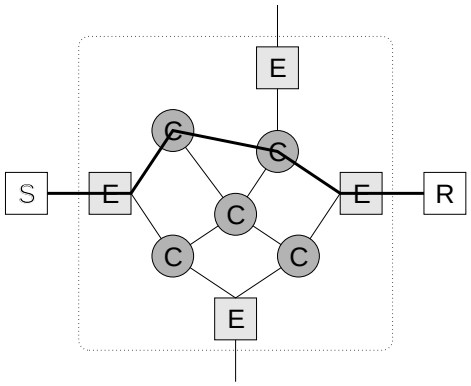

Figure 8.1: A single TCP connection traversing an island of routers with CSFQ-capabilities

fair share is estimated and packets of a flow are dropped according to Equation (7.8) if their label is larger than this fair share. This is exact the same algorithm used in CSFQ.

The new feature of FBA-TCP is that the edge router on the receiver-side of a flow do not remove the label from each packet. The edge router puts the label of a packet into a new option of an IPv4 header (or an extension header in IPv6) to transparently transfer this label to the TCP receiver through non-CSFQ-capable network parts. If the receiving end system of a TCP connection receives a packet with its label l or its estimated rate $\widehat{r}$, respectively, it delivers this value to the TCP receiver which

computes a new allowed sending window for the TCP sender

$$\text{allowed sending window} = \text{RTT} \cdot \widehat{r} \tag{8.1}$$

using its own estimated RTT. Due to the full-duplex capability of TCP, i.e., a TCP receiver is simultaneously a TCP sender and vice versa, it is assumed that a TCP receiver possesses an adequate estimation of the RTT in the network path. The minimum of this sending window and the number of bytes the TCP receiver is currently able to accept from the TCP sender is the new advertised receiver window of the TCP receiver:

$$\text{advertised receiver window} = \min\{\text{allowed sending window}, \text{advertised receiver window}\} \tag{8.2}$$

This advertised receiver window is sent to the TCP sender inside the next TCP acknowledgment.

Chapter 9

TCP Quick-Start (QS-TCP)

In contrast to all other approaches described in this chapter, TCP Quick-Start [9] uses feedback information from routers only at the beginning of a TCP connection to start this TCP connection with a larger initial congestion window if this is allowed by the routers in the network path. If the starting TCP connection is allowed by all routers in the network path to use a larger initial congestion window it performs a pacing mechanism to avoid sending its first segments in a burst.

The design principles of QS-TCP are:

- QS-TCP can only work if all routers in the path from the sending end system to the receiving end system are able to return explicit feedback.
- Only an underutilized router should allow a TCP sender to start with a larger initial congestion window than the standardized one. Otherwise, the queue at the router will show a transient behavior.
- Routers do not store any per-flow states to support QS-TCP. The only additional state a router has to store is its aggregated initial sending rate authorized over the most recent interval of time.

In the following Sections 9.1 and 9.2, the QS-TCP congestion-feedback transfer, its usage in the end systems, and the QS-TCP algorithm performed in the routers are described.

9.1 QS-TCP Congestion-Feedback Transfer and Usage in End Systems

If a TCP sender wants to use a larger initial congestion window it has to send a QS-TCP request (QSR) as a new IP option in its first IP packet which carries either the SYN or the SYN/ACK segment, respectively. The structure of a QSR is shown in Figure 9.1. The first two bytes of the QSR contain the IP option identifier and the length of this option (4 bytes). The third byte contains the QS-TCP time-to-live (TTL). It is randomly set by the TCP sender and decremented by one (modulo 256) from

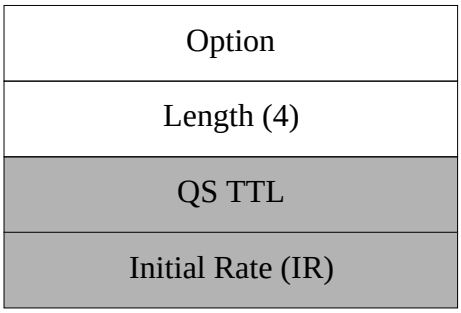

Figure 9.1: Quick-Start request option for IPv4

each router which approves the QSR. The QS TTL field is used by the sender to detect that all routers along the path can cope with the QSR. In addition, the QS TTL byte is used to partly protect the network against a misbehaving TCP receiver which otherwise could forge the explicit feedback from the routers to enable the TCP sender to send with a larger initial sending window than the network routers can deal with. For this reason, the TCP sender calculates and stores the difference of the TTL in the IP header and the randomly chosen QS TTL:

$$\Delta\text{TTL} = \text{TTL} - \text{QS TTL} \pmod{256} \tag{9.1}$$

This value is later used by the TCP sender to validate the Quick-Start Response. The fourth byte of the QSR is the initial rate (IR) requested by the TCP sender. This byte is set by the TCP sender to its desired initial rate and decreased by QS-TCP-capable routers if necessary. The current proposal of QS-TCP (cf. [9]) sets the unit of this byte to packets per 0.1 seconds. Thus, the initial rate is limited to 2550 packets per second.

After receiving a QSR in a SYN or SYN/ACK segment, respectively, the TCP receiver returns a QS-TCP response (see Figure 9.2) to the sender. This response is carried as a new option in the TCP header in the SYN/ACK or ACK for the SYN/ACK, respectively. The first two bytes of this

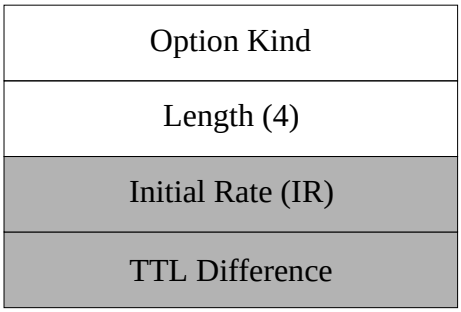

Figure 9.2: Quick-Start response option in the TCP header

QS-TCP response contain the TCP option identifier and the length of this option (4 bytes). The third

byte of the QS-TCP response contains the initial rate allowed by the network. And the fourth byte of the QS-TCP response contains the difference of the TTL in the IP header and the QS TTL in the received IP packet with the carried SYN or SYN/ACK segment, respectively, calculated according to Equation (9.1).

After receiving a QS-TCP response, the TCP sender checks the validity of this response. The response is valid if it contains the correct value for the TTL difference and if the initial rate carried in the response is less or equal to the initial rate requested by the TCP sender. Due to this validity check, the negative influence of a misbehaving TCP receiver is limited to the initial rate requested by the TCP sender.

