TCP Maintenance and Minor Extensions (tcpm)
Internet-Draft
Intended status: Informational
Expires: April 24, 2014

M. Kuehlewind, Ed.
University of Stuttgart
R. Scheffenegger
NetApp, Inc.

October 21, 2013

Problem Statement and Requirements for a More Accurate ECN Feedback draft-ietf-tcpm-accecn-reqs-04

Abstract

Explicit Congestion Notification (ECN) is an IP/TCP mechanism where network nodes can mark IP packets instead of dropping them to indicate congestion to the end-points. An ECN-capable receiver will feedback this information to the sender. ECN is specified for TCP in such a way that only one feedback signal can be transmitted per Round-Trip Time (RTT). Recently, new TCP mechanisms like ConEx or DCTCP need more accurate ECN feedback information in the case where more than one marking is received in one RTT. This document specifies requirements for an update to the TCP protocol to provide more accurate ECN feedback than one signal per RTT.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 24, 2014.

Table of Contents

1. Introduction

Explicit Congestion Notification (ECN) [RFC3168] is an IP/TCP mechanism where network nodes can mark IP packets instead of dropping them to indicate congestion to the end-points. An ECN-capable receiver will feedback this information to the sender. ECN is specified for TCP in such a way that only one feedback signal can be transmitted per Round-Trip Time (RTT). This is sufficient for pre-existing congestion control mechanisms that perform only one reduction in sending rate per RTT, independent of the number of ECN congestion marks. But recently proposed/deployed mechanisms like Congestion Exposure (ConEx) [RFC6789] or DCTCP [Ali10] need more fine-grained ECN feedback information to work correctly in the case where more than one marking is received in any one RTT.

This document lists requirements for a robust and interoperable more accurate TCP/ECN feedback protocol that all implementations of new TCP extension like ConEx and/or DCTCP can use. While a new feedback scheme should still deliver identical performance as classic ECN, this document also clarifies what has to be taken into consideration in addition. Thus the listed requirements should be addressed in the specification of a more accurate ECN feedback scheme. Moreover, as a

large set of proposals already exists, a few high level design choices are sketched and briefly discussed, to demonstrate some of the benefits and drawbacks of each of these potential schemes. A few solutions have already been proposed, so Section 5 demonstrates how to use the requirements to compare them, by briefly sketching their high level design choices and discussing the benefits and drawbacks of each.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

We use the following terminology from [RFC3168] and [RFC3540]:

The ECN field in the IP header:

CE: the Congestion Experienced codepoint,

ECT(0): the first ECN-capable Transport codepoint, and

ECT(1): the second ECN-capable Transport codepoint.

The ECN flags in the TCP header:

CWR: the Congestion Window Reduced flag,

ECE: the ECN-Echo flag, and

NS: ECN Nonce Sum.

In this document, the ECN feedback scheme as specified in [RFC3168] is called the 'classic ECN' and any new proposal the 'more accurate ECN feedback' scheme. A 'congestion mark' is defined as an IP packet where the CE codepoint is set. A 'congestion episode' refers to one or more congestion marks belonging to the same overload situation in the network (usually during one RTT). A TCP segment with the acknowledgment flag set is simply called ACK.

2. Recap of Classic ECN and ECN Nonce in IP/TCP

ECN requires two bits in the IP header. The ECN capability of a packet is indicated when either one of the two bits is set. An ECN

sender can set one or the other bit to indicate an ECN-capable transport (ECT) which results in two signals, ECT(0) and ECT(1). A network node can set both bits simultaneously when it experiences congestion. When both bits are set the packet is regarded as "Congestion Experienced" (CE).

In the TCP header the first two bits in byte 14 are defined as ECN feedback for each half-connection. A TCP receiver signals the reception of a congestion mark using the ECN-Echo (ECE) flag in the TCP header. For reliability, the receiver continues to set the ECE flag on every ACK. To enable the TCP receiver to determine when to stop setting the ECN-Echo flag, the sender sets the CWR flag upon reception of an ECE feedback signal. This always leads to a full RTT of ACKs with ECE set. Thus the receiver cannot signal back any additional CE markings arriving within the same RTT.

