Copyright Notice

Institute of Communication Networks and Computer Engineering
University of Stuttgart
Pfaffenwaldring 47, D-70569 Stuttgart, Germany
Phone: ++49-711-685-68026, Fax: ++49-711-685-67983
Email: mail@ikr.uni-stuttgart.de, http://www.ikr.uni-stuttgart.de

Design and Evaluation of Schemes for More Accurate ECN Feedback

Mirja Kühlewind
Institute of Communication Networks and Computer Engineering (IKR)
University of Stuttgart, Germany
mirja.kuehlewind@ikr.uni-stuttgart.de

Richard Scheffenegger
NetApp, Inc.
Vienna, Austria
rs@netapp.com

***Abstract*—Explicit Congestion Notification (ECN) is a mechanism in the Internet that allows network nodes to mark packets instead of (early) dropping them in case of overload. A receiver will detect this signal and inform the sender about the congestion situation. The current ECN specification does only allow one feedback signal from the receiver to the sender per Round-Trip Time (RTT). In case a network node is marking more than one packet per RTT, the sender will not notice. Up to now, congestion control using this signal as an input parameter was designed to not react more than once per RTT time. However, new mechanisms like Congestion Exposure (ConEx) or Data Center TCP (DCTCP) need a more accurate feedback than one signal per RTT. This paper proposes two approaches to address this problem by reusing the ECN TCP header bits without allocating further option space. Both approaches are developed based on very different protocol design choices, which makes it hard to evaluate both approaches against each other. This paper takes a first approach for evaluation based on simple simulations. Further evaluations in more realistic scenarios and a Linux kernel implementation are currently work in progress.**

I. Introduction

Explicit Congestion Notification (ECN) is a mechanism that was standardized in 2001 in RFC 3168 [1] (which obsoletes the experimental RFC 2481 from 1999) by the Internet Engineering Task Force (IETF). Today, most operation systems implement ECN but it is hardly used and very rarely supported by network nodes. With ECN, network nodes have the possibility to mark packets instead of dropping them in case of overload. To mark packets before a queue overflows, an Active Queue Management (AQM) algorithm needs to be implemented in the network node. The proper configuration of such an AQM strongly depends on the traffic conditions and thus is an open problem [2]. But in fact, the use of ECN could avoid packet losses due to congestion which would be beneficial for the network performance.

Congestion Exposure (ConEx) is a current activity in the IETF which is based on loss and ECN as a congestion signal. The ConEx working group has been formed in 2010 based on previous research work on the re-ECN mechanism [3]. ConEx aims to re-insert congestion information into the network by a sender. This congestion information given by loss and ECN is already known by the sender due to existing mechanisms in the Transmission Control Protocol (TCP). Exposing this information to the network can be regarded as estimation about the expected congestion on the network path. Based on this information a network node can introduce policing at the network ingress to management congestion later in the network. With a deployment of this additional functionality, ConEx will as well incentive ECN deployment. With the current ECN, only one congestion notification per RTT can be fed back from the receiver to the sender. In case of strong (short-time) congestion this is not sufficient to estimate the actual level of congestion on the link.

Another recent development in congestion control is DCTCP – an approach addressing the TCP incast problem in data centers proposed by Alizadeh *et al.* [4]. DCTCP is based on ECN and proposes a specific AQM configuration with a very low marking threshold to keep queues empty when long-lived flows are present. At the same time this mechanism will provide sufficient buffer space to avoid losses when many small flows hit the same link concurrently. Using such a marking scheme will increase the number of ECN marks. For DCTCP congestion control it is proposed to take the number of marks (within one RTT) into account when decreasing the sending rate. This is done to keep the link utilized. Again, the more accurate number of marks in one RTT would be needed.

Moreover, in congestion control research discussions recently started if the decrease behavior of the sender can be differently for ECN than for loss (see mailing list of the Internet Congestion Control Research Group (ICCRG) of the Internet Research Task Force (IRTF) [5]). The number of markings observed in one RTT could be a valuable input information for such new congestion control schemes.

We, the two authors of this paper, did work on these two different problems separately – ConEx and DCTCP – but discovered the same shortcoming with the ECN mechanism as it is standardized by the IETF. To address this problem we are aiming to replace the current ECN feedback mechanism and reuse the assigned ECN/ECN-Nonce bits in the TCP header, as such a new scheme should be able to fully replace the current one. An additional TCP handshake negotiation, that is backward compatible with the current ECN handshake negotiation mechanism, was already proposed by [3] and is further explained in [6]. Possible approaches to realize the actually feedback mechanism are discussed in this paper.