If the validity check for the QS-TCP response is successful, the TCP sender sets its initial congestion window to

$$\text{CWND} = \max\{\text{IR} \cdot \text{RTT} \cdot \text{MSS}, \text{default initial CWND}\} \tag{9.2}$$

using the initial rate from the response, the first measured RTT, and the maximum segment size (MSS). In order to avoid sending bursts of segments, the TCP sender has to perform a pacing mechanism. The QSR might not be able to use the fast path in routers, since routers have to handle these specific packets differently than normal IP packets. As a result, the first RTT measurement of a TCP sender might be larger than the real RTT between the sending and receiving end system. Therefore, the QS-TCP mode ends when the first acknowledgment arrives at the TCP sender. If in this point in time the congestion window has not been fully used, the congestion window is reduced to the current number of outstanding bytes before the first acknowledgment has been received.

If the validity check for the QS-TCP response is not successful, since the request has been failed or no QS-TCP response has been arrived at all, the TCP sender uses its default initial congestion window.

9.2 QS-TCP Algorithms in Routers

If a router receives an IP packet with a QSR, three different cases are possible:

- A router is not able to understand the QS-TCP request. In this case, the router forwards the QSR without changing any bytes in it.
- A router is able to understand the QS-TCP request but not willing or not in a position to accept a larger initial rate than the standardized one. In this case, the router either can delete the whole QSR from the IP packet or it can set the initial rate in the QSR to zero.
- A router is able to understand the QS-TCP request and is able to provide a larger initial rate than the standardized one. In this case, the router decrements the QS TTL by one, puts its own

accepted initial rate in the IR byte of the QSR if its own initial rate is lower than the IR value stored in the QSR, and forwards the IP packet with the changed QSR included to the next hop.

How a router determines whether it can accept a larger initial sending rate than the standardized one or not depends on a newly introduced congestion control algorithm in the router that calculates the explicit router feedback. In [9], only some general rules for this congestion controller in the router are mentioned but no precise algorithm has been described. Therefore, the QS-TCP mechanism can be supported by routers in the network working internally with different congestion-control algorithms.

Chapter 10

Summary

In the previous chapters, different Router Congestion Feedback (RCF) mechanisms have been described. The main properties and functionalities of these mechanisms are summarized and compared in Table 10.1. Which of these approaches should be used in future IP-based networks depends on the desired level of compatibility with the currently deployed transport protocols TCP and UDP in the end systems. XCP, for example, seems to be the most powerful approach to improve the overall performance of a network—at least if the network is a high-speed network. But XCP requires slight but necessary adaptations of the transport protocol in the end systems. In addition, the overhead per packet in the XCP-capable routers seems to be feasible but large compared to other approaches.

In contrast to XCP, (F)EWA does not require any changes in the end systems. But (F)EWA is less powerful than XCP, since with (F)EWA the sending window of TCP senders cannot be as accurately controlled as with XCP. ETCP as an extension of (F)EWA is able to compensate this in parts at the cost of slightly adapted TCP senders. The basic (F)EWA and ETCP approaches require a symmetrical routing at least in the bottleneck router(s) of the network. But if this necessary condition cannot be guaranteed in the network, (F)EWA and ETCP can be easily adapted to work with a new TCP option or an IP option header (cf. Section 3.4).

CSFQ does not provide an explicit feedback for (TCP) senders. It is mainly developed to increase the fairness between flows in a network (part). Thus, its performance gain is rather limited. But FBA-TCP as an extension of CSFQ might be a potential candidate for an improved congestion control in IP-based networks. From its design FBA-TCP should have a comparable power to increase the

Table 10.1: Summary and comparison of the main properties and functionalities of different RCF mechanisms in IP-based networks

	(F)EWA	ETCP	XCP	CSFQ	FBA-TCP	QS-TCP
supports TCP / UDP flows	yes / no	yes / no	no / no	yes / yes	yes / no	yes / no
is transparent for senders / receivers	yes / yes	no / yes	no / no	yes / yes	yes / no	no / no
direction of the RCF approach	one-way	one-way	two-way	one-way	one-way	two-way
provides explicit / implicit RCF	yes / yes	yes / yes	yes / yes	no / yes	yes / yes	yes / yes
RCF used to increase / decrease sending rate	no / yes	yes / yes	yes / yes	no / yes	no / yes	yes / yes
provides continuous RCF	yes	yes	yes	yes	yes	no
needs per-flow state in some routers	no	no	no	yes	yes	no
needs symmetrical routing	yes	yes	no	no	no	no
uses new IP / TCP header (option)	no / no	no / no	yes / no	no / no	yes / no	yes / yes
has built-in pacing mechanism	no	yes	no	no	no	yes
applicability in the current Internet[1]	yes	no	no	yes[2]	(yes[2])	no
complexity in end systems / routers	low / mid	low / mid	low / high	low / high	low / high	low / mid
expected performance gain[3]	(+)++	++++	+++++[4]	++	+++	+

efficiency in a network than (F)EWA. The main disadvantage of FBA-TCP is that the edge routers

[1]Here it is considered if a RCF approach is transparent for end systems (or only minor changes in end systems are necessary) and can be stepwise deployed, i.e., the RCF approach does not require new functionalities in all routers of the

in the CSFQ-capable part of the network have to store per-flow information to label the packets of a flow. As a result, this approach can only work if the number of flows passing an edge router is rather low. In a sense, the main difference of XCP and FBA-TCP is that XCP labels packets in the transport protocol of a sending end system while FBA-TCP (re)labels packets in the routers of a CSFQ-capable network part.

QS-TCP is able to allow TCP connections to start with a larger initial congestion window than standardized if possible. This aim of QS-TCP is too limited to deploy QS-TCP as the (main) approach to improve congestion control in future IP-based networks. The most interesting aspect of QS-TCP is the mechanism how the network can be saved from misbehaving TCP receivers that forge the congestion-control feedback from routers in order to enable their senders to start sending with a too large and not allowed initial congestion window.

Table 10.2 shows which of the RCF approaches considered in this technical report can be combined with each other.

Table 10.2: Possible combinations of RCF approaches

can be combined with:	(F)EWA	ETCP	XCP	CSFQ	FBA-TCP	QS-TCP
(F)EWA		yes	in parts[5]	yes	yes	yes
ETCP	yes		no	yes	yes	yes
XCP	in parts[5]	no		yes	no	no
CSFQ	yes	yes	yes		yes	yes
FBA-TCP	yes	yes	no	mandatory[6]		yes
QS-TCP	yes	yes	no	yes	yes	

Internet

[2]The expected performance gain of these approaches is limited if only a few routers or end systems in the Internet are CSFQ- or FBA-TCP-capable

[3]Compared to standard TCP

[4]At least in high-speed networks (the performance of XCP in other network scenarios, e.g., (radio) access networks, is currently unknown)

[5]It is planned to replace the Efficiency Controller of XCP with a FEWA-related fuzzy controller

[6]FBA-TCP is based on CSFQ

Chapter 11

Congestion Feedback from Routers: Applicability in the Current Internet Environment

In the previous chapters, different Router Congestion Feedback (RCF) mechanisms have been described and compared. The focus of this chapter is on the applicability of these RCF approaches in the current Internet environment. The chapter describes if and how these RCF approaches can be (gradually) deployed in the current Internet in order to improve the performance of the network. In addition, it is explained if and how these RCF approaches can deal with the security mechanisms of Internet Protocol Security (IPSec) [16, 3].

