TCP Maintenance and Minor Extensions (tcpm)
Internet-Draft
Intended status: Informational
Expires: August 16, 2014

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

February 12, 2014

Problem Statement and Requirements for a More Accurate ECN Feedback
draft-ietf-tcpm-accecn-reqs-05

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 feed this information back to the sender. ECN is specified for TCP in such a way that it can only feed back one congestion signal per Round-Trip Time (RTT). In contrast, ECN for other transport protocols, such as RTP/UDP and SCTP, is specified with more accurate ECN feedback. Recent new TCP mechanisms (like ConEx or DCTCP) need more accurate ECN feedback 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.

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 feed this information back 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 TCP congestion control mechanisms that perform only one reduction in sending rate per RTT, independent of the number of ECN congestion marks. But recently proposed or deployed mechanisms like Congestion Exposure (ConEx) [RFC6789] or Data Center TCP (DCTCP) [Ali10] need more accurate ECN feedback to work correctly in the case where more than one marking is received in any one RTT.

ECN is also defined for transport protocols beside TCP. ECN feedback as defined for RTP/UDP [RFC6679] provides a very detailed level of information, delivering individual counters for all four ECN

codepoints as well as lost and duplicate segments, but at the cost of high signaling overhead. ECN feedback for SCTP [I-D.stewart-tsvwg-sctpecn] delivers a counter for the number of CE marked segments between CWR chunks, but also comes at the cost of increased overhead.

Today, implementations of DCTCP already exist that alter TCP's ECN feedback protocol in proprietary ways (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 standardization activity, and they omit capability negotiation, relying instead on uniform configuration across a across all hosts and network devices with ECN capability. A primary motivation for this document is to intervene before each proprietary implementation invents its own non-interoperable handshake, which could lead to _de facto_ consumption of the few flags or codepoints that remain available for standardizing capability negotiation.

This document lists requirements for a robust and interoperable more accurate TCP/ECN feedback protocol that all implementations of new TCP extensions, like ConEx and/or DCTCP, can use. While a new feedback scheme should still deliver as much information 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. A few solutions have already been proposed. 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. Terminology

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:

Not-ECT: the not ECN-Capable Transport codepoint,

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 'classic ECN' and any new proposal is called a '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 that belong 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. A network node can set both bits simultaneously when it experiences congestion. This leads to the four codepoints (not-ECT, ECT(0), ECT(1), and CE) as listed above.

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 CE-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.

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               |           | N | C | E | U | A | P | R | S | F |
| Header Length | Reserved  | S | W | C | R | C | S | S | Y | I |
|               |           |   | R | E | G | K | H | T | N | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

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

However, as the ECN Nonce is a separate extension to ECN, even if a sender tries to protect itself with the ECN Nonce, any receiver wishing to conceal marked packets only has to pretend not to support the ECN Nonce and simply does not provide any nonce sum feedback.

An alternative for a sender to assure feedback integrity has been proposed where the sender occasionally inserts a CE mark itself (or reordering or loss), and checks that the receiver feeds it back faithfully [I-D.moncaster-tcpm-rcv-cheat]. This alternative requires no standardization 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 a sender to relay congestion feedback provided by the receiver into the network along the forward data path. ConEx information can be used for traffic management to limit traffic proportionate to the actual congestion being caused, rather than limiting traffic based on rate or volume [RFC6789]. A ConEx sender uses selective acknowledgements (SACK) [RFC2018] for accurate feedback of loss signals, but currently TCP offers no equivalent accurate feedback for ECN.

DCTCP offers very low and predictable queuing delay. DCTCP changes the reaction to congestion of a TCP sender and additionally requires switches/routers to have ECN enabled and configured with a low step threshold and no signal smoothing, so it is currently only used in private networks, e.g. internal to data centers. DCTCP was released in Microsoft Windows 8, and implementations exist for Linux and FreeBSD. To retrieve sufficient congestion information, the different DCTCP implementations use a proprietary ECN feedback protocol, but they omit capability negotiation. Moreover, the feedback protocol proposed in [Ali10] only works if there are no losses at all, and otherwise it gets very confused (see Appendix A). Therefore, if a generic more accurate ECN feedback scheme were available, it would solve two problems for DCTCP: i) need for a consistent variant of DCTCP to be deployed network-wide and ii) inability to cope with ACK loss.

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

A sender with standardised TCP congestion control 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 supporting ConEx:

Both the congestion control algorithm and ConEx use the more accurate ECN feedback mechanism.

As-yet-unspecified sender mechanisms:

The above are two examples of more general interest in sender mechanisms that respond to the extent of congestion feedback, not just its existence. It will greatly simplify incremental deployment if the sender can unilaterally deploy new behaviours, and rely on the presence of generic receivers that have already implemented more accurate feedback.

