Network Working Group
Internet-Draft
Intended status: Experimental
Expires: October 15, 2017

M. Bagnulo
UC3M
B. Briscoe
Simula Research Lab

April 13, 2017

Adding Explicit Congestion Notification (ECN) to TCP control packets and TCP retransmissions

draft-bagnulo-tcpm-generalized-ecn-01

Abstract

This document describes an experimental modification to ECN when used with TCP. It allows the use of ECN on the following TCP packets: SYNs, Pure ACKs, Window probes, FINs, RSTs and retransmissions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

RFC 3168 [RFC3168] specifies support of Explicit Congestion Notification (ECN) in IP (v4 and v6). By using the ECN capability, switches performing Active Queue Management (AQM) can use ECN marks instead of packet drops to signal congestion to the endpoints of a communication. This results in lower packet loss and increased performance. RFC 3168 also specifies support for ECN in TCP, but solely on data packets. For various reasons it precludes the use of

ECN on TCP control packets (TCP SYN, TCP SYN-ACK, pure ACKs, Window probes) and on retransmitted packets. RFC 3168 is silent about the use of ECN on RST and FIN packets. RFC 5562 [RFC5562] is an experimental modification to ECN that enables ECN support for TCP SYN-ACK packets.

This document defines an experimental modification to ECN [RFC3168] that enables ECN support on all the aforementioned types of TCP packet. [I-D.ietf-tsvwg-ecn-experimentation] is a standards track procedural device that updates RFC 3168 to allow the present experiment, which RFC 3168 would otherwise prohibit.

1.1. Motivation

The absence of ECN support on TCP control packets and retransmissions has a potential harmful effect. In any ECN deployment, non-ECN-capable packets suffer a penalty when they traverse a congested bottleneck. For instance, with a drop probability of 1%, 1% of connection attempts suffer a timeout of about 1 second before the SYN is retransmitted, which is highly detrimental to the performance of short flows. TCP control packets, such as TCP SYNs and pure ACKs, are important for performance, so dropping them is best avoided.

Non-ECN control packets particularly harm performance in environments where the ECN marking level is high. For example, [judd-nsdi] shows that in a data centre (DC) environment where ECN is used (in conjunction with DCTCP), the probability of being able to establish a new connection using a non-ECN SYN packet drops to close to zero even when there are only 16 ongoing TCP flows transmitting at full speed. In this data centre context, the issue is that DCTCP's aggressive response to packet marking leads to a high marking probability for ECN-capable packets, and in turn a high drop probability for non-ECN packets. Therefore non-ECN SYNs are dropped aggressively, rendering it nearly impossible to establish a new connection in the presence of even mild traffic load.

Finally, there are ongoing experimental efforts to promote the adoption of a slightly modified variant of DCTCP (and similar congestion controls) over the Internet to achieve low latency, low loss and scalable throughput (L4S) for all communications [I-D.briscoe-tsvwg-l4s-arch]. In such an approach, L4S packets identify themselves using an ECN codepoint. Preventing TCP control packets from obtaining the benefits of ECN would not only expose them to the prevailing level of congestion loss, but it would also stop them from being classified into the low latency (L4S) queue, which would greatly degrade L4S performance.

1.2. Experiment goals

The goal of the experimental modifications defined in this document is to allow the use of ECN (both ECT and CE codepoints) on all TCP packets. Experiments are expected in the public Internet as well as in controlled environments to understand the following issues:

- How SYNs, Window probes, pure ACKs, FINs, RSTs and retransmissions that carry the ECT(0), ECT(1) or CE codepoints are processed by the TCP endpoints and the network (including routers, firewalls and other middleboxes). In particular we would like to learn if these packets are frequently blocked or if these packets are usually forwarded and processed.

- The scale of deployment of the different flavours of ECN, including [RFC3168], [RFC5562], [RFC3540] and [I-D.ietf-tcpm-accurate-ecn].

- How much the performance of TCP communications is improved by allowing ECN marking of each packet type.

- To identify any issues (including security issues) raised by enabling ECN marking of these packets.

The data gathered through the experiments described in this document, particularly under the first 2 bullets above, will help in the design of the final mechanism (if any) for adding ECN support to the different packet types considered in this document. Whenever data input is needed to assist in a design choice, it is spelled out throughout the document.

Success criteria: The experiment will be a success if we obtain enough data to have a clearer view of the deployability and benefits of ECN marking all TCP packets, as well as any issues. If the results of the experiment show that it is feasible to deploy such changes; that there are gains to be achieved though the changes described in this specification; and that no other major issues may interfere with the deployment of the proposed changes; then it would be reasonable to adopt the proposed changes in a standards track specification that would update RFC 3168.

1.3. Document structure

The remainder of this document is structured as follows. In Section 2, we present the terminology used in the rest of the document. In Section 3, we specify the modifications to provide ECN support to TCP SYNs, pure ACKs, Window probes, FINs, RSTs and retransmissions. We describe both the network behaviour and the

endpoint behaviour. Section 4 discusses variations of the specification that will be necessary to interwork with a number of popular variants or derivatives of TCP. RFC 3168 provides a number of specific reasons why ECN support is not appropriate for each packet type. In Section 5, we revisit each of these arguments and explore the possibility of enabling the ECN capability for each packet type in turn.

2. Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

Pure ACK: A TCP segment with the ACK flag set and no data payload.

SYN: A TCP segment with the SYN (synchronize) flag set. It may carry data if TCP Fast Open is used.

