Transport Area Working Group
Internet-Draft
Intended status: Historic
Expires: January 17, 2014

B. Briscoe, Ed.
A. Jacquet
BT
T. Moncaster
Moncaster.com
A. Smith
BT

July 16, 2013

Re-ECN: Adding Accountability for Causing Congestion to TCP/IP
draft-briscoe-conex-re-ecn-tcp-02

Abstract

This document introduces re-ECN (re-inserted explicit congestion notification), which is intended to make a simple but far-reaching change to the Internet architecture. The sender uses the IP header to reveal the congestion that it expects on the end-to-end path. The protocol works by arranging an extended ECN field in each packet so that, as it crosses any interface in an internetwork, it will carry a truthful prediction of congestion on the remainder of its path. It can be deployed incrementally around unmodified routers. The purpose of this document is to specify the re-ECN protocol at the IP layer and to give guidelines on any consequent changes required to transport protocols. It includes the changes required to TCP both as an example and as a specification. It briefly gives examples of mechanisms that can use the protocol to ensure data sources respond sufficiently to congestion, but these are described more fully in a companion document.

Note concerning Intended Status: If this draft were ever published as an RFC it would probably have historic status. There is limited space in the IP header, so re-ECN had to compromise by requiring the receiver to be ECN-enabled otherwise the sender could not use re-ECN. Re-ECN was a precursor to chartering of the IETF's Congestion Exposure (ConEx) working group, but during chartering there were still too few ECN receivers enabled, therefore it was decided to pursue other compromises in order to fit a similar capability into the IP header.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on January 17, 2014.

Table of Contents

Authors' Statement: (to be removed by the RFC Editor)

The most immediate priority for the authors is to delay any move of the ECN nonce to Proposed Standard status, in order to leave options open for the future. The argument for this position is developed in Appendix E.

Changes from previous drafts (to be removed by the RFC Editor)

Full diffs from all previous versions (created using the rfcdiff tool) are available at <http://www.bobbriscoe.net/pubs.html#retcp>

From draft-briscoe-conex-...-01 to -02 (current version): Re-issued to keep alive; updated references

From draft-briscoe-conex-...-00 to -01: Re-issued to keep alive; updated references

From draft-briscoe-tsvwg-...-08 to draft-briscoe-conex-...-00:

Re-issued to keep alive for reference by ConEx working group

Changed working group tag in filename from tsvwg to conex

Changed intended status to historic and added explanatory note

Updated references. Also, now that RFC6040 has been published, the section on tunnelling required a re-write

Corrected name of CE(0) to Cancelled in Table 2

Noted errors and omissions (rather than spending time correcting them):

* Made a few 'ToDo' comments visible that had previously been comments within the document source

* Identified errors with 'ToDo' comments, referring to correct material where possible.

From -08 to -09:

Re-issued to keep alive for reference by ConEx working group.

Hardly any changes to content, even where it is out of date, except references updated.

From -07 to -08:

Minor changes and consistency checks.

References updated.

From -06 to -07:

Major changes made following splitting this protocol document from the related motivations document [I-D.re-ecn-motiv].

Significant re-ordering of remaining text.

New terminology introduced for clarity.

Minor editorial changes throughout.

1. Introduction

This document provides a complete specification for the addition of the re-ECN protocol to IP and guidelines on how to add it to transport layer protocols, including a complete specification of re-ECN in TCP as an example. The motivation behind this proposal is given in [I-D.re-ecn-motiv], but we include a brief summary here.

Re-ECN is intended to allow senders to inform the network of the level of congestion they expect their flows to see. This information is currently only visible at the transport layer. ECN [RFC3168] reveals the upstream congestion state of any path by monitoring the rate of CE marks. The receiver then informs the sender when they have seen a marked packet. Re-ECN builds on ECN by providing new codepoints that allow the sender to declare the level of congestion they expect on the forward path. It is closely related to ECN and indeed we define a compatibility mode to allow a re-ECN sender to communicate with an ECN receiver.

If a sender understates expected congestion compared to actual congestion then the network could discard packets or enact some other sanction. A policer can also be introduced at the ingress of networks that can limit the level of congestion being caused.

A general statement of the problem solved by re-ECN is to provide sufficient information in each IP datagram to be able to hold senders and whole networks accountable for the congestion they cause downstream, before they cause it. But the every-day problems that re-ECN can solve are much more recognisable than this rather generic statement: mitigating distributed denial of service (DDoS); simplifying differentiation of quality of service (QoS); policing

compliance to congestion control; and so on.

It is important to add a few key points.

- In any standard network it always takes one round trip before any feedback is received. For this reason a sender must make a conservative prediction by transmitting IP packets with a special Cautious marking when it is unsure of the state of the network.

- It should be noted that the prediction is carried in-band in normal data packets and for many transports feedback can be carried in the normal acknowledgements or control packets.

- The re-ECN protocol is independent of the transport. In TCP, acknowledgments are used to convey the feedback from receiver to sender. This memo concentrates on TCP as an example transport protocol, however the re-ECN protocol is compatible with any transport where feedback can be sent from receiver to sender.

This document is structured as follows. First an overview of the re-ECN protocol is given (Section 4), outlining its attributes and explaining conceptually how it works as a whole. The two main parts of the document follow. That is, the protocol specification divided into network (Section 5) and transport (Section 6) layers. Deployment issues discussed throughout the document are brought together in Section 7. Related work is discussed in (Section 8).

2. Requirements notation

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

3. Terminology

{ToDo: No attempt has been made to bring terminology into line with that agreed within the ConEx working group. For instance the term dropper remains unchanged, even though the ConEx w-g has decided to call it an audit function (which is actually a much better term).}

The following terminology is used throughout this memo. Some of this terminology has changed as this draft has been revised. Therefore, to help avoid confusion, Appendix F sets out all the alternative terminology that has been used in other re-ECN related documents.

- Neutral packet - a packet that is able to be congestion marked by an ECN or re-ECN queue.

- Negative packet - a Neutral packet that has been congestion marked by an ECN or re-ECN queue.

- Positive packet - a packet that has been marked by the sender to indicate the expected level of congestion along its path. In general Positive packets should only be sent in response to feedback received from the receiver.*

- Cancelled packet - a Positive Packet that has been congestion marked by an ECN or re-ECN queue.

- Cautious packet - a packet that has been marked by the sender to indicate the expected level of congestion along its path. In general Cautious packets should be used when there is insufficient feedback to be confident about the congestion state of the network.*

  * the difference between positive and cautious packets is explained in detail later in the document along with guidelines on the use of Cautious packets.

All the above terms have related IP codepoints as defined in (Section 5).

4. Protocol Overview

4.1. Simplified Re-ECN Protocol

We describe here the simplified re-ECN protocol. To simplify the description we assume packets and segments are synonymous.

Packets are sent from a sender to a receiver. In Figure 1 the queues (Q1 and Q2) are ECN enabled as per RFC 3168 [RFC3168]. If congestion occurs then packets are marked with the congestion experienced (CE) flag exactly as in the ECN protocol [RFC3168]; the routers do not need to be modified and do not need to know the re-ECN protocol. The receiver constantly informs the sender of the current count of Negative packets it has seen. The sender uses this information determine how many Positive packets it must send into the network. The receiver's aim is to balance the number of bytes that have been congestion marked with the number of Positive bytes it has sent.

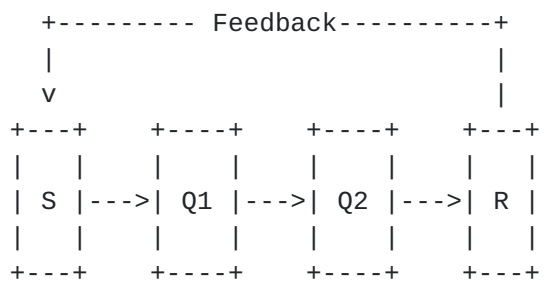

Figure 1: Simple Re-ECN

4.1.1. Congestion Control and Policing the Protocol

The arrangement of the protocol ensures that packets carry a declaration of the amount of congestion that will be experienced on the path. The re-ECN protocol is orthogonal to any congestion control algorithms, but can be used to ensure that congestion control is being applied by the sender.

In general we assume that there will be a policer at the network ingress which can rate limit traffic based on the amount of congestion declared.

At the network egress there is a dropper which can impose sanctions on flows that incorrectly declare congestion.

Policers and droppers are explained in more detail in [I-D.re-ecn-motiv].

4.1.2. Background and Applicability

The re-ECN protocol makes no changes and has no effect on the TCP congestion control algorithm or on other rate responses to congestion. Re-ECN is not a new congestion control protocol, rather it is orthogonal to congestion control itself. Re-ECN is concerned with revealing information about congestion so that users and networks can be held accountable for the congestion they cause, or allow to be caused.

Re-ECN builds on ECN so we briefly recap the essentials of the ECN protocol [RFC3168]. Two bits in the IP protocol (v4 or v6) are assigned to the ECN field. The sender clears the field to "00" (Not-ECT) if either end-point transport is not ECN-capable. Otherwise it indicates an ECN-capable transport (ECT) using either of the two code-points "10" or "01" (ECT(0) and ECT(1) resp.).

ECN-capable queues probabilistically set this field to "11" if

congestion is experienced (CE). In general this marking probability will increase with the length of the queue at its egress link (typically using the RED algorithm [RFC2309]). However, they still drop rather than mark Not-ECT packets. With multiple ECN-capable queues on a path, a flow of packets accumulates the fraction of CE marking that each queue adds. The combined effect of the packet marking of all the queues along the path signals congestion of the whole path to the receiver. So, for example, if one queue early in a path is marking 1% of packets and another later in a path is marking 2%, flows that pass through both queues will experience approximately 3% marking (see Appendix A for a precise treatment).

The choice of two ECT code-points in the ECN field [RFC3168] permitted future flexibility, optionally allowing the sender to encode the experimental ECN nonce [RFC3540] in the packet stream. The nonce is designed to allow a sender to check the integrity of congestion feedback. But Section 8.1 explains that it still gives no control over how fast the sender transmits as a result of the feedback. On the other hand, re-ECN is designed both to ensure that congestion is declared honestly and that the sender's rate responds appropriately.

Re-ECN is based on a feedback arrangement called `re-feedback' [Re-fb]. The word is short for either receiver-aligned, re-inserted or re-echoed feedback. But it actually works even when no feedback is available. In fact it has been carefully designed to work for single datagram flows. It also encourages aggregation of single packet flows by congestion control proxies. Then, even if the traffic mix of the Internet were to become dominated by short messages, it would still be possible to control congestion effectively and efficiently.

Changing the Internet's feedback architecture seems to imply considerable upheaval. But re-ECN can be deployed incrementally at the transport layer around unmodified queues using existing fields in IP (v4 or v6). However it does also require the last undefined bit in the IPv4 header, which it uses in combination with the 2-bit ECN field to create four new codepoints. Nonetheless, we RECOMMEND adding optional preferential drop to IP queues based on the re-ECN fields in order to improve resilience against DoS attacks. Similarly, re-ECN works best if both the sender and receiver transports are re-ECN-capable, but it can work with just sender support(Section 6.1.2).

4.2. Re-ECN Abstracted Network Layer Wire Protocol (IPv4 or v6)

The re-ECN wire protocol uses the two bit ECN field broadly as in RFC3168 [RFC3168] as described above, but with five differences of

detail (brought together in a list in Section 7). This specification defines a new re-ECN extension (RE) flag. We will defer the definition of the actual position of the RE flag in the IPv4 & v6 headers until Section 5. When we don't need to choose between IPv4 and v6 wire protocols it will suffice call it the RE flag.

