Workgroup: Network Working Group
Internet-Draft:
draft-ietf-tcpm-generalized-ecn-15
Obsoletes: 5562 (if approved)
Published: 2 December 2023
Intended Status: Experimental
Expires: 4 June 2024
Authors: M. Bagnulo   B. Briscoe, Ed.
UC3M   Independent

ECN++: Adding Explicit Congestion Notification (ECN) to TCP Control Packets

Abstract

This document specifies an experimental modification to ECN when used with TCP. It allows the use of ECN in the IP header of the following TCP packets: SYNs, SYN/ACKs, pure ACKs, Window probes, FINs, RSTs and retransmissions. This specification obsoletes RFC5562, which described a different way to use ECN on SYN/ACKs alone.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction
  1.1. Motivation
  1.2. Experiment Goals
  1.3. Document Structure
2. Terminology
3. Specification
  3.1. Network (e.g. Firewall) Behaviour
  3.2. Sender Behaviour
    3.2.1. SYN (Send)
    3.2.2. SYN-ACK (Send)
    3.2.3. Pure ACK (Send)
    3.2.4. Window Probe (Send)
    3.2.5. FIN (Send)
    3.2.6. RST (Send)
    3.2.7. Retransmissions (Send)
    3.2.8. General Fall-back for any Control Packet or Retransmission
  3.3. Receiver Behaviour
    3.3.1. Receiver Behaviour for Any TCP Control Packet or Retransmission
    3.3.2. SYN (Receive)
    3.3.3. Pure ACK (Receive)
    3.3.4. FIN (Receive)
    3.3.5. RST (Receive)
    3.3.6. Retransmissions (Receive)
4. Rationale
  4.1. The Reliability Argument
  4.2. SYNs
    4.2.1. Argument 1a: Unrecognized CE on the SYN
    4.2.2. Argument 1b: ECT Considered Invalid on the SYN
    4.2.3. Caching Strategies for ECT on SYNs
    4.2.4. Argument 2: DoS Attacks
  4.3. SYN-ACKs
    4.3.1. Possibility of Unrecognized CE on the SYN-ACK
    4.3.2. Response to Congestion on a SYN-ACK
    4.3.3. Fall-Back if ECT SYN-ACK Fails
  4.4. Pure ACKs
    4.4.1. Arguments against ECT on Pure ACKS
    4.4.2. Counter-arguments for ECT on Pure ACKs
    4.4.3. Summary: Enabling ECN on Pure ACKs
    4.4.4. Pure ACKs: DupACK Tests
  4.5. Window Probes
  4.6. FINs

4.7. RSTs
4.8. Retransmitted Packets.
4.9. General Fall-back for any Control Packet
5. Interaction with popular variants or derivatives of TCP
5.1. IW10
5.2. TFO
5.3. L4S
5.4. Other transport protocols
6. Security Considerations
7. IANA Considerations
8. References
8.1. Normative References
8.2. Informative References
Acknowledgments
Authors' Addresses

1. Introduction

RFC 3168 [RFC3168] specifies support of Explicit Congestion Notification (ECN) in IP (v4 and v6). By using the ECN capability, network elements (e.g. routers, 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 shall be called ECN++. It enables ECN support on all the aforementioned types of TCP packet. RFC 5562 (which was called ECN+) is obsoleted by the present specification, because it has the same goal of enabling ECT, but on only one type of control packet. The mechanisms proposed in this document have been defined conservatively and with safety in mind, possibly in some cases at the expense of performance.

ECN++ uses a sender-only deployment model. It works whether the two ends of the TCP connection use classic ECN feedback [RFC3168] or the updated scheme called Accurate ECN feedback (AccECN [I-D.ietf-tcpm-accurate-ecn]). {This is written assuming that AccECN will have been published as an RFC before ECN++, and that AccECN does indeed update RFC 3168, as intended at the time of writing. This note to be removed by the RFC Editor.}

Using ECN on initial SYN packets provides significant benefits, as we describe in the next subsection. However, only AccECN provides a way to feed back whether the SYN was CE marked, and RFC 3168 does not. Therefore, this spec recommends that implementers of ECN++ also implement AccECN. Conversely, if AccECN (or an equivalent safety mechanism) is not implemented with ECN++, this specification rules out ECN on the SYN.

ECN++ is designed for compatibility with a number of latency improvements to TCP such as TCP Fast Open (TFO [RFC7413]), initial window of 10 SMSS (IW10 [RFC6928]) and Low latency Low Loss Scalable Transport (L4S) [RFC9330], but they can all be implemented and deployed independently. [RFC8311] is a standards track procedural device that relaxes requirements in RFC 3168 and other standards track RFCs that would otherwise preclude the experimental modifications needed for ECN++ and other ECN experiments.

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, particularly TCP SYNs and SYN-ACKs, are important for performance, so dropping them is best avoided.

Not using ECN on control packets can be particularly detrimental to performance in environments where the ECN marking level is high. For example, [judd-nsdi] shows that in a controlled private data centre (DC) environment where ECN is used (in conjunction with DCTCP [RFC8257]), 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. The issue is that DCTCP exhibits a much more aggressive response to packet marking (which is why it is only applicable in controlled environments). This 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 [RFC9330]. In such an approach, L4S packets identify themselves using an ECN codepoint [RFC9331]. With L4S, 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 them into a different queue. Then only L4S data packets would be classified into the L4S queue that is expected to have lower latency, while the packets controlling and retransmitting these data packets would still get stuck behind the queue induced by non-L4S-enabled TCP traffic.

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.

* To conduct the specific experiments identified in the text by the strings "EXPERIMENTATION NEEDED" or "MEASUREMENTS NEEDED".

The data gathered through the experiments described in this document, particularly under the first 2 bullets above, will help in the redesign of the final mechanism (if needed) for adding ECN support to the different packet types considered in this 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 key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 [RFC9293], a window probe is a regular TCP segment, but with only one octet of new data that is 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.

TCP client: The initiating end of a TCP connection. Also called the initiator.

TCP server: The responding end of a TCP connection. Also called the responder or listener.

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 endpoints 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 has been redefined for use by L4S experiments [RFC8311] [RFC9331].

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

The experimental ECN++ changes to the specification of TCP over ECN [RFC3168] defined here primarily alter the behaviour of the sending host for each half-connection. However, there are subsections for forwarding elements and receivers below, which recommend that they accept the new packets - they should do already, but might not. This will prompt implementers to check the receive side code while they are altering the send-side code. All changes can be deployed at each endpoint independently of others and independent of any network behaviour.

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.

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 [RFC8311] updates RFC 3168 to remove this ambiguity. It requires 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 [RFC8311]).

3.2. Sender Behaviour

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	If AccECN, reduce IW
SYN-ACK	ECT	ECT	Reduce IW
Pure ACK	ECT	not-ECT	If AccECN, 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

W Probe and Re-XMT stand for Window Probe and Retransmission.
* For a SYN, "negotiated" means "requested".
** AccECN or equivalent safety (see Section 3.2.1.1.2)

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 we recommend against the sender setting ECT on the SYN if it is not requesting AccECN feedback. Therefore it is RECOMMENDED that the AccECN specification [I-D.ietf-tcpm-accurate-ecn] is implemented, along with the ECN++ experiment, 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 an ECN++ TCP sender is not using AccECN. This could be because it does not support AccECN or because its peer does not (AccECN can only be used if both ends of a connection support it).

Note that Table 1 does not imply any obligation to set any packet to ECT. ECN++ removes the restrictions that RFC 3168 places against setting ECT on these types of packets, and an implementation would normally be expected to take advantage of this, but it does not have to. Therefore, an implementation of the ECN++ experiment would be compliant if, for instance, it set ECT on some types of control packets but not others. If it did not set ECT on any control packets or retransmissions, it would not be compliant.