Window probe: Defined in [RFC1122], a window probe is a TCP segment with only one byte of data sent to learn if the receive window is still zero.

FIN: A TCP segment with the FIN (finish) flag set.

RST: A TCP segment with the RST (reset) flag set.

Retransmission: A TCP segment that has been retransmitted by the TCP sender because it determined that the original segment was lost, which may or may not be the case.

ECT: ECN-Capable Transport. One of the two codepoints ECT(0) or ECT(1) in the ECN field [RFC3168] of the IP header (v4 or v6). An ECN-capable sender sets one of these to indicate that both transport end-points support ECN. When this specification says the sender sets an ECT codepoint, by default it means ECT(0). Optionally, it could mean ECT(1), which is in the process of being redefined for use by L4S experiments [I-D.ietf-tsvwg-ecn-experimentation] [I-D.briscoe-tsvwg-ecn-l4s-id].

Not-ECT: The ECN codepoint that indicates that the transport is not ECN-capable.

CE: Congestion Experienced. The ECN codepoint that an intermediate node sets to indicate congestion [RFC3168]. A node sets an increasing proportion of ECT packets to CE as the level of congestion increases.

3. Specification

3.1. Network behaviour

Previously the specification of ECN for TCP [RFC3168] required the sender to set not-ECT on TCP control packets and retransmissions. Some readers might have erroneously interpreted this as a requirement for firewalls, intrusion detection systems, etc. to check and enforce this behaviour. Now that the present experimental specification allows TCP senders to set ECT on all TCP packets (control and data), it needs to be clear that a firewall (or any network node) SHOULD NOT treat any ECN-capable packet differently dependent on what type of TCP packet it is.

The previous sentence says "SHOULD NOT" rather than "MUST NOT" because one potential exception is envisaged. A security function that has detected an ongoing attack MAY drop more ECT marked SYNs than not-ECT marked SYNs. Such a policy MUST NOT be applied routinely. It can only be applied if an attack is detected, and preferably only if it is determined that the ECT capability is intensifying the attack.

3.2. Endpoint behaviour

The changes to the specification of TCP over ECN [RFC3168] defined here solely alter the behaviour of a sending host.

The feedback behaviour at the receiver depends on whether classic ECN TCP feedback [RFC3168] or Accurate ECN (AccECN) TCP feedback [I-D.ietf-tcpm-accurate-ecn] has been negotiated. Nonetheless, neither receiver feedback behaviour is altered by the present specification.

For each type of control packet or retransmission, the following sections detail changes to the sender's behaviour in two respects: i) whether it sets ECT; and ii) its response to congestion feedback. Table 1 summarises these two behaviours for each type of packet, but the relevant subsection below should be referred to for the detailed behaviour. The subsection on the SYN is more complex than the others, because it has to include fall-back behaviour if the ECT packet appears not to have got through, and caching of the outcome to detect persistent failures.

TCP packet type	ECN field if AccECN f/b negotiated*	ECN field if RFC 3168 f/b negotiated*	Congestion Response
SYN	ECT	not-ECT	Reduce IW
SYN-ACK	ECT	ECT	Reduce IW as in [RFC5562]
Pure ACK	ECT	ECT	None or optionally [RFC5690]
W Probe	ECT	ECT	Usual response
FIN	ECT	ECT	None or optionally [RFC5690]
RST	ECT	ECT	N/A
Re-XMT	ECT	ECT	Usual response

Window probe and retransmission are abbreviated to W Probe an Re-XMT.
* For a SYN, "negotiated" means "requested".

Table 1: Summary of sender behaviour. In each case the relevant section below should be referred to for the detailed behaviour

It can be seen that the sender can set ECT in all cases, except if it is not requesting AccECN feedback on the SYN. Therefore it is RECOMMENDED that the experimental AccECN specification [I-D.ietf-tcpm-accurate-ecn] is implemented, because it is expected that ECT on the SYN will give the most significant performance gain, particularly for short flows. Nonetheless, this specification also caters for the case where AccECN feedback is not implemented.

3.2.1. SYN

3.2.1.1. Setting ECT on the SYN

With classic [RFC3168] ECN feedback, the SYN was never expected to be ECN-capable, so the flag provided to feed back congestion was put to another use (it is used in combination with other flags to indicate that the responder supports ECN). In contrast, Accurate ECN (AccECN) feedback [I-D.ietf-tcpm-accurate-ecn] provides a codepoint in the

SYN-ACK for the responder to feed back that the SYN arrived marked CE.

Therefore, a TCP initiator MUST NOT set ECT on a SYN unless it also attempts to negotiate Accurate ECN feedback in the same SYN.

For the experiments proposed here, if the SYN is requesting AccECN feedback, the TCP sender will also set ECT on the SYN. It can ignore the prohibition in section 6.1.1 of RFC 3168 against setting ECT on such a SYN.

The following subsections about the SYN solely apply to this case where the initiator sent an ECT SYN.

3.2.1.2. Caching Failed Connection Attempts

Until AccECN servers become widely deployed, a TCP initiator that implements AccECN and sets ECT on a SYN SHOULD also maintain a cache per server to record any failure of the previous attempt. It SHOULD record whether a server does not support AccECN and MAY record whether the ECT SYN is persistently lost (see fall-back below). The TCP initiator will not subsequently attempt any behaviour recorded as persistently problematic. However, the cache should be arranged to expire so that the initiator will infrequently attempt to check whether each problem has been resolved.