After some initial discussion, we, the two authors, ended up with two different solutions how to actually feed the ECN information back in a more accurate way. A theoretically

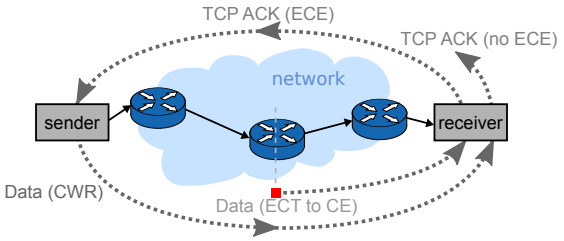


Fig. 1. Classic ECN Feedback Scheme

analysis of requirements on resilience, timeliness, integrity, accuracy and complexity, is discussed in [6]. This analysis could not provide a final evaluation. While the one approach can provide a better accuracy, the other is more resistant against (consecutive) losses of acknowledgements (ACK).

In this paper we provide further evaluation results by some initial simulations with different probabilities of congestion markings and ACK losses. These results give insights on the characteristic of the different protocol design choices. This is still work in progress. Currently, we are implementing both approaches in the Linux kernel. This will provide a better study on implementation complexity. Furthermore, we will uses this code in more realistic simulations, where the marking and loss patterns will be determined by real TCP connections using a certain congestion control and different load scenarios. The results in this paper show that both approach over-/underestimated congestion in certain (over-)load situations. The proposed codepoint-based approach performs well in an Internet-like scenario and as such is a good candidate to provide a more accurate ECN feedback.

This paper is structured the following: The next section explains background information on the current ECN feedback scheme and both newly proposed approaches. Section III explains the simulation setup and scenarios and Section IV provides first evaluation results. Section V draws conclusions on the current state of work and presents next steps.

II. ECN Feedback Schemes

A. Classic Explicit Congestion Notification

Explicit Congestion Notification (ECN) is a TCP/IP mechanism in the Internet that allows network nodes to mark packets instead of (early) dropping them. The ECN scheme as specified in RFC 3168 [1] we call the classic ECN.

In the TCP handshake an ECN sender can negotiate for ECN support. Then, if the receiver is ECN capable, a sender can mark each IP packet as ECN-capable transport (ECT) by setting one of the two IP ECN bits. Setting one or the other bit leads to two flags, the ECT(0) or ECT(1), which are used with ECN-Nonce to provide an integrity mechanism.

If an IP packet is marked as ECT(0) or ECT(1), a network node can mark this packet as Congestion Experienced (CE) by setting the other IP bit. A network node will use an AQM mechanism like e.g. Random Early Detection (RED) to mark packets before the queue overflows. RED calculates a marking probability depending on the queue length. If the queue grows, which means the congestion level increases, each packet will be marked with an increasing probability.

If a classic ECN receiver sees a CE flag, it will set the ECN-Echo bit in the TCP header of the ACK. The ECE bit is then set in all subsequent ACKs until a packet with the Congestion Window Reduced (CWR) bit set in the TCP header is received from the sender to acknowledge the ECE. Thus for one received CE a whole RTT of ECE marked packets will be sent. During this period additional received CE marks will have no influence. The original sender will set the CWR bit after reducing the sending rate/congestion window on reception of the first ECE bit as shown in Figure 1. The sender will not reduce more than once per RTT. ECN-None [7] uses one more TCP bit to signal a one-bit Nonce Sum (NS), which counts the number ECT(1) flags received.

To negotiate a more accurate scheme but keep backwards compatible to the classic ECN, we set the NS in the initial SYN of TCP handshake additionally to the ECN negotiation. The NS bit is currently unused in the SYN. Further details are explained in [6]. All TCP and IP bits used by ECN and ECN-Nonce are shown in Figure 2.

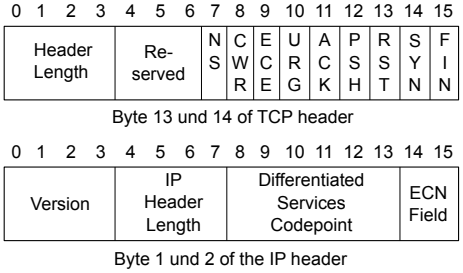


Fig. 2. ECN/ECN-Nonce TCP/IP header bits

B. One Bit Scheme

With the One Bit (1B) feedback scheme an ECN receiver will set the ECE bit only once when a CE flag was received. The ECE is set in the corresponding acknowledgment. Thus the sender is able to not only detect the number of CE markings but also the number of acknowledged payload bytes of the marked packet. To provide this additional functionality the 1B scheme uses a specific ACK'ing scheme.