The ECN Nonce [RFC3540] is an experimental addition to ECN that the TCP sender can use to protect itself against accidental or malicious concealment of marked or dropped packets. This addition defines the last bit of byte 13 in the TCP header as the Nonce Sum (NS) flag. The receiver maintains a nonce sum that counts the occurrence of ECT(1) packets, and signals the least significant bit of this sum on the NS flag.

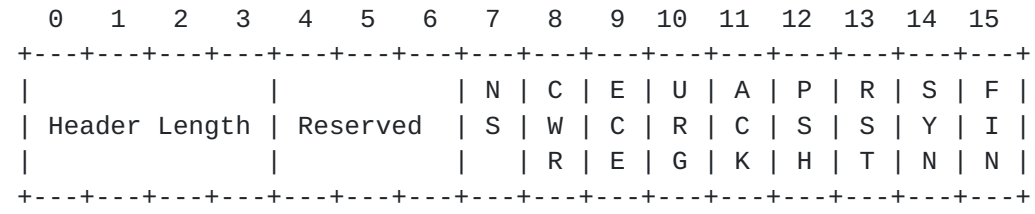

Figure 1: The (post-ECN Nonce) definition of the TCP header flags

However, as ECN is a seperate extension to ECN, even if a sender tries to protect itself with the nonce, any receiver wishing to conceal marked or dropped packets only has to pretent not supporting ECN Nonce and simply not provide any nonce feedback. An alternative for a sender to assure feedback integrity has been proposed where the sender occasionally inserts a CE mark, reordering or loss itself, and checks that the receiver feeds it back faithfully [I-D.moncaster-tcpm-rcv-cheat]. This alternative requires no standardisation and consumes no header bits or codepoints, as well as releasing the ECT(1) codepoint in the IP header and the NS flag in the TCP header for other uses.

3. Use Cases

ConEx is an experimental approach that allows the sender to re-insert the congestion feedback it sees into the forward data path. This is primarily so that any traffic management can be proportionate to actual congestion caused by traffic, rather than limiting traffic based on rate or volume in case it might cause congestion [RFC6789]. A ConEx sender uses selective acknowledgements (SACK [RFC2018]) for fine-grained feedback of loss signals, but currently TCP offers no equivalent fine-grained feedback for ECN.

DCTCP offers very low and predictable queueing delay. DCTCP requires switches/routers to have ECN enabled and configured with no signal smoothing, so it is currently only used in private networks, e.g. internal to data centres. DCTCP was released in Microsoft Windows 8, and implementations exist for Linux and FreeBSD.

The changes DCTCP makes to TCP are not currently the subject of any IETF standardisation activity. The different DCTCP implementations alter TCP's ECN feedback protocol [RFC3168] in unspecified proprietary ways, and they either omit capability negotiation, or they use non-interoperable negotiation. A primary motivation for this document is to prevent each proprietary implementation from inventing its own handshake, which could lead to _de facto_ consumption of the few flags that remain available for standardising capability negotiation. Also, those variants that use the feedback protocol proposed in [Ali10] only work if there are no losses at all, and otherwise they become confused.

The following scenarios should briefly show where the accurate feedback is needed or adds value:

An RFC5681 TCP sender that supports ConEx:
: In this case the ConEx mechanism uses the extra information per RTT to re-echo the precise congestion information, but the congestion control algorithm still ignores multiple marks per RTT [RFC5681].

A sender using DCTCP congestion control without ConEx:
: The congestion control algorithm uses the extra info per RTT to perform its decrease depending on the number of congestion marks.

A sender using DCTCP congestion control and supports ConEx:
: Both the congestion control algorithm and ConEx use the fine-grained ECN feedback mechanism.

