Network Working Group
Internet-Draft
Intended status: Experimental
Expires: November 20, 2017

M. Bagnulo
UC3M
B. Briscoe
Simula Research Lab
May 19, 2017

ECN++: Adding Explicit Congestion Notification (ECN) to TCP Control Packets
draft-bagnulo-tcpm-generalized-ecn-04

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

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 November 20, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Motivation 3
1.2. Experiment Goals 4
1.3. Document Structure 5
2. Terminology 5
3. Specification 6
3.1. Network (e.g. Firewall) Behaviour 6
3.2. Endpoint Behaviour 6
3.2.1. SYN 8
3.2.2. SYN-ACK 11
3.2.3. Pure ACK 12
3.2.4. Window Probe 13
3.2.5. FIN 13
3.2.6. RST 14
3.2.7. Retransmissions 14
4. Rationale 15
4.1. The Reliability Argument 15
4.2. SYNs 16
4.2.1. Argument 1a: Unrecognized CE on the SYN 16
4.2.2. Argument 1b: Unrecognized ECT on the SYN 18
4.2.3. Argument 2: DoS Attacks 20
4.3. SYN-ACKs 20
4.4. Pure ACKs 22
4.4.1. Cwnd Response to CE-Marked Pure ACKs 23
4.4.2. ACK Rate Response to CE-Marked Pure ACKs 24
4.4.3. Summary: Enabling ECN on Pure ACKs 25
4.5. Window Probes 25
4.6. FINs 26
4.7. RSTs 26
4.8. Retransmitted Packets. 27
5. Interaction with popular variants or derivatives of TCP . . . 28
5.1. SCTP 29
5.2. IW10 29
5.3. TFO 30
6. Security Considerations 30
7. IANA Considerations 30
8. Acknowledgments 30
9. References 31
9.1. Normative References 31
9.2. Informative References 31
Authors' Addresses 33

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 relaxes standards track requirements in RFC 3168 that would otherwise preclude these experimental modifications.

The present document also considers the implications for common derivatives and variants of TCP, such as SCTP [RFC4960], if the experiment is successful. One particular variant of TCP adds accurate ECN feedback (AccECN [I-D.ietf-tcpm-accurate-ecn]), without which ECN support cannot be added to SYNs. Nonetheless, ECN support can be added to all the other types of TCP packet whether or not AccECN is also supported.

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. With L4S and potentially other similar cases, 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 classify control packet into a different queue with different network treatment, which may also lead to reordering, further degrading TCP performance.

1.2. Experiment Goals

The goal of the experimental modifications defined in this document is to allow the use of ECN 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 enabling ECN on 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 through 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 5 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 4, we revisit each of these arguments for each packet type to justify why it is reasonable to conduct this experiment.

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.

Window probe: Defined in [RFC0793], 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.

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 set by senders 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 (e.g. Firewall) Behaviour

Previously the specification of ECN for TCP [RFC3168] required the sender to set not-ECT on TCP control packets and retransmissions. Some readers of RFC 3168 might have erroneously interpreted this as a requirement for firewalls, intrusion detection systems, etc. to check and enforce this behaviour. Section 4.3 of [I-D.ietf-tsvwg-ecn-experimentation] updates RFC 3168 to remove this ambiguity. It require firewalls or any intermediate nodes not to treat certain types of ECN-capable TCP segment differently (except potentially in one attack scenario). This is likely to only involve a firewall rule change in a fraction of cases (at most 0.4% of paths according to the tests reported in Section 4.2.2).

In case a TCP sender encounters a middlebox blocking ECT on certain TCP segments, the specification below includes behaviour to fall back to non-ECN. However, this loses the benefit of ECN on control packets. So operators are RECOMMENDED to alter their firewall rules to comply with the requirement referred to above (section 4.3 of [I-D.ietf-tsvwg-ecn-experimentation]).

3.2. Endpoint Behaviour

The changes to the specification of TCP over ECN [RFC3168] defined here solely alter the behaviour of the sending host for each half-connection. All changes can be deployed at each end-point independently of others.

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 RFC3168 f/b negotiated*	Congestion Response
SYN	ECT	not-ECT	Reduce IW
SYN-ACK [RFC5562]	ECT	ECT	Reduce IW as in [RFC5562]
Pure ACK	ECT	ECT	Usual cwnd response and optionally [RFC5690]
W Probe	ECT	ECT	Usual cwnd response
FIN	ECT	ECT	None or optionally [RFC5690]
RST	ECT	ECT	N/A
Re-XMT	ECT	ECT	Usual cwnd 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 (as well as the present specification), 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 two codepoints in the SYN-ACK for the responder to feed back whether or not 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.

