Implementation and Performance Evaluation of the re-ECN Protocol

Mirja Kühlewind[1] and Michael Scharf[2,⋆]

[1] Institute of Communication Networks and Computer Engineering (IKR)
University of Stuttgart, Germany
mirja.kuehlewind@ikr.uni-stuttgart.de
[2] Alcatel-Lucent Bell Labs, Stuttgart, Germany
michael.scharf@alcatel-lucent.com

Abstract. Re-inserted ECN (re-ECN) is a proposed TCP/IP extension that informs the routers on a path about the estimated level of congestion. The re-ECN protocol extends the Explicit Congestion Notification (ECN) mechanism and reinserts the obtained feedback into the network. This exposure of congestion information is a new economic traffic management mechanism that enables the network to share the available capacity more equally and to police the compliance of congestion control through e. g. a per-user congestion limitation.

This paper studies performance implications of the re-ECN mechanism. Our evaluation is based on simulations with an own re-ECN implementation in the Linux TCP/IP stack. Our results also confirm that congestion exposure generally works. But we also show that traffic characteristics such as the round-trip time (RTT), flow sizes, as well as the selection of congestion control algorithms have a significant impact on the congestion exposure information. These non-trivial effects have to be taken into account when using the re-ECN information as input parameter for congestion control mechanisms in end-systems or for routing/policing inside the network. Comparable results have not been published so far.

1 Introduction

In the Internet, the congestion control mechanisms of the Transmission Control Protocol (TCP) mainly decide how resources are shared [1]. Today's practice does not ensure a fair share of the available bandwidth in respect to all kind of emerging application requirements. This problem becomes increasingly critical as peer-to-peer file sharing and video streaming services consume already a large part of the available bandwidth with a strong upwards trend. In addition, more and more inelastic traffic emerges (e. g., television). A further challenge for resource sharing are new high-speed congestion control mechanisms (e. g., [2]) that are more aggressive than TCP's standard congestion control algorithms.

As a response, more and more network providers use e. g. Deep Packet Inspection in combination with traffic shapers to penalize bulk data applications and

⋆ The major part of this work was performed while the author was with IKR.

B. Stiller, T. Hoßfeld, and G.D. Stamoulis (Eds.): ETM 2010, LNCS 6236, pp. 39–50, 2010.

users that cause large traffic volumes [3]. But these approaches do not consider the actual congestion situation on the path, nor do they provide incentives for the users to react appropriately. A promising alternative is an accountability system that encourages the end-systems to share the bandwidth fairly. That means the available bandwidth should be used efficiently if no congestion occurs. But at the same time, time-critical applications should not significantly be affected even in a congestion situation. One solution to make end-systems accountable for their impact on the network is to expose the level of expected congestion on a network path to the network components on the path. This can be achieved thought the re-ECN protocol which has been proposed by Briscoe *et al.* [4]. It extends the Explicit Congestion Notification (ECN) mechanism [5] and reinserts the obtained feedback in the network, using the principle of re-feedback [6]. It is an incentive-oriented, end-to-end protocol that also enables congestion-based policing in the network, which has a significant potential for economic traffic management [7]. re-ECN is currently discussed in the Internet Engineering Task Force (IETF) as a solution for the congestion exposure problem [8].

The key contribution of this paper is an independent evaluation of the re-ECN protocol. So far there are only a few studies of Briscoe *et al.* [6]. In order to verify the specification, we implemented the protocol for the Internet Protocol version 4 (IPv4) and TCP in the Linux network stack. Our implementation is the only kernel implementation besides the one of Alan Smith used by Briscoe *et al.* The use of real kernel code with the Network Simulation Cradle (NSC) framework [9] allows us to perform realistic user-space simulations.

As this is an initial evaluation, our focus is to highlight the principle operation of the re-ECN protocol. Our results reveal several important characteristics of re-ECN that affect the performance of the proposed congestion-based traffic management: The congestion exposure information depends on the RTT, on the selected congestion control mechanism, and/or on the parameterization of the Active Queue Management (AQM) in a router. Furthermore, the TCP Slow-Start overshoot can cause congestion peaks that could discriminate flows with a certain length. These non-trivial dependencies have to be taken into account when using re-ECN information for economic traffic management. In the long term re-ECN is intented to encourage the use of transport protocols that adapt to the users' intentions.