Unlike the ECN field, the RE flag is intended to be set by the sender and SHOULD remain unchanged along the path, although it can be read by network elements that understand the re-ECN protocol. It is feasible that a network element MAY change the setting of the RE flag, perhaps acting as a proxy for an end-point, but such a protocol would have to be defined in another specification (e.g. [I-D.re-pcn-border-cheat]).

Although the RE flag is a separate, single bit field, it can be read as an extension to the two-bit ECN field; the three concatenated bits in what we will call the extended ECN field (EECN) giving eight codepoints. We will use the RFC3168 names of the ECN codepoints to describe settings of the ECN field when the RE flag setting is "don't care", but we also define the following six extended ECN codepoint names for when we need to be more specific.

One of re-ECN's codepoints is an alternative use of the codepoint set aside in RFC3168 for the ECN nonce (ECT(1)). Transports using re-ECN do not need to use the ECN nonce as long as the sender is also checking for transport protocol compliance [tcp-rcv-cheat]. The case for doing this is given in Appendix E. Two re-ECN codepoints are given compatible uses to those defined in RFC3168 (Not-ECT and CE). The other codepoint used by RFC3168 (ECT(0)) isn't used for re-ECN. Altogether this leave one codepoint of the eight unused by ECN or re-ECN and available for future use.

ECN field	RFC3168 codepoint	RE flag	EECN codepoint	re-ECN meaning
00	Not-ECT	0	Not-ECT	Not re-ECN-capable transport (Legacy)
00	---	1	FNE	Feedback not established (Cautious)
01	ECT(1)	0	Re-Echo	Re-echoed congestion and RECT (Positive)
01	---	1	RECT	Re-ECN capable transport (Neutral)
10	ECT(0)	0	ECT(0)	RFC3168 ECN use only
10	---	1	--CU--	Currently unused
11	CE	0	CE(0)	Re-Echo cancelled by CE (Cancelled)
11	---	1	CE(-1)	Congestion Experienced (Negative)

Table 1: Extended ECN Codepoints

4.3. Re-ECN Protocol Operation

In this section we will give an overview of the operation of the re-ECN protocol for TCP/IP, leaving a detailed specification to the following sections. Other transports will be discussed later.

{ToDo: This section to be updated to explain that the sender re-echoes losses in the same way as ECN markings.}

In summary, the protocol adds a third `re-echo' stage to the existing TCP/IP ECN protocol. Whenever the network adds CE congestion signalling to the IP header on the forward data path, the receiver feeds it back to the ingress using TCP, then the sender re-echoes it into the forward data path using the RE flag in the next packet.

Prior to receiving any feedback a sender will not know which setting of the RE flag to use, so it sends Cautious packets by setting the FNE codepoint. The network reads the FNE codepoint conservatively as equivalent to re-echoed congestion.

Specifically, once feedback from an ECN or re-ECN capable flow is established, a re-ECN sender always initialises the ECN field to ECT(1). And it usually sets the RE flag to "1" indicating a Neutral packet. Whenever a queue marks a packet to CE, the receiver feeds

back this event to the sender. On receiving this feedback, the re-ECN sender will clear the RE flag to "0" in the next packet it sends (indicating a Positive packet).

We chose to set and clear the RE flag this way round to ease incremental deployment (see Section 7). To avoid confusion we will use the term `blanking' (rather than marking) when the RE flag is cleared to "0". So, over a stream of packets, we will talk of the `RE blanking fraction' as the fraction of octets in packets with the RE flag cleared to "0".

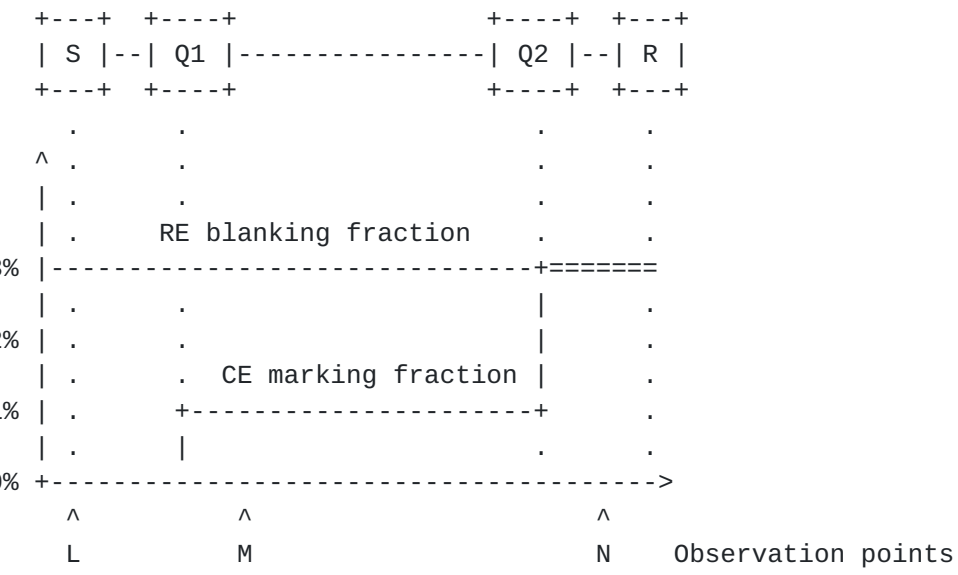

Figure 2: A 2-Queue Example (Imprecise)

Figure 2 uses a simple network to illustrate how re-ECN allows queues to measure downstream congestion. The receiver views a CE marking fraction of 3% which is fed back to the sender. The sender sets an RE blanking fraction of 3% to match this. This RE blanking fraction can be observed along the path as the RE flag is not changed by network nodes once set by the sender. This is shown by the horizontal line at 3% in the figure. The CE marked fraction is shown by the stepped line which rises to meet the RE blanking fraction line with steps at each queue where packets are marked. Two queues are shown (Q1 and Q2) that are currently congested. Each time packets pass through a fraction are marked; 1% at Q1 and 2% at Q2). The approximate downstream congestion can be measured at the observation points shown along the path by subtracting the CE marking fraction from the RE blanking fraction, as shown in the table below (Appendix A derives these approximations from a precise analysis). NB due to the unary nature of ECN marking and the equivalent unary

nature of re-ECN blanking, the precise fraction of marked bytes must be calculated by maintaining a moving average of the number of packets that have been marked as a proportion of the total number of packets.

Along the path the fraction of packets that had their RE field cleared remains unchanged so it can be used as a reference against which to compare upstream congestion. The difference predicts downstream congestion for the rest of the path. Therefore, measuring the fractions of each codepoint at any point in the Internet will reveal upstream, downstream and whole path congestion.

Note that we have introduced discussion of marking and blanking fractions solely for illustration. We are not saying any protocol handler will work with these average fractions directly. In fact the protocol actually requires the number of marked and blanked bytes to balance by the time the packet reaches the receiver.

4.4. Positive and Negative Flows

In Section 3 we introduced the terms Positive, Neutral, Negative, Cautious and Cancelled. This terminology is based on the requirement to balance the proportion of bytes marked as CE with the proportion of bytes that are re-echo marked. In the rest of this memo we will loosely talk of positive or negative flows, meaning flows where the moving average of the downstream congestion metric is persistently positive or negative. A negative flow is one where more CE marked packets than re-ECN blanked packets arrive. Likewise in positive flows more re-ECN blanked packets arrive than CE marked packets. The notion of a negative metric arises because it is derived by subtracting one metric from another. Of course actual downstream congestion cannot be negative, only the metric can (whether due to time lags or deliberate malice).

Therefore we will talk of packets having `worth' of +1, 0 or -1, which, when multiplied by their size, indicates their contribution to the downstream congestion metric. The worth of each type of packet is given below in Table 2. The idea is that most flows start with zero worth. Every time the network decrements the worth of a packet, the sender increments the worth of a later packet. Then, over time, as many positive octets should arrive at the receiver as negative. Note we have said octets not packets, so if packets are of different sizes, the worth should be incremented on enough octets to balance the octets in negative packets arriving at the receiver. It is this balance that will allow the network to hold the sender accountable for the congestion it causes.

If a packet carrying re-echoed congestion happens to also be

congestion marked, the +1 worth added by the sender will be cancelled out by the -1 network congestion marking. Although the two worth values correctly cancel out, neither the congestion marking nor the re-echoed congestion are lost, because the RE bit and the ECN field are orthogonal. So, whenever this happens, the receiver will correctly detect and re-echo the new congestion event as well.

The table below specifies unambiguously the worth of each extended ECN codepoint. Note the order is different from the previous table to better show how the worth increments and decrements.

ECN field	RE bit	Extended ECN codepoint	Worth	Re-ECN Term
00	0	Not-RECT	...	---
00	1	FNE	+1	Cautious
01	0	Re-Echo	+1	Positive
10	0	Legacy	...	RFC3168 ECN use only
11	0	CE(0)	0	Cancelled
01	1	RECT	0	Neutral
10	1	--CU--	...	Currently unused
11	1	CE(-1)	-1	Negative

Table 2: 'Worth' of Extended ECN Codepoints

5. Network Layer

5.1. Re-ECN IPv4 Wire Protocol

The wire protocol of the ECN field in the IP header remains largely unchanged from [RFC3168]. However, an extension to the ECN field we call the RE (Re-ECN extension) flag (Section 4.2) is defined in this document. It doubles the extended ECN codepoint space, giving 8 potential codepoints. The semantics of the extra codepoints are backward compatible with the semantics of the 4 original codepoints [RFC3168] (Section 7 collects together and summarises all the changes defined in this document).

For IPv4, this document proposes that the new RE control flag will be positioned where the `reserved' control flag was at bit 48 of the IPv4 header (counting from 0). Alternatively, some would call this bit 0 (counting from 0) of byte 7 (counting from 1) of the IPv4 header (Figure 3).

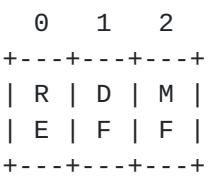

Figure 3: New Definition of the Re-ECN Extension (RE) Control Flag at the Start of Byte 7 of the IPv4 Header

The semantics of the RE flag are described in outline in Section 4 and specified fully in Section 6. The RE flag is always considered in conjunction with the 2-bit ECN field, as if they were concatenated together to form a 3-bit extended ECN field. If the ECN field is set to either the ECT(1) or CE codepoint, when the RE flag is blanked (cleared to "0") it represents a re-echo of congestion experienced by an early packet. If the ECN field is set to the Not-ECT codepoint, when the RE flag is set to "1" it represents the feedback not established (FNE) codepoint, which signals that the packet was sent without the benefit of congestion feedback.

It is believed that the FNE codepoint can simultaneously serve other purposes, particularly where the start of a flow needs distinguishing from packets later in the flow. For instance it would have been useful to identify new flows for tag switching and might enable similar developments in the future if it were adopted. It is similar to the state set-up bit idea designed to protect against memory exhaustion attacks. This idea was proposed informally by David Clark and documented by Handley and Greenhalgh [Steps DoS]. The FNE codepoint can be thought of as a `soft-state set-up flag', because it is idempotent (i.e. one occurrence of the flag is sufficient but further occurrences achieve the same effect if previous ones were lost).