In this chapter, each of the different RCF approaches is considered in more detail and with regard to the following point of views:

- Some of the considered RCF approaches require several network properties and changes in routers and/or end systems. Which of these requirements can be easily provided and which of these requirements are hard to reach in the current Internet environment? This viewpoint includes also the consideration of the additional complexity a RCF approach requires in the routers and/or in the end systems.
- Is it possible at all to gradually deploy the considered RCF approach in the current Internet environment? If this question can be approved there are other points which should be considered then:
    - How large is the influence on the expected performance gain of the RCF approach if only a few routers in the Internet, e.g., all routers in a network part, are equipped with additional RCF functionalities?

- If the expected performance gain of a RCF approach is negatively influenced by a gradual deployment, how can this negative influence on the expected performance gain be limited to reach an improved expected performance even with such a successive deployment? Here, it is considered which strategy should be used to choose the routers in the Internet that are primarily equipped with the new functionalities of the considered RCF approach.
- In addition, it is worth to investigate which RCF approaches are suited for deployment in parts of the Internet, e.g., a provider subnetwork, to improve congestion control at least in these parts of the Internet.

- Another interesting aspect is if and how a RCF approach is able to deal with existing security mechanisms in the Internet, e.g., IPSec. In IPSec, two different encryption methods can be distinguished:
  - In the transport mode of IPSec, the payload of an IP packet is encrypted. This mode provides a secure transfer of transport-layer data between two end systems. In this case, an intermediate router is not able to read or write fields in the TCP header.
  - In the tunnel mode of IPSec, a complete IP packet is encrypted and carried as payload of a new IP packet. This mode provides a secure transfer of IP packets between two routers, for example, to establish a virtual private network (VPN) in the Internet. In this case, an intermediate router is not able to find out how many separate IP flows are carried inside such an IP tunnel. Also in this case, an intermediate router is not able to read or write fields in the header of TCP segments transferred in some of these IP flows.

The remainder of this chapter is organized as follows. In the following Sections 11.1 to 11.6 each of the RCF approaches is considered with respect to the above stated viewpoints. Section 11.7 summarizes these considerations.

11.1 (Fuzzy) Explicit Window Adaptation ((F)EWA)

EWA and its improvement FEWA can only work if TCP segments and their TCP acknowledgments pass the same (F)EWA-capable routers in the network. Therefore, both RCF approaches can be only deployed in the current Internet if IP packets are symmetrically routed in the part of the network where routers are equipped with (F)EWA. Since a (F)EWA-capable router must be able to decrease the advertised receiver window in TCP acknowledgments to inform TCP senders about the current load in the router, parts of the TCP header must be changeable. Thus, (F)EWA in its basic variant cannot be deployed in the current Internet if (some of the) TCP connections are encrypted using the IPSec transport mode.

Both stated necessary conditions of supporting (F)EWA in the current Internet can be eliminated if the basic (F)EWA mechanism of manipulating TCP's built-in flow control is replaced by a mechanism based on a new option in an IP header or based on a new IPv6 extension header, respectively, if IPv6 will be widely deployed in future. Then, the congestion feedback information of the routers can be transferred in the IP header option or IPv6 extension header of an IP packet carrying a TCP segment. In addition, this (F)EWA variant can cooperate with the IPSec transport mode. But (F)EWA cannot be used together with the IPSec tunnel mode. This can be only reached if it is ensured by new IPSec mechanisms that the IP header option or the IPv6 extension header for (F)EWA is not encrypted in the IP tunnel and carried with the encapsulated IP packet after the IP tunnel. Since this (F)EWA variant requires that the receivers are able to handle the new IP option or new IPv6 extension header, the transparency of (F)EWA for receiving end systems is lost.

The main advantage of (F)EWA compared to other RCF approaches is that (F)EWA can be gradually deployed in the network. If this successive deployment is accurately done, i.e., the bottleneck routers in the network are primarily equipped with (F)EWA, the expected performance gain of (F)EWA is not reduced. In addition, (F)EWA can be separately supported in a network part, e.g., a provider subnetwork, to improve the congestion control and the overall performance of TCP connections at least in this network part. In addition, the complexity of the (F)EWA algorithm in the routers is comparatively low compared to XCP, for example.

11.2 Enhanced TCP (ETCP)

ETCP depends on the implementation of the FEWA algorithms in all routers or at least in the bottleneck routers of the network. Therefore, all statements about the applicability of FEWA in current Internet routers and the transport of FEWA congestion control information from routers to the end systems are valid for ETCP. A disadvantage of ETCP in its current version compared to, for example, QS-TCP is that an ETCP-capable sender is not able to verify whether the routers in a network are equipped with FEWA-capabilities or not. Thus, it is not possible to deploy ETCP senders in a network where not all bottleneck routers are equipped with FEWA. But with mechanisms similar to those used in QS-TCP, it is possible to provide ETCP senders with the feature to detect if routers are equipped with FEWA capabilities and agree with the new advertised receiver window carried in the TCP acknowledgments or not. Then, ETCP can be understood as an extension of QS-TCP where congestion feedback from routers is carried to a TCP sender during the whole lifetime of its connection. In addition, the shortcoming of the QS-TCP proposal [9] that no router algorithms are specified in detail is lapsed with this ETCP adaptation.

Since ETCP requires slight but fundamental changes (e.g., a pacing mechanism) in the congestion-control algorithm of each TCP senders, this RCF approach can only be hardly completely

deployed in the current Internet. But it is possible to gradually update existing TCP senders with this new improved congestion-control functionality after all (bottleneck) routers in the network have been equipped with FEWA.

So far, an open question is how the changed semantic of the advertised receiver window in ETCP senders influences the traffic characteristics in the Internet. This has to be carefully evaluated by simulations.

11.3 Explicit Control Protocol (XCP)

Although XCP has been identified as the most promising RCF approach in the last chapter, the deployment of XCP in the current Internet is hard to reach, since end systems and routers have to be equipped with new congestion-control algorithms. Particularly the complexity of these new control algorithms in routers seems to be high compared to other RCF approaches. An XCP-capable router, for example, has to perform several additions and multiplications per packet.