> MEASUREMENTS NEEDED: Measurements are needed to verify that if SYN packets with the ECT(0)/ECT(1)/CE codepoints are properly delivered by the network. We need to learn if there are cases if SYN packets are dropped because having the the ECT(0)/ECT(1)/CE codepoints. We also need to learn if the network clears SYN packet with the the ECT(0)/ECT(1)/CE codepoints. In addition, we need measurements to learn how current deployed base of servers react to SYN packets with ECT(0)/ECT(1)/CE codepoints whether they discard it, or process it an return a SYN/ACK packet proceeding with the connection. It would be also useful to measure how the network elements and the servers react to all possible combinations of ECN codepoints and NS/CWR/ECE flags.

3.2.1.2. Caching Lack of Support for ECT on SYNs

Until AccECN servers become widely deployed, a TCP initiator that sets ECT on a SYN (which implies the same SYN also requests AccECN, as required above) SHOULD also maintain a cache per server to record any failure of previous attempts.

The initiator will record any server's SYN-ACK response that does not support AccECN. Subsequently the initiator will not set ECT on a SYN to such a server, but it can still always request AccECN support (because the response will state any earlier stage of ECN evolution that the server supports with no performance penalty). The initiator will discover a server that has upgraded to support AccECN as soon as it next connects, then it can remove the server from its cache and subsequently always set ECT for that server.

If the initiator times out without seeing a SYN-ACK, it will also cache this fact (see fall-back in Section 3.2.1.4 for details).

There is no need to cache successful attempts, because the default ECT SYN behaviour performs optimally on success anyway. Servers that do not support ECN as a whole probably do not need to be recorded separately from non-support of AccECN because the response to a request for AccECN immediately states which stage in the evolution of ECN the server supports (AccECN [I-D.ietf-tcpm-accurate-ecn], classic ECN [RFC3168] or no ECN).

The above strategy is named "optimistic ECT and cache failures". It is believed to be sufficient based on initial measurements and assumptions detailed in Section 4.2.1, which also gives alternative strategies in case larger scale measurements uncover different scenarios.

3.2.1.3. SYN Congestion Response

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 Section 4.2.1). 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].

If an initial window of 10 (IW10 [RFC6928]) is implemented, Section 5 gives additional recommendations.

3.2.1.4. Fall-Back Following No Response to an ECT SYN

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. If other experimental fields or options were on the SYN, it will also be necessary to follow their specifications for fall-back too. It would make sense to co- ordinate all the strategies for fall-back in order to isolate the specific cause of the problem.

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. Even if it does eventually decide to give up on ECT on the SYN, it will probably not need to give up on AccECN on the SYN. In any case, the cache should be arranged to expire so that the initiator will infrequently attempt to check whether the problem has been resolved.

Other fall-back strategies MAY be adopted where applicable (see Section 4.2.2 for suggestions, and the conditions under which they would apply).

3.2.2. SYN-ACK

3.2.2.1. Setting ECT on the SYN-ACK

For the experiments proposed here, the TCP implementation will set ECT on SYN-ACKs. It can ignore the requirement in section 6.1.1 of RFC 3168 to set not-ECT on a SYN-ACK.

The feedback behaviour by the initiator in response to a CE-marked SYN-ACK from the responder depends on whether classic ECN feedback [RFC3168] or AccECN feedback [I-D.ietf-tcpm-accurate-ecn] has been negotiated. In either case no change is required to RFC 3168 or the AccECN specification.

Some classic ECN implementations might ignore a CE-mark on a SYN-ACK, or even ignore a SYN-ACK packet entirely if it is set to ECT or CE. This is a possibility because an RFC 3168 implementation would not necessarily expect a SYN-ACK to be ECN-capable.

> FOR DISCUSSION: To eliminate this problem, the WG could decide to prohibit setting ECT on SYN-ACKs unless AccECN has been negotiated. However, this issue already came up when the IETF first decided to experiment with ECN on SYN-ACKs [RFC5562] and it was decided to go ahead without any extra precautionary measures because the risk was low. This was because the probability of encountering the problem was believed to be low and the harm if the problem arose was also low (see Appendix B of RFC 5562).
>
> MEASUREMENTS NEEDED: Server-side experiments could determine whether this specific problem is indeed rare across the current installed base of clients that support ECN.