3.2.1. SYN (Send)

3.2.1.1. Setting ECT on the SYN

With classic [RFC3168] ECN feedback, the SYN was not 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 whether the SYN arrived marked CE. Therefore the setting of the IP/ECN field on the SYN is specified separately for each case in the following two subsections.

3.2.1.1.1. ECN++ TCP Client also Supports AccECN

For the ECN++ experiment, if the SYN is requesting AccECN feedback, the TCP sender will also set ECT on the SYN. Section 4.3 of [RFC8311] allows an ECN experiment to use ECT on a SYN, by updating Section 6.1.1 of RFC 3168 (which prohibited it).

3.2.1.1.2. ECN++ TCP Client does not Support AccECN

If the SYN sent by a TCP initiator does not attempt to negotiate Accurate ECN feedback, or does not use an equivalent safety mechanism, it MUST NOT set ECT on a SYN. This follows section 6.1.1 of [RFC3168], because, in this case, it would not be safe to use the extra flexibility to set ECT on a SYN that [RFC8311] allows.

The only envisaged examples of "equivalent safety mechanisms" are: a) some future TCP ECN feedback protocol, perhaps evolved from AccECN, that feeds back CE marking on a SYN; b) setting the initial window to 1 SMSS. IW=1 is NOT RECOMMENDED because it could degrade performance, but might be appropriate for certain lightweight TCP implementations.

See Section 4.2 for discussion and rationale.

If the TCP initiator does not set ECT on the SYN, the rest of Section 3.2.1 does not apply.

3.2.1.2. Caching where to use ECT on SYNs

This subsection only applies if the ECN++ TCP client sets ECT on the SYN and supports AccECN.

Until AccECN servers become widely deployed, a TCP initiator that sets ECT on a SYN (which typically implies the same SYN also requests AccECN, as above) SHOULD also maintain a cache entry per server to record servers that it is not worth sending an ECT SYN to, e.g. because they do not support AccECN and therefore have no logic for congestion markings on the SYN. Mobile hosts MAY maintain a cache entry per access network to record 'non-ECT SYN' entries against proxies (see Section 4.2.3). This cache can be implemented as part of the shared state across multiple TCP connections, if it is following [RFC9040].

Subsequently the initiator will not set ECT on a SYN to such a server or proxy, 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). If a server subsequently upgrades to support AccECN, the initiator will discover this as soon as it next connects, then it can remove the server from its cache and subsequently always set ECT for that server.

The client can limit the size of its cache of 'non-ECT SYN' servers. Then, while AccECN is not widely deployed, it will only cache the 'non-ECT SYN' servers that are most used and most recently used by the client. As the client accesses servers that have been expelled from its cache, it will simply use ECT on the SYN by default.

Servers that do not support ECN as a whole 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 three measurement studies and assumptions detailed in Section 4.2.3. However, Section 4.2.3 gives two other strategies and the choice between them depends on the implementer's goals (e.g., see Section 5.3 if using L4S) and the deployment prevalence of ECN variants in the network and on servers, not to mention the prevalence of some significant bugs.

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

3.2.1.3. SYN Congestion Response

As explained above, this subsection only applies if the ECN++ TCP client sets ECT on the initial SYN.

If the SYN-ACK returned to the TCP initiator confirms that the server supports AccECN, it will also be able to indicate whether or not the SYN was CE-marked. If the SYN was CE-marked, and if the initial window is greater than 1 MSS, then, the initiator MUST reduce its Initial Window (IW) and SHOULD reduce it to 1 SMSS (sender maximum segment size). The rationale is the same as that for the response to CE on a SYN-ACK (Section 4.3.2).

If the initiator has set ECT on the SYN and if the SYN-ACK shows that the server does not support feedback of a CE on the SYN (e.g. it does not support AccECN) and if the initial congestion window of the initiator is greater than 1 MSS, then 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 the 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 more than 3 segments is implemented (e.g. IW10 [RFC6928]), Section 5 gives additional recommendations.

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

As explained above, this subsection only applies if the ECN++ TCP client also sets ECT on the initial 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. Some evidence of this was found in a 2014 study [ecn-pam], but in a more recent study using 2017 data [Mandalari18] extensive measurements found no case where ECT on TCP control packets was treated any differently from ECT on TCP data packets. 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 loss might be the associated attempt to negotiate AccECN, or possibly other unrelated options on the SYN.

Therefore, if the timer expires after the TCP initiator has sent the first ECT SYN, it SHOULD make one more attempt to retransmit the SYN with ECT set (backing off the timer as usual). If the retransmission timer expires again, it SHOULD retransmit the SYN with the not-ECT codepoint in the IP header, to expedite connection set-up. 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 coordinate 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 a 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 setting ECT on the SYN, it will probably not need to give up on AccECN on the SYN. In any case, if a cache is used, it 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 (Send)

3.2.2.1. Setting ECT on the SYN-ACK

For the ECN++ experiment, the TCP implementation will set ECT on SYN-ACKs. Section 4.3 of [RFC8311] allows an ECN experiment to use ECT on a SYN-ACK, by updating Section 6.1.1 of RFC 3168 (which prohibited it).

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 (see Section 4.3.1). 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 4.3.2.

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 experimentation with an initial window of more than 3 segments (e.g. 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 retransmit one more SYN-ACK with ECT set (and back-off its timer as usual). If the timer expires again, it SHOULD retransmit the SYN-ACK with not-ECT in the IP header. If other experimental fields or options were on the initial SYN-ACK, it will also be necessary to follow their specifications for fall-back. It would make sense to co-ordinate all the strategies for fall-back in order to isolate the specific cause of the problem.

This fall-back strategy attempts to use ECT one more time than the strategy for ECT SYN-ACKs in [RFC5562] (which is made obsolete, being superseded by the present specification). Other fall-back strategies MAY be adopted if found to be more effective, e.g. fall-back to not-ECT on the first retransmission attempt.

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).

A client-based alternative to caching at the server is given in Section 4.3.3.

If either endpoint caches failed attempts, the cache SHOULD be arranged to expire so that the endpoint will infrequently attempt to check whether the problem has been resolved.

3.2.3. Pure ACK (Send)

A Pure ACK is an ACK packet that does not carry data, which includes the Pure ACK at the end of TCP's 3-way handshake.

For the ECN++ experiment, whether a TCP implementation sets ECT on a Pure ACK depends on whether or not Accurate ECN TCP feedback [I-D.ietf-tcpm-accurate-ecn] has been successfully negotiated for a particular TCP connection, as specified in the following two subsections.

3.2.3.1. Pure ACK without AccECN Feedback

If AccECN has not been successfully negotiated for a connection, ECT MUST NOT be set on Pure ACKs by either end.

3.2.3.2. Pure ACK with AccECN Feedback

For the ECN++ experiment, a host can only set ECT on outgoing Pure ACKs if it satisfies the following three conditions:

* it MUST have successfully negotiated AccECN feedback for the connection;

* it MUST have successfully negotiated SACK [RFC2018] for the connection;

* when it checks whether incoming pure ACKs are duplicates it MUST apply the additional test in Section 3.3.3.1.

If the host satisfies all these requirements, it can then set ECT on a pure ACK. Section 4.3 of [RFC8311] allows an ECN experiment to use ECT on a pure ACK, by updating Sections 5.2 and 6.1.4 of RFC 3168 (which prohibited it).

> MEASUREMENTS NEEDED: Measurements are needed to learn how the deployed base of network elements and RFC 3168 servers react to pure ACKs marked with the ECT(0)/ECT(1)/CE codepoints, i.e. whether they are dropped, codepoint cleared or processed and the congestion indication fed back on a subsequent packet.