A RFC5681 TCP sender without ConEx:

No accurate feedback is necessary here. The congestion control algorithm still reacts on only one signal per RTT. But it is best to have one generic feedback mechanism, whether it is used or not.

Using CE for checking integrity:
If a more accurate ECN feedback scheme would feed all occurrences of CE marks back, a sender could perform interigty checking based in the injection of CE marks. Thereby, a sender will send packets which are deterministically marked with CE (at a low frequency) and keep track if feedback is received for these packets. Of course, the congestion notification feedback for these self-injected markings, does not cause a congestion control react.

4. Requirements

The requirements of the accurate ECN feedback protocol, for the use of e.g. Conex or DCTCP, are to have a fairly accurate (not necessarily perfect), timely and protected signaling. This leads to the following requirements, which should be discussed for any proposed more accurate ECN feedback scheme:

Resilience
The ECN feedback signal is carried within the ACK. TCP ACKs can get lost. Moreover, delayed ACKs are commonly used with TCP. That means in most cases only every second data packet triggers an ACK. In a high congestion situation where most of the packets are marked with CE, an accurate feedback mechanism must still be able to signal sufficient congestion information. Thus the accurate ECN feedback extension has to take delayed ACK and ACK loss into account. Also, a more accurate feedback protocol would still work if delayed ACKs covered more than two packets.

Timeliness
a CE mark can be induced by a network node on the transmission path and is then echoed by the receiver in the TCP ACK. Thus when this information arrives at the sender, it is naturally already about one RTT old. With a sufficient ACK rate a further delay of a small number of ACKs can be tolerated. However, this information will become stale with large delays, given the dynamic nature of networks. TCP congestion control (which itself partly introduces these dynamics) operates on a time scale of one RTT. Thus, to be timely, congestion feedback information should be delivered within about one RTT.

Integrity

A more accurate ECN feedback scheme should assure the integrity of the feedback at least as well as the ECN Nonce or gives strong incentives for the receiver and network nodes to cooperate honestly.

Given there are known problems with the ECN nonce (as identified above), this document only requires that the integrity of the more accurate ECN feedback can be assured; it does not require that the ECN Nonce mechanism is employed to achieve this. Indeed, if integrity could be provided else-wise, a more accurate ECN feedback protocol might re-use the nonce sum (NS) flag in the TCP header.

If the accurate ECN feedback scheme provides sufficient information, the integrity check could e.g. be performed by deterministically setting the CE in the sender and monitoring the respective feedback (similar to ECT(1) and the ECN Nonce sum). If and what kind of enforcements a sender should do, when detecting wrong feedback information, is not part of this document.

Accuracy

Classic ECN feeds back one congestion notification per RTT, which is sufficient for classic TCP congestion control which reduces the sending rate at most once per RTT. The more accurate ECN feedback scheme has to ensure that if a congestion episode occurs, at least one congestion notification is echoed and received per RTT as classic ECN would do. Of course, the goal of a more accurate ECN extension is to reconstruct the number of CE markings more accurately and in the best case even to reconstruct the exact number of payload bytes that a CE marked packet was carrying. However, a sender should not assume to get the exact number of congestion markings or marked bytes in all situations. Moreover, the feedback scheme should preserve the order at which any ECN signal was received. And ideally, it would even be possible for the sender to determine which of the packets (covered by one delayed ACK) were congestion marked, e.g. if the flow consists of packets of different sizes, or to allow for future protocols where the order of the markings may be important.

In fact, the ECN field in the IP header provides four code points. In the best case, a sender that has the more accurate ECN feedback information, would be able to reconstruct the occurrence of any of the four code points. Assuming the sender marks all data packets will at least as

ECN-capable and ETC(0) will be the default setting, feeding the occurrence of CE and ECT(1) back might be sufficient.

