Transport Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 13, 2008

B. Briscoe
BT & UCL
A. Jacquet
T. Moncaster
A. Smith
BT

January 10, 2008

Re-ECN: Adding Accountability for Causing Congestion to TCP/IP
draft-briscoe-tsvwg-re-ecn-tcp-05

Status of this Memo

Copyright Notice

Abstract

This document introduces a new protocol for explicit congestion notification (ECN), termed re-ECN, which can be deployed incrementally around unmodified routers. The protocol arranges 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. Then the upstream party at any trust boundary in the internetwork can be held responsible for the congestion they cause, or allow to be caused. So, networks can introduce straightforward accountability and policing mechanisms for incoming traffic from end-customers or from neighbouring network domains. 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 also gives examples of mechanisms that can use the protocol to ensure data sources respond correctly to congestion. And it describes example mechanisms that ensure the dominant selfish strategy of both network domains and end-points will be to set the extended ECN field honestly.

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

Although the re-ECN protocol is intended to make a simple but far-reaching change to the Internet architecture, the most immediate priority for the authors is to delay any move of the ECN nonce to Proposed Standard status. The argument for this position is developed in Appendix I.

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

Full diffs created using the rfcdiff tool are available at <http://www.cs.ucl.ac.uk/staff/B.Briscoe/pubs.html#retcp>

From -04 to -05 (current version):

- Completed justification for packet marking with FNE during slow-start(Appendix D).

- Minor editorial changes throughout.

From -03 to -04:

- Clarified reasons for holding back ECN nonce (Section 3.2 & Appendix I).

- Clarified Figure 1.

- Added Section 4.1.1.1 on equivalence of drops and ECN marks.

- Improved precision of Section 5.6 on IP in IP tunnels.

Explained the RTT fairness is possible to enforce, but unlikely to be required (Section 6.1.3 & Appendix F).

Explained that bulk per-user policing should be adequate but per-flow policing is also possible if desired, though it is not likely to be necessary (Section 6.1.5 & Appendix G).

Reinforced need for passive policing at inter-domain borders to enable all-optical networking (Section 6.1.6).

Minor editorial changes throughout.

From -02 to -03:

Started guidelines for re-ECN support in DCCP and SCTP.

Added annex on limitations of nonce mechanism.

Minor editorial changes throughout.

From -01 to -02:

Explanation on informal terminology in Section 3.4 clarified.

IPv6 wire protocol encoding added (Section 5.2).

Text on (non-)issues with tunnels, encryption and link layer congestion notification added (Section 5.6 & Section 5.7).

Section added giving evolvability arguments against encouraging bottleneck policing (Section 6.1.2). And text on re-ECN's evolvability by design added to Section 6.1.3

Text on inter-domain policing (Section 6.1.6) and inter-domain fail-safes (Section 6.1.7) added.

From -00 to -01:

Encoding of re-ECN wire protocol changed for reasons given in Appendix B and consequently draft substantially re-written.

Substantial text added in sections on applications, incremental deployment, architectural rationale and security considerations.

Table of Contents

1. Introduction

This document aims:

- To provide a complete specification of 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;

- To show how a number of hard problems become much easier to solve once re-ECN is available in IP.

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.

Uniquely, re-ECN manages to enable solutions to these problems without unduly stifling innovative new ways to use the Internet. This was a hard balance to strike, given it could be argued that DDoS is an innovative way to use the Internet. The most valuable insight was to allow each network to choose the level of constraint it wishes to impose. Also re-ECN has been carefully designed so that networks that choose to use it conservatively can protect themselves against the congestion caused in their network by users on other networks with more liberal policies.

For instance, some network owners want to block applications like voice and video unless their network is compensated for the extra share of bottleneck bandwidth taken. These real-time applications tend to be unresponsive when congestion arises. Whereas elastic TCP-based applications back away quickly, ending up taking a much smaller share of congested capacity for themselves. Other network owners want to invest in large amounts of capacity and make their gains from simplicity of operation and economies of scale.