See Section 3.3.3 for the implications if a host receives a CE-marked Pure ACK.

3.2.3.2.1. Pure ACK Congestion Response

This subsection only applies for a host that is setting ECT on outgoing pure ACKs, which is conditional on it satisfying the three conditions in Section 3.2.3.2.

A host that sets ECT on pure ACKs SHOULD respond to the congestion signal resulting from pure ACKs being marked with the CE codepoint. The specific response will need to be defined as an update to each congestion control specification. Possible responses to congestion feedback include reducing the congestion window (CWND) and/or regulating the pure ACK rate (see Section 4.4.2.1).

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.

3.2.4. Window Probe (Send)

For the ECN++ experiment, the TCP sender will set ECT on window probes. Section 4.3 of [RFC8311] allows an ECN experiment to use ECT on a window probe, by updating Section 6.1.6 of RFC 3168 (which prohibited it).

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 receiving 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. This complements cwnd validation [RFC7661], which reduces cwnd as more time elapses without having used available capacity. 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 (Send)

A TCP implementation can set ECT on a FIN.

See Section 3.3.4 for the implications if a host receives a CE-marked 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.

> 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 (Send)

A TCP implementation can set ECT on a RST.

See Section 3.3.5 for the implications if a host receives a CE-marked RST.

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.

Implementers SHOULD ensure that RST packets are always sent out with the same ECN field regardless of the TCP state machine. Otherwise the ECN field could reveal internal TCP state. For instance, the ECN field on a RST ought not to reveal any distinction between a non-listening port, a recently in-use port, and a closed session port.

3.2.7. Retransmissions (Send)

For the ECN++ experiment, the TCP sender will set ECT on retransmitted segments. Section 4.3 of [RFC8311] allows an ECN experiment to use ECT on a retransmission, by updating Sections 6.1.5 of RFC 3168 (which prohibited it).

See Section 3.3.6 for the implications if a host receives a CE-marked retransmission.

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.

3.2.8. General Fall-back for any Control Packet or Retransmission

Extensive measurements in fixed and mobile networks [Mandalari18] have found no evidence of blockages due to ECT being set on any type of TCP control packet.

In case traversal problems arise in future, fall-back measures have been specified above, but only for the cases regarding the initial packet of a half-connection (SYN or SYN-ACK) where ECT is persistently failing to get through.

Fall-back measures for blockage of ECT on other TCP control packets MAY be implemented. However they are not specified here given the lack of any evidence they will be needed. Section 4.9 justifies this advice in more detail.

3.3. Receiver Behaviour

The present ECN++ specification primarily concerns the behaviour for sending TCP control packets or retransmissions. Below are a few changes to the receive side of an implementation that are recommended while updating its send side. Nonetheless, where deployment is concerned, ECN++ is still a sender-only deployment, because it does not depend on receivers complying with any of these recommendations.

3.3.1. Receiver Behaviour for Any TCP Control Packet or Retransmission

RFC8311 is a standards track update to RFC 3168 in order to (amongst other things) "...allow the use of ECT codepoints on SYN packets, pure acknowledgement packets, window probe packets, and retransmissions of packets..., provided that the changes from RFC 3168 are documented in an Experimental RFC in the IETF document stream."

Section 4.3 of RFC 8311 amends every statement in RFC 3168 that precludes the use of ECT on control packets and retransmissions to add "unless otherwise specified by an Experimental RFC in the IETF document stream". The present specification is such an Experimental RFC. Therefore, In order for the present RFC 8311 experiment to be useful, TCP receivers will need to satisfy the following requirements:

* Any TCP implementation SHOULD accept receipt of any valid TCP control packet or retransmission irrespective of its IP/ECN field. If any existing implementation does not, it SHOULD be updated to do so.

* A TCP implementation taking part in the experiments proposed here MUST accept receipt of any valid TCP control packet or retransmission irrespective of its IP/ECN field.

The following sections give further requirements specific to each type of control packet.

These measures are derived from the robustness principle of "... be liberal in what you accept from others", not only to ensure compatibility with the present experimental specification, but also any future protocol changes that allow ECT on any TCP packet.

3.3.2. SYN (Receive)

RFC 3168 negotiates the use of ECN for the connection end-to-end using the ECN flags in the TCP header. RFC 3168 originally said that "A host MUST NOT set ECT on SYN ... packets." but it was silent as to what a TCP server ought to do if it receives a SYN packet with a non-zero IP/ECN field anyway.

For the avoidance of doubt, the normative statements for all TCP control packets in Section 3.3.1 are interpreted for the specific case when a SYN is received as follows:

* Any TCP server implementation SHOULD accept receipt of a valid SYN that requests ECN support for the connection, irrespective of the IP/ECN field of the SYN. If any existing implementation does not, it SHOULD be updated to do so.

* A TCP implementation taking part in the ECN++ experiment MUST accept receipt of a valid SYN, irrespective of its IP/ECN field.

* If the SYN is CE-marked and the server has no logic to feed back a CE mark on a SYN-ACK (e.g. it does not support AccECN), it has to ignore the CE-mark (the client detects this case and behaves conservatively in mitigation - see Section 3.2.1.3).

Rationale: At the time of the writing, some implementations of TCP servers (see Section 4.2.2.2) assume that, if a host receives a SYN with a non-zero IP/ECN field, it must be due to network mangling, and they disable ECN for the rest of the connection. Section 4.2.2.2 cites a measurement study run in 2017 that found no occurrence of this type of network mangling. However, a year earlier, when ECN was enabled on connections from Apple clients, there was a case of a whole network that re-marked the ECN field of every packet to CE (it was rapidly fixed).

When ECN was not allowed on SYNs, it made sense to look for a non-zero ECN field on the SYN to detect this type of network mangling. But now that ECN is being allowed on a SYN, detection needs to be more nuanced. A server needs to disable the test on the SYN alone for AccECN SYNs (which was done for Linux RFC 3168 servers in 2019 [relax-strict-ecn]) and for RFC 3168 SYNs it needs to watch for three or four packets all set to CE at the start of a flow. If such mangling is indeed now so rare, it would also be preferable to log each case detected and manually report it to the responsible network, so that the problem will eventually be eliminated.

3.3.3. Pure ACK (Receive)

For the avoidance of doubt, the normative statements for all TCP control packets in Section 3.3.1 are interpreted for the specific case when a Pure ACK is received as follows:

* Any TCP implementation SHOULD accept receipt of a pure ACK with a non-zero IP-ECN field, as now allowed by [RFC8311], which updated section 6.1.4 of [RFC3168] (which previously required the ECN field to be cleared to zero on all pure ACKs).

* A TCP implementation taking part in the ECN++ experiment MUST accept receipt of a pure ACK with a non-zero ECN field.

The question of whether and how the receiver of pure ACKs is required to feed back any CE marks on them is outside the scope of the present specification because it is a matter for the relevant feedback specification ([RFC3168] or [I-D.ietf-tcpm-accurate-ecn]). AccECN feedback mandates counting of CE marks on any control packets including pure ACKs. Whereas RFC 3168 is silent on this point, so feedback of CE-markings might be implementation specific (see Section 4.4.2.1).

3.3.3.1. Additional DupACK Check

A TCP implementation that is sending ECN-capable pure ACKs MUST add the following check to all its algorithms that detect duplicate ACKs (defined in Section 2 of [RFC5681]):

* If there is no SACK option on an incoming pure ACK (ECN-capable or not) despite SACK having been negotiated, it is not counted as a duplicate ACK.