We are sure there will probably be other claims pending on the use of bit 48. We know of at least two [ARI05], [RFC3514] but neither have been pursued in the IETF, so far, although the present proposal would meet the needs of the latter.

The security flag proposal (commonly known as the evil bit) was published on 1 April 2003 as Informational RFC 3514, but it was not adopted due to confusion over whether evil-doers might set it inappropriately. The present proposal is backward compatible with RFC3514 because if re-ECN compliant senders were benign they would correctly clear the evil bit to honestly declare that they had just received congestion feedback. Whereas evil-doers would hide congestion feedback by setting the evil bit continuously, or at least more often than they should. So, evil senders can be identified, because they declare that they are good less often than they should.

5.2. Re-ECN IPv6 Wire Protocol

For IPv6, this document proposes that the new RE control flag will be positioned as the first bit of the option field of a new Congestion hop by hop option header (Figure 4).

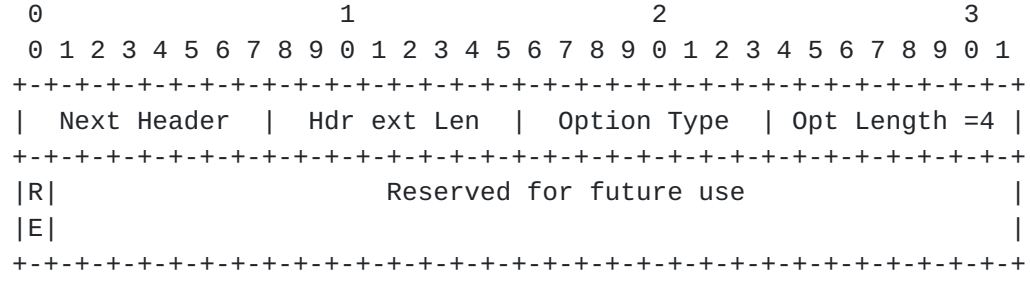

Figure 4: Definition of a New IPv6 Congestion Hop by Hop Option Header containing the re-ECN Extension (RE) Control Flag

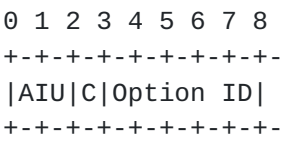

Figure 5: Congestion Hop by Hop Option Type Encoding

The Hop-by-Hop Options header enables packets to carry information to be examined and processed by routers or nodes along the packet's delivery path, including the source and destination nodes. For re-ECN, the two bits of the Action If Unrecognized (AIU) flag of the Congestion extension header MUST be set to "00" meaning if unrecognized `skip over option and continue processing the header'. Then, any routers or a receiver not upgraded with the optional re-ECN features described in this memo will simply ignore this header. But routers with these optional re-ECN features or a re-ECN policing function, will process this Congestion extension header.

The `C' flag MUST be set to "1" to specify that the Option Data (currently only the RE control flag) can change en-route to the packet's final destination. This ensures that, when an Authentication header (AH [RFC4302]) is present in the packet, for any option whose data may change en-route, its entire Option Data field will be treated as zero-valued octets when computing or verifying the packet's authenticating value.

Although the RE control flag should not be changed along the path, we expect that the rest of this option field that is currently `Reserved for future use' could be used for a multi-bit congestion notification

field which we would expect to change en route. Therefore, as changes to the RE flag could be detected end-to-end without authentication (see Section 9), we set the C flag to '1'.

5.3. Router Forwarding Behaviour

{ToDo: Consider a section on how whole protocol interworks with drop. Perhaps in Protocol Overview.}

Re-ECN works well without modifying the forwarding behaviour of any routers. However, below, two OPTIONAL changes to forwarding behaviour are defined which respectively enhance performance and improve a router's discrimination against flooding attacks. They are both OPTIONAL additions that we propose MAY apply by default to all Diffserv per-hop scheduling behaviours (PHBs) [RFC2475] and ECN marking behaviours [RFC3168]. Specifications for PHBs MAY define different forwarding behaviours from this default, but this is not required. [I-D.re-pcn-border-cheat] is one example.

FNE indicates ECT:

> The FNE codepoint tells a router to assume that the packet was sent by an ECN-capable transport (see Section 5.4). Therefore an FNE packet MAY be marked rather than dropped. Note that the FNE codepoint has been intentionally chosen so that, to RFC3168 compliant routers (which do not inspect the RE flag) an FNE packet appears to be Not-ECT so it will be dropped by legacy AQM algorithms.
>
> A network operator MUST NOT configure a queue to ECN mark rather than drop FNE packets unless it can guarantee that FNE packets will be rate limited, either locally or upstream. The ingress policers discussed in [I-D.re-ecn-motiv] would count as rate limiters for this purpose.

Preferential Drop: If a re-ECN capable router queue experiences very high load so that it has to drop arriving packets (e.g. a DoS attack), it MAY preferentially drop packets within the same Diffserv PHB using the preference order for extended ECN codepoints given in Table 3. Preferential dropping can be difficult to implement on some hardware, but if feasible it would discriminate against attack traffic if done as part of the overall policing framework of [I-D.re-ecn-motiv]. If nowhere else, routers at the egress of a network SHOULD implement preferential drop (stronger than the MAY above). For simplicity, preferences 4 & 5 MAY be merged into one preference level.

The tabulated drop preferences are arranged to preserve packets with more positive worth (Section 4.4), given senders of positive packets must have honestly declared downstream congestion. A full treatment of this is provided in the companion document describing the motivation and architecture for re-ECN [I-D.re-ecn-motiv] particularly when the application of re-ECN to protect against DDoS attacks is described.

ECN field	RE bit	Extended ECN codepoint	Worth	Drop Pref (1 = drop 1st)	Re-ECN meaning
01	0	Re-Echo	+1	5/4	Re-echoed congestion and RECT
00	1	FNE	+1	4	Feedback not established
11	0	CE(0)	0	3	Re-Echo canceled by congestion experienced
01	1	RECT	0	3	Re-ECN capable transport
11	1	CE(-1)	-1	3	Congestion experienced
10	1	--CU--	n/a	2	Currently Unused
10	0	---	n/a	2	RFC3168 ECN use only
00	0	Not-RECT	n/a	1	Not Re-ECN-capable transport

Table 3: Drop Preference of EECN Codepoints (Sorted by `Worth')

5.4. Justification for Setting the First SYN to FNE

the initial SYN MUST be set to FNE by Re-ECT client A (Section 6.1.4) and (Section 5.3) says a queue MAY optionally treat an FNE packet as ECN capable, so an initial SYN may be marked CE(-1) rather than dropped. This seems dangerous, because the sender has not yet established whether the receiver is a RFC3168 one that does not understand congestion marking. It also seems to allow malicious senders to take advantage of ECN marking to avoid so much drop when launching SYN flooding attacks. Below we explain the features of the protocol design that remove both these dangers.

ECN-capable initial SYN with a Not-ECT server: If the TCP server B is re-ECN capable, provision is made for it to feedback a possible congestion marked SYN in the SYN ACK (Section 6.1.4). But if the TCP client A finds out from the SYN ACK that the server was not ECN-capable, the TCP client MUST conservatively consider the first SYN as congestion marked before setting itself into Not-ECT mode. Section 6.1.4 mandates that such a TCP client MUST also set its initial window to 1 segment. In this way we remove the need to cautiously avoid setting the first SYN to Not-RECT. This will give worse performance while deployment is patchy, but better performance once deployment is widespread.

SYN flooding attacks can't exploit ECN-capability: Malicious hosts may think they can use the advantage that ECN-marking gives over drop in launching classic SYN-flood attacks. But Section 5.3 mandates that a router MUST only be configured to treat packets with the FNE codepoint as ECN-capable if FNE packets are rate limited somewhere. Introduction of the FNE codepoint was a deliberate move to enable transport-neutral handling of flow-start and flow state set-up in the IP layer where it belongs. It then becomes possible to protect against flooding attacks of all forms (not just SYN flooding) without transport-specific inspection for things like the SYN flag in TCP headers. Then, for instance, SYN flooding attacks using IPsec ESP encryption can also be rate limited at the IP layer.

It might seem pedantic going to all this trouble to enable ECN on the initial packet of a flow, but it is motivated by a much wider concern to ensure safe congestion control will still be possible even if the application mix evolves to the point where the majority of flows consist of a single window or even a single packet. It also allows denial of service attacks to be more easily isolated and prevented.

{ToDo: Give alternative where initial packet is Not-RECT and last ACK of three-way handshake is FNE. Explain this will give better performance while deployment is patchy, but worse performance once deployment is high.}

5.5. Control and Management

5.5.1. Negative Balance Warning

A new ICMP message type is being considered so that a dropper can warn the apparent sender of a flow that it has started to sanction the flow. The message would have similar semantics to the `Time exceeded' ICMP message type. To ensure the sender has to invest some work before the network will generate such a message, a dropper SHOULD only send such a message for flows that have demonstrated that

they have started correctly by establishing a positive record, but have later gone negative. The threshold is up to the implementation. The purpose of the message is to deconfuse the cause of drops from other causes, such as congestion or transmission losses. The dropper would send the message to the sender of the flow, not the receiver. If we did define this message type, it would be REQUIRED for all re-ECT senders to parse and understand it. Note that a sender MUST only use this message to explain why losses are occurring. A sender MUST NOT take this message to mean that losses have occurred that it was not aware of. Otherwise, spoof messages could be sent by malicious sources to slow down a sender (c.f. ICMP source quench).

However, the need for this message type is not yet confirmed, as we are considering how to prevent it being used by malicious senders to scan for droppers and to test their threshold settings. {ToDo: Complete this section.}

5.5.2. Rate Response Control

As discussed in [I-D.re-ecn-motiv] the sender's access operator will be expected to use bulk per-user policing, but they might choose to introduce a per-flow policer. In cases where operators do introduce per-flow policing, there may be a need for a sender to send a request to the ingress policer asking for permission to apply a non-default response to congestion (where TCP-friendly is assumed to be the default). This would require the sender to know what message format(s) to use and to be able to discover how to address the policer. The required control protocol(s) are outside the scope of this document, but will require definition elsewhere.

The policer is likely to be local to the sender and inline, probably at the ingress interface to the internetwork. So, discovery should not be hard. A variety of control protocols already exist for some widely used rate-responses to congestion. For instance DCCP congestion control identifiers (CCIDs [RFC4340]) fulfil this role and so does QoS signalling (e.g. and RSVP request for controlled load service is equivalent to a request for no rate response to congestion, but with admission control).

5.6. IP in IP Tunnels

Ideally, for re-ECN to work through IP in IP tunnels, the tunnel entry should copy both the RE flag and the ECN field from the inner to the outer IP header. Then at the tunnel exit, any CE marking of the outer ECN field should overwrite the inner ECN field (unless the inner field is Not-ECT in which case an alarm should be raised). The RE flag shouldn't change along a path, so the outer RE flag should be the same as the inner. If it isn't, a management alarm should be

raised.

This requirement is satisfied by the latest specification for handling ECN through IP tunnels [RFC6040] as well as by IPsec [RFC4301]. However, it is not satisfied by the ingress behaviour specified in [RFC3168] although at least the full-functionality variant of the egress behaviour is fine. RFC6040 updates RFC3168, but it is likely that many legacy non-IPsec IP-in-IP tunnels will exist.