Re-ECN allows the more conservative networks to police out flows that have not asked to be unresponsive to congestion---not because they are voice or video---just because they don't respond to congestion. But it also allows other networks to choose not to police. Crucially, when flows from liberal networks cross into a conservative network, re-ECN enables the conservative network to apply penalties to its neighbouring networks for the congestion they allow to be caused. And these penalties can be applied to bulk data, without

regard to flows.

Then, if unresponsive applications become so dominant that some of the more liberal networks experience congestion collapse [RFC3714], they can change their minds and use re-ECN to apply tighter controls in order to bring congestion back under control.

Re-ECN works by arranging that each packet arrives at each network element carrying a view of expected congestion on its own downstream path, albeit averaged over multiple packets. Most usefully, congestion on the remainder of the path becomes visible in the IP header at the first ingress. Many of the applications of re-ECN involve a policer at this ingress using the view of downstream congestion arriving in packets to police or control the packet rate.

Importantly, the scheme is recursive: a whole network harbouring users causing congestion in downstream networks can be held responsible or policed by its downstream neighbour.

This document is structured as follows. First an overview of the re-ECN protocol is given (Section 3), outlining its attributes and explaining conceptually how it works as a whole. The two main parts of the document follow, as described above. That is, the protocol specification divided into transport (Section 4) and network (Section 5) layers, then the applications it can be put to, such as policing DDoS, QoS and congestion control (Section 6). Although these applications do not require standardisation themselves, they are described in a fair degree of detail in order to explain how re-ECN can be used. Given re-ECN proposes to use the last undefined bit in the IPv4 header, we felt it necessary to outline the potential that re-ECN could release in return for being given that bit.

Deployment issues discussed throughout the document are brought together in Section 7, which is followed by a brief section explaining the somewhat subtle rationale for the design from an architectural perspective (Section 8). We end by describing related work (Section 9), listing security considerations (Section 10) and finally drawing conclusions (Section 12).

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].

This document first specifies a protocol, then describes a framework that creates the right incentives to ensure compliance to the

protocol. This could cause confusion because the second part of the document considers many cases where malicious nodes may not comply with the protocol. When such contingencies are described, if any of the above keywords are not capitalised, that is deliberate. So, for instance, the following two apparently contradictory sentences would be perfectly consistent: i) x MUST do this; ii) x may not do this.

3. Protocol Overview

3.1. Background and Applicability

First 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 routers probabilistically set "11" if congestion is experienced (CE), the marking probability increasing 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 routers on a path, a flow of packets accumulates the fraction of CE marking that each router adds. The combined effect of the packet marking of all the routers along the path signals congestion of the whole path to the receiver. So, for example, if one router early in a path is marking 1% of packets and another later in a path is marking 2%, flows that pass through both routers 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 9.2 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 routers 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, changes to IP routers are RECOMMENDED 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 7.1 summarises the incremental deployment strategy.

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 only concerned with enabling the ingress network to police that a source is complying with a congestion control algorithm, which is orthogonal to congestion control itself.

Before re-ECN can be considered worthy of using up the last bit in the IP header, we must be sure that all our claims are robust. We have gradually been reducing the list of outstanding issues, but the few that still remain are listed in Section 6.3. We expect new attacks may still be found, but we offer the re-ECN protocol on the basis that it is built on fairly solid theoretical foundations and, so far, it has proved possible to keep it relatively robust.

3.2. 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 7.1). 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. Until then it will suffice to use an abstraction of the IPv4 and v6 wire protocols by just calling it the RE flag.

Unlike the ECN field, the RE flag is intended to be set by the sender and 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. [Re-PCN]).

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) making 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.

RFC3168 ECN defines uses for all four codepoints of the two-bit ECN field. This memo widens the codepoint space to eight, and uses six codepoints. One of re-ECN's codepoints is an alternative use of the codepoint set aside in RFC3168 for the ECN nonce (ECT(1)). Transports not using re-ECN can still use the ECN nonce, while those using re-ECN do not need to as long as the sender is also checking for transport protocol compliance [I-D.moncaster-tcpm-rcv-cheat]. The case for doing this is given in Appendix I. 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 and available for future use.