In theory, XCP can be smoothly and incrementally deployed in current IP-based networks. Two cases have to be distinguished for that: (1) some routers and receivers are not XCP-capable and (2) a mix of XCP and non-XCP connections cooperate in the network. In the first case, the XCP sender must check that all routers in the path and the receiver are XCP-capable. This can be done with existing TCP and IP mechanisms. If at least one of these routers is not XCP-capable, the XCP sender cannot use the XCP protocol and has to switch to a conventional transport protocol, e.g., TCP. In the second case, an XCP-capable router should be able to fairly handle both types of traffic, i.e., XCP flows should behave TCP-friendly. To reach this, an XCP-capable router has to distinguish between XCP and non-XCP traffic and queues them separately. Then the packets in both queues are processed such that XCP flows from the one queue reach the same average throughput than non-XCP flows from the other queue. This can be reached with a weighted fair queueing mechanism with dynamically adapted weights according to a TCP-Friendly Rate Control (TFRC) [5] approach (cf. [15]). Thus, in practice the gradual deployment of XCP in the current Internet environment is still a challenging problem.

Note that the quality of a gradual deployment of XCP in the current Internet is different from the quality of a gradual deployment of (F)EWA. For example, if at least one router in the network path is not XCP-capable, i.e., XCP cannot be used, the whole benefit of XCP compared to standard TCP gets lost. On the other hand, if a few routers in a network path are not (F)EWA-capable the performance gain of (F)EWA compared to standard TCP can be maintained if at least the bottleneck routers in the network path are equipped with (F)EWA. Thus, a noticeable performance gain of XCP in a network can only be expected if most of the routers and end systems are deployed with the XCP algorithms.

XCP uses an additional congestion header for carrying congestion-control information from XCP

senders to XCP-capable routers and congestion feedback information in the inverse direction. If this congestion header can be separated from the remaining data of the transport protocol and excluded from the encryption of the transport protocol data (IPSec's transport mode), XCP is not affected at all by transport-layer security mechanisms in the current Internet. But XCP cannot be used together with the tunnel mode of IPSec (cf. notes on IPSec tunnel mode in Section 11.1).

11.4 Core-Stateless Fair Queueing (CSFQ)

CSFQ is mainly developed to improve the fairness between flows in a network (part) of CSFQ-capable routers. Therefore, it is possible to gradually deploy CSFQ in the current Internet. But CSFQ does not provide an explicit congestion feedback from routers to further improve the end-to-end performance of flows. Thus, its usability for flows which only traverse small CSFQ-capable network parts is somehow limited.

The main disadvantage of CSFQ compared to all other considered RCF approaches is that the edge routers of a CSFQ-capable network (part) need to store per-flow states in order to identify flows, to measure their current flow rate, and to mark packets of a flow according to the current measured flow rate. Thus, CSFQ is limited in its scalability and its applicability is restricted to network parts with a small number of flows traversing an edge router.

The identification of single flows in an edge router can be complicated or even prevented if some of these flows entering a CSFQ-capable network (part) are encrypted. If, for example, transport-layer flows between two end systems are encrypted using the transport mode of IPSec, the edge router is not able to separate these flows from each other. Then, the edge router can only determine the rate of the combined transport-layer flows between two end systems. And if flows of IP packets between two routers outside the CSFQ-network are encrypted in the tunnel mode of IPSec, the edge router can only determine a rate for this combined flow of IP flows. Thus, there may exist different types of flows inside a CSFQ-capable network. How CSFQ should work in this case is not considered by the developers of CSFQ. In the end, the CSFQ approach cannot be used with flows encrypted by IPSec.

11.5 Core-Stateless Fair Bandwidth Allocation for TCP (FBA-TCP)

FBA-TCP is an extension of CSFQ that provides an explicit feedback from a CSFQ-capable network to improve congestion control of TCP senders. Therefore, all properties and boundaries of CSFQ are valid for FBA-TCP. In addition, since the explicit feedback from the CSFQ-capable network has to be transferred to the TCP receiver, the FBA-TCP approach cannot cooperate with IPSec's tunnel mode.

11.6 TCP Quick-Start (QS-TCP)

As already mentioned, QS-TCP does not specify any algorithms in the routers, only some general design principles and rules for these algorithms are given (cf. Section 8 in [22], [9]). But it can be assumed that the implementation complexity of QS-TCP in routers is comparable to that of (F)EWA. QS-TCP is not transparent for end systems. It requires minor changes in TCP receivers and more substantial changes in TCP senders. Therefore, the deployment of QS-TCP in the current Internet is harder to reach than the deployment of (F)EWA, for example. And this comes with a limited expected performance gain of QS-TCP compared to (F)EWA, for example, since QS-TCP is only used to start a TCP connection with a larger initial congestion window than it is prescribed by the standard. In addition, a gradual deployment of QS-TCP in (parts of the) network has comparable limitations in usefulness and expected performance gain than a gradual deployment of XCP (cf. Section 11.3).

QS-TCP is able to cooperate with Internet security mechanisms based on IPSec's transport mode. But QS-TCP is not able to cooperate with encapsulated IP flows as it is used in the tunnel mode of IPSec.

11.7 Summary

In the previous sections, it has been described if and how the different RCF approaches (F)EWA, ETCP, XCP, CSFQ, FBA-TCP, and QS-TCP can be deployed in the current Internet environment. The focus of this consideration has been on the additional complexity of the RCF approaches in end systems and routers, their capability to be gradually deployed in the current Internet, their expected performance loss if they are only gradually deployed, and their ability to cooperate with security mechanisms in the Internet like IPSec. Table 11.1 summarizes these considerations.

Although XCP is the most promising approach in improving the performance of the network, its capability to be (gradually) deployed in the current Internet is limited. Therefore, implementing XCP in the Internet is rather a medium- to long-term task than a near-term one. The next best expected RCF approach regarding the performance gain is ETCP. But since ETCP requires some changes in TCP senders, its complete deployment in the current Internet is also hard to reach. In addition, the influence of ETCP senders on the current traffic characteristics in the Internet has been not investigated so far. If both the ability of a gradual deployment and the expected performance gain of a RCF approach is taken into account, FEWA in its variant with a new IP option or an IPv6 extension header should be considered as the first choice for a near-term improvement of the performance in the Internet. FBA-TCP has a comparable expected performance gain than FEWA, but the shortcomings of FBA-TCP are that per-flow state is required in some routers and that FBA-TCP cannot be used together with IPSec mechanisms in the current Internet environment. CSFQ and QS-TCP have limited expected

Table 11.1: Applicability of different RCF mechanisms in the current Internet environment