The remainder of this paper is structured as follows: Section 2 gives an overview of the re-ECN protocol. In Section 3, we highlight some of the implementation challenges, and we also outline some open issues in the specification. The setup of our simulations is explained in Section 4. Section 5 then presents selected results for the performance of re-ECN. Section 6 concludes the paper.

2 Overview of re-ECN

2.1 The re-ECN Protocol for TCP/IPv4

re-ECN is an extension of ECN [5], which signals congestion to the sender before packets get lost. Like ECN, it requires routers to use AQM, such as Random

Early Detection (RED) [10], to mark a packet instead of dropping it. Full re-ECN support requires modifications in the receiver as well (RECN mode). If the receiver is only ECN capable, re-ECN will operate in RECN-Co mode.

In order to expose the expected congestion to all components on a network path, the sender uses bit 48 of the IPv4 header, the so called re-Echo (RE) flag. The receiver counts the packets that have been marked as Congestion Experienced (CE) by an ECN router and returns this value back to the sender. For every congestion announcement the sender has to blank the RE flag of a data packet. Otherwise, the RE flag is always set to "1". The fraction of re-ECN capable packets with the RE flag blanked can be considered as an estimation of the expected congestion on the path.

The re-ECN extension flag together with the two-bit ECN IPv4 field provide 8 available codepoints shown in Table 1. The encoding conserves backwards compatibility to ECN. The Feedback Not Established (FNE) codepoint should be used in the TCP handshake and at the beginning of a data transmission or after an idle time when not sufficient feedback from the receiver is available. In [4] it is recommended to set the first and third data packet of a re-ECN transmission to FNE. Pure ACKs, re-transmissions, window probes, and partial ACKs are supposed to be Not-ECT marked.

Table 1. Description of the extended re-ECN codepoints

ECN field	**RE flag**	**RFC3168 codepoint**	**re-ECN codepoint**	**Worth**	**Description**
00	0	Not-ECT	Not-ECT	–	Not re-ECN capable
00	1	-	FNE	+1	Feedback not established
01	0	ECT(1)	Re-Echo	+1	Re-echoed congestion and RECT
01	1	-	RECT	0	re-ECN capable
10	0	ECT(0)	ECT(0)	–	ECN use only
10	1	-	–	–	unused
11	0	CE	CE(0)	0	Re-Echo cancelled
11	1	-	CE(-1)	-1	Congestion experienced

Table 1 also displays a worth for every re-ECN codepoint. Packets with the RE flag blanked, as well as FNE packets, can be considered as credits for congestion a sender might cause on the path. The FNE codepoint should provide initial credits when no feedback has been observed so far. In contrast, every CE marking of a router counts as debit. As the CE(0) codepoint has the RE flag blanked and is CE marked, it is neutral.

2.2 Overall re-ECN Framework

A re-ECN framework with a policer at network ingress and a so-called dropper at network egress is proposed in [6]. The bottom part of Fig. 1 illustrates the congestion level signaling with re-ECN. The fraction of positive (RE blanked or FNE) packets, illustrated in black, is the whole-path congestion level declared

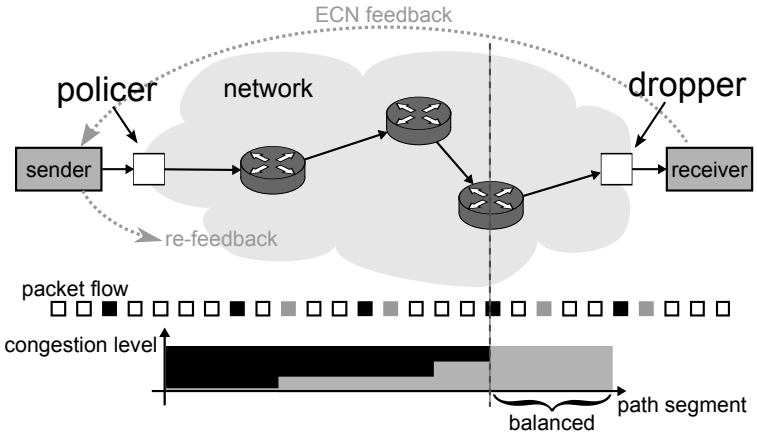

Fig. 1. re-ECN framework and the congestion level resulting from the balance of marks