ECN field	RFC3168 codepoint	RE flag	Extended ECN codepoint	Re-ECN meaning
00	Not-ECT	0	Not-RECT	Not re-ECN-capable transport
00	Not-ECT	1	FNE	Feedback not established
01	ECT(1)	0	Re-Echo	Re-echoed congestion and RECT
01	ECT(1)	1	RECT	Re-ECN capable transport
10	ECT(0)	0	---	Legacy ECN use only
10	ECT(0)	1	--CU--	Currently unused
11	CE	0	CE(0)	Re-Echo canceled by congestion experienced
11	CE	1	CE(-1)	Congestion experienced

Table 1: Extended ECN Codepoints

3.3. 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.

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 sets the feedback not established (FNE) codepoint. The network reads the FNE codepoint conservatively as equivalent to re-echoed congestion.

Specifically, once a flow is established, a re-ECN sender always initialises the ECN field to ECT(1). And it usually sets the RE flag to "1". Whenever a router re-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.

We chose to set and clear the RE flag this way round to ease incremental deployment (see Section 7.1). 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".

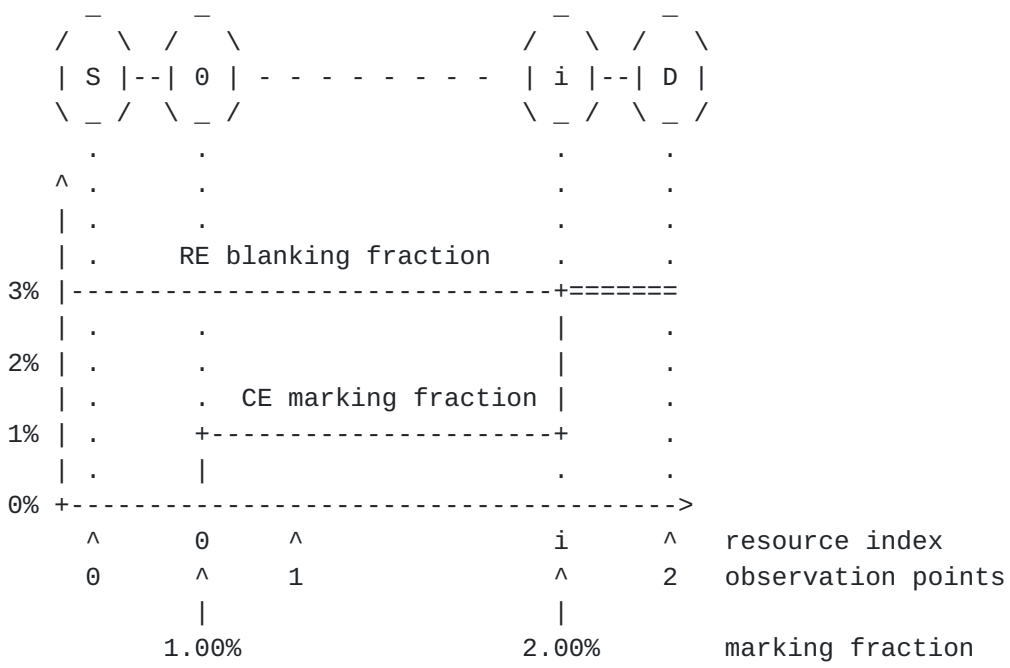

Figure 1: A 2-Router Example (Imprecise)

Figure 1 uses a simple network to illustrate how re-ECN allows routers to measure downstream congestion. The horizontal axis represents the index of each congestible resource (typically queues) along a path through the Internet. There may be many routers on the path, but we assume only two are currently congested (those with resource index 0 and i). The two superimposed plots show the fraction of each extended ECN codepoint in a flow observed along this path. Given about 3% of packets reaching the destination are marked CE, in response to feedback the sender will blank the RE flag in about 3% of packets it sends. Then 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).