	(F)EWA	ETCP	XCP	CSFQ	FBA-TCP	QS-TCP
complexity in end systems / routers	low / mid	low / mid	low / high	low / high	low / high	low / mid
can be gradually deployed	yes	yes	yes	yes	yes	yes
performance losses if gradually deployed[1]	(no[2])	yes	yes	(no[3])	(no[3])	yes
can cooperate with IPSec transport / tunnel mode	(yes[4]) / no	(yes[4]) / no	(yes[5]) / no	no / no	no / no	yes / no

performance gains compared to the other RCF approaches. In addition, CSFQ cannot be used in cooperation with IPSec mechanisms and a gradual deployment of QS-TCP in the current Internet environment is hard to reach. Therefore, CSFQ and QS-TCP should not be considered as the primary choice of a RCF approach in the Internet. But it is conceivable to use a QS-TCP-related approach working over the whole lifetime of a TCP connection in combination with, for example, FEWA to benefit from properties of both approaches. Then, the slightly adapted QS-TCP sender is able (1) to faster increase its sending window to the currently available bandwidth and (2) to accordingly decrease its sending window to the current load in the network path than it is possible with a standard TCP sender.

Since there exist a trade-off between the capability to be simply (and gradually) deployed in the current Internet environment and the expected performance gain a RCF approach has, it is decided to select the further and in more detail considered RCF approach with regard to the following two aspects: (a) easy (gradual) deployment in the current Internet environment and (b) maximum expected performance gain. Therefore, the following RCF mechanisms are the most promising candidates for implementing an improved congestion control in the current Internet and in future IP-based networks.

(a) **FEWA (ETCP, extended QS-TCP+FEWA)**
FEWA can be simply deployed in the current Internet environment, since the end systems can be kept unchanged and the additional algorithms in the FEWA-capable routers have a low com-

[1] Compared to a full deployment of a RCF approach in the Internet
[2] If at least the bottleneck router is equipped with (F)EWA
[3] If the bottleneck router is in the CSFQ-capable part of the network
[4] Only if the variant with a new IP header option or a new IPv6 extension header is used
[5] If the XCP congestion header is not part of the encrypted transport data

plexity. In addition, it is sufficient to equip only the bottleneck routers in a network (part) with FEWA capabilities, i.e., FEWA can be gradually deployed to reduce or even prevent congestion in a network (part).

The expected performance gain of FEWA in networks with a high dynamical load is moderate compared to standard TCP. Feedback information about a suddenly decreased available bandwidth in a FEWA-capable router can be transferred to the TCP senders within a half round trip time. The TCP senders are then able to faster and more appropriate react on (impending) congestion in the network. Feedback information about an increased available bandwidth in such a router can also be transferred within a half round trip time. But this information is only used to perform standard TCP's probing mechanisms for available bandwidth (slow start or congestion avoidance) in the sending end systems. Therefore, FEWA is mainly used to reduce or prevent congestion in the network and not to higher utilize the links in the network (although this is done if packet losses due to congestion are significantly reduced). But if all routers or at least the bottleneck routers in a network are equipped with FEWA, this shortcoming of FEWA can be eliminated if the ARWND-semantic of standard TCP in the sending end systems is replaced by the new ARWND-semantic of ETCP. Alternatively, FEWA can be combined with ideas related to QS-TCP to extend TCP's network-probing mechanisms with an explicit non-congestion feedback from the routers that allows such a slightly adapted TCP sender to be more aggressive than a standard TCP sender if it is allowed by all routers in the network path.

(b) **XCP (FXCP)**

The main disadvantage of XCP is that it has the highest complexity compared to the other RCF approaches. In addition, XCP cannot be gradually deployed in parts of a network. If at least one router or end system in a network path is not able to cope with XCP, XCP is not performed and standard TCP is used instead of it.

But XCP promises the highest expected performance gain compared to the other RCF approaches—at least in high-speed networks. Its performance in other network scenarios like (radio) access networks with much lower available bandwidth has to be carefully investigated. Nevertheless, XCP should be investigated as an a-priori best case for the performance gain that can be reached with existing RCF approaches in IP-based networks. In addition to this investigation, in future it will be intended to evaluate the performance of an XCP adaptation where the efficiency controller of XCP in the routers is replaced by a fuzzy-based controller related to the controller used in FEWA. This new XCP-variant is called FXCP.

Chapter 12

Conclusions and Outlook

A-priori, the most promising candidate of existing router congestion feedback mechanisms to improve congestion control in IP-based networks is XCP. Therefore, the XCP performance should be investigated in more detail for different network scenarios and variable traffic loads.

It has been analytically shown that the algorithms of the fairness controller of XCP are well chosen. But the efficiency controller of XCP could be suboptimal. To investigate other algorithms for this part of XCP, e.g., a modified fuzzy controller of the FEWA mechanism or a CSFQ-related estimation of the aggregated rate in edge or core routers, might be promising. In addition, the overall performance of XCP or its adaptations could be improved if a pacing mechanism in an XCP sender is introduced. Since the XCP approach has a much higher complexity per packet in the routers than ETCP, for example, it might be interesting to investigate also the performance of the ETCP approach compared to standard TCP and XCP or its adaptations, respectively. It would be also of interest to investigate the performance gain non-transparent RCF approaches like XCP or ETCP can have compared to transparent RCF approaches like FEWA.

XCP and ETCP do not support UDP flows. If also UDP flows should be optionally supported by a router congestion feedback mechanism in future IP-based networks, these mechanisms have to be extended to operate with rates as input information for the routers and changes in rates as congestion-feedback information from the routers instead of windows. This approach can be combined with a CSFQ-related packet loss criterion to limit the rate of flows that are not able to understand or to penalize flows that are not willing to accept this congestion feedback from the network. Both the flow information for the routers and the feedback information from the routers can be carried in a new IP option if IPv4 is used or in an IP extension header if IPv6 is used, respectively. The protocol instances running in the end systems are then responsible to set and evaluate the values in this IP option. If the semantic of this new IP option is standardized, it will be possible to perform a congestion feedback mechanism between the routers and the end systems of an IP-based network based on router-specific

developed RCF algorithms running in different types of routers, e.g., backbone or gateway routers.

Appendix A

Parameters of Fuzzy Explicit Window Adaptation (FEWA)

In this appendix, the linguistic rules of the FEWA algorithm are presented. Furthermore, the chosen parameters of the membership functions $m_{\Delta Q}$ and $m_{\Delta G}$ of the linguistic variables ΔQ and ΔG are stated.