by the sender. The fraction of negative (CE marked) packets, in gray, increases over the network path as more and more packets get marked by congested routers. At the end of the path the rate of black and gray packets should be about the same if the sender declares honestly the expected congestion level. Hence, the difference between the fraction of positive (Re-Echo or FNE) and negative (CE(-1)) marks of all packets gives the expected congestion level for the remaining path at every point in the network.

The re-ECN framework sketched in Fig. 1 requires a dropper component at network egress to ensure that the sender honestly declares its expected whole-path congestion. If the number of negative marked packets is larger than the number of positive marked ones, the dropper can detect that a sender underestimates its congestion and subsequently penalize its flow through packet drops.

At the same time a policer at network ingress can restrict how much resources one end-system can allocate on congested links, based on end-system's whole-path congestion estimation. Such a policer encourages senders to only cause congestion if really needed or to decrease the sending rate instead. This can for example be implemented through a token bucket mechanism with a certain allowed congestion rate and maximum congestion volume per end-system. If the given limits are exceeded, the policer will enforce the restrictions, e. g., through packet drops. As this paper focuses on the signaling mechanism of re-ECN, the design and implications of droppers and policers are left for further study.

The re-ECN framework provides incentives to end-systems to perform appropriate congestion control. In particular, it creates a motivation to use less than best effort congestion control schemes [11] for bulk data transfers that are not time-critical. Furthermore, it could be one component of a disruptive new Internet capacity sharing architecture that uses new congestion control algorithms not designed to be TCP-friendly (e. g., [12]).

3 re-ECN Implementation

3.1 Implementation of re-ECN in Linux

We have implemented re-ECN for IPv4 and TCP in the Linux kernel 2.6.26. Our implementation conforms to the protocol specification in [4], except that it does not send re-ECN marks after lost packets, which is not required in our studies since all components are ECN capable. In our implementation the CE marks are counted on a per-packet base. In contrast, the implementation of Briscoe *et al.* uses a per-segment solution. Our implementation is completely independent from this code. As all ECN functionality is implemented in separate methods in the Linux kernel, most of the re-ECN processing was simple to realize.

3.2 Lessons Learned

During the development of our re-ECN implementation, we found issues that are not addressed by the existing specification. First, the re-ECN specification does not address how to deal with fragmentation of IP packets. A router that has to fragment IP packets does not know how to set the markings correctly. As a remedy, in our implementation every fragmented IP packet is set to Not-ECT. A similar problem occurs for Generic Segmentation Offload (GSO) which would require the transfer of TCP state information. In addition, the specification [4] recommands to set the first and thrid packet to FNE. This is based on the assumption that the level of congestion is constant, which is not correct. The problems that result form this recommandation are explained in Section 5.2.

A number of specific design choices of the Linux TCP/IP stack also complicates the implementation of re-ECN: SYN cookies as implemented in the Linux kernel cannot be used with ECN and therefore with re-ECN neither. The check of re-ECN information in our implementation is supported by both, the fast and slow path processing of TCP ACKs in the Linux kernel. Slow path processing only would result in significant delays of some positive markings. Furthermore, the Linux stack is not well prepared to handle a new usage of the unused bit 48 in the IP header. Finally, the implementation of the RECN-Co mode is non-trivial, because there is a problem with tracing the transition from '1' to '0' of the ECN Expected Congestion Echo (ECE) flag in the Linux kernel implementation. Anyway, the RECN-Co mode is not able to provide feedback on more than one observed congestion event per RTT.

4 Simulation Setup

4.1 Simulation Tool

In order to investigate the re-ECN performance, the TCP/IP network stack of the Linux kernel was used within a simulation environment. The use of operating system code more realistically models the behavior of a TCP/IP stack, whereas abstract TCP models in simulation tools are often overly simplified. Our tool is based on the IKR Simulation Library [13], and the NSC [9] is used to integrate the kernel code within simulation components.