Observation point	Approx downstream congestion
0	3% - 0% = 3%
1	3% - 1% = 2%
2	3% - 3% = 0%

Table 2: Downstream Congestion Measured at Example Observation Points

All along the path, whole-path congestion 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. To be absolutely clear, these fractions are averages that would result from the behaviour of a TCP protocol handler mechanically blanking outgoing packets in direct response to incoming feedback---we are not saying any protocol handler works with these average fractions directly.

3.4. Informal Terminology

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. 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).

Just as we will loosely talk of positive and negative flows, we will also talk of positive or negative packets, meaning packets that contribute positively or negatively to the downstream congestion metric.

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.

Figure 2 shows the main state transitions of the system once a flow is established, showing the worth of packets in each state. When the network congestion marks a packet it decrements its worth (moving from the left of the main square to the right). When the sender blanks the RE flag in order to re-echo congestion it increments the worth of a packet (moving from the bottom of the main square to the top).

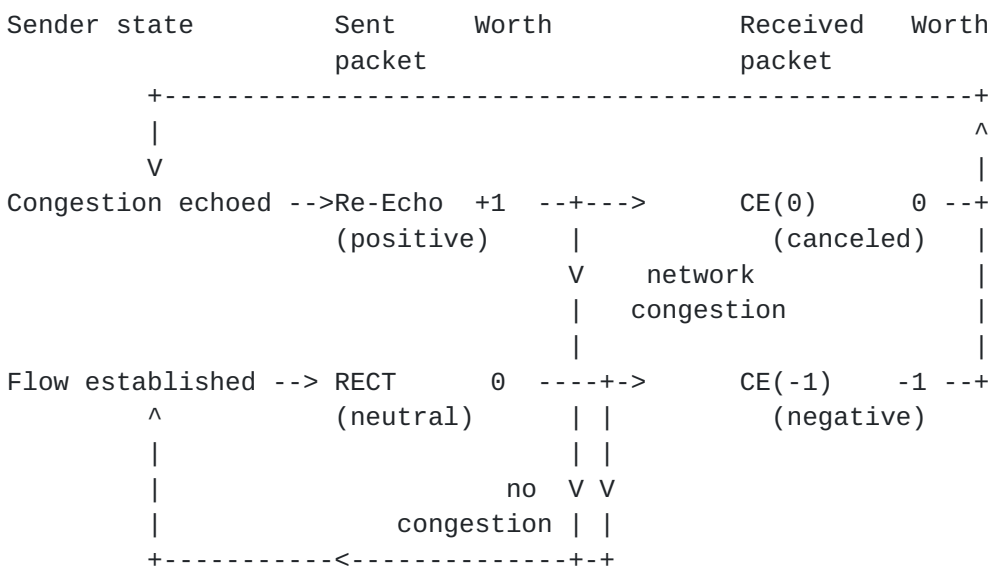

Figure 2: Re-ECN System State Diagram (bootstrap not shown)

The idea is that 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, as we shall see. The informal outline below uses TCP as an example transport, but the idea would be broadly similar for any transport that adapts its rate to congestion.

We will start with the sender in `flow established' state. Normally, as acknowledgements of earlier packets arrive that don't feedback any congestion, the congestion window can be opened, so the sender goes round the smaller sub-loop, sending RECT packets (worth 0) and returning to the flow established state to send another one. If a router congestion marks one of the packets, it decrements the packet's worth. The sender will have been continuing to traverse round the smaller feedback loop every time acknowledgements arrive. But when congestion feedback returns from this packet that was marked with -1 worth (the largest loop in the figure) the sender jumps to the congestion echoed state in order to re-echo the congestion, incrementing the worth of the next packet to +1 by blanking its RE flag. The sender then returns to the flow established state and continues round the smaller loop, sending packets worth 0. Note that the size of the loops is just an artefact of the figure; it is not meant to imply that one loop is slower than the other - they are both the same end to end feedback loop.

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 top sub-loop). When we need to distinguish, we will sometimes call a packet marked RECT 'neutral' (0 worth), while we will call the CE(0) marking 'canceled' (also 0 worth). If a re-echoed packet isn't unlucky enough to be further congestion marked, the sender will return to the flow established state and continue to send RECT packets (worth 0).