A.1 Linguistic Rules of FEWA

If the queue (length) is empty,
then the utilization factor should be very high. (R1)

If the queue (length) is short and
the rate of change is decreasing fast,
then the utilization factor should be high. (R2)

If the queue (length) is short and
the rate of change is decreasing slowly,
then the utilization factor should be high. (R3)

If the queue (length) is short and
the rate of change is zero,
then the utilization factor should be high. (R4)

If the queue (length) is short and
the rate of change is increasing slowly,
then the utilization factor should be high. (R5)

If the queue (length) is short and
the rate of change is increasing fast, (R6)
then the utilization factor should be medium.

If the queue (length) is moderate and
the rate of change is decreasing fast, (R7)
then the utilization factor should be high.

If the queue (length) is moderate and
the rate of change is decreasing slowly, (R8)
then the utilization factor should be medium.

If the queue (length) is moderate and
the rate of change is zero, (R9)
then the utilization factor should be medium.

If the queue (length) is moderate and
the rate of change is increasing slowly, (R10)
then the utilization factor should be medium.

If the queue (length) is moderate and
the rate of change is increasing fast, (R11)
then the utilization factor should be little.

If the queue (length) is long and
the rate of change is decreasing fast, (R12)
then the utilization factor should be little.

If the queue (length) is long and
the rate of change is decreasing slowly, (R13)
then the utilization factor should be little.

If the queue (length) is long and
the rate of change is zero, (R14)
then the utilization factor should be very little.

If the queue (length) is long and
the rate of change is increasing slowly,
then the utilization factor should be very little. (R15)

If the queue (length) is long and
the rate of change is increasing fast,
then the utilization factor should be very little. (R16)

If the queue (length) is full and
the rate of change is decreasing fast,
then the utilization factor should be very little. (R17)

If the queue (length) is full and
the rate of change is decreasing slowly,
then the utilization factor should be very little. (R18)

If the queue (length) is full and
the rate of change is zero,
then the utilization factor should be very little. (R19)

If the queue (length) is full and
the rate of change is increasing slowly,
then the utilization factor should be very little. (R20)

If the queue (length) is full and
the rate of change is increasing fast,
then the utilization factor should be very little. (R21)

If the queue (length) is congested,
then the utilization factor should be very very little. (R22)

A.2 Parameters of FEWA

Each row k of the following tables A.1, A.2 shows the angle points $(x_{k,1}; y_{k,1}), \ldots, (x_{k,4}; y_{k,4})$ of the membership function m_{v_k} of the linguistic value v_k, $1 \leq k \leq n_v$.

Table A.1: Parameters of the linguistic variable $\Delta Q = Q/QT$ (see Figure 4.1)

k	$m_{\Delta Q_k}$
1	$(\ 0.00; 1), (\ 0.00; 1), (\ 0.20; 1), (\ 0.40; 0)$
2	$(\ 0.30; 0), (\ 0.50; 1), (\ 0.80; 1), (\ 0.90; 0)$
3	$(\ 0.80; 0), (\ 0.90; 1), (\ 1.10; 1), (\ 1.20; 0)$
4	$(\ 1.10; 0), (\ 1.20; 1), (\ 1.40; 1), (\ 1.60; 0)$
5	$(\ 1.50; 0), (\ 1.70; 1), (\ 1.90; 1), (\ 2.00; 0)$
6	$(\ 1.90; 0), (\ 2.00; 1), (\ 2.00; 1), (\ 2.00; 1)$

Table A.2: Parameters of the linguistic variable $\Delta G = G/B$ (see Figure 4.2)

k	$m_{\Delta G_k}$
1	$(\text{-}\,1.00; 1), (\text{-}\,1.00; 1), (\text{-}\,0.20; 1), (\text{-}\,0.15; 0)$
2	$(\text{-}\,0.20; 0), (\text{-}\,0.15; 1), (\text{-}\,0.10; 1), (\text{-}\,0.05; 0)$
3	$(\text{-}\,0.10; 0), (\text{-}\,0.05; 1), (\ 0.05; 1), (\ 0.10; 0)$
4	$(\ 0.05; 0), (\ 0.10; 1), (\ 0.15; 1), (\ 0.20; 0)$
5	$(\ 0.15; 0), (\ 0.20; 1), (\ 1.00; 1), (\ 1.00; 1)$

Notice:

- If $(x_{k,1}; y_{k,1})$ is equal to $(x_{k,2}; y_{k,2})$, then $m_{v_k}(x) = y_{k,1}$ for all $x \leq x_{k,1}$.
- If $(x_{k,3}; y_{k,3})$ is equal to $(x_{k,4}; y_{k,4})$, then $m_{v_k}(x) = y_{k,4}$ for all $x \geq x_{k,4}$.

Appendix B

Selection of FEWA α_i's for Different Maximum Queue Lengths

B.1 The Rule of Thumb

In this section, a rule of thumb for the calculation of the α_i's will be derived that can be used for queues with a maximum queue length different from $B = 99$. One assumption for this rule of thumb is that only B is changed. All other parameters, linguistic variables, and linguistic rules of FEWA are unchanged, at least in relation to the new maximum queue length $B^{'}$.

Let $\alpha = (\alpha_1, \ldots, \alpha_6)$ be the parameter set for the FEWA fuzzy controller of a queue with a maximum queue length B, e.g., $B = 99$. For another queue with a maximum queue length $B^{'}$, e.g., $B^{'} = 999$, the parameter set $\alpha^{'} = (\alpha_1^{'}, \ldots, \alpha_6^{'})$ of this queue is selected that the control surface of the new FEWA fuzzy controller matches the control surface shown in Figure 4.3. Thus,

$$\alpha \cdot \log_2 (B - Q) = \alpha^{'} \cdot \log_2 \left(B^{'} - Q^{'}\right) \tag{B.1}$$

for comparable current queue lengths $Q = f_m \cdot B$ and $Q^{'} = f_m \cdot B^{'}$, $0 \leq f_m \leq 1$, expressed using the maximum queue length or $Q = f_t \cdot QT$ and $Q^{'} = f_t \cdot QT^{'}$, $0 \leq f_t \leq T = B/QT = B^{'}/QT^{'}$, expressed using the target queue length. Then

$$\alpha \cdot \log_2 ((1 - f_m) \cdot B) = \alpha^{'} \cdot \log_2 \left((1 - f_m) \cdot B^{'}\right) \tag{B.2}$$

This is equivalent–at least for queue lengths where only one membership function applies–to

$$\alpha_k \cdot \log_2 ((1 - f_k) \cdot B) = \alpha_k^{'} \cdot \log_2 \left((1 - f_k) \cdot B^{'}\right) \qquad 1 \leq k \leq 6 \tag{B.3}$$

with $x_{k,1} \cdot T^{-1} \leq f_k \leq x_{k,4} \cdot T^{-1}$ (cf. Table A.1). It follows that

$$\alpha_k^{'} = \frac{\log_2((1 - f_k) \cdot B)}{\log_2((1 - f_k) \cdot B^{'})} \cdot \alpha_k = \frac{\log_2(1 - f_k) + \log_2(B)}{\log_2(1 - f_k) + \log_2(B^{'})} \cdot \alpha_k \qquad 1 \leq k \leq 6 \tag{B.4}$$