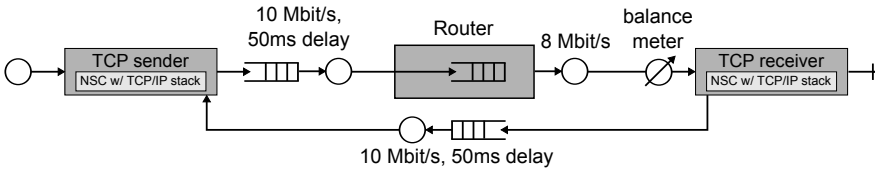

Fig. 2. Simple re-ECN simulation scenario with one TCP sender

4.2 Evaluation Scenario

For a first evaluation of the implications of the re-ECN protocol, a simulation scenario was used with just two end-systems both supporting re-ECN. The setup is displayed in Fig. 2. In the simplest case, one TCP endpoint sends bulk data traffic generated by a greedy source. Alternatively, we use a burst traffic generator with on/off traffic in order to evaluate the impact of small flows on the congestion exposure with re-ECN. We also use simulations with constant bit rate (CBR) or TCP cross traffic. Finally, to emulate a realistic workload scenario [14], we replay Internet traffic traces that are provided in [15]. The flows then start with an negative exponential distributed inter-arrival time of 100 ms.

In the simple setup, the first link between sender and router has a link capacity of 10 Mbit/s and a delay of 50 ms. The second link between router and receiver provides just 8 Mbit/s and is thus the bottleneck. The reverse path also has a capacity of 10 Mbit/s and a delay of 50 ms. This scenario reveals how the information exposed by re-ECN depends on the AQM mechanism. When replaying the real Internet traces, all transmissions share one bottleneck link of 10 Mbit/s in both directions and have an access bandwidth of 100 Mbit/s. We use 9 different RTTs, as recommended in [14].

In every scenario the router on the path uses RED with a queue length of 100 packets. RED's minimum threshold to mark packets is 10 packets. If the average queue size exceeds this threshold, packets are marked with a certain increasing probability. There is a maximum (hard mark) threshold to mark every packet if the average queue size is above 50 packets. The RED algorithm requires a weight factor to calculate the average queue size, which is always set to 1/32 in our simulations. As our RED implementation is based on the Linux kernel code, there is a parameter for the number of random bits which determines the maximum mark probability. This parameter is always set to 9.

In order to study the performance of re-ECN, a meter in front of every receiver counts the total number of negative and positive marked packets. The difference gives the current re-ECN balance. At network egress it should usually be zero as explained in Sect. 2.2. In our calculation we always measure all observed marks from the beginning of the transfer. As there are three FNE packets (SYN, first and third data packet) at the beginning of every data transmission, the rest-of-path congestion is balanced with a value of 3, instead of 0. An alternative to determine the balance would have been a weighted moving average. But for our studies the actual total balance reveals more precise information.

5 Performance Results

5.1 Principle Behavior of re-ECN

As a first step, the fundamental behavior of the re-ECN protocol is evaluated. For this purpose we use the simple scenario described in Sect. 4.2 with one TCP bulk data sender. The sender honestly exposes its observed congestion and can use different congestion control mechanisms. Fig. 3 plots both the congestion window (CWND) size and the re-ECN balance measured by the meter if the sender either uses the Reno [1] or the CUBIC [2] congestion control algorithm.

The diagrams in Fig. 3 show TCP's typical sawtooth pattern in the CWND evolution over time. In the trace of the re-ECN balance, several aspects can be observed: First, the balance is 3 most of the time due to the initial FNE marks. Second, each time packets get marked by the RED queue, there is a negative peak in the re-ECN balance, which is caused by delayed re-insertion of the re-ECN information. The peak size depends on the number of marked