3.2.2.2. SYN-ACK Congestion Response

A host that sets ECT on SYN-ACKs MUST reduce its initial window in response to any congestion feedback, whether using classic ECN or AccECN. It SHOULD reduce it to 1 SMSS. This is different to the behaviour specified in an earlier experiment that set ECT on the SYN-ACK [RFC5562]. This is justified in Section 4.3.

The responder does not have to back off its retransmission timer because the ECN feedback proves that the network is delivering packets successfully and is not severely overloaded. Also the responder does not have to leave slow start or reduce ssthresh, which is not even required when a SYN-ACK has been lost.

The congestion response to CE-marking on a SYN-ACK for a server that implements either the TCP Fast Open experiment (TFO [RFC7413]) or the

initial window of 10 experiment (IW10 [RFC6928]) is discussed in Section 5.

3.2.2.3. Fall-Back Following No Response to an ECT SYN-ACK

After the responder sends a SYN-ACK with ECT set, if its retransmission timer expires it SHOULD resend a SYN-ACK with not-ECT set. If other experimental fields or options were on the SYN, it will also be necessary to follow their specifications for fall-back too. It would make sense to co-ordinate all the strategies for fall-back in order to isolate the specific cause of the problem.

The server MAY cache failed connection attempts, e.g. per client access network. If the TCP server is caching failed connection attempts, it SHOULD NOT give up using ECT on the first SYN-ACK of subsequent connection attempts until it is clear that the blockage persistently and specifically affects ECT on SYN-ACKs. This is because loss is so commonplace for other reasons (see Section 3.2.1.4). The cache should be arranged to expire so that the server will infrequently attempt to check whether the problem has been resolved.

This fall-back strategy is the same as that for ECT SYN-ACKs in [RFC5562]. Other fall-back strategies MAY be adopted if found to be more effective, e.g. one retransmission attempt using ECT before reverting to not-ECT.

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.

A host that sets ECT on pure ACKs MUST reduce its congestion window in response to any congestion feedback, in order to regulate any data segments it might be sending amongst the pure ACKs. It MAY also implement AckCC [RFC5690] to regulate the pure ACK rate, but this is not required. Note that, in comparison, TCP Congestion Control [RFC5681] does not require a TCP to detect or respond to loss of pure ACKs at all; it requires no reduction in congestion window or ACK rate.

The question of whether the receiver of pure ACKs is required to feed back any CE marks on them is a matter for the relevant feedback specification ([RFC3168] or [I-D.ietf-tcpm-accurate-ecn]). It is outside the scope of the present specification. Currently AccECN feedback is required to count CE marking of any control packet including pure ACKs. Whereas RFC 3168 is silent on this point, so

feedback of CE-markings might be implementation specific (see Section 4.4.1).

DISCUSSION: An AccECN deployment or an implementation of RFC 3168 that feeds back CE on pure ACKs will be at a disadvantage compared to an RFC 3168 implementation that does not. To solve this, the WG could decide to prohibit setting ECT on pure ACKs unless AccECN has been negotiated. If it does, the penultimate sentence of the Introduction will need to be modified.

MEASUREMENTS NEEDED: Measurements are needed to learn how the deployed base of network elements and servers react to pure ACKs marked with the ECT(0)/ECT(1)/CE codepoints, i.e. whether they are dropped, codepoint cleared or processed.

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 the duration between window probes doubles [RFC1122] as long as the receiver is advertising a zero window (currently minimum 1 second, maximum at least 1 minute [RFC6298]).

MEASUREMENTS NEEDED: Measurements are needed to learn how the deployed base of network elements and servers react to Window probes marked with the ECT(0)/ECT(1)/CE codepoints, i.e. whether they are dropped, codepoint cleared or processed.

3.2.5. FIN

A TCP implementation can set ECT on a FIN.

The TCP data receiver MUST ignore the CE codepoint on incoming FINs that fail any validity check. The validity check in section 5.2 of [RFC5961] is RECOMMENDED.

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.

> MEASUREMENTS NEEDED: Measurements are needed to learn how the deployed base of network elements and servers react to FIN packets marked with the ECT(0)/ECT(1)/CE codepoints, i.e. whether they are dropped, codepoint cleared or processed.

3.2.6. RST

A TCP implementation can set ECT on a RST.