There is no need to cache successful attempts, because the default ECT SYN behaviour performs optimally on success.

Servers that do not support ECN as a whole can be recorded as non-support of AccECN and do not need to be distinguished, because there is no performance penalty in always attempting to negotiate classic [RFC3168] ECN support.

3.2.1.3. SYN Congestion Response

Here, we use IW0 to denote the initial window of the TCP initiator [RFC5681].

If the SYN-ACK returned to the TCP initiator confirms that the server supports AccECN, it will also indicate whether or not the SYN was CE-marked. If the SYN was CE-marked, the initiator MUST reduce its Initial Window (IW) and SHOULD reduce it to 1 SMSS (sender maximum segment size).

If the SYN-ACK shows that the server does not support AccECN, the TCP initiator MUST conservatively reduce its Initial Window and SHOULD reduce it to 1 SMSS. A reduction to greater than 1 SMSS MAY be

appropriate (see discussion below). Conservatism is necessary because a non-AccECN SYN-ACK cannot show whether the SYN was CE-marked.

If the TCP initiator (host A) receives a SYN from the remote end (host B) after it has sent a SYN to B, it indicates the (unusual) case of a simultaneous open. Host A will respond with a SYN-ACK. Host A will probably then receive a SYN-ACK in response to its own SYN, after which it can follow the appropriate one of the two paragraphs above.

In all the above cases, the initiator does not have to back off its retransmission timer as it would in response to a timeout following no response to its SYN [RFC6298], because both the SYN and the SYN-ACK have been successfully delivered through the network. Also, the initiator does not need to exit slow start or reduce ssthresh, which is not even required when a SYN is lost [RFC5681],

> DISCUSSION: In the case where the server does not support AccECN, because we impose a conservative reduction in initial window, we are penalizing those that deploy AccECN with ECT SYNs, rather than improving performance as intended. Nonetheless, if such cases are cached, performance will only suffer on the first attempt to access a non-AccECN server. Also, the data sent initially by a TCP client is often a small request that usually fits within 1 SMSS anyway {ToDo: reference? (this information was given informally by Yuchung Cheng)}.

See Section 4 for cases where TCP Fast Open (TFO [RFC7413]) or an initial window of 10 (IW10 [RFC6928]) are also implemented.

3.2.1.4. Fall-back Following a Lost ECT SYN (or SYN-ACK))

An ECT SYN might be lost due to an over-zealous path element (or server) blocking ECT packets that do not conform to RFC 3168. However, loss is commonplace for numerous other reasons, e.g. congestion loss at a non-ECN queue on the forward or reverse path, transmission errors, etc. Alternatively, the cause of the blockage might be the attempt to negotiate AccECN, or possibly other unrelated options on the SYN.

To expedite connection set-up if, after sending an ECT SYN, the retransmission timer expires, the TCP initiator SHOULD send a SYN with the not-ECT codepoint in the IP header and not attempt to negotiate AccECN. It would make sense to also remove any other experimental fields or options on the SYN, but that will depend on the specification of the other option(s). Other fall-back strategies that are considered to improve performance MAY be adopted.

If the TCP initiator is caching failed connection attempts, it SHOULD NOT give up using ECT on the first SYN of subsequent connection attempts until it is clear that the blockage persistently and specifically affects ECT on SYNs. This is because loss is so commonplace for other reasons.

DISCUSSION: If initial experiments show that blocking of ECT on SYNs is widespread, it MAY be necessary to cache successful attempts as well as failures. Then, if there is no entry in the cache for a particular server, the TCP initiator could send a not-ECT SYN soon after the first ECT SYN. This would reduce the performance penalty for those deploying ECT SYN support.

3.2.2. SYN-ACK

To comply with the present specification, the responder (server) part of a TCP implementation MUST also comply with [RFC5562], which defines the use of ECT on a SYN-ACK and the congestion response of the TCP listener if a SYN-ACK is CE-marked.

Feedback by the initiator in response to a CE-marked SYN-ACK from the responder depends on whether classic ECN feedback or AccECN feedback [I-D.ietf-tcpm-accurate-ecn] has been negotiated. In either case no change is required to RFC 5562 or the AccECN specification respectively.

3.2.3. Pure ACK

For the experiments proposed here, the TCP implementation will set ECT on Pure ACKs. It can ignore the requirement in section 6.1.4 of RFC 3168 to set not-ECT on a Pure ACK.

TCP does not normally detect or respond to loss of pure ACKs. Therefore, any response to CE markings on Pure ACKs is not required in order to comply with the present specification. Nonetheless, a congestion response is not precluded either. It could be arranged using any one of the following approaches.

TCP never acknowledges Pure ACKs. So classic [RFC3168] ECN provides no mechanism to feed back a CE marking on a Pure ACK, unless the feedback is added to the ACK of a later data packet (if one arises).

In contrast, an AccECN receiver [I-D.ietf-tcpm-accurate-ecn] continually feeds back a count of the number of CE-marked packets that it has received (and, if possible, a count of CE-marked bytes). So a TCP sender that has negotiated AccECN and is setting ECT on pure ACKs will receive congestion feedback if any Pure ACKs are CE-marked in transit.