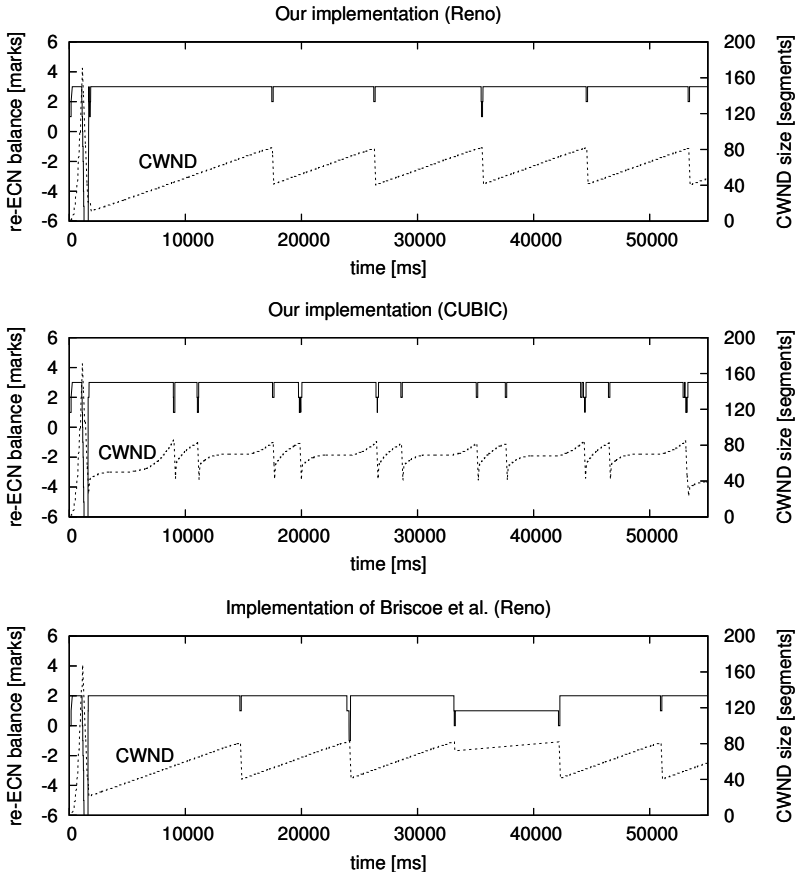

Fig. 3. re-ECN balance of a TCP connection at egress

packets. In all diagrams, there is a significant first negative peak of about -60 in the re-ECN balance, which occurs a couple of RTTs after the connection setup. This is caused by the well-known overshoot effect in TCP Slow-Start. This effect needs to be considered in the design of droppers and might result in a need to identify flow starts. Thereby, the dropper design may get more complicated and the potential for attacks increases. In the long term re-ECN is indented to foster congestion-improved start-up mechanisms.

It must be noted that it takes more than one RTT after the first negative marks until the re-ECN balance is recovered. One RTT is already needed to transport the congestion exposure information. Moreover, the congestion window and respectively the sending rate get reduced as soon as the sender receives the congestion signal. In the examples in Fig. 3 having a one-way delay of 50 ms, the recovery of balance after the first negative peak takes 766 ms for Reno and 622 ms for CUBIC, since several CE marking have appeared in a row.

The first two diagrams in Fig. 3 also show that the amount of congestion depends on the congestion control algorithm of the sender. If the CWND is increased more aggressively, the number of marked packets increases, too. re-ECN correctly exposes the higher congestion level of a more aggressive sender.

The third diagram displays the behavior of the re-ECN implementation of Briscoe *et al.*, using the same simulation scenario. The general behavior is similar. However, the implementation of Briscoe *et al.* results in an incorrect re-ECN balance between two congestion events. This problem seems to be caused by the lack of checking the re-ECN TCP flags in the fast path processing.

5.2 re-ECN Dependencies on Path and Traffic Charateristics

The presented results show that the re-ECN protocol in principle works and correctly informs all network components on the path about the amount of congestion, which would otherwise be known by the sender only. Still, the re-ECN balance will be incorrect if positively marked packets get lost or negatively marked ones get dropped in the receiver, e. g., because of a checksum error. A loss of four subsequent ACKs will cause a mismatch of 8 between the receiver and sender, as the 3 bit counter then may wrap around. Furthermore, the exposure information is at least one RTT delayed, which can result in a permanent negative offset when a congestion situation persists.