The "challenge ACK" approach to checking the validity of RSTs (section 3.2 of [RFC5961] is RECOMMENDED at the data receiver.

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

> MEASUREMENTS NEEDED: Measurements are needed to learn how the deployed base of network elements and servers react to RST packets marked with the ECT(0)/ECT(1)/CE codepoints, i.e. whether they are dropped, codepoint cleared or processed.

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 TCP data receiver MUST ignore the CE codepoint on incoming segments that fail any validity check. The validity check in section 5.2 of [RFC5961] is RECOMMENDED. This will effectively mitigate an attack that uses spoofed data packets to fool the receiver into feeding back spoofed congestion indications to the

sender, which in turn would be fooled into continually halving its congestion window.

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.

> MEASUREMENTS NEEDED: Measurements are needed to learn how the deployed base of network elements and servers react to retransmissions marked with the ECT(0)/ECT(1)/CE codepoints, i.e. whether they are dropped, codepoint cleared or processed.

4. Rationale

This section is informative, not normative. It presents counter-arguments against the justifications in the RFC series for disabling ECN on TCP control segments and retransmissions. It also gives rationale for why ECT is safe on control segments that have not, so far, been mentioned in the RFC series. First it addresses over-arching arguments used for most packet types, then it addresses the specific arguments for each packet type in turn.

4.1. 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 misplaced. TCP does not deliver most 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 (see motivation in Section 1.1). ECN also improves the reliability and latency of delivery of any congestion notification on control packets, particularly because TCP does not detect the loss of most types of control packet anyway. Both these points outweigh by far the concern that a CE marking applied to a control packet by one node might subsequently be dropped by another node.

The principle to determine whether a packet can be 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 that would otherwise have been dropped 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.

The concern about a CE marking being dropped at a subsequent node might be motivated by the idea that ECN-marking a packet at the first node does not remove the packet, so it could go on to worsen congestion at a subsequent node. However, it is not useful to reason about congestion by considering single packets. The departure rate from the first node will generally be the same (fully utilized) with or without ECN, so this argument does not apply.

4.2. 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.

4.2.1. Argument 1a: Unrecognized CE 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 and cache successes: 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. No cache: The optimistic ECT strategy ought to work fairly well without caching any responses.

B. Cache failures: 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 it is assumed that each tenant mostly communicates with its own VMs.

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

- The "pessimistic ECT and cache successes" 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 and cache failures" 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 {ToDo: reference? (this information was given informally by Yuchung Cheng)}.

The normative specification for ECT on a SYN in Section 3.2.1 uses the "optimistic ECT and cache failures" strategy (S2B) on the assumption that an initial window of 1 SMSS is usually sufficient for client requests anyway. Clients that often initially send more than 1 SMSS of data could use strategy (S1) 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, caching successes (S1) starts off small then grows, while caching failures (S2B) becomes large at first, then shrinks.

> MEASUREMENTS NEEDED: Measurements are needed to determine whether one or the other strategy would be sufficient for any particular client, or whether a particular client would need both strategies in different circumstances.

4.2.2. Argument 1b: Unrecognized ECT on the SYN

Given, until now, ECT-marked SYN packets have 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%.

> MEASUREMENTS NEEDED: In order to get these failures fixed, data will be needed on which of the possible causes below is behind them.

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. For the future, [I-D.ietf-tsvwg-ecn-experimentation] updates RFC 3168 to clarify 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 and cache failures" strategy in Section 3.2.1.

> MEASUREMENTS NEEDED: Experiments are needed to determine whether blocking of ECT on SYNs is widespread, and how many occurrences of problems would be masked by how few cache entries.

If blocking is too widespread for the "optimistic ECT and cache failures" strategy (S2B), the "pessimistic ECT and cache successes" strategy (Section 4.2.1) would be better.

> MEASUREMENTS NEEDED: Then measurements would be needed on whether failures were still widespread on the second connection attempt after the more careful ("pessimistic") first connection.

If so, it might be necessary to send a not-ECT SYN soon after the first ECT SYN (possibly with a delay between them - effectively reducing the retransmission timeout) and only accept the non-ECT connection if it returned first. This would reduce the performance penalty for those deploying ECT SYN support.

> FOR DISCUSSION: If this becomes necessary, how much delay ought to be required before the second SYN? Certainly less than the standard RTO (1 second). But more or less than the maximum RTT expected over the surface of the earth (roughly 250ms)? Or even back-to-back?