Complexity

The implementation should be as simple as possible and only a minimum of additional state information should be needed. This will enable the more accurate ECN feedback to be used as the default feedback mechanism, even if only one ECN feedback signal per RTT is needed. Furthermore, the receiver should not take assumptions about the mechanism that was used to set the markin nor about any interpretation or reaction to the congestion signal. The receiver should only feed the information back as accurate as possible.

Overhead

A more accurate ECN feedback signal should limit the additional network load, because ECN feedback is ultimately not critical information (in the worst case, loss will still be available as a congestion signal of last resort). As feedback information has to be provided frequently and in a timely fashion, potentially all or a large fraction of TCP acknowledgments might carry this information. Ideally, no additional segments should be exchanged compared to an RFC3168 TCP session, and the overhead in each segment should be minimised.

Backward and forward compatibility

Given more accurate ECN feedback will involve a change to the TCP protocol, it will need to be negotiated between the two TCP endpoints. If either end does not support the more accurate feedback, they should both be able to fall-back to classic ECN feedback.

A more accurate ECN feedback extension should aim to be able to traverse most existing middleboxes. Further, a feedback mechanism should provide a method to fall-back to classic RFC3168 signaling if the new signal is suppressed by certain middleboxes.

In order to avoid a fork in the TCP protocol specifications, if experiments with the new ECN feedback protocol are successful, it is intended to eventually update RFC3168 for any TCP/ECN sender, not just for ConEx or DCTCP senders. Therefore, even if only one ECN feedback signal per RTT is needed, it should be possible to use the more accuarte ECN feedback.

5. Design Approaches

All approaches presented below (and proposed so far) are able to provide accurate ECN feedback information as long as no ACK loss occurs and the congestion rate is reasonable. In case of high a high ACK loss rate or very high congestion (CE marking) rate, the proposed schemes have different resilience characteristics depending on the number of used bits for the encoding. While classic ECN provides a reliable (inaccurate) feedback of a maximum of one congestion signal per RTT, the proposed schemes do not implement an explicit acknowledgement mechanism.

5.1. Re-use of ECN/NS Header Bits

The three ECN/NS header, ECE, CWR and NS are re-used (not only for additional capability negotiation during the TCP handshake exchange but) to signal the current value of an CE counter at the receiver. This approach only provides a limited resilience against ACK lost depending of the number of used bits.

Several codings have been proposed so far:

- A one bit scheme sends one ECE for each CE received (to increase the robustness against ACK loss CWR could be used to introduce redundant information on the next ACK);
- A 3-bit counter scheme continuously feeds back the three least significant bits of a CE counter;
- A 3-bit codepoint scheme encodes either a CE counter or an ECT(1) counter in 8 codepoints.

The proposed schemes provide accumulated information on ECN-CE-marking feedback, similar to the number of acknowledged bytes in the TCP header. Due to the limited number of bits the ECN feedback information will wrap much more often than the acknowledgement field. Thus feedback information could be lost due to a relatively small sequence of pure-ACK losses. Resilience could be increased by introducing redundancy, e.g. send each counter increase two or more times. Of course any of these additional mechanisms will increasee the complexity. If the congestion rate is larger that the ACK rate (multiplied by the number of congestion marks that can be signaled per ACK), the congestion information cannot correctly fed back. Thus an accurate ECN feedback mechanism needs to be able to also cover the worst case situation where every packet is CE marked. This can potentially be realized by dynamically adapting the ACK rate and redundancy, which again increases complexity and perhaps the signaling overhead as well. Scheme that do not re-use the ECN NS bit, can still support ECN Nonce.

5.2. Using Other Header Bits

As seen in Figure 1, there are currently three unused flag bits in the TCP header. The proposed 3 bit or codepoint schemes could be extended by one or more bits, to add higher resilience against ACK loss. The relative gain would be proportionally higher resilience against ACK loss, while the respective drawbacks would remain identical.