Most TCP endpoints will only send one ACK for every second data packet received, so-called delayed ACKs [8]. The receiver will acknowledge the payload bytes of two packets within one ACK. In contrast the 1B scheme needs to send separate ACKs for CE marked packets and not-marked once. Whenever an CE flag is received and the previous packet was not CE marked and not ack'ed yet, the endpoint has to send a separate 'late' ACK for the previous packet. Similar action should be triggered when the currently received packet is not CE but the previous was and it not ack'ed yet. Thus an endpoint needs to send a late ACK with the ECE bit set and acknowledge the new packet separately (and potential delay it until the next packet arrives). A similar scheme has been proposed by Alizadeh *et al.* with DCTCP [4]. For the 1B

ECN feedback scheme the proposed mechanism was extended to send a late ACK for previously unack'ed data when the congestion status changes.

With the 1B scheme the loss of one ACK (having the ECE bit set) can cause an information loss. To provide a slightly higher reliability the CWR will be set in the subsequent ACK. If now the ACK with the ECE bit set gets loss, the subsequent ACK with CWR set will acknowledge all bytes of the previous (lost) ACK and the current ACK. An sender seeing such an ACK cannot exactly know how many bytes where supposed to be ack'ed by the previous, lost ACK with ECE set. Thus the sender might account all bytes of the current ACK as CE marked. We propose to account the number of ack'ed minus the Maximum Segment Size (MSS) as at least one packet need to be not CE marked if only CWR but not ECE is set.

```
if (ACK & ECE)
    account(acked_bytes)
else if ((ACK & CWR) & !(pre_ACK & ECE))
    account(acked_bytes - MSS)
```

To be even more conservative, all ack'ed bytes could also be accounted if two consecutive ACK losses or more have been detected. Such a detection could be done if the ack'ing scheme is known, as e.g. TCP delayed ACKs where at least every second data packet should be acknowledged. Otherwise, if more than two ACKs in a row get lost, the endpoint might still miss a congestion notification. If ECN is used as an input for congestion control, the endpoint will not adapt its sending appropriately. If the congestion is persistent, the number of ECN marks will increase and the probability that the feedback is received by the endpoint increases as well. This will preclude the escalation to a congestion collapse. If the congestion situation is missed completely, potentially other flows using the same link have reduced their sending rate based on their feedback information. Thus the flow which missed the congestion feedback has now a higher sending rate then the other flows on the same link. But this also means that this flow has a larger marking probability when the next congestion situation occurs as more packets of this flow will be on the link. In average the flows will still share the link equally.

The 1B uses only the ECE and CWR but not the NS. Such it is compatible with ECN-Nonce. The 1B scheme allows a more accurate ECN feedback and provides even an additional functionality to reconstruct the number of CE marked payload bytes. The 1B does not guarantee a reliable ECN feedback if ACK loss occurs. Whereas the classic ECN scheme provides reliably one feedback signal per RTT, as the reception need to be confirmed by the CWR bit. Due to the new proposes ACK scheme the number of ACKs might increase compared to the usual delayed ACK scheme. This can decrease the network performance slightly depending on the congestion level.

C. Codepoint Scheme

With the Codepoint (CP) scheme all three TCP header bits (ECE, CWR, NS) are used as one field. Such a scheme has first been proposed by Briscoe *et al.* with re-ECN [3]. Briscoe *et al.* proposed to use this field to feedback a 3-bit counter of all CE marks received during the connection.

We extend this idea by not sending the CE counter value directly but using the header field to encode 8 codepoints. In addition, we introduce a counter to sum up the number of ECT(1) codepoints received. We call these counters the Congestion Indication (CI) counter and the ECT(1) counter (E1), respectively. The codepoints are used to encode either the current CE counter value or the current ECT(1) counter value. The codepoints are shown in Figure I.