If legacy tunnels are left as specified in [RFC3168], whether the limited or full-functionality variants is used, a problem arises with re-ECN if a tunnel crosses an inter-domain boundary, because the difference between positive and negative markings will not be correctly accounted for. In a limited functionality ECN tunnel, the flow will appear to be RFC3168 compliant traffic, and therefore may be wrongly rate limited. In a full-functionality ECN tunnel, the result will depend whether the tunnel entry copies the inner RE flag to the outer header or the RE flag in the outer header is always cleared. If the former, the flow will tend to be too positive when accounted for at borders. If the latter, it will be too negative. If the rules set out in [RFC6040] are followed then this will not be an issue.

5.7. Non-Issues

The following issues might seem to cause unfavourable interactions with re-ECN, but we will explain why they don't:

- Various link layers support explicit congestion notification, such as Frame Relay and ATM. Explicit congestion notification is proposed to be added to other link layers, such as Ethernet (802.3ar Ethernet congestion management) and MPLS [RFC5129];

- Encryption and IPsec.

In the case of congestion notification at the link layer, each particular link layer scheme either manages congestion on the link with its own link-level feedback (the usual arrangement in the cases of ATM and Frame Relay), or congestion notification from the link layer is merged into congestion notification at the IP level when the frame headers are decapsulated at the end of the link (the recommended arrangement in the Ethernet and MPLS cases). Given the RE flag is not intended to change along the path, this means that downstream congestion will still be measurable at any point where IP is processed on the path by subtracting positive from negative markings.

In the case of encryption, as long as the tunnel issues described in Section 5.6 are dealt with, payload encryption itself will not be a problem. The design goal of re-ECN is to include downstream congestion in the IP header so that it is not necessary to bury into inner headers. Obfuscation of flow identifiers is not a problem for re-ECN policing elements. Re-ECN doesn't ever require flow identifiers to be valid, it only requires them to be unique. So if an IPsec encapsulating security payload (ESP [RFC4835]) or an authentication header (AH [RFC4302]) is used, the security parameters index (SPI) will be a sufficient flow identifier, as it is intended to be unique to a flow without revealing actual port numbers.

In general, even if endpoints use some locally agreed scheme to hide port numbers, re-ECN policing elements can just consider the pair of source and destination IP addresses as the flow identifier. Re-ECN encourages endpoints to at least tell the network layer that a sequence of packets are all part of the same flow, if indeed they are. The alternative would be for the sender to make each packet appear to be a new flow, which would require them all to be marked FNE in order to avoid being treated with the bulk of malicious flows at the egress dropper. Given the FNE marking is worth +1 and networks are likely to rate limit FNE packets, endpoints are given an incentive not to set FNE on each packet. But if the sender really does want to hide the flow relationship between packets it can choose to pay the cost of multiple FNE packets, which in the long run will compensate for the extra memory required on network policing elements to process each flow.

{ToDo: Add a note about it being useful that the AH header does not cover the RE flag, referring to Section 9.}

6. Transport Layers

6.1. TCP

Re-ECN capability at the sender is essential. At the receiver it is optional, as long as the receiver has a basic RFC3168-compliant ECN-capable transport (ECT) [RFC3168]. Given re-ECN is not the first attempt to define the semantics of the ECN field, we give a table below summarising what happens for various combinations of capabilities of the sender S and receiver R, as indicated in the first four columns below. The last column gives the mode a half-connection should be in after the first two of the three TCP handshakes.

Re-ECT	ECT-Nonce (RFC3540)	ECT (RFC3168)	Not-ECT	S-R Half-connection Mode
SR				RECN
S	R			RECN-Co
S		R		RECN-Co
S			R	Not-ECT

Table 4: Modes of TCP Half-connection for Combinations of ECN Capabilities of Sender S and Receiver R

We will describe what happens in each mode, then describe how they are negotiated. The abbreviations for the modes in the above table mean:

RECN: Full re-ECN capable transport

RECN-Co: Re-ECN sender in compatibility mode with a RFC3168 compliant [RFC3168] ECN receiver or an [RFC3540] ECN nonce-capable receiver. Implementation of this mode is OPTIONAL.

Not-ECT: Not ECN-capable transport, as defined in [RFC3168] for when at least one of the transports does not understand even basic ECN marking.

Note that we use the term Re-ECT for a host transport that is re-ECN-capable but RECN for the modes of the half connections between hosts when they are both Re-ECT. If a host transport is Re-ECT, this fact alone does NOT imply either of its half connections will necessarily be in RECN mode, at least not until it has confirmed that the other host is Re-ECT.

6.1.1. RECN mode: Full Re-ECN capable transport

In full RECN mode, for each half connection, both the sender and the receiver each maintain an unsigned integer counter we will call ECC (echo congestion counter). The receiver maintains a count of how many times a CE marked packet has arrived during the half-connection. Once a RECN connection is established, the three TCP option flags (ECE, CWR & NS) used for ECN-related functions in other versions of ECN are used as a 3-bit field for the receiver to repeatedly tell the sender the current value of ECC, modulo 8, whenever it sends a TCP ACK. We will call this the echo congestion increment (ECI) field. This overloaded use of these 3 option flags as one 3-bit ECI field is shown in Figure 7. The actual definition of the TCP header,

including the addition of support for the ECN nonce, is shown for comparison in Figure 6. This specification does not redefine the names of these three TCP option flags, it merely overloads them with another definition once a flow is established.

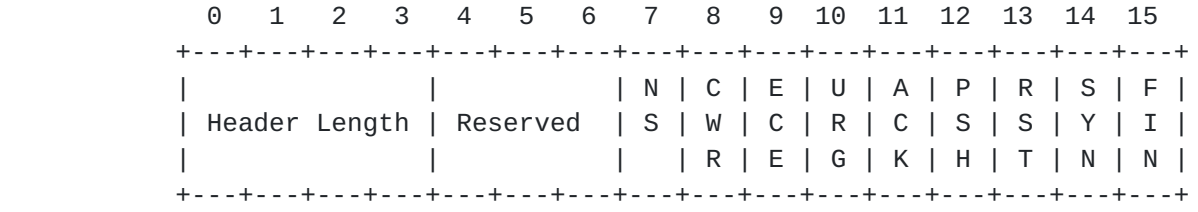

Figure 6: The (post-ECN Nonce) definition of bytes 13 and 14 of the TCP Header

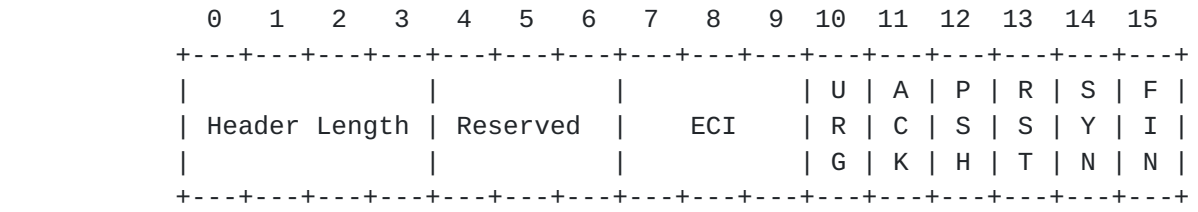

Figure 7: Definition of the ECI field within bytes 13 and 14 of the TCP Header, overloading the current definitions above for established RECN flows.

Receiver Action in RECN Mode

Every time a CE marked packet arrives at a receiver in RECN mode, the receiver transport increments its local value of ECC and MUST echo its value, modulo 8, to the sender in the ECI field of the next ACK. It MUST repeat the same value of ECI in every subsequent ACK until the next CE event, when it increments ECI again.

The increment of the local ECC values is modulo 8 so the field value simply wraps round back to zero when it overflows. The least significant bit is to the right (labelled bit 9).

A receiver in RECN mode MAY delay the echo of a CE to the next delayed-ACK, which would be necessary if ACK-withholding were implemented.

Sender Action in RECN Mode

On the arrival of every ACK, the sender compares the ECI field with its own ECC value, then replaces its local value with that from the ACK. The difference D (D = (ECI + 8 - ECC mod 8) mod 8) is assumed to be the number of CE marked packets that arrived at the receiver since it sent the previously received ACK (but see below for the sender's safety strategy). Whenever the ECI field increments by D (and/or d drops are detected), the sender MUST clear the RE flag to "0" in the IP header of the next D' data packets it sends (where D' = D + d), effectively re-echoing each single increment of ECI. Otherwise the data sender MUST send all data packets with RE set to "1".

As a general rule, once a flow is established, as well as setting or clearing the RE flag as above, a data sender in RECN mode MUST always set the ECN field to ECT(1). However, the settings of the extended ECN field during flow start are defined in Section 6.1.4.

As we have already emphasised, the re-ECN protocol makes no changes and has no effect on the TCP congestion control algorithm. So, the first increment of ECI (or detection of a drop) in a RTT triggers the standard TCP congestion response, no more than one congestion response per round trip, as usual. However, the sender re-echoes every increment of ECI irrespective of RTTs.

A TCP sender also acts as the receiver for the other half-connection. The host will maintain two ECC values S.ECC and R.ECC as sender and receiver respectively. Every TCP header sent by a host in RECN mode will also repeat the prevailing value of R.ECC in its ECI field. If a sender in RECN mode has to retransmit a packet due to a suspected loss, the re-transmitted packet MUST carry the latest prevailing value of R.ECC when it is re-transmitted, which will not necessarily be the one it carried originally.

6.1.2. RECN-Co mode: Re-ECT Sender with a RFC3168 compliant ECN Receiver

If the half-connection is in RECN-Co mode, ECN feedback proceeds no differently to that of RFC3168 compliant ECN. In other words, the receiver sets the ECE flag repeatedly in the TCP header and the sender responds by setting the CWR flag. Although RECN-Co mode is used when the receiver has not implemented the re-ECN protocol, the sender can infer enough from its RFC3168 compliant ECN feedback to set or clear the RE flag reasonably well. Specifically, every time the receiver toggles the ECE field from "0" to "1" (or a loss is detected), as well as setting CWR in the TCP flags, the re-ECN sender

MUST blank the RE flag of the next packet to "0" as it would do in full RECN mode. Otherwise, the data sender SHOULD send all other packets with RE set to "1". Once a flow is established, a re-ECN data sender in RECN-Co mode MUST always set the ECN field to ECT(1).

If a CE marked packet arrives at the receiver within a round trip time of a previous mark, the receiver will still be echoing ECE for the last CE mark. Therefore, such a mark will be missed by the sender. Of course, this isn't of concern for congestion control, but it does mean that very occasionally the RE blanking fraction will be understated. Therefore flows in RECN-Co mode may occasionally be mistaken for very lightly cheating flows and consequently might suffer a small number of packet drops through an egress dropper. We expect re-ECN would be deployed for some time before policers and droppers start to enforce it. So, given there is not much ECN deployment yet anyway, this minor problem may affect only a very small proportion of flows, reducing to nothing over the years as RFC3168 compliant ECN hosts upgrade. The use of RECN-Co mode would need to be reviewed in the light of experience at the time of re-ECN deployment.

RECN-Co mode is OPTIONAL. Re-ECN implementers who want to keep their code simple, MAY choose not to implement this mode. If they do not, a re-ECN sender SHOULD fall back to RFC3168 compliant ECT mode in the presence of an ECN-capable receiver. It MAY choose to fall back to the ECT-Nonce mode, but if re-ECN implementers don't want to be bothered with RECN-Co mode, they probably won't want to add an ECT-Nonce mode either.