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. The FNE codepoint is an exception. It is used in the flow bootstrap process (explained later) and has the same positive (+1) worth as a packet with the Re-Echo codepoint.

ECN field	RE bit	Extended ECN codepoint	Worth	Re-ECN meaning
00	0	Not-RECT	...	Not re-ECN-capable transport
01	0	Re-Echo	+1	Re-echoed congestion and RECT
10	0	---	...	Legacy ECN use only
11	0	CE(0)	0	Re-Echo canceled by congestion experienced
00	1	FNE	+1	Feedback not established
01	1	RECT	0	Re-ECN capable transport
10	1	--CU--	...	Currently unused
11	1	CE(-1)	-1	Congestion experienced

Table 3: 'Worth' of Extended ECN Codepoints

4. Transport Layers

4.1. TCP

Re-ECN capability at the sender is essential. At the receiver it is optional, as long as the receiver has a basic (`vanilla flavour')

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 vanilla [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.

4.1.1. 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, modulo 8,

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 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 4. The actual definition of the TCP header, including the addition of support for the ECN nonce, is shown for comparison in Figure 3. 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.

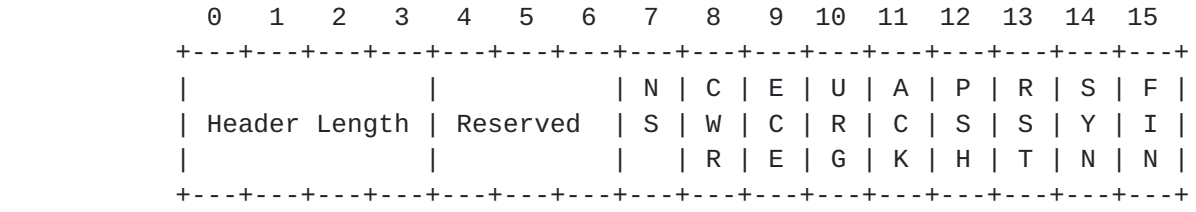

Figure 3: The (post-ECN Nonce) definition of bytes 13 and 14 of the TCP Header

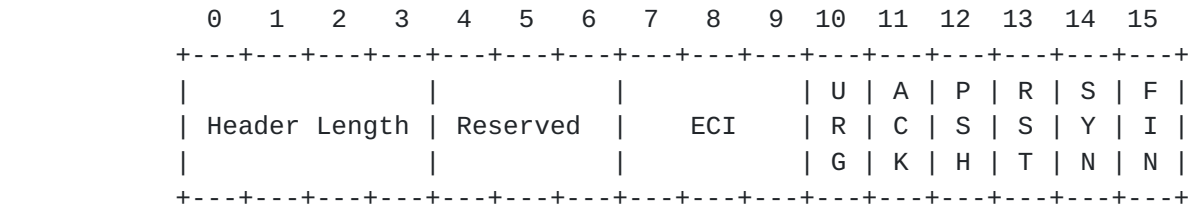

Figure 4: Definition of the ECI field within bytes 13 and 14 of the TCP Header, overloading the current definitions above for established RECN flows.

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 modulo 8 and MUST echo its value 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 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 4.1.4.

As we have already emphasised, the re-ECN protocol makes no changes and has no effect on the TCP congestion control algorithm. So, each increment of ECI (or detection of a drop) also triggers the standard TCP congestion response, but with no more than one congestion response per round trip, as usual.

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.

4.1.1.1. Drops and Marks

Re-ECN is based on the ECN protocol [RFC3168] which in turn is typically based on the RED algorithm [RFC2309]. This algorithm marks packets as CE with a probability that increases as the size of the router queue increases. Howeverif the queue becomes too full then it will revert to dropping packets. Because of this it is important

that re-ECN treats each packet drop it detects as if it were actually a CE mark. This ensures that it can continue to correctly echo congestion even through a highly congested path.

In order to ensure that drops are correctly echoed the sender needs to add the number of drops detected per RTT to the difference in ECI value waiting to be echoed. A drop is defined as set out in [RFC2581] -- if the connection is in slow start then a single duplicate aknowledgement will be treated as an indication of a drop. When the system is in the congestion avoidance stage then 3 duplicate acknowledgements will be treated as a sign of a drop. In all cases, if a re-transmission time-out occurs then that will be treatd as a drop.