However, based on the data above from [ecn-pam], 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 sending two SYNs will be necessary, particularly given failures at well-maintained sites could reduce further once ECT SYNs are standardized.

4.2.3. 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.

4.3. SYN-ACKs

The proposed approach in Section 3.2.2 for experimenting with ECN-capable SYN-ACKs is identical to the scheme called ECN+ [ECN-PLUS]. In 2005, the ECN+ paper demonstrated that it could reduce the average Web response time by an order of magnitude. It also argued that adding ECT to SYN-ACKs did not raise any new security vulnerabilities.

The IETF has already specified an experiment with ECN-capable SYN-ACK packets [RFC5562]. It was inspired by the ECN+ paper, but it

specified a much more conservative congestion response to a CE-marked SYN-ACK, called ECN+/TryOnce. This required the server to reduce its initial window to 1 segment (like ECN+), but then the server had to send a second SYN-ACK and wait for its ACK before it could continue with its initial window of 1 SMSS. The second SYN-ACK of this 5-way handshake had to carry no data, and had to disable ECN, but no justification was given for these last two aspects.

The present ECN experiment uses the ECN+ congestion response, not ECN+/TryOnce. First we argue against the rationale for ECN+/TryOnce given in sections 4.4 and 6.2 of [RFC5562]. It starts with a rather too literal interpretation of the requirement in RFC 3168 that says TCP's response to a single CE mark has to be "essentially the same as the congestion control response to a *single* dropped packet." TCP's response to a dropped initial (SYN or SYN-ACK) packet is to wait for the retransmission timer to expire (currently 1s). However, this long delay assumes the worst case between two possible causes of the loss: a) heavy overload; or b) the normal capacity-seeking behaviour of other TCP flows. When the network is still delivering CE-marked packets, it implies that there is an AQM at the bottleneck and that it is not overloaded. This is because an AQM under overload will disable ECN (as recommended in section 7 of RFC 3168 and repeated in section 4.2.1 of RFC 7567). So scenario (a) can be ruled out. Therefore, TCP's response to a CE-marked SYN-ACK can be similar to its response to the loss of _any_ packet, rather than backing off as if the special _initial_ packet of a flow has been lost.

How TCP responds to the loss of any single packet depends what it has just been doing. But there is not really a precedent for TCP's response when it experiences a CE mark having sent only one (small) packet. If TCP had been adding one segment per RTT, it would have halved its congestion window, but it hasn't established a congestion window yet. If it had been exponentially increasing it would have exited slow start, but it hasn't started exponentially increasing yet so it hasn't established a slow-start threshold.

Therefore, we have to work out a reasoned argument for what to do. If an AQM is CE-marking packets, it implies there is already a queue and it is probably already somewhere around the AQM's operating point - it is unlikely to be well below and it might be well above. So, it does not seem sensible to add a number of packets at once. On the other hand, it is highly unlikely that the SYN-ACK itself pushed the AQM into congestion, so it will be safe to introduce another single segment immediately (1 RTT after the SYN-ACK). Therefore, starting to probe for capacity with a slow start from an initial window of 1 segment seems appropriate to the circumstances. This is the approach adopted in Section 3.2.2.

4.4. Pure ACKs

Section 5.2 of RFC 3168 gives the following arguments for not allowing the ECT marking of pure ACKs (ACKs not piggy-backed on data):

> "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 4.1.

The second argument says that ECN ought not to be enabled unless there is a mechanism to respond to it. However, actually there _is_ a mechanism to respond to congestion on a pure ACK that RFC 3168 has overlooked - the congestion window mechanism. When data segments and pure ACKs are interspersed, congestion notifications ought to regulate the congestion window, whether they are on data segments or

on pure ACKs. Otherwise, if ECN is disabled on Pure ACKs, and if (say) 70% of the segments in one direction are Pure ACKs, about 70% of the congestion notifications will be missed and the data segments will not be correctly regulated.

So RFC 3168 ought to have considered two congestion response mechanisms - reducing the congestion window (cwnd) and reducing the ACK rate - and only the latter was missing. Further, RFC 3168 was incorrect to assume that, if one ACK was a pure ACK, all segments in the same direction would be pure ACKs. Admittedly a continual stream of pure ACKs in one direction is quite a common case (e.g. a file download). However, it is also common for the pure ACKs to be interspersed with data segments (e.g. HTTP/2 browser requests controlling a web application). Indeed, it is more likely that any congestion experienced by pure ACKs will be due to mixing with data segments, either within the same flow, or within competing flows.