TABLE I
CODEPOINTS

#	TCP ECN field	CI	E1	Description
0	000	0	–	CI is 0, no info on E1
1	001	1	–	CI is 1, no info on E1
2	010	2	–	CI is 2, no info on E1
3	011	3	–	CI is 3, no info on E1
4	100	4	–	CI is 4, no info on E1
5	101	–	0	E1 is 0, no info on CI
6	110	–	1	E1 is 1, no info on CI
7	111	–	2	E1 is 2, no info on CI

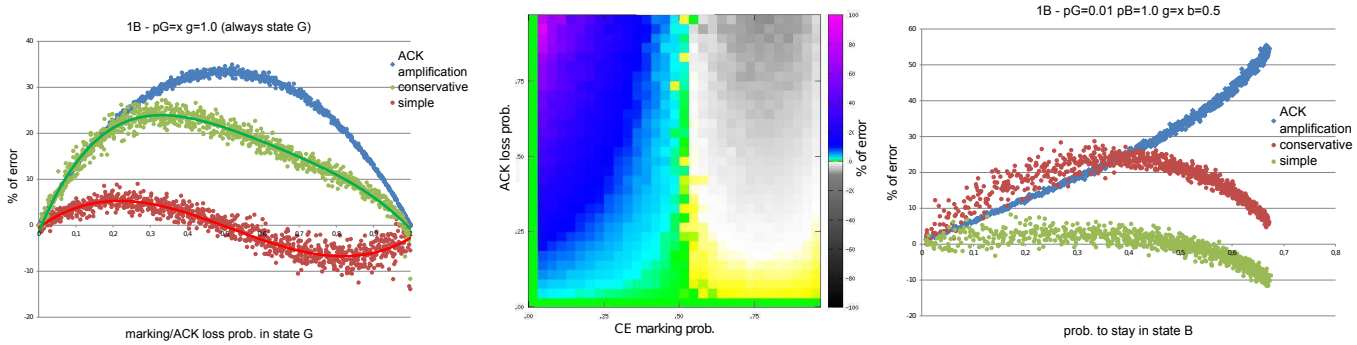
Using the CP scheme, a ECN receiver will feed back the current CI value by default in every ACK, using the codepoints #0 to #4. Only if an ECT(1) is received the E1 value will be signaled, using codepoint #5, #6 or #7, once in the next ACK. Thus an ECN sender can trigger the signaling of the current E1 value by sending an ECT(1) flag. If the sender sets an ECT(1) flag but a network node overwrites it by the CE flag, the receiver will not notice the ECT(1) anymore and will, of course, not signal the E1 value. In this case the ECN-Nonce sender will need to re-synchronize anyway.

The CP scheme uses the NS bit additionally to the ECE and CWR but includes at the same time a way to provide an integrity check as proposed by ECN-Nonce in RFC 3540. This RFC is experimental but is the only proposal to provide integrity in ECN. Such an mechanism is mainly needed if ECN is used for congestion control as the receiver can have an advantage by not feeding back the CE marks.

III. SIMULATION SETUP

In this first setup, we want to investigate the characteristic behavior of both proposals depending on the number of congestion marks and the number of ACK losses. Both schemes provide a more accurate but not exact feedback of the number of CE marks. We used a simple base scenario as illustrated below in Figure 4. A sender is sending packets at a constant bit rate continuously during the whole simulation. With the shown simulation results, all packets are equal sized as most likely in the Internet for an individual one-way data transmission.

The packets are sent over a forward channel and marked with a probability p_{mark}. The receiver, in case of the CP scheme, maintains the CI and E1 counter and, in both cases, acknowledgments are sent with the respective TCP header bits set. Usually only every second packet is acknowledged.



(a) Scenario 1: Gaussian mark/loss prob. (b) CE marking prob. vs. ACK loss prob. (c) Scenario 2: Burst marks/losses

Fig. 3. Simulation results of 1-bit (1B) scheme

In case of B1, a different ack'ing scheme is realized when the ECN state changes. Additionally, the receiver maintains a counter CE_{rcv} that counts the total number of marked packets during the whole simulation. On the feedback channel the acknowledgements have an loss probability of p_{loss}. The sender will reconstruct the number of marked bytes. For the evaluation, we compare the estimated number of marked bytes at the sender CE_{snd} with the exact number CE_{rcv}.