A RFC5681 TCP sender without ConEx:

No accurate feedback is necessary here. The congestion control algorithm still reacts to only one signal per RTT. But it is best to feed back all the information the receiver gets, whether the sender uses it or not -- at least as long as overhead is low or zero.

Using CE for checking integrity:

If a more accurate ECN feedback scheme feeds all occurrences of CE marks back, a sender could perform integrity checking by occasionally injecting CE marks itself. Specifically, a sender can send packets which it randomly marks with CE (at low frequency), then check if feedback is received for these packets. The congestion notification feedback for these self-injected markings, would not require a congestion control reaction [I-D.moncaster-tcpm-rcv-cheat].

4. Requirements

The requirements of the accurate ECN feedback protocol are to have fairly accurate (not necessarily perfect), timely and protected signaling. This leads to the following requirements, which MUST be discussed for any proposed more accurate ECN feedback scheme:

Resilience

The ECN feedback signal is carried within the ACK. Pure TCP ACKs can get lost without recovery (not just due to congestion, but also due to deliberate ACK thinning). Moreover, delayed ACKs are commonly used with TCP. Typically, an ACK is triggered after two data segments (or more e.g., due to receive segment coalescing, ACK compression, ACK congestion control [RFC5690] or other phenomena). In a high congestion situation where most of the packets are marked with CE, an accurate feedback mechanism should still be able to signal sufficient congestion information. Thus the accurate ECN feedback extension has to take delayed ACKs and ACK loss into account. Also, a more accurate feedback protocol should 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 packets 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

It should be possible to assure the integrity of the feedback in a more accurate ECN feedback scheme, at least as well as the ECN Nonce. Alternatively, it should at least be possible to give 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 as an inherent part of the new more accurate ECN feedback

protocol; 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-purpose the nonce sum (NS) flag in the TCP header.

If the more 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). Whether a sender should enforce when it detects wrong feedback information, and what kind of enforcement it should apply, are policy issues that need not be specified as part of more accurate ECN feedback scheme.

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. Thus the more accurate ECN feedback scheme should 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. In the best case the new scheme should even allow reconstruction of the exact number of payload bytes that a CE marked packet was carrying. However, it is accepted that it may be too complex for a sender to get the exact number of congestion markings or marked bytes in all situations. Ideally, the feedback scheme should preserve the order in which any (of the four) ECN signals were 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 the best case, a sender that sees more accurate ECN feedback information would be able to reconstruct the occurrence of any of the four code points (non-ECT, CE, ECT(0), ECT(1)). However, assuming the sender marks all data packets as ECN-capable and uses the default setting of ECT(0), solely feeding back the occurrence of CE and ECT(1) might be sufficient. Thus a more accurate ECN feedback scheme should at least provide information on these two signals, CE and ECT(1).

If a more accurate ECN scheme can reliably deliver feedback in most but not all circumstances, ideally the scheme should

at least not introduce bias. In other words, undetected loss of some ACKs should be as likely to increase as decrease the sender's estimate of the probability of ECN marking.

Complexity

Implementation should be as simple as possible and only a minimum of additional state information should be needed. This will enable 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 make assumptions about the mechanism that was used to set the markings nor about any interpretation or reaction to the congestion signal. The receiver only needs to faithfully reflect congestion information back to the sender.

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 minimized.

Backward and forward compatibility

Given more accurate ECN feedback will involve a change to the TCP protocol, it should 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 ECN 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. Then future senders will be able to unilaterally deploy new behaviours that exploit the existence of more accurate ECN feedback in receivers (forward compatibility). Conversely, even if another sender only needs one ECN feedback signal per

RTT, it should be able to use more accurate ECN feedback, and simply ignore the excess information.

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 a high ACK loss rate or very high congestion (CE marking) rate, the proposed schemes have different resilience characteristics depending on the number of bits used for the encoding. While classic ECN provides reliable (but inaccurate) feedback of a maximum of one congestion signal per RTT, the proposed schemes do not implement an explicit acknowledgement mechanism for the feedback (as e.g. the ECE / CWR exchange of [RFC3168]).

5.1. Re-Definition of ECN/NS Header Bits

Schemes in this category can additionally use the NS bit for capability negotiation during the TCP handshake exchange. Thus a more accurate ECN could be negotiated without changing the classic ECN negotiation and thus being backwards compatible.

Schemes in this category can simply re-define the ECN header flags, ECE and CWR, to encode the occurrence of a CE marking at the receiver. This approach provides very limited resilience against loss of ACK, particularly pure ACKs (no payload and therefore delivered unreliably).

A couple of schemes have been proposed so far:

- o A naive one-bit scheme that sends one ECE for each CE received could use CWR to increase robustness against ACK loss by introducing redundant information on the next ACK, but this is still highly vulnerable to ACK loss.