This insight swings the argument towards enabling ECN on pure ACKs so that CE marks can drive the cwnd response to congestion (whenever data segments are interspersed with the pure ACKs). Then to separately decide whether an ACK rate response is also required (when they are ECN-enabled). The two types of response are addressed separately in the following two subsections, then a final subsection draws conclusions.

4.4.1. Cwnd Response to CE-Marked Pure ACKs

If the sender of pure ACKs sets them to ECT, the bullets below assess whether the three stages of the congestion response mechanism will all work for each type of congestion feedback (classic ECN [RFC3168] and AccECN [I-D.ietf-tcpm-accurate-ecn]):

Detection: The receiver of a pure ACK can detect a CE marking on it:

* Classic feedback: the receiver will not expect CE marks on pure ACKs, so it will be implementation-dependent whether it happens to check for CE marks on all packets.

* AccECN feedback: the AccECN specification requires the receiver of any TCP packets to count any CE marks on them (whether or not control packets are ECN-capable).

Feedback: TCP never ACKs a pure ACK, but the receiver of a CE-mark on a pure ACK can feed it back when it sends a subsequent data segment (if it ever does):

* Classic feedback: the receiver (of the pure ACKs) would set the echo congestion experienced (ECE) flag in the TCP header as normal.

* AccECN feedback: the receiver 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).

Congestion response: 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.

4.4.2. ACK Rate Response to CE-Marked Pure ACKs

Reducing the congestion window will have no effect on the rate of pure ACKs. The worst case here is if the bottleneck is congested solely with pure ACKs, but it could also be problematic if a large fraction of the load was from unresponsive ACKs, leaving little or no capacity for the load from responsive data.

Since RFC 3168 was published, Acknowledgement Congestion Control (AckCC) techniques have been documented in [RFC5690] (informational). So any pair of TCP end-points can choose to agree to regulate the delayed ACK ratio in response to lost or CE-marked pure ACKs. However, the protocol has a number of open deployment issues (e.g. it relies on two new TCP options, one of which is required on the SYN where option space is at a premium and, if either option is blocked by a middlebox, no fall-back behaviour is specified). The new TCP options addressed two problems, namely that TCP had: i) no mechanism to allow ECT to be set on pure ACKs; and ii) 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.

Setting aside the practicality of RFC 5690, the need for AckCC has not been conclusively demonstrated. It has been argued that the Internet has survived so far with no mechanism to even detect loss of pure ACKs. However, it has also been argued that ECN is not the same as loss. Packet discard can naturally thin the ACK load to whatever the bottleneck can support, whereas ECN marking does not (it queues the ACKs instead). Nonetheless, RFC 3168 (section 7) recommends that an AQM switches over from ECN marking to discard when the marking

probability becomes high. Therefore discard can still be relied on to thin out ECN-enabled pure ACKs as a last resort.

4.4.3. Summary: Enabling ECN on Pure ACKs

In the case when AccECN has been negotiated, the arguments for ECT (and CE) on pure ACKs heavily outweigh those against. ECN is always more and never less reliable for delivery of congestion notification. The cwnd response has been overlooked as a mechanism for responding to congestion on pure ACKs, so it is incorrect not to set ECT on pure ACKs when they are interspersed with data segments. And when they are not, packet discard still acts as the "congestion response of last resort". 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. Separately, AckCC (or perhaps an improved variant exploiting AccECN) could optionally be used to regulate the spacing between pure ACKs. However, it is not clear whether AckCC is justified.

In the case when Classic ECN has been negotiated, there is still an argument for ECT (and CE) on pure ACKs, but it is less clear-cut. Some existing RFC 3168 implementations might happen to (unintentionally) provide the correct feedback to support a cwnd response. Even for those that did not, setting ECT on pure ACKs would still be better for performance than not setting it and do no extra harm. If AckCC was required, it is designed to work with RFC 3168 ECN.

4.5. Window Probes

Section 6.1.6 of RFC 3168 presents only the reliability argument for prohibiting ECT on Window probes:

> "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 4.1.

Allowing ECT on window probes could considerably improve performance because, once the receive window has reopened, if a window probe is lost the sender will stall until the next window probe reaches the

receiver, which might be after the maximum retransmission timeout (at least 1 minute [RFC6928]).

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.