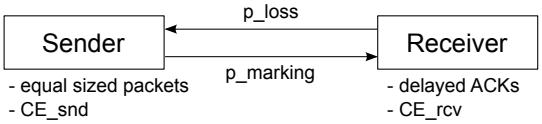


Fig. 4. Simple simulation scenario

In the Internet the predominant flows are TCP flows using some kind of congestion control. That means if a queue is overloaded all TCP flows using the respective link will react to the congestion signal within one RTT. The actual marking/loss pattern will also strongly depend on the used AQM mechanism and its parameters. This can lead to burst of congestion marks/losses. We modeled this scenario with a two state Gilbert Model. We have a certain ACK loss/marking probability in each state. In the scenarios evaluated below, the ACK loss probability and the marking probability always have the same value, which models the same congestion level on the forward and feedback channel. In state G (good) a packet/ACK has a marking/loss probability of pG and a probability of g to stay in this state. Respectively, in state B (Bad/Burst) there is a marking probability of pB and a probability of b to stay in this state. That means

State G (good): $p_{loss} = p_{mark} = pG$; $G \rightarrow B$: $1 - g$
State B (bad/burst): $p_{loss} = p_{mark} = pB$; $B \rightarrow G$: $1 - b$

Based on this model we evaluated two scenarios. Scenario 1 where we always stay in state G with a variable, Gaussian marking/loss probability of pG.

Scenario 1: $pG = x$, $gB = 1.0$, $b = 1.0$ and $g = 0.0$.

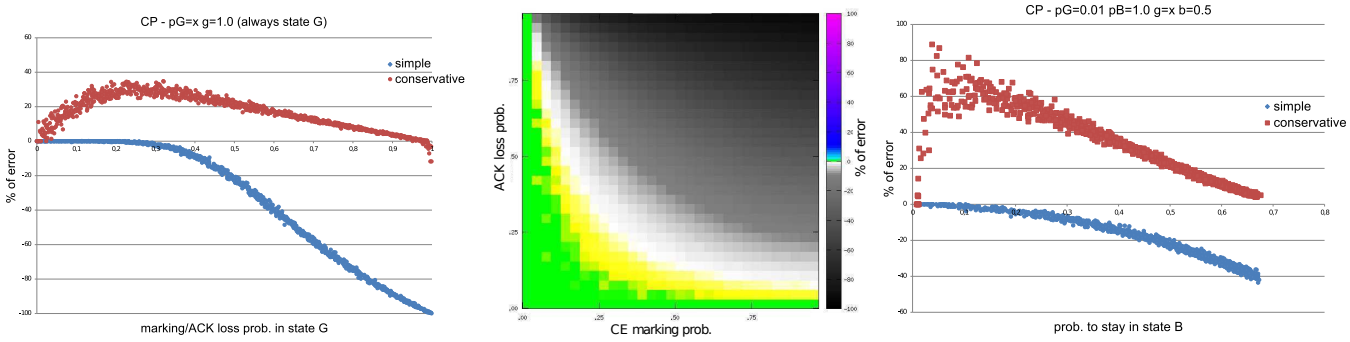
In scenario 2 a probability $b = 0.5$ in the state B, where all packets get marked ($pB = 1.0$), will give us a mean burst size of 2. This provides a critical scenario to investigate where two consecutive ACK losses appear. We evaluated this for a low marking/loss probability $pG = 0.01$ in the non-bursty state G and a variable probability to enter the burst state B, that means with different burst accumulations.

Scenario 2: $pG = 0.01$, $gB = 1.0$, $b = 0.5$ and $g = x$.

IV. Evaluation

We evaluated the accuracy of the proposed more accurate ECN feedback schemes by monitoring the percentage of over-/underestimation of the actually congestion level, given as positive/negative % of error in all figures. For both schemes we evaluated one variant without ACK loss detection, labeled as 'simple' and a more conservative variant, labeled as 'conservative' as explained in section III. This variant always overestimates as loss ACKs are assumed to carry marks.

Figure 3 shows the evaluation of the 1B scheme. We also evaluated the ACK amplification, shown by the blue dots, as this scheme requires a specific ACK response scheme. Figure 3a shows scenario 1. The ACK amplification has a maximum at a marking probability of 0.5 percent. This is expected because the worst case is if every second packet is marked and thus every single packet would need to be ACK'ed separately (which means there are no delayed ACKs at all anymore). In our simulation with a Gaussian marking probability of 0.5, there is an ACK amplification of about 34%. Looking at the simple 1B variant, it overestimates when the marking/loss probability is smaller than 0.5 and underestimates otherwise. This is due to the increasing congestion marking probability. As more ACKs carrying feedback information as larger the chance to lose an ACK that holds valuable information. Losing two consecutive ACKs already will induce a feedback information loss. This can be seen more clearly in Figure 3b where the percentage of error is coded with different colors over different (Gaussian) ACK loss and different independent ECN marking probabilities. In scenario 2, shown in Figure 3c, we have burst markings/losses with mean length of 2. Here the ACK amplification increases and the (under)estimation gets worse with an increasing number of ACK loss bursts.