SACK blocks are known to be stripped on some paths through the public Internet, or some implementations are known to negotiate SACK but never send SACK blocks. Therefore, if a TCP implementation is sending ECN-capable pure ACKs, it SHOULD check for these pathologies (unless it is intended solely for an environment known to be clear of them), as follows:

* If an incoming pure ACK appears to be a duplicate, but:

  - it does not carry any SACK blocks (despite SACK having been negotiated)

  - and AccECN [I-D.ietf-tcpm-accurate-ecn] has been negotiated but the ACE field has increased by less than 3 (after allowing for wrap) then the implementation can deem that SACK blocks are being stripped. In this case, for the rest of the connection, it SHOULD:

- either disable the sending of ECN-capable pure ACKs;

- or replace any instance of the above additional DupACK check with heuristics such as the following examples:

  - If the ACE field [I-D.ietf-tcpm-accurate-ecn] has incremented by at least 3, an incoming ACK is not counted as a duplicate ACK;

  - if the Timestamp option (TSopt) is available [RFC7323], and the echoed timestamp (TSecr) increments relative to the previous ACK, an incoming ACK is not counted as a Duplicate ACK.

An additional DupACK check is one of the mandatory conditions specified in Section 3.2.3.2 for sending ECN-capable pure ACKs (the others are AccECN mode and SACK-negotiated mode).

See Section 4.4.4 for rationale.

3.3.4. FIN (Receive)

The normative statements for all TCP control packets in Section 3.3.1 apply for the specific case when a FIN is received, with 'valid' defined as follows:

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.

3.3.5. RST (Receive)

The normative statements for all TCP control packets in Section 3.3.1 apply for the specific case when a RST is received, with 'valid' defined as follows:

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

3.3.6. Retransmissions (Receive)

The normative statements for all TCP control packets in Section 3.3.1 apply for the specific case when a retransmission is received, with 'valid' defined as follows:

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 reducing its congestion window.

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 (FINs and RSTs). 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 where TCP does not detect the loss of certain 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.

It will help to first compare with the case of a reliably delivered packet (e.g. a SYN or data packet) that is made ECN-capable. If it is CE-marked at two buffers in succession, it is not discarded by the first buffer so it goes on to help congest the second. But it delivers only one congestion signal. Similarly, if instead it is marked at the first buffer and dropped at the second, it still helps congest the second buffer, but it still delivers only one congestion signal (the loss).

Some non-ECN TCP control packets (e.g. pure ACKs or FINs) certainly do not reliably deliver a congestion signal if they are discarded. But, making such control packets ECN-capable upgrades their ability to deliver a congestion signal from a buffer with ECN support. However, as before, they still cannot reliably deliver a loss signal from a non-ECN buffer. This includes the case where one congested buffer CE-marks such a packet, then a second congested buffer without ECN support discards it.

Thus ECN is always more and never less reliable for delivery of congestion notification.

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 a CE marking on a SYN and, even if logic were added, there was no field in the SYN-ACK to feed it back. 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.

The accurate ECN (AccECN) protocol [I-D.ietf-tcpm-accurate-ecn] has since been designed to solve this problem. Two features are important here:

1. An AccECN server uses the 3 AccECN flags in the TCP header of the SYN-ACK to respond to the client. 4 of the possible 8 codepoints provide enough space for the server to feed back which of the 4 IP/ECN codepoints was on the incoming SYN (including CE of course).

2. If any of these 4 codepoints are in the SYN-ACK, it confirms that the server supports AccECN and, if another codepoint is returned, it confirms that the server does not support AccECN.

This still does not seem to allow a client to set ECT on a SYN, it only finds out whether the server would have supported it afterwards. The trick the client uses for ECN++ is to set ECT on the SYN optimistically then, if the SYN-ACK reveals that the server wouldn't have understood CE on the SYN, the client responds conservatively as if the SYN was marked with CE.

The recommended conservative congestion response is to reduce the initial window, which does not affect the performance of very popular protocols such as HTTP, since it is currently extremely rare for an HTTP client to send more than one packet as its initial request anyway (for data on HTTP/1 & HTTP/2 request sizes see Fig 3 in [Manzoor17]). Any clients that do frequently use a larger initial window for their first message to the server can cache which servers will not understand ECT on a SYN (see Section 4.2.3 below). If caching is not practical, such clients could reduce the initial window to say IW2 or IW3.

> EXPERIMENTATION NEEDED: Experiments will be needed to determine any better strategy for reducing IW in response to congestion on a SYN, when the server does not support congestion feedback on the SYN-ACK (whether cached or discovered explicitly).

4.2.2. Argument 1b: ECT Considered Invalid on the SYN

Given that prior to [RFC8311] ECT-marked SYN packets were prohibited, it cannot be assumed they will be accepted, by TCP middleboxes or servers.

4.2.2.1. ECT on SYN Considered Invalid by Middleboxes

According to a study using 2014 data [ecn-pam] from a limited range of fixed vantage points, for the top 1M Alexa web sites, adding the ECN capability to SYNs increased connection establishment failures by about 0.4%.

From a wider range of fixed and mobile vantage points, a more recent study in Jan-May 2017 [Mandalari18] found no occurrences of blocking of ECT on SYNs. However, in more than half the mobile networks tested it found wiping of the ECN codepoint at the first hop.

MEASUREMENTS NEEDED: As wiping at the first hop is remedied, measurements will be needed to check whether SYNs with ECT are sometimes blocked deeper into the path.

Silent failures introduce a retransmission timeout delay (default 1 second) at the initiator before it attempts any fall back strategy (whereas explicit RSTs can be dealt with immediately). Ironically, making SYNs ECN-capable is intended to avoid the timeout when a SYN is lost due to congestion. Fortunately, if there is any discard of ECN-capable SYNs due to policy, it will occur predictably, not randomly like congestion. So the initiator should be able to avoid it by caching paths or servers that do not support ECN-capable SYNs (see the last paragraph of Section 3.2.1.2).

4.2.2.2. ECT on SYN Considered Invalid by Servers

A study conducted in Nov 2017 [Kuehlewind18] found that, of the 82% of the Alexa top 50k web servers that supported ECN, 84% disabled ECN if the IP/ECN field on the SYN was ECT0, CE or either. Given most web servers use Linux, this behaviour can most likely be traced to a patch contributed in May 2012 that was first distributed in v3.5 of the Linux kernel [strict-ecn]. The comment says "RFC3168 : 6.1.1 SYN packets must not have ECT/ECN bits set. If we receive a SYN packet with these bits set, it means a network is playing bad games with TOS bits. In order to avoid possible false congestion notifications, we disable TCP ECN negociation." Of course, some of the 84% might be due to similar code in other OSs.

For brevity we shall call this the "over-strict" ECN test, because it is over-conservative with what it accepts, contrary to Postel's robustness principle. A robust protocol will not usually assume network mangling without comparing with the value originally sent, and one packet is not sufficient to make an assumption with such irreversible consequences anyway.

Ironically, networks rarely seem to alter the IP/ECN field on a SYN from zero to non-zero anyway. In a study conducted in Jan-May 2017 over millions of paths from vantage points in a few dozen mobile and fixed networks [Mandalari18], no such transition was observed. With such a small or non-existent incidence of this sort of network mangling, it would be preferable to report any residual problem paths so that they can be fixed.

Whatever, the widespread presence of this 'over-strict' test proves that RFC 5562 was correct to expect that ECT would be considered invalid on SYNs. Nonetheless, it is not an insurmountable problem - the over-strict test in Linux was patched in Apr 2019 [relax-strict-ecn] and caching can work round it where previous versions of Linux are running. The prevalence of these "over-strict" ECN servers makes it challenging to cache them all. However, Section 4.2.3 below explains how a cache of limited size can alleviate this problem for a client's most popular sites.