6.1.2.1. Re-ECN support for the ECN Nonce

A TCP half-connection in RECN-Co mode MUST NOT support the ECN Nonce [RFC3540]. This means that the sending code of a re-ECN implementation will never need to include ECN Nonce support. Re-ECN is intended to provide wider protection than the ECN nonce against congestion control misbehaviour, and re-ECN only requires support from the sender, therefore it is preferable to specifically rule out the need for dual sender implementations. As a consequence, a re-ECN capable sender will never set ECT(0), so it will be easier for network elements to discriminate re-ECN traffic flows from other ECN traffic, which will always contain some ECT(0) packets.

However, a re-ECN implementation MAY OPTIONALLY include receiving code that complies with the ECN Nonce protocol when interacting with a sender that supports the ECN nonce (rather than re-ECN), but this support is not required.

RFC3540 allows an ECN nonce sender to choose whether to sanction a

receiver that does not ever set the nonce sum. Given re-ECN is intended to provide wider protection than the ECN nonce against congestion control misbehaviour, implementers of re-ECN receivers MAY choose not to implement backwards compatibility with the ECN nonce capability. This may be because they deem that the risk of sanctions is low, perhaps because significant deployment of the ECN nonce seems unlikely at implementation time.

6.1.3. Capability Negotiation

During the TCP hand-shake at the start of a connection, an originator of the connection (host A) with a re-ECN-capable transport MUST indicate it is Re-ECT by setting the TCP flags NS=1, CWR=1 and ECE=1 in the initial SYN.

A responding Re-ECT host (host B) MUST return a SYN ACK with flags CWR=1 and ECE=0. The responding host MUST NOT set this combination of flags unless the preceding SYN has already indicated Re-ECT support as above. Normally a Re-ECT server (B) will reply to a Re-ECT client with NS=0, but if the initial SYN from Re-ECT client A is marked CE(-1), a Re-ECT server B MUST increment its local value of ECC. But B cannot reflect the value of ECC in the SYN ACK, because it is still using the 3 bits to negotiate connection capabilities. So, server B MUST set the alternative TCP header flags in its SYN ACK: NS=1, CWR=1 and ECE=0.

These handshakes are summarised in Table 5 below, with X indicating NS can be either 1 or 0 depending respectively on whether congestion had been experienced or not. The handshakes used for the other flavours of ECN are also shown for comparison. To compress the width of the table, the headings of the first four columns have been severely abbreviated, as follows:

R: *R*e-ECT

N: ECT-*N*once (RFC3540)

E: *E*CT (RFC3168)

I: Not-ECT (*I*mplicit congestion notification).

These correspond with the same headings used in Table 4. Indeed, the resulting modes in the last two columns of the table below are a more comprehensive way of saying the same thing as Table 4.

R	N	E	I	SYN A-B			SYN ACK B-A			A-B Mode	B-A Mode
				NS	CWR	ECE	NS	CWR	ECE		
AB				1	1	1	X	1	0	RECN	RECN
A	B			1	1	1	1	0	1	RECN-Co	ECT-Nonce
A		B		1	1	1	0	0	1	RECN-Co	ECT
A			B	1	1	1	0	0	0	Not-ECT	Not-ECT
B	A			0	1	1	0	0	1	ECT-Nonce	RECN-Co
B		A		0	1	1	0	0	1	ECT	RECN-Co
B			A	0	0	0	0	0	0	Not-ECT	Not-ECT

Table 5: TCP Capability Negotiation between Originator (A) and Responder (B)

As soon as a re-ECN capable TCP server receives a SYN, it MUST set its two half-connections into the modes given in Table 5. As soon as a re-ECN capable TCP client receives a SYN ACK, it MUST set its two half-connections into the modes given in Table 5. The half-connections will remain in these modes for the rest of the connection, including for the third segment of TCP's three-way hand-shake (the ACK).

{ToDo: Consider delaying mode changes if using SYN cookies (will also affect next section).}

{ToDo: consider RSTs within a connection.}

Recall that, if the SYN ACK reflects the same flag settings as the preceding SYN (because there is a broken RFC3168 compliant implementation that behaves this way), RFC3168 specifies that the whole connection MUST revert to Not-ECT.

Also note that, whenever the SYN flag of a TCP segment is set (including when the ACK flag is also set), the NS, CWR and ECE flags (i.e the ECI field of the SYN-ACK) MUST NOT be interpreted as the 3-bit ECI value, which is only set as a copy of the local ECC value in non-SYN packets.

6.1.4. Extended ECN (EECN) Field Settings during Flow Start or after Idle Periods

If the originator (A) of a TCP connection supports re-ECN it MUST set the extended ECN (EECN) field in the IP header of the initial SYN packet to the feedback not established (FNE) codepoint.

FNE is a new extended ECN codepoint defined by this specification

(Section 4.2). The feedback not established (FNE) codepoint is used when the transport does not have the benefit of ECN feedback so it cannot decide whether to set or clear the RE flag.

If after receiving a SYN the server B has set its sending half-connection into RECN mode or RECN-Co mode, it MUST set the extended ECN field in the IP header of its SYN ACK to the feedback not established (FNE) codepoint. Note the careful wording here, which means that Re-ECT server B MUST set FNE on a SYN ACK whether it is responding to a SYN from a Re-ECT client or from a client that is merely ECN-capable. This is because FNE indicates the transport is ECN capable as well as re-ECN capable.

The original ECN specification [RFC3168] required SYNs and SYN ACKs to use the Not-ECT codepoint of the ECN field. The aim was to prevent well-known DoS attacks such as SYN flooding being able to gain from the advantage that ECN capability afforded over drop at ECN-capable routers.

For a SYN ACK, Kuzmanovic [RFC5562] has shown that this caution was unnecessary, and allows a SYN ACK to be ECN-capable to improve performance. By stipulating the FNE codepoint for the initial SYN, we comply with RFC3168 in word but not in spirit, because we have indeed set the ECN field to Not-ECT, but we have extended the ECN field with another bit. And it will be seen (Section 5.3) that we have defined one setting of that bit to mean an ECN-capable transport. Therefore, by proposing that the FNE codepoint MUST be used on the initial SYN of a connection, we have gone further by proposing to make the initial SYN ECN-capable too. Section 5.4 justifies deciding to make the initial SYN ECN-capable.

Once a TCP half connection is in RECN mode or RECN-Co mode, FNE will have already been set on the initial SYN and possibly the SYN ACK as above. But each re-ECN sender will have to set FNE cautiously on a few data packets as well, given a number of packets will usually have to be sent before sufficient congestion feedback is received. The behaviour will be different depending on the mode of the half-connection:

RECN mode: Given the constraints on TCP's initial window [RFC3390] and its exponential window increase during slow start phase [RFC5681], it turns out that the sender SHOULD set FNE on the first and third data packets in its flow after the initial 3-way handshake, assuming equal sized data packets once a flow is established. Appendix D presents the calculation that led to this conclusion. Below, after running through the start of an example TCP session, we give the intuition learned from that calculation. {ToDo: unfortunately the calculation was based on erroneous

assumptions; see [I-D.conex-tcp-mods] for a better approach.}

RECN-Co mode: A re-ECT sender that switches into re-ECN compatibility mode or into Not-ECT mode (because it has detected the corresponding host is not re-ECN capable) MUST limit its initial window to 1 segment. The reasoning behind this constraint is given in Section 5.4. Having set this initial window, a re-ECN sender in RECN-Co mode SHOULD set FNE on the first and third data packets in a flow, as for RECN mode.

	Data	TCP A(Re-ECT)	IP A	IP B	TCP B(Re-ECT)	Data
	Byte	SEQ ACK CTL	EECN	EECN	SEQ ACK CTL	Byte
--	----	----------	-----	-----	----------	----
1		0100 SYN	FNE	-->	R.ECC=0	
		CWR,ECE,NS				
2		R.ECC=0	<--	FNE	0300 0101	
					SYN,ACK,CWR	
3		0101 0301 ACK	RECT	-->	R.ECC=0	
4	1000	0101 0301 ACK	FNE	-->	R.ECC=0	
5		R.ECC=0	<--	FNE	0301 1102 ACK	1460
6		R.ECC=0	<--	RECT	1762 1102 ACK	1460
7		R.ECC=0	<--	FNE	3222 1102 ACK	1460
8		1102 1762 ACK	RECT	-->	R.ECC=0	
9		R.ECC=0	<--	RECT	4682 1102 ACK	1460
10		R.ECC=0	<--	RECT	6142 1102 ACK	1460
11		1102 3222 ACK	RECT	-->	R.ECC=0	
12		R.ECC=0	<--	RECT	7602 1102 ACK	1460
13		R.ECC=1	<*-	RECT	9062 1102 ACK	1460
		...				

Table 6: TCP Session Example #1

Table 6 shows an example TCP session, where the server B sets FNE on its first and third data packets (lines 5 & 7) as well as on the initial SYN ACK as previously described. The left hand half of the table shows the relevant settings of headers sent by client A in three layers: the TCP payload size; TCP settings; then IP settings. The right hand half gives equivalent columns for server B. The only TCP settings shown are the sequence number (SEQ), acknowledgement number (ACK) and the relevant control (CTL) flags that the relevant sending host sets in the TCP header. The IP columns show the setting of the extended ECN (EECN) field.

Also shown on the receiving side of the table is the value of the receiver's echo congestion counter (R.ECC) after processing the

incoming EECN header. Note that, once a host sets a half-connection into RECN mode, it MUST initialise its local value of ECC to zero.

The intuition that Appendix D gives for why a sender should set FNE on the first and third data packets is as follows. At line 13, a packet sent by B is shown with an '*', which means it has been congestion marked by an intermediate queue from RECT to CE(-1). On receiving this CE marked packet, client A increments its ECC counter to 1 as shown. This was the 7th data packet B sent, but before feedback about this event returns to B, it might well have sent many more packets. Indeed, during exponential slow start, about as many packets will be in flight (unacknowledged) as have been acknowledged. So, when the feedback from the congestion event on B's 7th segment returns, B will have sent about 7 further packets that will still be in flight. At that stage, B's best estimate of the network's packet marking fraction will be 1/7. So, as B will have sent about 14 packets, it should have already marked 2 of them as FNE in order to have marked 1/7; hence the need to have set the first and third data packets to FNE.

Client A's behaviour in Table 6 also shows FNE being set on the first SYN and the first data packet (lines 1 & 4), but in this case it sends no more data packets, so of course, it cannot, and does not need to, set FNE again. Note that in the A-B direction there is no need to set FNE on the third part of the three-way hand-shake (line 3---the ACK).

Note that in this section we have used the word SHOULD rather than MUST when specifying how to set FNE on data segments before positive congestion feedback arrives (but note that the word MUST was used for FNE on the SYN and SYN ACK). FNE is only RECOMMENDED for the first and third data segments to entertain the possibility that the TCP transport has the benefit of other knowledge of the path, which it re-uses from one flow for the benefit of a newly starting flow. For instance, one flow can re-use knowledge of other flows between the same hosts if using a Congestion Manager [RFC3124] or when a proxy host aggregates congestion information for large numbers of flows.

{ToDo: There is probably scope for re-writing the above in a different way so that it says MUST unless some other knowledge of the path is available. See earlier note pointing out FNE on 1st & 3rd is too few.}

After an idle period of more than 1 second, a re-ECN sender transport MUST set the EECN field of the packet that resumes the connection to FNE. Note that this next packet may be sent a very long time later, a packet does NOT have to be sent after 1 second of idling. In order that the design of network policers can be deterministic, this

specification deliberately puts an absolute lower limit on how long a connection can be idle before the packet that resumes the connection must be set to FNE, rather than relating it to the connection round trip time. We use the lower bound of the retransmission timeout (RTO) [RFC6298], which is commonly used as the idle period before TCP must reduce to the restart window [RFC5681]. Note our specification of re-ECN's idle period is NOT intended to change the idle period for TCP's restart, nor indeed for any other purposes.