In either case (classic or AccECN feedback), if the TCP sender does receive feedback about CE-markings on Pure ACKs, it will react in the usual way by reducing its congestion window accordingly. This will regulate the rate of any data packets it is sending amongst the Pure ACKs. However, reducing the congestion window will have no effect on the rate of Pure ACKs. So while it is only sending Pure ACKs the sender will not be responding to congestion.

Any pair of TCP end-points can already choose to regulate the rate of Pure ACKs by agreeing to regulate the delayed ACK ratio in response to loss or CE-marking of Pure ACKs, using the Acknowledgement Congestion Control (AckCC) techniques documented in [RFC5690] (informational). However, AckCC is not required.

RFC 5690 proposed new TCP options to address the problems that TCP had no mechanism to allow ECT to be set on Pure ACKs and no mechanism to feed back loss or CE-marking of Pure ACKs. A combination of the present specification and AccECN addresses both these problems, at least for ECN marking. So it might now be possible to design an ECN-specific ACK congestion control scheme without the extra TCP options proposed in RFC 5690. However, such a mechanism is out of scope of the present document.

3.2.4. Window Probe

For the experiments proposed here, the TCP sender will set ECT on window probes. It can ignore the prohibition in section 6.1.6 of RFC 3168 against setting ECT on a window probe.

A window probe contains a single octet, so it is no different from a regular TCP data segment. Therefore a TCP receiver will feed back any CE marking on a window probe as normal (either using classic ECN feedback or AccECN feedback). The sender of the probe will then reduce its congestion window as normal.

A receive window of zero indicates that the application is not consuming data fast enough and does not imply anything about network congestion. Once the receive window opens, the congestion window might become the limiting factor, so it is correct that CE-marked probes reduce the congestion window. However, CE-marking on window probes does not reduce the rate of the probes themselves. This is unlikely to present a problem, given a window probe is sent only every 2 minutes [RFC0793] as long as the receiver is advertising a zero window.

3.2.5. FIN

A TCP implementation can set ECT on a FIN.

A congestion response to a CE-marking on a FIN is not required.

After sending a FIN, the endpoint will not send any more data in the connection. Therefore, even if the FIN-ACK indicates that the FIN was CE-marked (whether using classic or AccECN feedback), reducing the congestion window will not affect anything.

After sending a FIN, a host might send one or more pure ACKs. If it is using one of the techniques in Section 3.2.3 to regulate the delayed ACK ratio for Pure ACKs, it could equally be applied after a FIN. But this is not required.

3.2.6. RST

A TCP implementation can set ECT on a RST.

A congestion response to a CE-marking on a RST is not required (and actually not possible).

The host generating the RST message does not have an open connection after sending it (either because there was no such connection when the packet that triggered the RST message was received or because the packet that triggered the RST message also triggered the closure of the connection).

Moreover, the receiver of a CE-marked RST message can either: i) accept the RST message and close the connection; ii) emit a so-called challenge ACK in response (with suitable throttling) [RFC5961] and otherwise ignore the RST (e.g. because the sequence number is in-window but not the precise number expected next); or iii) discard the RST message (e.g. because the sequence number is out-of-window). In the first two cases there is no point in echoing any CE mark received because the sender closed its connection when it sent the RST. In the third case it makes sense to discard the CE signal as well as the RST. So, in all these cases it does not make sense to generate feedback about a CE mark on a RST message.

The following factors have been considered before deciding whether ECT ought to be allowed on a RST message:

o As explained above, a congestion response by the sender of a CE-marked RST message is not possible;

- So the only reason for the sender setting ECT on a RST would be to improve the reliability of the message's delivery;

- RST messages are used to both mount and mitigate attacks:

  * Spoofed RST messages are used by attackers to terminate ongoing connections, although the mitigations in RFC 5961 have considerably raised the bar against off-path RST attacks;

  * Legitimate RST messages allow endpoints to inform their peers to eliminate existing state that correspond to non existing connections, liberating resources e.g. in DoS attacks scenarios;

- AQMs are advised to disable ECN marking during persistent overload, so:

  * it is harder for an attacker to exploit ECN to intensify an attack;

  * it is harder for a legitimate user to exploit ECN to more reliably mitigate an attack

- Prohibiting ECT on a RST would deny the benefit of ECN to legitimate RST messages, but not to attackers who can disregard RFCs;

- If ECT were prohibited on RSTs, security middleboxes could discard any RSTs that were exploiting ECN to intensify an attack;

- However, unlike a SYN flood, a RST flood is easier to distinguish from legitimate traffic, so it is easier to ignore or eliminate without harming legitimate traffic.

So, on balance, it has been decided that it is not necessary to prohibit ECT on RSTs. However, there is always the possibility that someone might demonstrate a new RST attack that proves this decision to be unwise.

3.2.7. Retransmissions

For the experiments proposed here, the TCP sender will set ECT on retransmitted segments. It can ignore the prohibition in section 6.1.5 of RFC 3168 against setting ECT on retransmissions. Nonetheless, the requirement in RFC 3168 that "the TCP data receiver SHOULD ignore the CE codepoint on out-of-window packets" still holds.

If the TCP sender receives feedback that a retransmitted packet was CE-marked, it will react as it would to any feedback of CE-marking on a data packet.

4. Interaction with popular variants or derivatives of TCP

The following subsections specify additional behaviour necessary when setting ECT on all data and control packets while using the following popular variants or derivatives of TCP: SCTP, TFO, IW10. The subsection on IW10 discusses changes to specifications but does not recommend any, because the specification as it stands is safe, and there is only a corner-case where performance could be occasionally improved.