For the future, [RFC8311] updates RFC 3168 to clarify that the IP/ECN field does not have to be zero on a SYN if documented in an experimental RFC such as the present ECN++ specification.

4.2.3. Caching Strategies for ECT on SYNs

Given the server handling of ECN on SYNs outlined in Section 4.2.2.2 above, an initiator might combine AccECN with three candidate strategies for setting ECT on a SYN and caching the outcome:

(S1): Pessimistic ECT and cache successes: The initiator always requests AccECN, but by default without ECT on the SYN. Then it caches those servers that confirm that they support AccECN as 'ECT SYN OK'. On a subsequent connection to any server that supports AccECN, the initiator can then set ECT on the SYN. When connecting to other servers (non-ECN or classic ECN) it will not set ECT on the SYN, so it will not fail the 'over-strict' ECN test.

Longer term, as servers upgrade to AccECN, the initiator is still requesting AccECN, so it will add them to the cache and use ECT on subsequent SYNs to those servers. However, assuming it has to cap the size of the cache, the client will not have the benefit of ECT SYNs to those less frequently used AccECN servers expelled from its cache.

(S2): Optimistic ECT: The initiator always requests AccECN and by default sets ECT on the SYN. 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. Two caching sub-strategies are feasible:

a. No cache.

b. Cache failures: The optimistic ECT strategy can be improved by caching solely those servers that do not support AccECN as 'ECT SYN NOK'. This would include non-ECN servers and all Classic ECN servers whether 'over-strict' or not. On subsequent connections to these non-AccECN servers, the initiator will still request AccECN but not set ECT on the SYN. Then, the connection can still fall back to Classic ECN, if the server supports it, and 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.

Where an access network operator mediates Internet access via a proxy that does not support AccECN, the optimistic ECT strategy will always fail. This scenario is more likely in mobile networks. Therefore, a mobile host could cache lack of AccECN support per attached access network operator. Whenever it attached to a new operator, it could check a well-known AccECN test server and, if it found no AccECN support, it would add a cache entry for the attached operator. It would only use ECT when neither network nor server were cached. It would only populate its per server cache when not attached to a non-AccECN proxy.

(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), (S2A) and (S2B). The normative specification for ECT on a SYN in Section 3.2.1 recommends the "optimistic ECT and cache failures" strategy (S2B) but the choice depends on the implementer's motivation for using ECN++, and the deployment prevalence of different technologies and bug-fixes.

* 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. If AccECN becomes widely deployed on servers, SYNs to those AccECN servers that are less frequently used by the client and therefore don't fit in the cache will not benefit from ECN protection at all.

* The "optimistic ECT without a cache" strategy (S2A) is the simplest. It would satisfy the goal of an implementer who is solely interested in low latency using AccECN and ECN++ and is not concerned about fall-back to Classic ECN.

* The "optimistic ECT and cache failures" strategy (S2B) exploits ECT on SYNs from the very first attempt. But if the server turns out to be 'over-strict' it will disable ECN for the connection, but only for the first connection if it's one of the client's more popular servers that fits in the cache. If the server turns out not to support AccECN, the initiator has to conservatively limit its initial window, but again only for the first connection if it's one of the client's more popular servers (and anyway this rarely makes any difference when most client requests fit in a single packet).

Note that, if AccECN deployment grows, storage for 'caching successes' (S1) starts off small then grows, while with 'caching failures' (S2B) it is large at first, then shrinks. At half-way, the size of the cache has to be capped with either approach, so the default behaviour for all the servers that do not fit in the cache is as important as the behaviour for the popular servers that do fit.

> MEASUREMENTS NEEDED: Measurements are needed to determine which strategy would be sufficient for any particular client, whether a particular client would need different strategies in different circumstances and how many occurrences of problems would be masked by how few cache entries.

Another strategy would be to send a not-ECT SYN a short delay (below the typical lowest RTT) after an ECT SYN and only accept the non-ECT connection if it returned first. This would reduce the performance penalty for those deploying ECT SYN support. However, this 'happy eyeballs' approach becomes complex when multiple optional features are all tried on the first SYN (or on multiple SYNs), so it is not recommended.

4.2.4. 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.

[ecn-overload] showed that ECT can only slightly augment flooding attacks relative to a non-ECT attack. It was hard to overload the link without causing the queue to grow, which in turn caused the AQM to disable ECN and switch to drop, thus negating any advantage of using ECT. This was true even with the switch-over point set to 25% drop probability (i.e. the arrival rate was 133% of the link rate).

4.3. SYN-ACKs

The proposed approach in Section 3.2.2 for experimenting with ECN-capable SYN-ACKs is effectively 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.

4.3.1. Possibility of Unrecognized CE on the 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 client 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. 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. 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).

4.3.2. Response to Congestion on a SYN-ACK

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++ experimental specification obsoletes RFC 5562 because it 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, the more data packets that the client sends in its IW, the more likely at least one will be CE marked, leading it to exit slow-start early. 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.

EXPERIMENTATION NEEDED: Experiments will be needed to check the above reasoning and determine any better strategy for reducing IW in response to congestion on a SYN-ACK (or a SYN).

4.3.3. Fall-Back if ECT SYN-ACK Fails

Section 3.2.2.3 describes how a server could cache failed connection attempts. As an alternative, the server could rely on the client to cache failed attempts (on the basis that the client would cache a failure whether ECT was blocked on the SYN or the SYN-ACK). This strategy cannot be used if the SYN does not request AccECN support.

It works as follows. If a server would rather not maintain its own cache, when it receives a SYN that requests AccECN support but is set to not-ECT, the server replies with a SYN-ACK also set to not-ECT. This gives the client the power to disable ECT on both the SYN and SYN-ACK, if it has cached knowedge that there were previous problems.

If a middlebox only blocks ECT on SYNs, not SYN-ACKs, this strategy might disable ECN on a SYN-ACK when it did not need to, but at least it saves the server from maintaining a cache. However, a client cannot rely on all non-caching servers suppressing ECT on SYN/ACKs when they might not need to.

Therefore, a more practical way for a client to cache failures on behalf of the server would be for the client to initially fall-back to Not-ECT on the SYN after multiple timeouts, then if that doesn't resolve the problem, the client could disable ECN completely at the TCP layer, and if the connection then works, it could cache to disable ECN in future. With this approach, it is still preferable but optional for the server to also cache failure to deliver ECN-capable SYN-ACKs.

4.4. Pure ACKs

4.4.1. Arguments against ECT on Pure ACKS

After the general reliability argument already quoted in Section 4.1, Section 5.2 of RFC 3168 goes on to use ECT marking of pure ACKs as a specific example of the reliability argument:

"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."

4.4.2. Counter-arguments for ECT on Pure ACKs

The first argument above 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 on Pure ACKs unless there is a mechanism to respond to it. Although the above passage from RFC 3168 envisages the possibility of ECN on pure ACKs in the future, it is silent on how its ECN feedback mechanisms would be used if CE markings did arrive on pure ACKs. In contrast, the position of AccECN with respect to the three parts of a congestion response mechanism is as follows:

Detection: The AccECN spec requires the receiver of any TCP packets including pure ACKs to count any CE marks on them (whether or not it sends ECN-capable control packets itself).

Feedback: The AccECN Data Receiver continually feeds back a count of the number of CE-marked packets (including pure ACKs) that it has received.

Even if the receiver of a CE-mark on a pure ACK does not feed it back immediately, it still includes it within subsequent feedback, for instance when it later sends a data segment. Even if an AccECN host has no data outstanding, it is still required to send an 'increment-triggered' pure ACK after every 'n' CE marks it receives, where 'n' is at least 3.