Fig. 4 shows the mean negative peak size, apart from the first negative peak in Slow-Start, as a function of the one-way delay, where a negative peak is a temporarily negative balance as to be observed in Fig. 3. The diagram reveals that the latency influences the number of re-ECN markings if the sender uses CUBIC. The larger the RTT of a flow, the larger is the temporary mismatch of the balance. As a result, it does not make sense to police a flow if its re-ECN balance exceeds a fixed negative threshold. Instead, a dropper has to observe a flow over a certain time period to ensure that the balance will not recover. However, the recovery time depends again on the RTT, the congestion control, as well as the number of markings, which complicates dropper design.

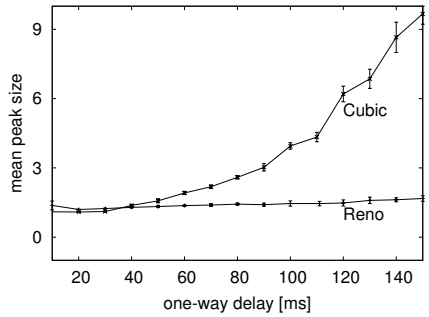

Fig. 4. Mean negative peak size with different one-way delays

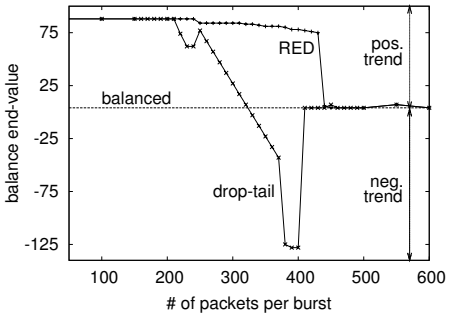

Fig. 5. Balance after 1 minute for burst traffic with variable burst length

The size of a data transfer can also have an impact on the re-ECN balance. In order to illustrate this, we considered traffic of several data bursts with an idle time of about 2 s in between. We also compared the default RED configuration in this paper with an alternative setup, which has a hard mark limit at 10 packets. The latter case emulates a drop-tail marking behavior. Fig. 5 prints the re-ECN balance end-value after 1 minute. The diagram shows two effects.

First, if the burst length is rather short, the total balance of a flow has a large positive value. The re-ECN specification mandates that after an idle period of 1 s some data packets should be FNE marked as the congestion information in the sender is expired. Thus, each burst increases the balance. Such a bursty traffic will consume a lot of the congestion volume in a policer even though no congestion occurs. As a solution to this problem, we suggest not to set FNE for the first and third data packets. TCP SYN packets should still be FNE marked to indicate flow start and prevent end-systems from by-passing the re-insertion of a RE marking by starting a new flow when congestions occurs.

Second, if the marking starts close to the end of one burst, the re-insertion of positive markings cannot be sent any more. If this event occurs several times, the balance at the end of a flow gets more and more negative, as in Fig. 5 for the drop-tail-like RED parametrization. The problem is particularly relevant when the bottleneck marks a large number of packets, e. g., in Slow-Start overshoot. Such a permanent negative balance would cause a re-ECN dropper to penalize a flow even if the sender honestly declares the congestion.

The re-ECN balance also depends on the AQM parameterization, which is still a research challenge [16]. For ECN the total number of marking is mostly irrelevant, as the CWND is reduced at most once per RTT. In contrast, within the re-ECN framework, every marking may cost credits in a policer. Accordingly, different amounts of marks can affect the share of resources obtained by a flow.

5.3 re-ECN Accuracy with Inelastic Cross Traffic

We also investigated a setup where CBR cross traffic competes with a TCP connection. This scenario studies the impact of unresponsive cross traffic on