{ToDo: Describe how the sender falls back to RFC3168 modes if packets don't appear to be getting through (to work round firewalls discarding packets they consider unusual).}

{ToDo: Possible future capabilities for changing Slow Start}

6.1.5. Pure ACKS, Retransmissions, Window Probes and Partial ACKs

A re-ECN sender MUST clear the RE flag to "0" and set the ECN field to Not-ECT in pure ACKs, retransmissions and window probes, as specified in [RFC3168]. Our eventual goal is for all packets to be sent with re-ECN enabled, and we believe the semantics of the ECI field go a long way towards being able to achieve this. However, we have not completed a full security analysis for these cases, therefore, currently we merely re-state current practice.

We must also reconcile the facts that congestion marking is applied to packets but acknowledgements cover octet ranges and acknowledged octet boundaries need not match the transmitted boundaries. The general principle we work to is to remain compatible with TCP's congestion control which is driven by congestion events at packet granularity while at the same time aiming to blank the RE flag on at least as many octets in a flow as have been marked CE.

Therefore, a re-ECN TCP receiver MUST increment its ECC value as many times as CE marked packets have been received. And that value MUST be echoed to the sender in the first available ACK using the ECI field. This ensures the TCP sender's congestion control receives timely feedback on congestion events at the same packet granularity that they were generated on congested queues.

Then, a re-ECN sender stores the difference D between its own ECC value and the incoming ECI field by incrementing a counter R. Then, R is decremented by 1 each subsequent packet that is sent with the RE flag blanked, until R is no longer positive. Using this technique, whenever a re-ECN transport sends a not re-ECN capable packet (e.g. a retransmission), the remaining packets required to have the RE flag blanked will be automatically carried over to subsequent packets, through the variable R.

This does not ensure precisely the same number of octets have RE blanked as were CE marked. But we believe positive errors will cancel negative over a long enough period. {ToDo: However, more research is needed to prove whether this is so. If it is not, it may be necessary to increment and decrement R in octets rather than packets, by incrementing R as the product of D and the size in octets of packets being sent (typically the MSS).}

6.2. Other Transports

6.2.1. General Guidelines for Adding Re-ECN to Other Transports

As a general rule, Re-ECT sender transports that have established the receiver transport is at least ECN-capable (not necessarily re-ECN capable) MUST blank the RE codepoint for at least as many octets as arrive at receiver with the CE codepoint set. Re-ECN-capable sender transports should always initialise the ECN field to the ECT(1) codepoint once a flow is established.

If the sender transport does not have sufficient feedback to even estimate the path's CE rate, it SHOULD set FNE continuously. If the sender transport has some, perhaps stale, feedback to estimate that the path's CE rate is nearly definitely less than E%, the transport MAY blank RE in packets for E% of sent octets, and set the RECT codepoint for the remainder.

The following sections give guidelines on how re-ECN support could be added to RSVP or NSIS, to DCCP, and to SCTP - although separate Internet drafts will be necessary to document the exact mechanics of re-ECN in each of these protocols.

{ToDo: Give a brief outline of what would be expected for each of the following:

o UDP fire and forget (e.g. DNS)

o UDP streaming with no feedback

o UDP streaming with feedback

}

6.2.2. Guidelines for adding Re-ECN to RSVP or NSIS

A separate I-D has been submitted [I-D.re-pcn-border-cheat] describing how re-ECN can be used in an edge-to-edge rather than end-to-end scenario. It can then be used by downstream networks to police whether upstream networks are blocking new flow reservations

when downstream congestion is too high, even though the congestion is in other operators' downstream networks. This relates to current IETF work on Admission Control over Diffserv using Pre-Congestion Notification (PCN) [RFC5559].

6.2.3. Guidelines for adding Re-ECN to DCCP

Beside adjusting the initial features negotiation sequence, operating re-ECN in DCCP [RFC4340] could be achieved by defining a new option to be added to acknowledgments, that would include a multibit field where the destination could copy its ECC.

6.2.4. Guidelines for adding Re-ECN to SCTP

Appendix A in [RFC4960] gives the specifications for SCTP to support ECN. Similar steps should be taken to support re-ECN. Beside adjusting the initial features negotiation sequence, operating re-ECN in SCTP could be achieved by defining a new control chunk, that would include a multibit field where the destination could copy its ECC

7. Incremental Deployment

The design of the re-ECN protocol started from the fact that the current ECN marking behaviour of queues was sufficient and that re-feedback could be introduced around these queues by changing the sender behaviour but not the routers. Otherwise, if we had required routers to be changed, the chance of encountering a path that had every router upgraded would be vanishingly small during early deployment, giving no incentive to start deployment. Also, as there is no new forwarding behaviour, routers and hosts do not have to signal or negotiate anything.

However, networks that choose to protect themselves using re-ECN do have to add new security functions at their trust boundaries with others. They distinguish legacy traffic by its ECN field. Traffic from Not-ECT transports is distinguishable by its Not-ECT marking. Traffic from RFC3168 compliant ECN transports is distinguished from re-ECN by which of ECT(0) or ECT(1) is used. We chose to use ECT(1) for re-ECN traffic deliberately. Existing ECN sources set ECT(0) on either 50% (the nonce) or 100% (the default) of packets, whereas re-ECN does not use ECT(0) at all. We can use this distinguishing feature of RFC3168 compliant ECN traffic to separate it out for different treatment at the various border security functions: egress dropping, ingress policing and border policing.

The general principle we adopt is that an egress dropper will not drop any legacy traffic, but ingress and border policers will limit the bulk rate of legacy traffic (Not-ECT, ECT(0) and those marked

with the unused codepoint) that can enter each network. Then, during early re-ECN deployment, operators can set very permissive (or non-existent) rate-limits on legacy traffic, but once re-ECN implementations are generally available, legacy traffic can be rate-limited increasingly harshly. Ultimately, an operator might choose to block all legacy traffic entering its network, or at least only allow through a trickle.

Then, as the limits are set more strictly, the more RFC3168 ECN sources will gain by upgrading to re-ECN. Thus, towards the end of the voluntary incremental deployment period, RFC3168 compliant transports can be given progressively stronger encouragement to upgrade.

The following list of minor changes, brings together all the points where re-ECN semantics for use of the two-bit ECN field are different compared to RFC3168:

- A re-ECN sender sets ECT(1) by default, whereas an RFC3168 sender sets ECT(0) by default (Section 4.3);

- No provision is necessary for a re-ECN capable source transport to use the ECN nonce (Section 6.1.2.1);

- Routers MAY preferentially drop different extended ECN codepoints (Section 5.3);

- Packets carrying the feedback not established (FNE) codepoint MAY optionally be marked rather than dropped by routers, even though their ECN field is Not-ECT (with the important caveat in Section 5.3);

- Packets may be dropped by policing nodes because of apparent misbehaviour, not just because of congestion ;

- Tunnel entry behaviour is still to be defined, but may have to be different from RFC3168 (Section 5.6).

None of these changes REQUIRE any modifications to routers. Also none of these changes affect anything about end to end congestion control; they are all to do with allowing networks to police that end to end congestion control is well-behaved.

8. Related Work

8.1. Congestion Notification Integrity

The choice of two ECT code-points in the ECN field [RFC3168] permitted future flexibility, optionally allowing the sender to encode the experimental ECN nonce [RFC3540] in the packet stream. This mechanism has since been included in the specifications of DCCP [RFC4340].

{ToDo: DCCP provides nonce support - how does this affect the RFC?}

The ECN nonce is an elegant scheme that allows the sender to detect if someone in the feedback loop - the receiver especially - tries to claim no congestion was experienced when in fact congestion led to packet drops or ECN marks. For each packet it sends, the sender chooses between the two ECT codepoints in a pseudo-random sequence. Then, whenever the network marks a packet with CE, if the receiver wants to deny congestion happened, she has to guess which ECT codepoint was overwritten. She has only a 50:50 chance of being correct each time she denies a congestion mark or a drop, which ultimately will give her away.

The purpose of a network-layer nonce should primarily be protection of the network, while a transport-layer nonce would be better used to protect the sender from cheating receivers. Now, the assumption behind the ECN nonce is that a sender will want to detect whether a receiver is suppressing congestion feedback. This is only true if the sender's interests are aligned with the network's, or with the community of users as a whole. This may be true for certain large senders, who are under close scrutiny and have a reputation to maintain. But we have to deal with a more hostile world, where traffic may be dominated by peer-to-peer transfers, rather than downloads from a few popular sites. Often the `natural' self-interest of a sender is not aligned with the interests of other users. It often wishes to transfer data quickly to the receiver as much as the receiver wants the data quickly.

In contrast, the re-ECN protocol enables policing of an agreed rate-response to congestion (e.g. TCP-friendliness) at the sender's interface with the internetwork. It also ensures downstream networks can police their upstream neighbours, to encourage them to police their users in turn. But most importantly, it requires the sender to declare path congestion to the network and it can remove traffic at the egress if this declaration is dishonest. So it can police correctly, irrespective of whether the receiver tries to suppress congestion feedback or whether the sender ignores genuine congestion feedback. Therefore the re-ECN protocol addresses a much wider range of cheating problems, which includes the one addressed by the ECN nonce.

{ToDo: Ensure we address the early ACK problem.}

9. Security Considerations

{ToDo: Describe attacks by networks on flows and by spoofing sources.} {ToDo: Re-ECN & DNS servers}

This whole memo concerns the deployment of a secure congestion control framework. However, below we list some specific security issues that we are still working on:

- Malicious users have ability to launch dynamically changing attacks, exploiting the time it takes to detect an attack, given ECN marking is binary. We are concentrating on subtle interactions between the ingress policer and the egress dropper in an effort to make it impossible to game the system.

- There is an inherent need for at least some flow state at the egress dropper given the binary marking environment, which leads to an apparent vulnerability to state exhaustion attacks. An egress dropper design with bounded flow state is in write-up.

- A malicious source can spoof another user's address and send negative traffic to the same destination in order to fool the dropper into sanctioning the other user's flow. To prevent or mitigate these two different kinds of DoS attack, against the dropper and against given flows, we are considering various protection mechanisms.

- A malicious client can send requests using a spoofed source address to a server (such as a DNS server) that tends to respond with single packet responses. This server will then be tricked into having to set FNE on the first (and only) packet of all these wasted responses. Given packets marked FNE are worth +1, this will cause such servers to consume more of their allowance to cause congestion than they would wish to. In general, re-ECN is deliberately designed so that single packet flows have to bear the cost of not discovering the congestion state of their path. One of the reasons for introducing re-ECN is to encourage short flows to make use of previous path knowledge by moving the cost of this lack of knowledge to sources that create short flows. Therefore, we in the long run we might expect services like DNS to aggregate single packet flows into connections where it brings benefits. However, this attack where DNS requests are made from spoofed addresses genuinely forces the server to waste its resources. The only mitigating feature is that the attacker has to set FNE on each of its requests if they are to get through an egress dropper to a DNS server. The attacker therefore has to consume as many

resources as the victim, which at least implies re-ECN does not unwittingly amplify this attack.