Congestion response: Once an AccECN Data Sender receives feedback about CE-markings on pure ACKs, it will be able to reduce the congestion window (cwnd) and/or the ACK rate. The specific response to congestion feedback is out of scope of both AccECN and the present spec, because it would be defined in a base TCP congestion control spec [RFC5681] [RFC9438] or a variant. Nonetheless, in order to decide whether the present ECN++ experimental spec should allow a host to set ECT on pure ACKs, we only need to know whether a congestion response would be feasible - we do not have to standardize it. Possible responses are discussed in the following subsections,

4.4.2.1. Congestion Window Response to CE-Marked Pure ACKs

This subsection explores issues that congestion control designers will need to consider when defining a cwnd response to CE-marked Pure ACKs.

A CE-mark on a Pure ACK does not mean that only Pure ACKs are causing congestion. It only means that the marked Pure ACK is part of an aggregate that is collectively causing a bottleneck queue to randomly CE-mark a fraction of the packets. A CE-mark on a Pure ACK might be due to data packets in other flows through the same bottleneck, due to data packets interspersed between Pure ACKs in the same half-connection, or just due to the rate of Pure ACKs alone. (RFC 3168 only considered the last possibility, which led to the argument that standardization of ECN-enabled Pure ACKs had to be deferred, because ACK congestion control was a research issue.)

If a host has been sending a mix of Pure ACKs and data, it doesn't need to work out whether a particular CE mark was on a Pure ACK or not; it just needs to respond to congestion feedback as a whole by reducing its congestion window (cwnd), which limits the data it can launch into flight through the congested bottleneck. If a host is solely receiving data and sending only Pure ACKs, reducing cwnd will have no immediate effect (the next subsection addresses that). Nonetheless, reducing cwnd at one moment would limit its rate if it was given something to send at a later moment.

When a host is sending data as well as Pure ACKs, it would not be right for CE-marks on Pure ACKs and on data packets to induce the same reduction in cwnd. A possible way to address this issue would be to weight the response by the size of the marked packets (assuming the congestion control supports a weighted response, e.g. [RFC8257]). For instance, one could calculate the fraction of

CE-marked bytes (headers and data) over each round trip (say) as follows:

(CE-marked header bytes + CE-marked data bytes) / (all header bytes + all data bytes)

Even if the exact header size is not known, header bytes could be calculated by multiplying a packet count by a nominal header size, which is possible with AccECN feedback, because it gives a count of CE-marked packets (as well as CE-marked bytes). The above simple aggregate calculation caters for the full range of scenarios; from all Pure ACKs to just a few interspersed with data packets.

Note that any mechanism that reduces cwnd due to CE-marked Pure ACKs would need to be integrated with the congestion window validation mechanism [RFC7661], which already conservatively reduces cwnd over time because cwnd becomes stale if it is not used to fill the pipe.

4.4.2.2. ACK Rate Response to CE-Marked Pure ACKs

Reducing the congestion window will have little effect if the bottleneck is congested mostly by unresponsive pure ACKs. This could leave little or no capacity for data transfers that would be responsive to the congestion.

Since RFC 3168 was published, experimental Acknowledgement Congestion Control (AckCC) techniques have been documented in [RFC5690] (informational), which describes how two new TCP options could allow any pair of TCP endpoints to regulate the delayed ACK ratio in response to lost or CE-marked pure ACKs. However, this spec did not ask IANA to actually allocate any option numbers, because the intention was to describe the scheme and document the unresolved complications.

AckCC addressed three main problems, namely that TCP had: i) no mechanism to feed back loss or CE-marking of pure ACKs; ii) consequently, no mechanism to allow ECT to be set on pure ACKs; and iii) no mechanism to regulate the ACK rate. A combination of AccECN and the present specification addresses the first two problems, at least for ECN marking. So, with the addition of an ACK rate mechanism, it might now be possible to design an ECN-specific ACK congestion control scheme along similiar lines to RFC 5690. However, such a mechanism is out of scope of the present document.

Setting aside the unfinished nature 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, it provides a feasible congestion response mechanism for Pure ACKs, so the arguments for ECT on pure ACKs outweigh those against. ECN is always more and never less reliable for delivery of congestion notification. A cwnd reduction needs to be considered by congestion control designers as a response to congestion on pure ACKs. Separately, AckCC (or 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. If it is not, packet discard will still act as the "congestion response of last resort" by thinning out the ACK traffic. 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.

In the case when Classic ECN has been negotiated, the argument for ECT on pure ACKs is less clear-cut. Some of the installed base of RFC 3168 implementations might happen to (unintentionally) provide a feedback mechanism to support a cwnd response. For those that did not, setting ECT on pure ACKs would be better for the flow's own performance than not setting it. However, where there was no feedback mechanism, setting ECT could do slightly more harm than not setting it. AckCC could provide a complementary response mechanism, because it is designed to work with RFC 3168 ECN, but it is incomplete. In summary, a congestion response mechanism for Pure ACKs is unlikely to be feasible with the installed base of classic ECN.

Section 3.2.3 of this specification uses a safe approach where it allows ECT on Pure ACKs if AccECN feedback has been negotiated, but not with classic RFC 3168 ECN feedback. Allowing hosts to set ECT on Pure ACKs without a feasible response mechanism could result in risk. It would certainly improve the flow's own performance, but it would slightly increase potential harm to others. Morevoer, if would set an undesirable precedent for setting ECT on packets with no mechanism to respond to any resulting congestion signals.

4.4.4. Pure ACKs: DupACK Tests

This section justifies the requirement for a host to use the additional test for incoming duplicate pure ACKs in Section 3.3.3.1, which is one of the mandatory conditions for a host to set ECT on its outgoing pure ACKs. See Section 3.2.3.2 for all three conditions, the other two being to have successfully negotiated SACK and AccECN feedback.

The AccECN spec [I-D.ietf-tcpm-accurate-ecn] mandates the 'increment-triggered ACK' rule where, "an AccECN Data Receiver MUST emit an ACK if 'n' CE marks have arrived since the previous ACK." The value of 'n' depends on whether there is newly delivered data to acknowledge. If there is not, "'n' MUST be no less than 3".

Use of ECN-capable pure ACKs by the ECN++ experiment combined with congestion at an ECN AQM at the bottleneck of the ACK path can cause AccECN to acknowledge ACKs that carry new CE information, by the above rule. This leads to repetition of the ACK of a segment, which is an exception to the requirement in the last paragraph of Sec 4.2 of [RFC5681].

This exception is justified because, although there is no new data for the receiver to acknowledge, there is new IP-ECN information to feed back. When such new IP-ECN information arrives on an ACK, it is not possible to feed it back without also repeating the acknowledgement of the latest data segment (if ECN++ had been part of TCP from the start, a logical approach would have been to clear the ACK flag, but nowadays such an unorthodox segment would be rejected). Therefore, to distinguish these ACKs of ACKs from genuine duplicate ACKs, it was necessary to introduce the test for the absence of SACK.

To justify adding this test, we use the same scenario as in Section 3.2.2.5.1 of the AccECN spec [I-D.ietf-tcpm-accurate-ecn] with volleys of unidirectional data initially from host A to B then from B to A. In response to the first volley, B emits ECN-capable pure ACKs as feedback (as per the present ECN++ spec). If the ACK stream from B experiences congestion at an ECN-enabled buffer, some of these ACKs will be CE-marked once they arrive at A.