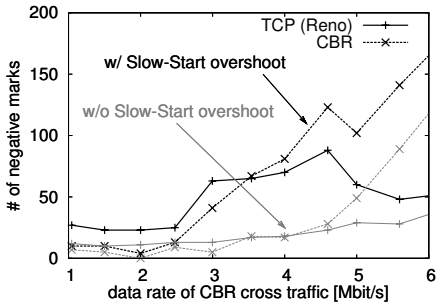

Fig. 6. Number of negative marks with CBR traffic and a TCP Reno sender

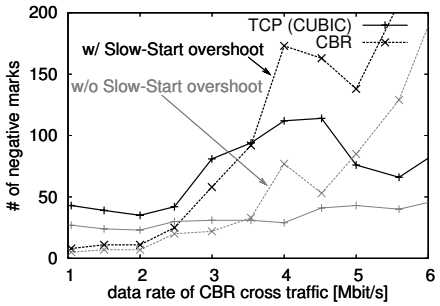

Fig. 7. Number of negative marks with CBR traffic and a TCP CUBIC sender

congestion exposure. It is assumed that the CBR traffic also honestly declares its congestion. As congestion exposure should motivate using an appropriate congestion control, one could assume that the CBR traffic will get more re-ECN marks. However, Fig. 6 and 7 show that this is not necessarily true. In order to explore this effect the CBR data rate has been varied in several simulation runs with one minute simulation time each. The results show that CBR traffic can get less markings than a TCP sender when its data rate is small. Only if the CBR data rate is larger than the mean TCP data rate, the CBR traffic requires more re-ECN credits than a TCP connection. This means that re-ECN will not necessarily achieve an enforcement of congestion control.

Furthermore, the higher the data rate of the CBR traffic, the larger is the amount of markings, since the TCP flow then more frequently exceeds the available bandwidth by probing. The gray lines in the diagram show the re-ECN balance values of the same simulation runs but without the markings caused by the Slow-Start overshoot. Again, the number of markings seen by the CBR traffic increases strongly for high CBR rates, as the TCP connection reduces its sending rate more quickly after a congestion event. Thus, even without the Slow-Start effect, the CBR traffic gets a larger share of the overall markings. Apart from that, Figures 6 and 7 also show that, as to be expected, a more aggressive congestion control such as CUBIC results in more re-ECN markings.

5.4 re-ECN Characteristics in a Realistic Internet Traffic Scenario

Finally, we studied a scenario with real Internet traces. Our setup is characterized by a mean load of 45 % in one hour simulation time. Fig. 8 plots the number of negative marks from the bottleneck router for every marked flow in response direction. As most flows in the replayed traces are short, only very few of them actually get marks. As to be expected, the total number of markings is larger for longer flows, simply because they send more packets. However, for a given flow length, the actual number varies over more than one order of magnitude. Because of the statistical nature, one cannot easily estimate the aggressiveness of a user based on the information of one flow only.

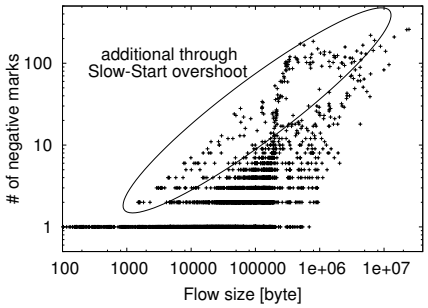

Fig. 8. Number of negative marks over flow size

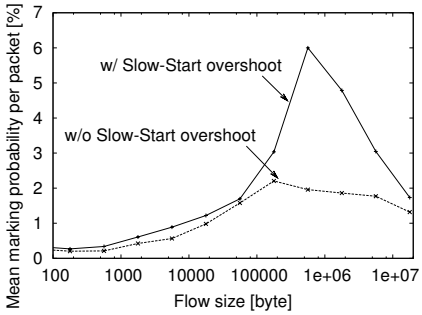

Fig. 9. Mean probability for a negative marking per data packet over flow size

But there is a further dependency on the flow size: Figure 9 shows the mean number of markings divided by the total number of data packets, as a function of the flow length. Ideally, one would expect about the same marking probability for every packet, but this is not true. This effect is again caused by Slow-Start overshooting. A higher per-packet marking probability can be observed for flows that are just long enough to overshoot. Accordingly, fair capacity sharing through policing might not be achieved, as those flows consume proportionally more of the available congestion volume.

We also analyzed the re-ECN balance at the end of each connection. Only 85.5 % of the flows complete with a re-ECN value of 3 or 2, i. e., they are indeed balanced. 10.7 % of the flows have a larger end balance with a maximum value of +2385, and for 3.51 % the final value is smaller, with a minimum value of -53. These numbers reveal again that non-persistent flows can be permanently unbalanced, e. g. by sending sporadically one to three (FNE marked) packets.