Having highlighted outstanding security issues, we now explain the design decisions that were taken based on a security-related rationale. It may seem that the six codepoints of the eight made available by extending the ECN field with the RE flag have been used rather wastefully to encode just five states. In effect the RE flag has been used as an orthogonal single bit, using up four codepoints to encode the three states of positive, neutral and negative worth. The mapping of the codepoints in an earlier version of this proposal used the codepoint space more efficiently, but the scheme became vulnerable to network operators bypassing congestion penalties by focusing congestion marking on positive packets. Appendix B explains why fixing that problem while allowing for incremental deployment, would have used another codepoint anyway. So it was better to use this orthogonal encoding scheme, which greatly simplified the whole protocol and brought with it some subtle security benefits (see the last paragraph of Appendix B).

With the scheme as now proposed, once the RE flag is set or cleared by the sender or its proxy, it should not be written by the network, only read. So the endpoints can detect if any network maliciously alters the RE flag. IPsec AH integrity checking does not cover the IPv4 option flags (they were considered mutable---even the one we propose using for the RE flag that was `currently unused' when IPsec was defined). But it would be sufficient for a pair of endpoints to make random checks on whether the RE flag was the same when it reached the egress as when it left the ingress. Indeed, if IPsec AH had covered the RE flag, any network intending to alter sufficient RE flags to make a gain would have focused its alterations on packets without authenticating headers (AHs).

The security of re-ECN has been deliberately designed to not rely on cryptography.

10. IANA Considerations

This memo includes no request to IANA (yet).

If this memo was to progress to standards track, it would list:

- The new RE flag in IPv4 (Section 5.1) and its extension with the ECN field to create a new set of extended ECN (EECN) codepoints;

- The definition of the EECN codepoints for default Diffserv PHBs (Section 4.2)

- The Hop-by-Hop option ID for the new extension header for IPv6 (Section 5.2);

- The new combinations of flags in the TCP header for capability negotiation (Section 6.1.3);

11. Conclusions

{ToDo:}

12. Acknowledgements

Sebastien Cazalet and Andrea Soppera contributed to the idea of re-feedback. All the following have given helpful comments: Andrea Soppera, David Songhurst, Peter Hovell, Louise Burness, Phil Eardley, Steve Rudkin, Marc Wennink, Fabrice Saffre, Cefn Hoile, Steve Wright, John Davey, Martin Koyabe, Carla Di Cairano-Gilfedder, Alexandru Murgu, Nigel Geffen, Pete Willis, John Adams (BT), Sally Floyd (ICIR), Joe Babiarz, Kwok Ho-Chan (Nortel), Stephen Hailes, Mark Handley (who developed the attack with canceled packets), Adam Greenhalgh (who developed the attack on DNS) (UCL), Jon Crowcroft (Uni Cam), David Clark, Bill Lehr, Sharon Gillett, Steve Bauer (who complemented our own dummy traffic attacks with others), Liz Maida (MIT), Meral Shirazipour (Ericsson) and comments from participants in the CRN/CFP Broadband and DoS-resistant Internet working groups.A special thank you to Alessandro Salvatori for coming up with fiendish attacks on re-ECN.

13. Comments Solicited

Comments and questions are encouraged and very welcome. They can be addressed to the IETF Congestion Exposure (ConEx) working group's mailing list <conex@ietf.org>, and/or to the authors.

14. References

14.1. Normative References

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

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

[RFC3390] Allman, M., Floyd, S., and C. Partridge,

"Increasing TCP's Initial Window", RFC 3390, October 2002.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, March 2006.

[RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, March 2006.

[RFC4835] Manral, V., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4835, April 2007.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

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

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

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

14.2. Informative References

[ARI05] Adams, J., Roberts, L., and A. IJsselmuiden, "Changing the Internet to

Support Real-Time Content Supply from a Large Fraction of Broadband Residential Users", BT Technology Journal (BTTJ) 23(2), April 2005.

[I-D.conex-tcp-mods] Kuehlewind, M. and R. Scheffenegger, "TCP modifications for Congestion Exposure", draft-ietf-conex-tcp-modifications-04 (work in progress), July 2013.

[I-D.re-ecn-motiv] Briscoe, B., Jacquet, A., Moncaster, T., and A. Smith, "Re-ECN: A Framework for adding Congestion Accountability to TCP/IP", draft-briscoe-conex-re-ecn-motiv-02 (work in progress), July 2013.

[I-D.re-pcn-border-cheat] Briscoe, B., "Emulating Border Flow Policing using Re-PCN on Bulk Data", draft-briscoe-re-pcn-border-cheat-03 (work in progress), October 2009.

[RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.

[RFC3514] Bellovin, S., "The Security Flag in the IPv4 Header", RFC 3514, April 2003.

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

[RFC4301] Kent, S. and K. Seo, "Security

Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, January 2008.

[RFC5559] Eardley, P., "Pre-Congestion Notification (PCN) Architecture", RFC 5559, June 2009.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, June 2011.

[Re-fb] Briscoe, B., Jacquet, A., Di Cairano-Gilfedder, C., Salvatori, A., Soppera, A., and M. Koyabe, "Policing Congestion Response in an Internetwork Using Re-Feedback", ACM SIGCOMM CCR 35(4)277--288, August 2005, <http://www.acm.org/sigs/sigcomm/sigcomm2005/techprog.html#session8>.

[Savage99] Savage, S., Cardwell, N., Wetherall, D., and T. Anderson, "TCP congestion control with a misbehaving receiver", ACM SIGCOMM CCR 29(5), October 1999, <http://citeseer.ist.psu.edu/savage99tcp.html>.

[Steps_DoS] Handley, M. and A. Greenhalgh, "Steps towards a DoS-resistant Internet Architecture", Proc. ACM SIGCOMM workshop on Future directions in network architecture (FDNA'04) pp 49--56, August 2004.

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

Appendix A. Precise Re-ECN Protocol Operation

The protocol operation in Section 4.3 was described as an approximation. In fact, standard ECN marking at a queue combines 1% and 2% marking into slightly less than 3% whole-path marking, because queues deliberately mark CE whether or not it has already been marked

by another queue upstream. So the combined marking fraction would actually be 100% - (100% - 1%)(100% - 2%) = 2.98%.

To generalise this we will need some notation.

- j represents the index of each resource (typically queues) along a path, ranging from 0 at the first queue to n-1 at the last.

- m_j represents the fraction of octets to be *m*arked CE by a particular queue (whether or not they are already marked) because of congestion of resource j.

- u_j represents congestion signals arriving from *u*pstream of resource j, being the fraction of CE marking in arriving packet headers (before marking).

- p_j represents *p*ath congestion, being the fraction of packets arriving at resource j with the RE flag blanked (excluding Not-RECT packets).

- v_j denotes expected congestion downstream of resource j, which can be thought of as a *v*irtual marking fraction, being derived from two other marking fractions.

Observed fractions of each particular codepoint (u, p and v) and queue marking rate m are dimensionless fractions, being the ratio of two data volumes (marked and total) over a monitoring period. All measurements are in terms of octets, not packets, assuming that line resources are more congestible than packet processing.

The path congestion (RE blanking fraction) set by the sender should reflect upstream congestion (CE marking fraction) from the viewpoint of the destination, which it feeds back to the sender. Therefore in the steady state

$$p_0 = u_n = 1 - (1 - m_1)(1 - m_2)...$$

Similarly, at some point j in the middle of the network, given $p = 1 - (1 - u_j)(1 - v_j)$, then

$$v_j = 1 - (1 - p)/(1 - u_j)$$

$$\sim= p - u_j; \quad \text{if } u_j << 100\%$$

So, between the two routers in the example in Section 4.3, congestion downstream is

```
v_1  = 100.00% - (100% - 2.98%) / (100% - 1.00%)
     = 2.00%,
```

or a useful approximation of downstream congestion is

```
v_1 ~= 2.98% - 1.00%
    ~= 1.98%.
```

Appendix B. Justification for Two Codepoints Signifying Zero Worth Packets

It may seem a waste of a codepoint to set aside two codepoints of the Extended ECN field to signify zero worth (RECT and CE(0) are both worth zero). The justification is subtle, but worth recording.

The original version of Re-ECN ([Re-fb] and draft-00 of this memo) used three codepoints for neutral (ECT(1)), positive (ECT(0)) and negative (CE) packets. The sender set packets to neutral unless re-echoing congestion, when it set them positive, in much the same way that it blanks the RE flag in the current protocol. However, routers were meant to mark congestion by setting packets negative (CE) irrespective of whether they had previously been neutral or positive.

However, we did not arrange for senders to remember which packet had been sent with which codepoint, or for feedback to say exactly which packets arrived with which codepoints. The transport was meant to inflate the number of positive packets it sent to allow for a few being wiped out by congestion marking. We (wrongly) assumed that routers would congestion mark packets indiscriminately, so the transport could infer how many positive packets had been marked and compensate accordingly by re-echoing. But this created a perverse incentive for routers to preferentially congestion mark positive packets rather than neutral ones.

We could have removed this perverse incentive by requiring Re-ECN senders to remember which packets they had sent with which codepoint. And for feedback from the receiver to identify which packets arrived as which. Then, if a positive packet was congestion marked to negative, the sender could have re-echoed twice to maintain the balance between positive and negative at the receiver.

Instead, we chose to make re-echoing congestion (blanking RE) orthogonal to congestion notification (marking CE), which required a second neutral codepoint. Then the receiver would be able to detect and echo a congestion event even if it arrived on a packet that had originally been positive.

If we had added extra complexity to the sender and receiver

transports to track changes to individual packets, we could have made it work, but then routers would have had an incentive to mark positive packets with half the probability of neutral packets. That in turn would have led router algorithms to become more complex. Then senders wouldn't know whether a mark had been introduced by a simple or a complex router algorithm. That in turn would have required another codepoint to distinguish between RFC3168 ECN and new Re-ECN router marking.

Once the cost of IP header codepoint real-estate was the same for both schemes, there was no doubt that the simpler option for endpoints and for routers should be chosen. The resulting protocol also no longer needed the tricky inflation/deflation complexity of the original (broken) scheme. It was also much simpler to understand conceptually.

A further advantage of the new orthogonal four-codepoint scheme was that senders owned sole rights to change the RE flag and routers owned sole rights to change the ECN field. Although we still arrange the incentives so neither party strays outside their dominion, these clear lines of authority simplify the matter.

Finally, a little redundancy can be very powerful in a scheme such as this. In one flow, the proportion of packets changed to CE should be the same as the proportion of RECT packets changed to CE(-1) and the proportion of Re-Echo packets changed to CE(0). Double checking using such redundant relationships can improve the security of a scheme (cf. double-entry book-keeping or the ECN Nonce). Alternatively, it might be necessary to exploit the redundancy in the future to encode an extra information channel.

Appendix C. ECN Compatibility

The rationale for choosing the particular combinations of SYN and SYN ACK flags in Section 6.1.3 is as follows.