Host A would normally inform B about the congested ACK path by piggybacking AccECN feedback on the data packets from A to B. However, once A stops sending data, it still needs to inform B about any CE-marked ACKs continuing to arrive from B in the subsequent round trip (so that B has up to date congestion information, in case it starts sending data). AccECN's 'increment-triggered ACK' rule ensures that A emits a pure ACK at least every third incoming CE mark. However, as each 'increment-triggered ACK' from A arrives at B, it will not only feed back CE markings, but it will also repeatedly acknowledge whatever the last sequence number was from B.

Then, if SACK had not been negotiated for the connection, and if B started sending its volley of new data packets, the ACKs from A could imply to B that its first new data packet had been lost and A was then emitting duplicate ACKs triggered by the rest of B's volley arriving. Then, as well as detecting CE marking, B could falsely detect loss, leading to spurious retransmission and potentially incorrect congestion response. Similarly, false detection of duplicate ACKs could confuse other algorithms, such as Limited Transmit, Fast Recovery, PRR, etc.

This is why Section 3.3.3.1 requires B to negotiate SACK and to use lack of SACK options as an additional check for a duplicate ACK.

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, given the RST is deemed invalid, any CE marking on it could also be invalid, so 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 unnecessary retransmissions. 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 ECT or 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.

Finally, the third argument is about unnecessary retransmissions. For those retransmitted packets that arrive at the receiver after the original packet has been properly received (so-called spurious retransmissions), RFC 3168 raises the concern that any CE marking will be ignored, because any spurious retransmission is out of window and CE markings on out of window packets will be ignored (by the above rule). In mitigation against this argument, the fact that the original packet has been delivered implies that the sender's original congestion response (when it deemed the packet to be lost and retransmitted it) was unnecessary. However, omitting a congestion response to the CE one round trip later does not strictly compensate for the previous unnecessary response, because the response should be in the same round as the congestion occurs. Nonetheless, there is a stronger argument against the concern in RFC 3168: TCP does not detect the loss of a spurious retransmission, and therefore does not respond to this congestion loss. So not responding to CE on ECN-capable supurious retransmissions is no worse than TCP's existing lack of response to loss of spurious retransmissions.

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

4.9. General Fall-back for any Control Packet

Extensive experiments have found no evidence of any traversal problems with ECT on any TCP control packet [Mandalari18]. Nonetheless, Sections 3.2.1.4 and 3.2.2.3 specify fall-back measures if ECT on the first packet of either half-connection (SYN or SYN-ACK) appears to be blocking progress. Here, the question of fall-back measures for ECT on other control packets is explored. It supports the advice given in Section 3.2.8, paraphrased here as, "Until there's evidence that something's broken, don't fix it."

If an implementation has had to disable ECT to ensure the first packet of a flow (SYN or SYN-ACK) gets through, the question arises whether it ought to disable ECT on all subsequent control packets within the same TCP connection. Without evidence of any such problems, this seems unnecessarily cautious. Particularly given it would be hard to detect loss of most other types of TCP control packets that are not ACK'd. And particularly given that unnecessarily removing ECT from other control packets could lead to performance problems, e.g. by directing them into another queue [RFC9331] or over a different path, because some broken multipath equipment (erroneously) routes based on all 8 bits of the ex-Traffic Class octet (IPv6) or the ex-ToS octet (IPv4).

In the case where a connection starts without ECT on the SYN (perhaps because problems with previous connections had been cached), there will have been no test for ECT traversal in the client-server direction until the pure ACK that completes the handshake. It is possible that some middlebox might block ECT on this pure ACK or on later retransmissions of lost packets. Similarly, after a route change, the new path might include some middlebox that blocks ECT on some or all TCP control packets.

However, without evidence of such problems, the complexity of a fix does not seem worthwhile.

MORE MEASUREMENTS NEEDED (?): If further two-ended measurements do find evidence for these traversal problems, measurements would be needed to check for correlation of ECT traversal problems between different control packets. It might then be necessary to introduce a catch-all fall-back rule that disables ECT on certain subsequent TCP control packets based on some criteria developed from these measurements.

5. Interaction with popular variants or derivatives of TCP

The designs of the following TCP variants have been assessed and found not to interact adversely with ECT on TCP control packets: SYN cookies (see Appendix A of [RFC4987] and section 3.1 of [RFC5562]), Initial Window of ten (IW10 [RFC6928], TCP Fast Open (TFO [RFC7413]), DCTCP [RFC8257] and L4S [RFC9330].

The following subsections assess the interaction between setting ECT on all packets and each of these variants (except SYN cookies where no detail is necessary). A final subsection briefly notes the possibility that the principles applied here should translate to protocols derived from TCP.

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.

5.1. 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 will typically 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, preferably 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, a 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, at least for web sessions, it is extremely rare for a TCP initiator (client) to have more than one data segment to send at the start of a TCP connection (see Fig 3 in [Manzoor17]) - IW10 is primarily exploited by TCP servers.

If a responder receives feedback that the SYN-ACK was CE-marked, Section 3.2.2.2 recommends that it reduces its initial window, preferably to 1 SMSS. When the responder also implements IW10, it might again seem rather over-conservative to reduce IW from 10 to 1. But in this case the rationale is somewhat different:

* The IW10 spec [RFC6928] recommends not reducing cwnd to 1 segment on the grounds that it is uncertain whether absence of feedback implies loss. However, in contrast, explicit feedback that the SYN-ACK was CE-marked is a positive indication that the queue has been building.

* Given it is now likely that a queue already exists, the more data packets that the server sends in its IW, the more likely at least one will be CE marked, leading it to exit slow-start early.

Experimentation will be needed to determine the best strategy.

It should be noted that experience from recent congestion avoidance experiments where the window is reduced by less than half in response to ECN marking [RFC8511] is not necessarily applicable to a flow start scenario.

5.2. 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 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].

5.3. L4S

A Low Latency, Low Loss, and Scalable throughput (L4S) variant of TCP such as TCP Prague [I-D.briscoe-iccrg-prague-congestion-control] requires AccECN feedback and uses ECN++. Also the spec of the L4S-ECN protocol [RFC9331] mentions ECN++ as a useful performance optimization.

Therefore, the L4S experiment and the present ECN++ experiment can be combined without altering any of the specifications. The only difference would be in the recommendation of the best SYN cache strategy.

The normative specification for ECT on a SYN in Section 3.2.1.2 recommends the "optimistic ECT and cache failures" strategy (S2B defined in Section 4.2.3) for the general Internet. However, if a user's Internet access bottleneck supported L4S ECN but not Classic ECN, the "optimistic ECT without a cache" strategy (S2A) would make most sense, because there would be little point trying to avoid the 'over-strict' test and negotiate Classic ECN, if L4S ECN but not Classic ECN was available on that user's access link (as is the case with Low Latency DOCSIS [DOCSIS3.1]).

Strategy (S2A) is the simplest, because it requires no cache. It would satisfy the goal of an implementer who is solely interested in ultra-low latency using AccECN and ECN++ (e.g. accessing L4S servers) and is not concerned about fall-back to Classic ECN (e.g. when accessing other servers).

5.4. Other transport protocols

Experience from experiments on adding ECN support to all TCP packets ought to be directly transferable between TCP and derivatives of TCP, like SCTP.

Stream Control Transmission Protocol (SCTP) [RFC9260] is a standards track transport protocol derived from TCP. SCTP currently does not include ECN support, but Appendix A of an obsoleted earlier version of the spec [RFC4960] 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]. The question of whether SCTP ought to set ECT on retransmissions and control packets is work in progress.

QUIC [RFC9000] is another standards track transport protocol offering similar services to TCP but intended to exploit some of the benefits of running over UDP. Building on the arguments in the current draft, a QUIC sender sets ECT on all packets unless it fails the test for path support.