TCP variants that have been assessed and found not to interact adversely with ECT on TCP control packets are: SYN cookies (see Appendix A of [RFC4987]) and L4S [I-D.briscoe-tsvwg-l4s-arch].

4.1. SCTP

Stream Control Transmission Protocol (SCTP [RFC4960]) is a standards track protocol derived from TCP. SCTP currently does not include ECN support, but a draft on the addition of ECN to SCTP has been produced [I-D.stewart-tsvwg-sctpecn]. This draft avoids setting ECT on control packets and retransmissions, closely following the arguments in RFC 3168. When ECN is finally added to SCTP, experience from experiments on adding ECN support to all TCP packets ought to be directly transferable to SCTP.

4.2. TFO

TCP Fast Open (TFO [RFC7413]) is an experiment to remove the round trip delay of TCP's 3-way hand-shake (3WHS). A TFO initiator caches a cookie from a previous connection with a TFO-enabled server. Then, for subsequent connections to the same server, any data included on the SYN and any other data segments sent directly after the SYN (up to the initial window limit) can be passed directly to the server application, which can then return response data with the SYN-ACK (again, up to the initial window limit).

If a TFO initiator has cached that the server supported ECN in the previous connection, it would be safe to set ECT on any data segments it sends before a SYN-ACK returns from the responder (server). Note that there is no space in the SYN-ACK itself (whether classic or AccECN feedback has been negotiated) to include feedback about any CE on data packets. Nonetheless, it is safe to set ECT on data packets within the handshake because any CE-marking on these data segments can be fed back by the responder on the first data segment it sends

after the SYN-ACK (or on an additional Pure ACK if it has no more data to send).

Note that the prohibition in Section 3.2.1.1 against setting ECT on the SYN if the same SYN is not requesting AccECN feedback still applies.

Strictly even a non-TFO TCP initiator can send up to an initial window of data segments straight after the SYN. However, this is rare because a non-TFO TCP server will not deliver them to the application until the 3WHS completes. Therefore the question of ECT on data segments within the handshake only becomes important with TFO. A TFO initiator's first ever connection with a server never uses a fast open, so the initiator always has a chance to cache whether a server supports ECN before it uses a fast open.

4.3. IW10

IW10 is an experiment to determine whether it is safe for TCP to use an initial window of 10 SMSS [RFC6928].

This subsection does not recommend any additions to the present specification in order to interwork with IW10. The specifications as they stand are safe, and there is only a corner-case where performance could be occasionally improved, as explained below.

As specified in Section 3.2.1.1, a TCP initiator can only set ECT on the SYN if it requests AccECN support. If, however, the SYN-ACK tells the initiator that the responder does not support AccECN, Section 3.2.1.1 advises the initiator to conservatively reduce its initial window to 1 SMSS because, if the SYN was CE-marked, the SYN-ACK has no way to feed that back.

If the initiator implements IW10, it seems rather over-conservative to reduce IW to 1 in this scenario. Nonetheless, it will rarely hit performance if we leave the advice at 1 SMSS, because:

- as long as the initiator is caching failures to negotiate AccECN, subsequent attempts to access the same server will not use ECT on the SYN anyway, so there will no longer be any need to conservatively reduce IW;

- currently it is not common for a TCP initiator (client) to have more than one segment to send {ToDo: evidence/reference?} - IW10 is primarily exploited by TCP servers.

5. Discussion of the arguments in RFC 3168

This section is informative, not normative. It presents counter-arguments against the justifications in the RFC series for disabling ECN marking on each type of packet. First it addresses over-arching arguments used for most packet types, then it addresses the specific arguments for each packet type in turn.

5.1. The reliability argument

Section 5.2 of RFC 3168 states:

> "To ensure the reliable delivery of the congestion indication of the CE codepoint, an ECT codepoint MUST NOT be set in a packet unless the loss of that packet [at a subsequent node] in the network would be detected by the end nodes and interpreted as an indication of congestion."

We believe this argument is overly conservative. The principle to determine whether a packet is ECN-capable ought to be "do no extra harm", meaning that the reliability of a congestion signal's delivery ought to be no worse with ECN than without. In particular, setting the CE codepoint on the very same packet fulfills this criterion, since either the packet is delivered and the CE signal is delivered to the endpoint, or the packet is dropped and the original congestion signal (packet loss) is delivered to the endpoint.

TCP does not deliver control packets reliably. So it is more important to allow control packets to be ECN-capable, which greatly improves reliable delivery of the control packets themselves. This outweighs by far the concern that a CE marking applied to a control packet by one node might subsequently be dropped by another node. Particularly given that, without ECN, the transport does not attempt to detect the drop of most control packets anyway.

5.2. SYNs

RFC 5562 presents two arguments against ECT marking of SYN packets (quoted verbatim):

> "First, when the TCP SYN packet is sent, there are no guarantees that the other TCP endpoint (node B in Figure 2) is ECN-Capable, or that it would be able to understand and react if the ECN CE codepoint was set by a congested router.
>
> Second, the ECN-Capable codepoint in TCP SYN packets could be misused by malicious clients to "improve" the well-known TCP SYN attack. By setting an ECN-Capable codepoint in TCP SYN packets, a

malicious host might be able to inject a large number of TCP SYN packets through a potentially congested ECN-enabled router, congesting it even further."

The first point actually describes two subtly different issues. So below three arguments are countered in turn.