Alternatively, the receiver could use bits in the Urgent Pointer field to signal more bits of its congestion signal counter, but only whenever it does not set the Urgent Flag. As this is often the case, resilience could be increased without additional header overhead.

Any proposal to use such bits would need to check the likelihood that some middleboxes might discard or 'normalise' the currently unused flag bits or a non-zero Urgent Pointer when the Urgent Flag is cleared.

5.3. Using a TCP Option

Alternatively, a new TCP option could be introduced, to help maintaining the accuracy and integrity of the ECN feedback between receiver and sender. Such an option could provide higher resilience and even more information. E.g. ECN for RTP/UDP provides explicit the number of ECT(0), ECT(1), CE, non-ECT marked and lost packets. However, deploying new TCP options has its own challenges. Moreover, to actually achieve a high resilience, this option would need to be carried by either all or a large number ACKs. Thus this approach would introduce considerable signaling overhead while ECN feedback is

not such a critical information (as in the worst case, loss will still be available to provide a strong congestion feedback signal). Anyway, such a TCP option could also be used in addition to a more accurate ECN feedback scheme in the TCP header or in addition to classic ECN, only when available and needed.

6. Acknowledgements

Thanks to Bob Briscoe for reviewing and providing valuable additons on DCTCP and ConEx. Moreover, thanks to Gorry Fairhurst as well as Bob Briscoe for ideas on CE-based interigty checking.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

Given ECN feedback is used as input for congestion control, the respective algorithm would not react appropriately if ECN feedback were lost and the resilience mechanism to recover it was inadequate. This resilience requirement is articulated in Section 4. However, it should be noted that ECN feedback is not the last resort against congestion collapse, because if there is insufficient response to ECN, loss will ensue, and TCP will still react appropriately to loss.

A receiver could suppress ECN feedback information leading to its connections consuming excess sender or network resources. This problem is similar to that seen with the classic ECN feedback scheme and should be addressed by integrity checking as required in Section 4.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3540] Spring, N., Wetherall, D., and D. Ely, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces", RFC 3540, June 2003.

9.2. Informative References

[Ali10] Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and M. Sridharan, "DCTCP: Efficient Packet Transport for the Commoditized Data Center", Jan 2010.

[I-D.briscoe-tsvwg-re-ecn-tcp] Briscoe, B., Jacquet, A., Moncaster, T., and A. Smith, "Re-ECN: Adding Accountability for Causing Congestion to TCP/IP", draft-briscoe-tsvwg-re-ecn-tcp-09 (work in progress), October 2010.

[I-D.kuehlewind-tcpm-accurate-ecn-option] Kuehlewind, M. and R. Scheffenegger, "Accurate ECN Feedback Option in TCP", draft-kuehlewind-tcpm-accurate-ecn-option-01 (work in progress), July 2012.

[I-D.moncaster-tcpm-rcv-cheat] Moncaster, T., Briscoe, B., and A. Jacquet, "A TCP Test to Allow Senders to Identify Receiver Non-Compliance", draft-moncaster-tcpm-rcv-cheat-02 (work in progress), November 2007.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[RFC5562] Kuzmanovic, A., Mondal, A., Floyd, S., and K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets", RFC 5562, June 2009.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[RFC5690] Floyd, S., Arcia, A., Ros, D., and J. Iyengar, "Adding Acknowledgement Congestion Control to TCP", RFC 5690, February 2010.

[RFC6789] Briscoe, B., Woundy, R., and A. Cooper, "Congestion Exposure (ConEx) Concepts and Use Cases", RFC 6789, December 2012.

Authors' Addresses

Mirja Kuehlewind (editor)
University of Stuttgart
Pfaffenwaldring 47
Stuttgart 70569
Germany

Email: mirja.kuehlewind@ikr.uni-stuttgart.de

Richard Scheffenegger
NetApp, Inc.
Am Euro Platz 2
Vienna 1120
Austria

Phone: +43 1 3676811 3146
Email: rs@netapp.com