6. Security Considerations

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 countered in Section 4, particularly Section 4.2.4 and Section 4.8 on DoS attacks using spoofed ECT-marked SYNs and spoofed CE-marked retransmissions. In both cases, RFC 3168 attempted to legislate against the sender setting ECT, which degrades the performance of genuine senders, but will not be heeded by attackers. Instead, the approach adopted is to specify reinforced defences against such attacks that already reside in the network and at the receiver.

Section 3.2.6 on sending TCP RSTs considers the question of whether ECT on RSTs will allow RST attacks to be intensified. It also points out that implementers need to take care to ensure that the ECN field on a RST does not depend on TCP's state machine. Otherwise the internal information revealed could be of use to potential attackers. This point actually applies more generally to all control packets (unless it has become necessary to disable ECN to evade path traversal problems).

If each TCP implementation chooses to make different control packets ECN-capable, it could contribute to easier fingerprinting of TCP stacks.

7. IANA Considerations

This section is to be removed before publishing as an RFC.

There are no IANA considerations in this memo.

8. References

8.1. Normative References

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, DOI 10.17487/RFC2018, October 1996, <https://www.rfc-editor.org/info/rfc2018>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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, <https://www.rfc-editor.org/info/rfc3168>.

[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, <https://www.rfc-editor.org/info/rfc5961>.

[I-D.ietf-tcpm-accurate-ecn] Briscoe, B., Kühlewind, M., and R. Scheffenegger, "More Accurate Explicit Congestion Notification (ECN) Feedback in TCP", Work in Progress, Internet-Draft, draft-ietf-tcpm-accurate-ecn-28, 17 November 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tcpm-accurate-ecn-28>.

[RFC8311] Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <https://www.rfc-editor.org/info/rfc8311>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

8.2. Informative References

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://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, <https://www.rfc-editor.org/info/rfc3540>.

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

[RFC9260] Stewart, R., Tüxen, M., and K. Nielsen, "Stream Control Transmission Protocol", RFC 9260, DOI 10.17487/RFC9260, June 2022, <https://www.rfc-editor.org/info/rfc9260>.

[I-D.stewart-tsvwg-sctpecn] Stewart, R. R. and M. Tüxen, "Explicit Congestion Notification for the Stream Control Transmission Protocol", Work in Progress, Internet-Draft, draft-stewart-tsvwg-sctpecn-06, 20 October 2023, <https://datatracker.ietf.org/doc/html/draft-stewart-tsvwg-sctpecn-06>.

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

[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, <https://www.rfc-editor.org/info/rfc5562>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6928>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High

Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <https://www.rfc-editor.org/info/rfc7323>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <https://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, <https://www.rfc-editor.org/info/rfc7567>.

[RFC7661] Fairhurst, G., Sathiaseelan, A., and R. Secchi, "Updating TCP to Support Rate-Limited Traffic", RFC 7661, DOI 10.17487/RFC7661, October 2015, <https://www.rfc-editor.org/info/rfc7661>.

[RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers", RFC 8257, DOI 10.17487/RFC8257, October 2017, <https://www.rfc-editor.org/info/rfc8257>.

[RFC8511] Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)", RFC 8511, DOI 10.17487/RFC8511, December 2018, <https://www.rfc-editor.org/info/rfc8511>.

[RFC9040] Touch, J., Welzl, M., and S. Islam, "TCP Control Block Interdependence", RFC 9040, DOI 10.17487/RFC9040, July 2021, <https://www.rfc-editor.org/info/rfc9040>.

[RFC9331] De Schepper, K. and B. Briscoe, Ed., "The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S)", RFC 9331, DOI 10.17487/RFC9331, January 2023, <https://www.rfc-editor.org/info/rfc9331>.

[RFC9330] Briscoe, B., Ed., De Schepper, K., Bagnulo, M., and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture", RFC 9330, DOI 10.17487/RFC9330, January 2023, <https://www.rfc-editor.org/info/rfc9330>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI

10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9438] Xu, L., Ha, S., Rhee, I., Goel, V., and L. Eggert, Ed., "CUBIC for Fast and Long-Distance Networks", RFC 9438, DOI 10.17487/RFC9438, August 2023, <https://www.rfc-editor.org/info/rfc9438>.

[I-D.briscoe-iccrg-prague-congestion-control] De Schepper, K., Tilmans, O., Briscoe, B., and V. Goel, "Prague Congestion Control", Work in Progress, Internet-Draft, draft-briscoe-iccrg-prague-congestion-control-03, 14 October 2023, <https://datatracker.ietf.org/doc/html/draft-briscoe-iccrg-prague-congestion-control-03>.

[judd-nsdi] Judd, G.J., "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, <https://www.usenix.org/node/188966>.

[ecn-pam] Trammell, B., Kühlewind, 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, <https://link.springer.com/chapter/10.1007/978-3-319-15509-8_15>.

[ECN-PLUS] Kuzmanovic, A., "The Power of Explicit Congestion Notification", ACM SIGCOMM 35(4):61--72, 2005, <https://dl.acm.org/citation.cfm?id=1080100>.

[Mandalari18] Mandalari, A., Lutu, A., Briscoe, B., Bagnulo, M., and Ö. Alay, "Measuring ECN++: Good News for ++, Bad News for ECN over Mobile", IEEE Communications Magazine , March 2018, <https://ieeexplore.ieee.org/document/8316790>.

[Manzoor17] Manzoor, J., Drago, I., and R. Sadre, "How HTTP/2 is changing Web traffic and how to detect it", In Proc: Network Traffic Measurement and Analysis Conference (TMA) 2017 pp.1-9, June 2017, <https://ieeexplore.ieee.org/document/8002899>.

[Kuehlewind18] Kühlewind, M., Walter, M., Learmonth, I., and B. Trammell, "Tracing Internet Path Transparency", In Proc: Network Traffic Measurement and Analysis Conference (TMA)

2018 , June 2018, <https://ieeexplore.ieee.org/document/8506532>.

[strict-ecn] Dumazet, E., "tcp: be more strict before accepting ECN negociation", Linux netdev patch list , 4 May 2012, <https://patchwork.ozlabs.org/patch/156953/>.

[relax-strict-ecn] Tilmans, O., "tcp: Accept ECT on SYN in the presence of RFC8311", Linux netdev patch list , 3 April 2019, <https://lore.kernel.org/patchwork/patch/1057812/>.

[ecn-overload] Steen, H., "Destruction Testing: Ultra-Low Delay using Dual Queue Coupled Active Queue Management", Masters Thesis, Uni Oslo , May 2017, <https://urn.nb.no/URN:NBN:no-60155>.

[DOCSIS3.1] CableLabs, "MAC and Upper Layer Protocols Interface (MULPI) Specification, CM-SP-MULPIv3.1", Data-Over-Cable Service Interface Specifications DOCSIS® 3.1 Version i17 or later, 21 January 2019, <https://specification-search.cablelabs.com/CM-SP-MULPIv3.1>.

Acknowledgments

Thanks to Mirja Kühlewind, David Black, Padma Bhooma, Gorry Fairhurst, Michael Scharf, Yuchung Cheng and Christophe Paasch for their useful reviews. Richard Scheffenegger provided useful advice gained from implementing ECN++ for FreeBSD.

The work of Marcelo Bagnulo has been partially funded by EU under projects Stand-ICT CCI and H2020-ICT-2014-2 5G NORMA.

Bob Briscoe's contribution was partly funded by Apple Inc, partly by the Research Council of Norway through the TimeIn project, partly by CableLabs and partly by the Comcast Innovation Fund. The views expressed here are solely those of the authors.

Authors' Addresses

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

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

Bob Briscoe (editor)

Independent
United Kingdom

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