4.1.1.2. Safety against Long Pure ACK Loss Sequences

The ECI method was chosen for echoing congestion marking because a re-ECN sender needs to know about every CE mark arriving at the receiver, not just whether at least one arrives within a round trip time (which is all the ECE/CWR mechanism supported). And, as pure ACKs are not protected by TCP reliable delivery, we repeat the same ECI value in every ACK until it changes. Even if many ACKs in a row are lost, as soon as one gets through, the ECI field it repeats from previous ACKs that didn't get through will update the sender on how many CE marks arrived since the last ACK got through.

The sender will only lose a record of the arrival of a CE mark if all the ACKS are lost (and all of them were pure ACKs) for a stream of data long enough to contain 8 or more CE marks. So, if the marking fraction was p, at least 8/p pure ACKs would have to be lost. For example, if p was 5%, a sequence of 160 pure ACKs would all have to be lost. To protect against such extremely unlikely events, if a re-ECN sender detects a sequence of pure ACKs has been lost it SHOULD assume the ECI field wrapped as many times as possible within the sequence.

Specifically, if a re-ECN sender receives an ACK with an acknowledgement number that acknowledges L segments since the previous ACK but with a sequence number unchanged from the previously received ACK, it SHOULD conservatively assume that the ECI field incremented by D' = L - ((L-D) mod 8), where D is the apparent increase in the ECI field. For example if the ACK arriving after 9 pure ACK losses apparently increased ECI by 2, the assumed increment of ECI would still be 2. But if ECI apparently increased by 2 after 11 pure ACK losses, ECI should be assumed to have increased by 10.

A re-ECN sender MAY implement a heuristic algorithm to predict beyond reasonable doubt that the ECI field probably did not wrap within a

sequence of lost pure ACKs. But such an algorithm is NOT REQUIRED. Such an algorithm MUST NOT be used unless it is proven to work even in the presence of correlation between high ACK loss rate on the back channel and high CE marking rate on the forward channel.

Whatever assumption a re-ECN sender makes about potentially lost CE marks, both its congestion control and its re-echoing behaviour SHOULD be consistent with the assumption it makes.

4.1.2. RECN-Co mode: Re-ECT Sender with a Vanilla or Nonce ECT Receiver

If the half-connection is in RECN-Co mode, ECN feedback proceeds no differently to that of vanilla 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 vanilla 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 (Section 6.1.4). 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 vanilla 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 vanilla 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.

4.1.2.1. 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.

4.1.3. 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 options 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. A Re-ECT server (B) can use either setting of the NS flag combined with this type of SYN ACK in response to a SYN from a Re-ECT client (A). Normally a Re-ECT server will reply to a Re-ECT client with NS=0, but in the special circumstance below it can return a SYN ACK with NS=1.

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 meaning `don't care'. 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 SYNs within a connection.}

Recall that, if the SYN ACK reflects the same flag settings as the preceding SYN (because there is a broken legacy 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 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.

4.1.4. 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 3.2). 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.

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 [I-D.ietf-tcpm-ecnsyn] has shown that this caution was unnecessary, and proposes to allow a SYN ACK to be ECN-capable to improve performance. We have gone further by proposing to make the initial SYN ECN-capable too. 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 (deliberately) made the initial SYN ECN-capable. 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 [RFC2581], it turns out that the sender SHOULD set FNE on the first and third data packets in its flow, 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.

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 CWR,ECE,NS	FNE	-->	R.ECC=0	
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 A 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 router 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.

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) [RFC2988], which is commonly used as the idle period before TCP must reduce to the restart window [RFC2581]. 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 legacy modes if packets don't appear to be getting through (to work round firewalls discarding packets they consider unusual).}

4.1.5. 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 routers.

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 (NRECN) 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).}

4.2. Other Transports

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

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 in packets carrying 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:

- UDP fire and forget (e.g. DNS)
- UDP streaming with no feedback
- UDP streaming with feedback

}

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

A separate I-D has been submitted [Re-PCN] 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) [PCN-arch].