5.2.1. Argument 1a: Loss of congestion notification on the SYN

This argument certainly applied at the time RFC 5562 was written, when no ECN responder mechanism had any logic to recognize or feed back a CE marking on a SYN. The problem was that, during the 3WHS, the flag in the TCP header for ECN feedback (called Echo Congestion Experienced) had been overloaded to negotiate the use of ECN itself. So there was no space for feedback in a SYN-ACK.

The accurate ECN (AccECN) protocol [I-D.ietf-tcpm-accurate-ecn] has since been designed to solve this problem, using a two-pronged approach. First AccECN uses the 3 ECN bits in the TCP header as 8 codepoints, so there is space for the responder to feed back whether there was CE on the SYN. Second a TCP initiator can always request AccECN support on every SYN, and any responder reveals its level of ECN support: AccECN, classic ECN, or no ECN. Therefore, if a responder does indicate that it supports AccECN, the initiator can be sure that, if there is no CE feedback on the SYN-ACK, then there really was no CE on the SYN.

An initiator can combine AccECN with three possible strategies for setting ECT on a SYN:

(S1): Pessimistic ECT with positive cache: The initiator always requests AccECN in the SYN, but without setting ECT. Then it records those servers that confirm that they support AccECN in a cache. On a subsequent connection to any server that supports AccECN, the initiator can then set ECT on the SYN.

(S2): Optimistic ECT: The initiator always sets ECT optimistically on the initial SYN and it always requests AccECN support. Then, if the server response shows it has no AccECN logic (so it cannot feed back a CE mark), the initiator conservatively behaves as if the SYN was CE-marked, by reducing its initial window.

A. With no cache: The optimistic ECT strategy ought to work pretty well without caching any responses.

B. With negative cache: The optimistic ECT strategy can be improved by recording solely those servers that do not

support AccECN. On subsequent connections to these non-AccECN servers, the initiator will still request AccECN but not set ECT on the SYN. Then, the initiator can use its full initial window (if it has enough request data to need it). Longer term, as servers upgrade to AccECN, the initiator will remove them from the cache and use ECT on subsequent SYNs to that server.

(S3): ECT by configuration: In a controlled environment, the administrator can make sure that servers support ECN-capable SYN packets. Examples of controlled environments are single-tenant DCs, and possibly multi-tenant DCs if we assume that each tenant mostly communicates with its own VMs.

For unmanaged environments like the public Internet, the choice is between strategies (S1) and (S2B):

- The "pessimistic ECT with positive cache" strategy (S1) suffers from exposing the initial SYN to the prevailing loss level, even if the server supports ECT on SYNs, but only on the first connection to each AccECN server.

- The "optimistic ECT with negative cache" strategy (S2B) exploits a server's support for ECT on SYNs from the very first attempt. But if the server turns out not to support AccECN, the initiator has to conservatively limit its initial window - usually unnecessarily. Nonetheless, initiator request data (as opposed to server response data) is rarely larger than 1 SMSS anyway (see Section 4.3).

The normative specification for ECT on a SYN in Section 3.2.1 uses the "optimistic ECT with negative cache" strategy on the assumption that an initial window of 1 SMSS is usually sufficient for client requests anyway. For clients that often initially send more than 1 SMSS of data, strategy (S1) could be used during initial deployment and strategy (S2B) later (when the probability of servers supporting AccECN and the likelihood of seeing some CE marking is higher). Also, as deployment proceeds a positive cache (S1) starts off small then grows, while a negative cache (S2B) becomes large at first, then shrinks.

5.2.2. Argument 1b: Unknown Handling of Unexpected ECN

GIven ECT-marked SYN packets have previously been prohibited, it cannot be assumed they will be accepted. According to a study using 2014 data [ecn-pam] from a limited range of vantage points, out of the top 1M Alexa web sites, 4791 (0.82%) IPv4 sites and 104 (0.61%) IPv6 sites failed to establish a connection when they received a TCP

SYN with any ECN codepoint set in the IP header and the appropriate ECN flags in the TCP header. Of these, about 41% failed to establish a connection due to the ECN flags in the TCP header even with a Not-ECT ECN field in the IP header (i.e. despite full compliance with RFC 3168). Therefore adding the ECN-capability to SYNs was increasing connection establishment failures by about 0.4%.

We will need to investigate which of numerous possible causes is leading to these failures. RFC 3168 says "a host MUST NOT set ECT on SYN [...] packets", but it does not say what the responder should do if an ECN-capable SYN arrives. So perhaps some responder implementations are checking that the SYN complies with RFC 3168, then silently ignoring non-compliant SYNs (or perhaps returning a RST). Also some middleboxes (e.g. firewalls) might be discarding non-compliant SYNs themselves. For the future, [I-D.ietf-tsvwg-ecn-experimentation] clarifies that middleboxes "SHOULD NOT" do this, but that does not alter the past.

Whereas RSTs can be dealt with immediately, silent failures introduce a retransmission timeout delay (default 1 second) at the initiator before it attempts any fall back strategy. Ironically, making SYNs ECN-capable is intended to avoid the timeout when a SYN is lost due to congestion. Fortunately, where discard of ECN-capable SYNs is due to policy it will occur predictably, not randomly like congestion. So the initiator can avoid it by caching those sites that do not support ECN-capable SYNs.

This further justifies the use of the "optimistic ECT with negative cache" strategy in Section 3.2.1.