But this equation has no unique solution for possible values of f_k if B and $B^{'}$ differ. What can be done is to solve this equation by selecting a single value as a representative for every validity interval of a membership function, e.g.,

$$f_k^* = \frac{x_{k,2} + x_{k,3}}{2} \cdot T^{-1} \qquad 1 \leq k \leq 6, \tag{B.5}$$

and calculate an approximation of

$$\alpha_k^{'} \approx \frac{\log_2(1 - f_k^*) + \log_2(B)}{\log_2(1 - f_k^*) + \log_2(B')} \cdot \alpha_k = \theta_k^{'} \cdot \alpha_k \qquad 1 \leq k \leq 6 \tag{B.6}$$

for this membership function by using the single value. This is the rule of thumb.

B.2 Example: Persistent and WWW Traffic Traversing a Single Bottleneck Router

For a maximum queue length $B = 99$ the parameter set $\alpha = (1, 2, 4, 6, 9, 15)$ has been identified as a good choice. Based on this result, the parameter set of a queue with a maximum queue length $B^{'} = 999$ is $\alpha^{'} = (0.66, 1.31, 2.60, 3.87, 5.70, 9.42)$.

The following Figure B.1 shows the histograms of the queue with $B = 999$ of a highly loaded router for the two RCF approaches EWA and FEWA compared to standard TCP. This example simulation scenario considers a traffic mixture of persistent and WWW-based TCP connections which is related to the simulation scenarios considered in [11, 12]. But even in this good-case scenario for EWA, FEWA is able to outperform EWA with a performance gain of more than 25 % considering the overall mean throughput.

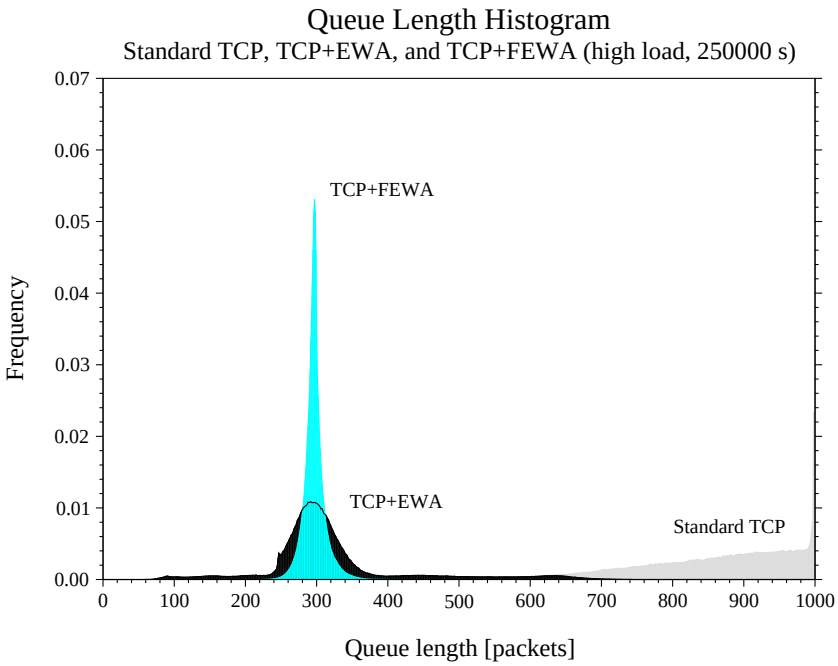

Figure B.1: Queue length process in a highly loaded bottleneck router for standard TCP, TCP+EWA, and TCP+FEWA

Appendix C

Variables and Parameters of the Congestion Feedback Approaches

In this chapter, an overview is given about the variables and parameters used in the RCF approaches considered in this technical report. Some of the variables and parameters of the RCF algorithms are used with indices to further specify a flow or a time interval. These indices are omitted in the following tables.

C.1 EWA

Table C.1: Variables and Parameters of EWA

Variable / Parameter	Description	Unit	(Initial) Value
B	maximum queue length	packets	e.g. 99
i	control interval	s	e.g. 0.010
Q	current queue length	packets	0
$\overline{Q}$	average current queue length	packets	0
α	utilization factor	1	1
w_{up}	AI-parameter	1	1/8
w_{down}	MD-parameter	1	31/32
$\text{threshold}_{\text{low}}$	lower queue threshold	packets	$0.20 \cdot B$
$\text{threshold}_{\text{high}}$	upper queue threshold	packets	$0.60 \cdot B$
MSS	maximum segment size	bytes	e.g. 1460

C.2 FEWA

Table C.2: Variables and Parameters of FEWA

Variable / Parameter	Description	Unit	(Initial) Value
B	maximum queue length	packets	e.g. 99
i	control interval	s	e.g. 0.010
Q	current queue length	packets	0
QT	target queue length	packets	24
$(\Delta)Q$	(fractional) queue length	1	see Table A.1
$(\Delta)G$	(fractional) queue length growth rate	1	see Table A.2
α	utilization factor	1	15
α_k	parameters in the FLC	1	1, 2, 4, 6, 9, 15
MSS	maximum segment size	bytes	e.g. 1460

C.3 XCP

Table C.3: Variables and Parameters of XCP

Variable / Parameter	Description	Unit	(Initial) Value
r	desired sending rate	bytes/s	
cwnd	congestion window	bytes	
s	packet size	bytes	e.g. 1500
rtt	round trip time	s	
d	average RTT of flows	s	
S	spare bandwidth	bytes	
α	weight for $d \cdot S$	1	0.4
Q	persistent queue size	bytes	0
β	weight for Q	1	0.226
ϕ	aggregated congestion feedback	bytes	
h	amount of shuffled bandwidth	bytes	
y	input traffic in d	bytes	
γ	fraction of bandwidth for shuffling	1	0.1
p_i	positive feedback	bytes	
n_i	negative feedback	bytes	
ξ_p	variable for p-computing	bytes/s^2	
ξ_n	variable for n-computing	1/s	

C.4 CSFQ

Table C.4: Variables and Parameters of CSFQ

Variable / Parameter	Description	Unit	(Initial) Value
$\widehat{r}$	estimated arrival rate	bytes/s	
t	arrival time of a packet	s	
l	length of a packet	bytes	
K, K_α, K_c	measurement intervals	s	$\approx 2 \cdot$ max queueing-delay
$\widehat{A}$	estimated aggregated arrival rate	bytes/s	
C	output link speed	bytes/s	e.g. 100 Mbps
$\widehat{F}$	accepted aggregated traffic rate	bytes/s	
$\widehat{\alpha}$	estimated fair share	bytes/s	
P	packet drop probability	1	$0 \leq P \leq 1$
label	(new) packet label	bytes/s	1 kbps...65 Mbps ± 6.25 %