4.2.3. 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.

4.2.4. Guidelines for adding Re-ECN to SCTP

Annex 1 in [RFC2960] 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

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 3.2) 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 7.1 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 5).

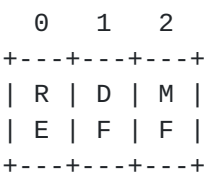

Figure 5: New Definition of the Re-ECN Extension (RE) Control Flag at the Start of Byte 7 of the IPv4 Header

The semantics of the RE flag are described in outline in Section 3 and specified fully in Section 4. 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 former.

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 6).

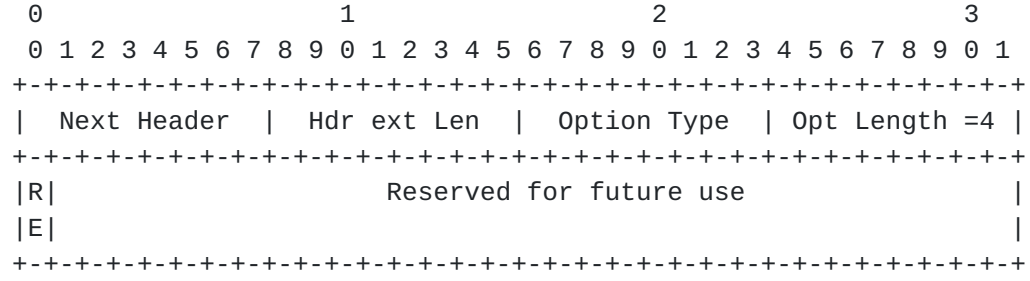

Figure 6: Definition of a New IPv6 Congestion Hop by Hop Option Header containing the Re-ECN Extension (RE) Control Flag

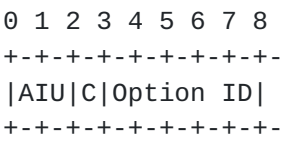

Figure 7: Congestion Hop by Hop Option Type Encoding

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 [RFC2402]) 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. As the RE flag does not need end-to-end authentication, we set the C flag to '1'.

{ToDo: A Congestion Hop by Hop Option ID will need to be registered with IANA.}

5.3. Router Forwarding Behaviour

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. [Re-PCN] 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 legacy 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 router 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 Section 6.1.5 would count as rate limiters for this purpose.

Preferential Drop: If a re-ECN capable router 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 7. 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 Section 6.1.3. 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.

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	Legacy ECN use only
00	0	Not-RECT	n/a	1	Not re-ECN-capable transport

Table 7: Drop Preference of EECN Codepoints (Sorted by `Worth')

The above drop preferences are arranged to preserve packets with more positive worth (Section 3.4), given senders of positive packets must have honestly declared downstream congestion. This is explained fully in Section 6 on applications, particularly when the application of re-ECN to protect against DDoS attacks is described.

5.4. Justification for Setting the First SYN to FNE

Congested routers may mark an FNE packet to CE(-1) (Section 5.3), and the initial SYN MUST be set to FNE by Re-ECT client A (Section 4.1.4). 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 legacy 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 4.1.4). But if the TCP client A finds out from the SYN ACK that the server was not ECN-capable, the TCP client MUST consider the first SYN as congestion marked before setting itself into Not-ECT mode. Section 4.1.4 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. 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.

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 Section 6.1.5 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

For re-ECN to work correctly through IP in IP tunnels, it needs slightly different tunnel handling to regular ECN [RFC3168]. Currently there is some incosistency between how the handling of IP in IP tunnels is defined in [RFC3168] and how it is defined in [RFC4301], but re-ECN would work fine with the IPsec behaviour. This inconsistency is addressed in a new Internet Draft [ECN-tunnel] that proposes to update RFC3168 tunnel behaviour to bring it into line with IPsec. Ideally, for re-ECN to work through a tunnel, 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 congestion 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 behaviour is the same as the full-functionality variant of [RFC3168] at tunnel exit, but different at tunnel entry.