It might seem tempting to first send an ECT SYN and then a non-ECT SYN (possibly with a small delay between them) and only accept the non-ECT connection if it returned first. However, even a cache of a dozen or so sites ought to avoid all ECN-related performance problems with roughly the Alexa top thousand. So it is questionable whether the level of failure of ECT on SYNs warrants always sending two SYNs, particularly given failures at well-maintained sites could reduce if ECT SYNs are standardized.

5.2.3. Argument 2: DoS attacks.

[RFC5562] says that ECT SYN packets could be misused by malicious clients to augment "the well-known TCP SYN attack". It goes on to say "a malicious host might be able to inject a large number of TCP SYN packets through a potentially congested ECN-enabled router, congesting it even further."

We assume this is a reference to the TCP SYN flood attack (see https://en.wikipedia.org/wiki/SYN_flood), which is an attack against a responder end point. We assume the idea of this attack is to use ECT to get more packets through an ECN-enabled router in preference to other non-ECN traffic so that they can go on to use the SYN flooding attack to inflict more damage on the responder end point. This argument could apply to flooding with any type of packet, but we assume SYNs are singled out because their source address is easier to spoof, whereas floods of other types of packets are easier to block.

Mandating Not-ECT in an RFC does not stop attackers using ECT for flooding. Nonetheless, if a standard says SYNs are not meant to be ECT it would make it legitimate for firewalls to discard them. However this would negate the considerable benefit of ECT SYNs for compliant transports and seems unnecessary because RFC 3168 already provides the means to address this concern. In section 7, RFC 3168 says "During periods where ... the potential packet marking rate would be high, our recommendation is that routers drop packets rather then set the CE codepoint..." and this advice is repeated in [RFC7567] (section 4.2.1). This makes it harder for flooding packets to gain from ECT.

Further experiments are needed to test how much malicious hosts can use ECT to augment flooding attacks without triggering AQMs to turn off ECN support (flying "just under the radar"). If it is found that ECT can only slightly augment flooding attacks, the risk of such attacks will need to be weighed against the performance benefits of ECT SYNs.

5.3. Pure ACKs.

RFC 3168 gives the following arguments for not allowing the ECT marking of pure ACKs (ACKs not piggy-backed on data). In section 5.2 it reads:

> "To ensure the reliable delivery of the congestion indication of the CE codepoint, an ECT codepoint MUST NOT be set in a packet unless the loss of that packet in the network would be detected by the end nodes and interpreted as an indication of congestion.
>
> Transport protocols such as TCP do not necessarily detect all packet drops, such as the drop of a "pure" ACK packet; for example, TCP does not reduce the arrival rate of subsequent ACK packets in response to an earlier dropped ACK packet. Any proposal for extending ECN- Capability to such packets would have to address issues such as the case of an ACK packet that was marked with the CE codepoint but was later dropped in the network. We believe that this aspect is still the subject of research, so

> this document specifies that at this time, "pure" ACK packets MUST NOT indicate ECN-Capability."

Later on, in section 6.1.4 it reads:

> "For the current generation of TCP congestion control algorithms, pure acknowledgement packets (e.g., packets that do not contain any accompanying data) MUST be sent with the not-ECT codepoint. Current TCP receivers have no mechanisms for reducing traffic on the ACK-path in response to congestion notification. Mechanisms for responding to congestion on the ACK-path are areas for current and future research. (One simple possibility would be for the sender to reduce its congestion window when it receives a pure ACK packet with the CE codepoint set). For current TCP implementations, a single dropped ACK generally has only a very small effect on the TCP's sending rate."

We next address each of the arguments presented above.

The first argument is a specific instance of the reliability argument for the case of pure ACKs. This has already been addressed by countering the general reliability argument in Section 5.1.

The second argument mentions that a sender does not reduce the load of a stream of pure ACKs even if they are contributing to congestion. Again, given that current TCP does not respond to pure ACK loss, setting ECT on pure ACKs to allow them to carry congestion marks would be no worse than not doing so (and not doing so would be detrimental from a performance perspective).

The proposed AccECN modification to TCP feedback [I-D.ietf-tcpm-accurate-ecn] involves a data receiver repeatedly sending a count of received congestion marks. So AccECN could include marks on pure ACKs in this count, even though it does not ACK pure ACKs themselves. Then the sender of the pure ACKs will reduce its congestion window, which will (correctly) reduce the rate at which it sends any subsequent data. Nonetheless, even if the original sender of the pure ACK does not respond to this feedback, or if it is decided that AccECN will not provide this information, it will still make sense to set ECT on pure ACKs, because the congestion situation will be no worse than it is today with non-ECT pure ACKs.

In summary, allowing ECT (and CE) to be set on pure ACKs is no worse than not doing so (and dropping the pure ACK). In contrast, not setting ECT on pure ACKs is certainly detrimental to performance because when a pure ACK is lost it can prevent the release of new data.

5.4. Window probes

RFC 3168 presents only the reliability argument for preventing setting the ECT codepoint in Window Probe packets. Specifically, Section 6.1.6 states:

> "If a window probe packet is dropped in the network, this loss is not detected by the receiver. Therefore, the TCP data sender MUST NOT set either an ECT codepoint or the CWR bit on window probe packets.
>
> However, because window probes use exact sequence numbers, they cannot be easily spoofed in denial-of-service attacks. Therefore, if a window probe arrives with the CE codepoint set, then the receiver SHOULD respond to the ECN indications."