(a) Scenario 1: Gaussian mark/loss prob. (b) CE marking prob. vs. ACK loss prob. (c) Scenario 2: Burst marks/losses

Fig. 5. Simulation results of codepoint (CP) scheme

Figure 5 shows the evaluation results of the codepoint (CP) scheme. The characteristic is strongly different from the 1B scheme. With the simple variant the estimation is quite accurate up to a certain ACK loss/congestion marking probability and then it always underestimates. In scenario 1 it only underestimates with an high loss/marking level above 20% but in scenario 2 this threshold is much lower. In comparison, the B1 scheme underestimates less. Underestimation can be a more serious problem if e.g. all congestion markings within one RTT are missed completely. In contrast, the conservative variant strongly overestimates but becomes more accurate with high ACK loss/congestion marking probabilities. In the Internet we expect low loss/marking probabilities, where the CP scheme performs well as it can be seen by the green area in Figure 5b. In a data center scenario we expect a very low ACK loss probability but eventual high marking probabilities due to new congestion control/AQM schemes, like DCTCP. For this scenario both schemes perform sufficiently well.

It might be valuable to e.g. use a more conservative estimation for congestion control that ensures to be always TCP-friendly or even less aggressive than TCP. And at the same time one could use the simple estimation variant for e.g. ConEx, where more accurate information are needed but other action can be taken if the estimation is not good enough.

V. Conclusion

In this paper we presented two schemes to achieve a more accurate ECN feedback. In various areas of research limitations of the current ECN feedback scheme were detected. We addressed this problem by reusing the existing ECN/ECN-Nonce bits in the TCP header. Both approaches came up based on different preliminary work and have been evaluated in some first simulations. An implementation in the Linux kernel and, based on this, more realistic simulations are work in progress.

The current first evaluation results show that the proposed codepoint mechanism works well in an Internet-like scenario with low congestion as well as in a data-center-like scenario with potentially large number of congestion markings but a very low ACK loss rate. As ACKs are not transmitted reliable, a scheme like the proposed codepoint (CP) scheme, that retransmits counter values with every ACK, seems to be more robust and as such more suitable. But it still does not provide reliable ECN feedback. To improve the robustness, new counter values could be delayed to not increase the counter within one ACK too much.

The 1-bit (1B) scheme provides an additional functionality by not only announcing a CE mark but also the byte-size of the marked packet. This is valuable information for ConEx and also for future congestion control scheme. Optionally, also with the CP scheme a different ack'ing scheme could be used to provide byte-wise feedback. But this would be orthogonal to the design choices to 1) not rely on the delivery of one certain acknowledgment and 2) not limit the use to one specific ack'ing scheme. Such an ack'ing scheme would be to implemented by the receiver and as such it is not under control of the sender. This might be not desirable as sometimes other ack'ing scheme are already implemented today e.g. ACK accumulation by network device cards.

Currently, we see the CP scheme as good candidate for more accurate ECN feedback. To arrive at a final conclusion, we will do further evaluations in more realistic scenarios and investigate effects on current and future TCP congestion control.

References

[1] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," IETF, RFC 3168, Sep. 2001.
[2] S. Ryu, C. Rump, and C. Qiao, "Advances in congestion control," in *IEEE Communications Surveys*, 2001.
[3] B. Briscoe, A. Jacquet, T. Moncaster, and A. Smith, "Re-ECN: Adding Accountability for Causing Congestion to TCP/IP," IETF, Internet Draft, Sep. 2009.
[4] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "DCTCP: Efficient Packet Transport for the Commoditized Data Center," in *Microsoft Research publications*, January 2010.
[5] "[Iccrg] Incentives for ECN," Thread on ICCRG ,mailing list, 2011. [Online]. Available: http://oakham.cs.ucl.ac.uk/pipermail/iccrg/2011-July/000926.html
[6] M. Kühlewind and R. Scheffenegger, "Accurate ECN Feedback in TCP: draft-kuehlewind-conex-accurate-ecn," IETF, Internet-Draft, July 2011.
[7] N. Spring, D. Wetherall, and D. Ely, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces," IETF, RFC 3540, June 2003.
[8] M. Allman, V. Paxson, and E. Blanton., "TCP Congestion Control," IETF, RFC 5681, Sep. 2009.