4.6. FINs

RFC 3168 is silent on whether a TCP sender can set ECT on a FIN. A FIN is considered as part of the sequence of data, and the rate of pure ACKs sent after a FIN could be controlled by a CE marking on the FIN. Therefore there is no reason not to set ECT on a FIN.

4.7. RSTs

RFC 3168 is silent on whether a TCP sender can set ECT on a RST. 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.

Although a congestion response following a CE-marking on a RST does not appear to make sense, the following factors have been considered before deciding whether the sender ought to set ECT on a RST message:

- 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

    * it would be easy for security middleboxes to discard all ECN-capable RSTs;

    * However, unlike a SYN flood, it is already easy for a security middlebox (or host) to distinguish a RST flood from legitimate traffic [RFC5961], and even if a some legitimate RSTs are accidentally removed as well, legitimate connections still function.

So, on balance, it has been decided that it is worth experimenting with ECT on RSTs. During experiments, if the ECN capability on RSTs is found to open a vulnerability that is hard to close, this decision can be reversed, before it is specified for the standards track.

4.8. 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 4.1.

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 section 6.1.5 of RFC 3168 is solely afforded by the requirement that "the TCP data receiver SHOULD ignore the CE codepoint on out-of-window packets". Therefore in Section 3.2.7 the sender is allowed to set ECT on retransmitted packets, in order to reduce the chance of them being dropped. We also strengthen the receiver's requirement from "SHOULD ignore" to "MUST ignore". And we generalize the receiver's requirement to include failure of any validity check, not just out-of-window checks, in order to include the more stringent validity checks in RFC 5961 that have been developed since RFC 3168.

A consequence is that, for those retransmitted packets that arrive at the receiver after the original packet has been properly received (so-called spurious retransmissions), any CE marking will be ignored. There is no problem with that because the fact that the original packet has been delivered implies that the sender's original congestion response (when it deemed the packet lost and retransmitted it) was unnecessary.

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.

5. Interaction with popular variants or derivatives of TCP

The following subsections discuss any interactions between setting ECT on all all packets and using the following popular variants or derivatives of TCP: SCTP, IW10 and TFO. This section is informative not normative, because no interactions have been identified that require any change to specifications. The subsection on IW10 discusses potential changes to specifications but recommends that no changes are needed.

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 section 3.1 of [RFC5562]), TCP Fast Open (TFO [RFC7413]) and L4S [I-D.briscoe-tsvwg-l4s-arch].

5.1. SCTP

Stream Control Transmission Protocol (SCTP [RFC4960]) is a standards track protocol derived from TCP. SCTP currently does not include ECN support, but Appendix A of RFC 4960 broadly describes how it would be supported and 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.

5.2. 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 with ECT on the SYN 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 from 10 to 1 just in case a congestion marking was missed. Nonetheless, the reduction to 1 SMSS will rarely harm performance, 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 data segment to send {ToDo: evidence/reference?} - IW10 is primarily exploited by TCP servers.

If a responder receives feedback that the SYN-ACK was CE-marked, Section 3.2.2.2 mandates that it reduces its initial window to 1 SMSS. When the responder also implements IW10, it is particularly important to adhere to this requirement in order to avoid overflowing a queue that is clearly already congested.

5.3. 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 can be passed directly to the server application, which can then return up to an initial window of response data on the SYN-ACK and on data segments straight after it, without waiting for the ACK that completes the 3WHS.

The TFO experiment and the present experiment to add ECN-support for TCP control packets can be combined without altering either specification, which is justified as follows:

- The handling of ECN marking on a SYN is no different whether or not it carries data.

- In response to any CE-marking on the SYN-ACK, the responder adopts the normal response to congestion, as discussed in Section 7.2 of [RFC7413].

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 4, particularly Section 4.2.3 and Section 4.8 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.

The work of Marcelo Bagnulo has been performed in the framework of the H2020-ICT-2014-2 project 5G NORMA. His contribution reflects the

consortiums view, but the consortium is not liable for any use that may be made of any of the information contained therein.

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>.

[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>.

[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-02 (work in progress), April 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>.

[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", USENIX Symposium on Networked Systems Design and Implementation (NSDI'15) pp.145-157, May 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 Passive and Active Network Measurement (PAM'15) pp193-205, 2015.

[ECN-PLUS]
Kuzmanovic, A., "The Power of Explicit Congestion Notification", ACM SIGCOMM 35(4):61--72, 2005.

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/