6 Conclusion and Outlook

re-ECN is a proposed TCP/IP extension that exposes the expected congestion on a network path. In this paper we present an own implementation of the re-ECN protocol in the Linux TCP/IP network stack and evaluation results of different simulation scenarios. As re-ECN is just a signaling protocol, we demonstrated that it correctly signals congestion. However, our results show that the information depends on many factors, such as the flow size, the RTT, or AQM parameters. As soon as network components react to re-ECN information, this could imply disadvantages for certain data transports, e. g., with a certain length. We conclude that it is crucial to take such effects into account when interpreting the re-ECN information. In particular, when using the re-ECN information for congestion policing, the absolute number of markings is relevant. Market competition should prevent network providers from overstating congestion and the control mechanism enforced by the dropper should restrain end-systems from understating. Both need further research.

In order to show the advantage of congestion exposure, more research is needed in various subtopics. This includes the design of the proposed network elements (dropper, policer), as well as the usage of the exposed information in the network, e. g., for multi-path routing. Another open issue is the interaction of re-ECN with applications if those would be responsible for realizing an appropriate congestion control, e. g., by adapting the data rate of a video stream or by delaying a file sharing transfer. All these design decisions depend on the investigated characteristics of the re-ECN protocol. The re-ECN protocol also leaves open how a receiver could be made accountable for the congestion caused by a download from a server. Finally, the overall economic implications of congestion exposure for accounting require further studies, e. g., concerning the usage across different network domains and corresponding business models.

References

[1] Allman, M., Paxson, V., Blanton, E.: TCP Congestion Control. RFC 5681, IETF (September 2009)
[2] Ha, S., Rhee, I., Xu, L.: CUBIC: A new TCP-friendly high-speed TCP variant. ACM SIGOPS Operating System Review 42(5), 64–74 (2008)
[3] Bastian, C., Klieber, T., Livingood, J., Mills, J., Woundy, R.: Comcast's Protocol-Agnostic Congestion Management System. Internet draft, IETF (2009)
[4] Briscoe, B., Jacquet, A., Moncaster, T., Smith, A.: Re-ECN: Adding Accountability for Causing Congestion to TCP/IP. Internet draft, IETF (September 2009)
[5] Floyd, S.: TCP and explicit congestion notification. ACM SIGCOMM Computer Communication Review 24(5), 8–23 (1994)
[6] Briscoe, B., Jacquet, A., Di Cairano-Gilfedder, C., Salvatori, A., Soppera, A., Koyabe, M.: Policing Congestion Response in an Internetwork using Re-feedback. ACM SIGCOMM Computer Communication Review 35(4), 277–288 (2005)
[7] Kostopoulos, A. (ed.): D10 - Initial evaluation of social and commercial control progress. Trilogy, EU project FP7 (2009)
[8] Moncaster, T., Krug, L., Menth, M., Araujo, J., Woundy, R.: The Need for Congestion Exposure in the Internet. Internet draft, IETF (2009)
[9] Jansen, S., McGregor, A.: Simulation with Real World Network Stacks. In: Proc. Winter Simulation Conference, pp. 2454–2463 (2005)
[10] Floyd, S., Jacobson, V.: Random Early Detection gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, 397–413 (August 1993)
[11] Shalunov, S.: Low Extra Delay Background Transport (LEDBAT). Internet draft, work in progress, IETF (March 2009)
[12] Mathis, M.: Relentless Congestion Control. In: Proc. PFLDNeT (2009)
[13] IKR, University of Stuttgart: IKR Simulation and Emulation Library (December 2009), `http://www.ikr.uni-stuttgart.de/Content/IKRSimLib/`
[14] Andrew, L., Marcondes, C., Floyd, S., Dunn, L., Guillier, R., Gang, W., Eggert, L., Ha, S., Rhee, I.: Towards a common TCP evaluation suite. In: Proc. PFLDnet (2008)
[15] Website: WAN in Lab – Traffic Traces for TCP Evaluation, `http://wil.cs.caltech.edu/suit/TrafficTraces.php`
[16] Chen, W., Yang, S.-H.: The mechanism of adapting RED parameters to TCP traffic. Computer Communications 32, 1525–1530 (2009)