The reliability argument has already been addressed in Section 5.1.

Allowing ECT on window probes could considerably improve performance because, if a window probe is lost in conditions when the Silly Window Syndrome applies, the sender will stall until the next window probe reaches the receiver (at least 2 minutes later).

On the bright side, RFC 3168 at least specifies the receiver behaviour if a CE-marked window probe arrives, so changing the behaviour ought to be less painful than for other packet types.

5.5. Retransmitted packets.

RFC 3168 says the sender "MUST NOT" set ECT on retransmitted packets. The rationale for this consumes nearly 2 pages of RFC 3168, so the reader is referred to section 6.1.5 of RFC 3168, rather than quoting it all here. There are essentially three arguments namely, reliability, DoS attacks and over-reaction to congestion. We address them in order below.

The reliability argument has already been addressed in Section 5.1.

Protection against DoS attacks is not afforded by prohibiting ECT on retransmitted packets. An attacker can set CE on spoofed retransmissions whether or not it is prohibited by an RFC. Protection against the DoS attack described in RFC 3168 is solely afforded by the requirement that "the TCP data receiver SHOULD ignore the CE codepoint on out-of-window packets". Therefore we propose to allow ECT marking of retransmitted packets, in order to reduce the chance of them being dropped.

Nonetheless, it is important to keep the RFC 3168 advice to ignore the CE codepoint in out-of-window packets. This means that, for those retransmitted packets that arrive at the receiver after the original packet has been properly received, any CE marking will be ignored. There is no problem with that because the delivery of the original packet implies that the sender's original congestion response (when it deemed the packet lost and retransmitted it) was unnecessary. The data receiver is also advised to use the more stringent input check for incoming segments in section 5.2 of [RFC5961].

Finally, the third argument is about over-reacting to congestion. The argument goes that, if a retransmitted packet is dropped, the sender will not detect it, so it will not react again to congestion (it would have reduced its congestion window already when it retransmitted the packet). Whereas, if retransmitted packets can be CE tagged instead of dropped, senders could potentially react more than once to congestion. However, we argue that it is legitimate to respond again to congestion if it still persists in subsequent round trip(s).

Therefore, in all three cases, it is not incorrect to set ECT on retransmissions.

6. Security considerations

Section 3.2.6 considers the question of whether ECT on RSTs will allow RST attacks to be intensified. There are several security arguments presented in RFC 3168 for preventing the ECN marking of TCP control packets and retransmitted segments. We believe all of them have been properly addressed in Section 5, particularly Section 5.2.3 and Section 5.5 on DoS attacks using spoofed ECT-marked SYNs and spoofed CE-marked retransmissions.

7. IANA Considerations

There are no IANA considerations in this memo.

8. Acknowledgments

Thanks to Mirja Kuehlewind and David Black for their useful reviews.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC5562] Kuzmanovic, A., Mondal, A., Floyd, S., and K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets", RFC 5562, DOI 10.17487/RFC5562, June 2009, <http://www.rfc-editor.org/info/rfc5562>.

[I-D.ietf-tcpm-accurate-ecn] Briscoe, B., Kuehlewind, M., and R. Scheffenegger, "More Accurate ECN Feedback in TCP", draft-ietf-tcpm-accurate-ecn-02 (work in progress), October 2016.

[I-D.ietf-tsvwg-ecn-experimentation] Black, D., "Explicit Congestion Notification (ECN) Experimentation", draft-ietf-tsvwg-ecn-experimentation-01 (work in progress), March 2017.

9.2. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC3540] Spring, N., Wetherall, D., and D. Ely, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces", RFC 3540, DOI 10.17487/RFC3540, June 2003, <http://www.rfc-editor.org/info/rfc3540>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <http://www.rfc-editor.org/info/rfc4987>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC5961] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", RFC 5961, DOI 10.17487/RFC5961, August 2010, <http://www.rfc-editor.org/info/rfc5961>.

[RFC5690] Floyd, S., Arcia, A., Ros, D., and J. Iyengar, "Adding Acknowledgement Congestion Control to TCP", RFC 5690, DOI 10.17487/RFC5690, February 2010, <http://www.rfc-editor.org/info/rfc5690>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <http://www.rfc-editor.org/info/rfc6298>.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <http://www.rfc-editor.org/info/rfc6928>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://www.rfc-editor.org/info/rfc7413>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

[I-D.briscoe-tsvwg-ecn-l4s-id] Schepper, K., Briscoe, B., and I. Tsang, "Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay", draft-briscoe-tsvwg-ecn-l4s-id-02 (work in progress), October 2016.

[I-D.briscoe-tsvwg-l4s-arch] Briscoe, B., Schepper, K., and M. Bagnulo, "Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture", draft-briscoe-tsvwg-l4s-arch-02 (work in progress), March 2017.

[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.

[judd-nsdi]
Judd, G., "Attaining the promise and avoiding the pitfalls of TCP in the Datacenter", NSDI 2015, 2015.

[ecn-pam] Trammell, B., Kuehlewind, M., Boppart, D., Learmonth, I., Fairhurst, G., and R. Scheffenegger, "Enabling Internet-Wide Deployment of Explicit Congestion Notification", Int'l Conf. on on Passive and Active Network Measurement (PAM'15) pp193-205, 2015.

Authors' Addresses

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Bob Briscoe
Simula Research Lab

Email: ietf@bobbriscoe.net
URI: http://bobbriscoe.net/