Choice of SYN flags: A Re-ECN sender can work with RFC3168 compliant ECN receivers so we wanted to use the same flags as would be used in an ECN-setup SYN [RFC3168] (CWR=1, ECE=1). But at the same time, we wanted a server (host B) that is Re-ECT to be able to recognise that the client (A) is also Re-ECT. We believe also setting NS=1 in the initial SYN achieves both these objectives, as it should be ignored by RFC3168 compliant ECT receivers and by ECT-Nonce receivers. But senders that are not Re-ECT should not set NS=1. At the time ECN was defined, the NS flag was not defined, so setting NS=1 should be ignored by existing ECT receivers (but testing against implementations may yet prove otherwise). The ECN Nonce RFC [RFC3540] is silent on what the NS

field might be set to in the TCP SYN, but we believe the intent was for a nonce client to set NS=0 in the initial SYN (again only testing will tell). Therefore we define a Re-ECN-setup SYN as one with NS=1, CWR=1 & ECE=1

Choice of SYN ACK flags: Choice of SYN ACK: The client (A) needs to be able to determine whether the server (B) is Re-ECT. The original ECN specification required an ECT server to respond to an ECN-setup SYN with an ECN-setup SYN ACK of CWR=0 and ECE=1. There is no room to modify this by setting the NS flag, as that is already set in the SYN ACK of an ECT-Nonce server. So we used the only combination of CWR and ECE that would not be used by existing TCP receivers: CWR=1 and ECE=0. The original ECN specification defines this combination as a non-ECN-setup SYN ACK, which remains true for RFC3168 compliant and Nonce ECTs. But for Re-ECN we define it as a Re-ECN-setup SYN ACK. We didn't use a SYN ACK with both CWR and ECE cleared to 0 because that would be the likely response from most Not-ECT receivers. And we didn't use a SYN ACK with both CWR and ECE set to 1 either, as at least one broken receiver implementation echoes whatever flags were in the SYN into its SYN ACK. Therefore we define a Re-ECN-setup SYN ACK as one with CWR=1 & ECE=0.

Choice of two alternative SYN ACKs: the NS flag may take either value in a Re-ECN-setup SYN ACK. Section 5.4 REQUIRES that a Re-ECT server MUST set the NS flag to 1 in a Re-ECN-setup SYN ACK to echo congestion experienced (CE) on the initial SYN. Otherwise a Re-ECN-setup SYN ACK MUST be returned with NS=0. The only current known use of the NS flag in a SYN ACK is to indicate support for the ECN nonce, which will be negotiated by setting CWR=0 & ECE=1. Given the ECN nonce MUST NOT be used for a RECN mode connection, a Re-ECN-setup SYN ACK can use either setting of the NS flag without any risk of confusion, because the CWR & ECE flags will be reversed relative to those used by an ECN nonce SYN ACK.

{ToDo: include the text below, either here, or in the algorithm sections} At an egress dropper, well-behaved RFC3168 compliant flows will appear to consist mostly of ECT(0) packets, with a few CE(0) packet. And, if the legacy source is setting the ECN nonce, the majority of packets will be an equal mix of ECT(0) and ECT(1) packets (the latter appearing to be Re-Echo packets in Re-ECN terms). None of these three packet markings is negative, so an egress dropper can handle all legacy flows in bulk and, as long as they don't send any packets using Re-ECN markings, it need not drop any legacy packets. So, as soon as an ECT(0) packet is seen, its flow ID can be added to the set of known legacy flows (a single Bloom filter would suffice). But, if any packets in flows classified as RFC3168 compliant are marked with any other marking than the three expected, the flow can

be removed from the RFC3168 set, to be treated in bulk with mis-behaving Re-ECN flows---the remainder of flow IDs that require no flow state to be held.

To an ingress Re-ECN policer, legacy ECN flows will appear as very highly congested paths. When policers are first deployed they can be configured permissively, allowing through both `RFC3168' ECN and misbehaving Re-ECN flows. Then, as the threshold is set more strictly, the more RFC3168 ECN sources will gain by upgrading to Re-ECN. Thus, towards the end of the voluntary incremental deployment period, RFC3168 transports can be given progressively stronger encouragement to upgrade.

Appendix D. Packet Marking with FNE During Flow Start

FNE (feedback not established) packets have two functions. Their main role is to announce the start of a new flow when feedback has not yet been established. However they also have the role of balancing the expected feedback and can be used where there are sudden changes in the rate of transmission. Whilst this should not happen under TCP their use as speculative marking is used in building the following argument as to why the first and third packets should be set to FNE.

The proportion of FNE packets in each round-trip should be a high estimate of the potential error in the balance of number of congestion marked packets versus number of re-echo packets already issued.

Let's call:

S: the number of the TCP segments sent so far

F: the number of FNE packets sent so far

R: the number of Re-Echo packets sent so far

A: the number of acknowledgments received so far

C: the number of acknowledgments echoing a CE packet

In normal operation, when we want to send packet S+1, we first need to check that enough Re-Echo packets have been issued:

If R<C, then S+1 will be a Re-echo packet

Next we need to estimate the amount of congestion observed so far. If congestion was stationary, it could be estimated as C/A. A

pessimistic bound is (C+1)/(A+1) which assumes that the next acknowledgment will echo a CE packet; we'll use that more pessimistic estimate to drive the generation of FNE packets.

The number of CE packets expected when (S+1) will be acknowledged is therefore (S+1)*(C+1)/(A+1). Packet S+1 should be set to FNE if that expected value exceeds the sum of FNE and Re-Echo packets sent so far.

```
If  (F+R)<(S+1)*(C+1)/(A+1),
  then S+1 will be set to FNE
  else S+1 will be set to RECT
```

So the full test should be:

```
When packet (S+1) is about to be sent...
  If R<C,
     then S+1 will be set to Re-Echo
  Else if  (F+R)<(S+1)*(C+1)/(A+1),
    then S+1 will be set to FNE
  Else S+1 will be set to RECT
```

This means that at any point, given A, R, F, C, the source could send another k RECT packets, so that k < (F+R)*(A+1)/(C+1)-S

The above scheme is independent of the actions of both the dropper and policer and doesn't depend on the rate adaptation discipline of the source. It only defines Re-Echo packets as notification of effective end-to-end congestion (as witnessed at the previous round-trip), and FNE packets as notification of speculative end-to-end congestion based on a high estimate of congestion

In practice, for any source:

- for the first packet, A=R=F=C=S=0 ==> 1 FNE
- if the acknowledgment doesn't echo a mark
  - for the second packet, A=F=S=1 R=C=0 ==> 1 RECT
  - for the third packet, S=2 A=F=1 R=C=0 ==> 1 FNE
- if no acknowledgement for these two packets echoes a congestion mark, then {A=S=3 F=2 R=C=0} which gives k<2*4/1-3, so the source
- if no acknowledgement for these four packets echoes a congestion mark, then {A=S=7 F=2 R=C=0} which gives k<2*8/1-7, so the source could send another 8 RECT packets. ==> 8 RECT

This behaviour happens to match TCP's congestion window control in slow start, which is why for TCP sources, only the first and third packet need be FNE packets.

A source that would open the congestion window any quicker would have to insert more FNE packets. As another example a UDP source sending VBR traffic might need to send several FNE packets ahead of the traffic peaks it generates.

Appendix E. Argument for holding back the ECN nonce

The ECN nonce is a mechanism that allows a /sending/ transport to detect if drop or ECN marking at a congested router has been suppressed by a node somewhere in the feedback loop---another router or the receiver.

Space for the ECN nonce was set aside in [RFC3168] (currently proposed standard) while the full nonce mechanism is specified in [RFC3540] (currently experimental). The specifications for [RFC4340] (currently proposed standard) requires that "Each DCCP sender SHOULD set ECN Nonces on its packets...". It also mandates as a requirement for all CCID profiles that "Any newly defined acknowledgement mechanism MUST include a way to transmit ECN Nonce Echoes back to the sender.", therefore:

- The CCID profile for TCP-like Congestion Control [RFC4341] (currently proposed standard) says "The sender will use the ECN Nonce for data packets, and the receiver will echo those nonces in its Ack Vectors."

- The CCID profile for TCP-Friendly Rate Control (TFRC) [RFC4342] recommends that "The sender [use] Loss Intervals options' ECN Nonce Echoes (and possibly any Ack Vectors' ECN Nonce Echoes) to probabilistically verify that the receiver is correctly reporting all dropped or marked packets."

The primary function of the ECN nonce is to protect the integrity of the information about congestion: ECN marks and packet drops. However, when the nonce is used to protect the integrity of information about packet drops, rather than ECN marks, a transport layer nonce will always be sufficient (because a drop loses the transport header as well as the ECN field in the network header), which would avoid using scarce IP header codepoint space. Similarly, a transport layer nonce would protect against a receiver sending early acknowledgements [Savage99].

If the ECN nonce reveals integrity problems with the information about congestion, the sending transport can use that knowledge for

two functions:

- to protect its own resources, by allocating them in proportion to the rates that each network path can sustain, based on congestion control,

- and to protect congested routers in the network, by slowing down drastically its connection to the destination with corrupt congestion information.

If the sending transport chooses to act in the interests of congested routers, it can reduce its rate if it detects some malicious party in the feedback loop may be suppressing ECN feedback. But it would only be useful to congested routers when /all/ senders using them are trusted to act in interest of the congested routers.

In the end, the only essential use of a network layer nonce is when sending transports (e.g. large servers) want to allocate their /own/ resources in proportion to the rates that each network path can sustain, based on congestion control. In that case, the nonce allows senders to be assured that they aren't being duped into giving more of their own resources to a particular flow. And if congestion suppression is detected, the sending transport can rate limit the offending connection to protect its own resources. Certainly, this is a useful function, but the IETF should carefully decide whether such a single, very specific case warrants IP header space.

In contrast, Re-ECN allows all routers to fully protect themselves from such attacks, without having to trust anyone - senders, receivers, neighbouring networks. Re-ECN is therefore proposed in preference to the ECN nonce on the basis that it addresses the generic problem of accountability for congestion of a network's resources at the IP layer.

Delaying the ECN nonce is justified because the applicability of the ECN nonce seems too limited for it to consume a two-bit codepoint in the IP header. It therefore seems prudent to give time for an alternative way to be found to do the one function the nonce is essential for.

Moreover, while we have re-designed the Re-ECN codepoints so that they do not prevent the ECN nonce progressing, the same is not true the other way round. If the ECN nonce started to see some deployment (perhaps because it was blessed with proposed standard status), incremental deployment of Re-ECN would effectively be impossible, because Re-ECN marking fractions at inter-domain borders would be polluted by unknown levels of nonce traffic.

The authors are aware that Re-ECN must prove it has the potential it claims if it is to displace the nonce. Therefore, every effort has been made to complete a comprehensive specification of Re-ECN so that its potential can be assessed. We therefore seek the opinion of the Internet community on whether the Re-ECN protocol is sufficiently useful to warrant standards action.

Appendix F. Alternative Terminology Used in Other Documents

A number of alternative terms have been used in various documents describing re-feedback and re-ECN. These are set out in the following table

Current Terminology	EECN codepoint	Colour
Cautious	FNE	Green
Positive	Re-Echo	Black
Neutral	RECT	Grey
Negative	CE(-1)	Red
Cancelled	CE(0)	Red-Black
Legacy ECN	ECT(0)	White
Currently Unused	--CU--	Currently unused
Legacy	Not-ECT	White

Table 7: Alternative re-ECN Terminology

Authors' Addresses

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

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

Arnaud Jacquet
BT
B54/70, Adastral Park
Martlesham Heath
Ipswich IP5 3RE
UK

Phone: +44 1473 647284
EMail: arnaud.jacquet@bt.com
URI:

Toby Moncaster
Moncaster.com
Dukes
Layer Marney
Colchester CO5 9UZ
UK

EMail: toby@moncaster.com

Alan Smith
BT
B54/76, Adastral Park
Martlesham Heath
Ipswich IP5 3RE
UK

Phone: +44 1473 640404
EMail: alan.p.smith@bt.com