- o The scheme defined for DCTCP [Ali10], which toggles the ECE feedback on an immediate ACK whenever the CE marking changes, and otherwise feeds back delayed ACKs with the ECE value unchanged. Appendix A demonstrates that this scheme is still highly ambiguous to the sender if the ACKs are pure ACKs, and if some may have been lost.

Alternatively, the receiver uses the three ECN/NS header flags, ECE, CWR and NS to represent a counter that signals the accumulated number of CE markings it has received. Resilience against loss is better than the flag-based schemes, but still not ideal.

A couple of coding schemes have been proposed so far in this category:

- o A 3-bit counter scheme continuously feeds back the three least significant bits of a CE counter;

- o A scheme that defines a standardised lookup table to map the 8 codepoints onto either a CE counter or an ECT(1) counter.

These 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 increase the complexity. If the congestion rate is greater than the ACK rate (multiplied by the number of congestion marks that can be signaled per ACK), the congestion information cannot correctly be fed back. Covering the worst case where every packet is CE marked can potentially be realized by dynamically adapting the ACK rate and redundancy. This again increases complexity and perhaps the signaling overhead as well. Schemes that do not re-purpose the ECN NS bit, could still support the ECN Nonce.

5.2. Using Other Header Bits

As seen in Figure 1, there are currently three unused flags in the TCP header. The proposed 3-bit counter or codepoint schemes could be extended by one or more bits to add higher resilience against ACK loss. The relative gain would be exponentially 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 'normalize' 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 maintain the accuracy and integrity of ECN feedback between receiver and sender. Such an option could provide higher resilience and even more information. E.g. ECN for RTP/UDP [RFC6679] explicitly provides the number of ECT(0), ECT(1), CE, non-ECT marked and lost packets, and SCTP counts the number of ECN marks [I-D.stewart-tsvwg-sctpecn] between CWR chunks. However, deploying new TCP options has its own challenges. Moreover, to actually achieve high resilience, this option would need to be carried by most or all ACKs. Thus this approach would introduce considerable signaling overhead even though ECN feedback is not extremely critical information (in the worst case, loss will still be available to provide a strong congestion feedback signal). Whatever, such a TCP option could be used in addition to a more accurate ECN feedback scheme in the TCP header or in addition to classic ECN, only when needed and when space is available.

6. Acknowledgements

Thanks to Gorry Fairhurst for ideas on CE-based integrity checking and to Mohammad Alizadeh for suggesting the need to avoid bias. Moverover, thanks to Michael Welzl and Michael Scharf for their feedback.

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, "Data Center TCP (DCTCP)", ACM SIGCOMM CCR 40(4)63-74, October 2010, <http://portal.acm.org/citation.cfm?id=1851192>.

[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.

[I-D.stewart-tsvwg-sctpecn]
Stewart, R., Tuexen, M., and X. Dong, "ECN for Stream Control Transmission Protocol (SCTP)", draft-stewart-tsvwg-sctpecn-05 (work in progress), January 2014.

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

[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.

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.

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

Appendix A. Ambiguity of the More Accurate ECN Feedback in DCTCP

As defined in [Ali10], a DCTCP receiver feeds back ECE=0 on delayed ACKs as long as CE remains 0, and also immediately sends an ACK with ECE=0 when CE transitions to 1. Similarly, it continually feeds back ECE=1 on delayed ACKs while CE remains 1 and immediately feeds back ECE=1 when CE transitions to 0. A sender can unambiguously decode this scheme if there is never any ACK loss, and the sender assumes there will never be any ACK loss.

The following two examples show that the feedback sequence becomes highly ambiguous to the sender, if either of these conditions is broken. Below, '0' will represent ECE=0, '1' will represent ECE=1 and '.' will represent a gap of one segment between delayed ACKs. Now imagine that the sender receives the following sequence of feedback on 3 pure ACKs:

```
0.0.0
```

When the receiver sent this sequence it could have been any of the following four sequences:

a. 0.0.0 (0 x CE)

b. 010.0 (1 x CE)

c. 0.010 (1 x CE)

d. 01010 (2 x CE)

where any of the 1s represent a possible pure ACK carrying ECE feedback that could have been lost. If the sender guesses (a), it might be correct, or it might miss 1 or 2 congestion marks over 5 packets. Therefore, when confronted with this simple sequence (that is not contrived), a sender can guess that congestion might have been 0%, 20% or 40%, but it doesn't know which.

Sequences with a longer gap (e.g. 0...0.0) become far more ambiguous. It helps a little if the sender knows the distance the receiver uses between delayed ACKs, and it helps a lot if the distance is 1, i.e. no delayed ACKs, but even then there will still be ambiguity whenever there are pure ACK losses.

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

Bob Briscoe
BT
B54/77, Adastral Park
Martlesham Heath
Ipswich IP5 3RE
UK

Phone: +44 1473 645196
Email: bob.briscoe@bt.com
URI: http://bobbriscoe.net/