C.5 FBA-TCP

Table C.5: Variables and Parameters of FBA-TCP

Variable / Parameter	Description	Unit	(Initial) Value
$\widehat{r}$	estimated arrival rate	bytes/s	
t	arrival time of a packet	s	
l	length of a packet	bytes	
K, K_α, K_c	measurement intervals	s	$\approx 2 \cdot$ max queueing-delay
$\widehat{A}$	estimated aggregated arrival rate	bytes/s	
C	output link speed	bytes/s	e.g. 100 Mbps
$\widehat{F}$	accepted aggregated traffic rate	bytes/s	
$\widehat{\alpha}$	estimated fair share	bytes/s	
P	packet drop probability	1	$0 \leq P \leq 1$
label	(new) packet label	bytes/s	1 kbps...65 Mbps ± 6.25 %
RTT	round trip time	s	

C.6 QS-TCP

Table C.6: Variables and Parameters of QS-TCP

Variable / Parameter	Description	Unit	(Initial) Value
QS TTL	TTL in QS request	1	$\in_R [0, 255]$
IR	initial rate	packets/s	≤ 2550
RTT	round trip time	s	
MSS	maximum segment size	bytes	e.g. 1460
CWND	congestion window	bytes	

Acronyms

ABR Available Bit Rate

AIMD Additive Increase Multiplicative Decrease

ARWND Advertised Receiver Window

ATM Asynchronous Transfer Mode

COG Center-of-Gravity

CSFQ Core-Stateless Fair Queueing

CWND Congestion Window

EC Efficiency Controller

ECN Explicit Congestion Notification

ETCP Enhanced TCP

EWA Explicit Window Adaptation

FBA-TCP Core-Stateless Fair Bandwidth Allocation for TCP

FC Fairness Controller

FCC Fuzzy Congestion Controller

FERM Fuzzy Explicit Rate Marking

FERMA FERM Adaptation

FERMAM FERMA-Modification

FEWA Fuzzy Explicit Window Adaptation

FLC Fuzzy Logic Controller

FXCP Fuzzy Explicit Control Protocol

IP Internet Protocol

IPv6 Internet Protocol Version 6

IPSec Internet Protocol Security

MSS Maximum Segment Size

NIS Network-Information Sharing

PDU Protocol Data Unit

QSR TCP Quick-Start Request

QS-TCP TCP Quick-Start

RCF Router Congestion Feedback

RED Random Early Detection

RLF Router Load Feedback

RTT Round Trip Time

SRTT Smoothed RTT

TCP Transmission Control Protocol

TFRC TCP-Friendly Rate Control

TOS Type of Service

TTL Time-To-Live

UDP User Datagram Protocol

WWW World Wide Web

XCP Explicit Control Protocol

Bibliography

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

[2] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the Internet. RFC 2309, April 1998.

[3] D. Comer. *Internetworking withTCP/IP—Principles, Protocols, and Architectures*, volume 1. Prentice Hall, 2000.

[4] M. de Pruecker. *Asynchronous Transfer Mode - Die Lösung für Breitband-ISDN*, volume 2. Prentice Hall, 1994.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM'00*, pages 43–56, 2000.

[6] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, aug 1993.

[7] The ATM Forum. Traffic management specification version 4.1. AF-TM-0121.000, March 1999.

[8] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323, May 1992.

[9] A. K. Jain and S. Floyd. Quick-start for TCP and IP. http://www.ietf.org/internet-drafts/draft-amit-quick-start-02.txt, work in progress, October 2002.

[10] R. Jain. Congestion control and traffic management in ATM networks: Recent advances and a survey. *Computer Networks and ISDN Systems*, 28(13):1723–1738, 1996.

[11] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. Explicit window adaptation: A method to enhance TCP performance. In *In Proceedings of IEEE INFOCOM'98*, pages 242–251, 1998.

[12] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. Explicit window adaptation: A method to enhance TCP performance. *IEEE/ACM Transactions on Networking*, 10(3):338–350, June 2002.

[13] R. Kapoor, C. Casetti, and M. Gerla. Core-stateless fair bandwidth allocation for TCP flows. In *Proceedings of IEEE ICC 2001*, 2001.

[14] D. Katabi. *Decoupling Congestion Control and Bandwidth Allocation Policy With Application to High Bandwidth-Delay Product Networks*. PhD thesis, Massachussetts Institute of Technology, March 2003.

[15] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *Proceedings of ACM SIGCOMM'02*, pages 89–102, August 2002.

[16] S. Kent and R. Atkinson. Security architecture for the Internet protocol. RFC 2401, November 1998.

[17] C. C. Lee. Fuzzy logic in control systems: Fuzzy logic controller—part I. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–418, March 1990.

[18] C. C. Lee. Fuzzy logic in control systems: Fuzzy logic controller—part II. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):419–435, March 1990.

[19] A. Pitsillides, Y. A. Şekercioğlu, and G. Ramamurthy. Effective control of traffic flow in ATM networks using fuzzy explicit rate marking (FERM). *IEEE Journal on Selected Areas in Communications*, 15(2):209–225, 1997.

[20] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, September 2001.

[21] M. Savorić. Untersuchung von Überlastabwehrverfahren für ABR-Dienste in drahtlosen ATM-Netzen. Master's thesis, University of Frankfurt, September 1998.

[22] M. Savorić. Description and comparison of distributed congestion management approaches in IP-based networks. project report EF-TUB-D6, March 2003.

[23] M. Savorić and U. R. Krieger. Adaptation of the fuzzy explicit rate marking to ABR flow-control in a wireless ATM network. In *Proc. of MMB99*, pages 25–30, September 1999.

[24] M. Savorić and U. R. Krieger. The performance of a fuzzy flow-control scheme for WWW data transfer in a wireless ATM network. In *Proc. of ITC Specialist Seminar on Mobile Systems and Mobility*, pages 145–156, March 2000.

[25] M. Savorić, U. R. Krieger, and A. Wolisz. The impact of handover protocols on the performance of ABR flow-control algorithms in a wireless ATM network. *European Transactions on Telecommunications (ETT)*, 11:419–430, August 2000. Special Issue on Service Quality Control in Multimedia Wireless Networks.

[26] G. Siegmund. *ATM – Die Technik des Breitband-ISDN*, volume 2. R. v. Decker, 1994.

[27] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *Proceedings of ACM SIGCOMM'98*, pages 118–130, 1998.