If tunnels are left as they are specified in [RFC3168], whether the limited or full-functionality variants are 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 legacy 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 [ECN-tunnel] 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 [ECN-MPLS];

- 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 measureable 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 [RFC2406]) or an authentication header (AH [RFC2402]) 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.

6. Applications

6.1. Policing Congestion Response

6.1.1. The Policing Problem

The current Internet architecture trusts hosts to respond voluntarily to congestion. Limited evidence shows that the large majority of end-points on the Internet comply with a TCP-friendly response to congestion. But telephony (and increasingly video) services over the best effort Internet are attracting the interest of major commercial operations. Most of these applications do not respond to congestion at all. Those that can switch to lower rate codecs, still have a lower bound below which they must become unresponsive to congestion.

Of course, the Internet is intended to support many different application behaviours. But the problem is that this freedom can be exercised irresponsibly. The greater problem is that we will never be able to agree on where the boundary is between responsible and irresponsible. Therefore re-ECN is designed to allow different networks to set their own view of the limit to irresponsibility, and to allow networks that choose a more conservative limit to push back against congestion caused in more liberal networks.

As an example of the impossibility of setting a standard for fairness, mandating TCP-friendliness would set the bar too high for unresponsive streaming media, but still some would say the bar was too low. Even though all known peer-to-peer filesharing applications are TCP-compatible, they can cause a disproportionate amount of congestion, simply by using multiple flows and by transferring data continuously relative to other short-lived sessions. On the other hand, if we swung the other way and set the bar low enough to allow streaming media to be unresponsive, we would also allow denial of service attacks, which are typically unresponsive to congestion and consist of multiple continuous flows.

Applications that need (or choose) to be unresponsive to congestion can effectively take (some would say steal) whatever share of bottleneck resources they want from responsive flows. Whether or not such free-riding is common, inability to prevent it increases the risk of poor returns for investors in network infrastructure, leading to under-investment. An increasing proportion of unresponsive or free-riding demand coupled with persistent under-supply is a broken economic cycle. Therefore, if the current, largely co-operative consensus continues to erode, congestion collapse could become more common in more areas of the Internet [RFC3714].

While we have designed re-ECN so that networks can choose to deploy stringent policing, this does not imply we advocate that every network should introduce tight controls on those that cause congestion. Re-ECN has been specifically designed to allow different networks to choose how conservative or liberal they wish to be with respect to policing congestion. But those that choose to be conservative can protect themselves from the excesses that liberal networks allow their users.

6.1.2. The Case Against Bottleneck Policing

The state of the art in rate policing is the bottleneck policer, which is intended to be deployed at any forwarding resource that may become congested. Its aim is to detect flows that cause significantly more local congestion than others. Although operators might solve their immediate problems by deploying bottleneck policers, we are concerned that widespread deployment would make it extremely hard to evolve new application behaviours. We believe the IETF should offer re-ECN as the preferred protocol on which to base solutions to the policing problems of operators, because it would not harm evolvability and, frankly, it would be far more effective (see later for why).

Approaches like [XCHOKe] & [pBox] are nice approaches for rate policing traffic without the benefit of whole path information (such

as could be provided by re-ECN). But they must be deployed at bottlenecks in order to work. Unfortunately, a large proportion of traffic traverses at least two bottlenecks (in two access networks), particularly with the current traffic mix where peer-to-peer file-sharing is prevalent. If ECN were deployed, we believe it would be likely that these bottleneck policers would be adapted to combine ECN congestion marking from the upstream path with local congestion knowledge. But then the only useful placement for such policers would be close to the egress of the internetwork.

But then, if these bottleneck policers were widely deployed (which would require them to be more effective than they are now), the Internet would find itself with one universal rate adaptation policy (probably TCP-friendliness) embedded throughout the network. Given TCP's congestion control algorithm is already known to be hitting its scalability limits and new algorithms are being developed for high-speed congestion control, embedding TCP policing into the Internet would make evolution to new algorithms extremely painful. If a source wanted to use a different algorithm, it would have to first discover then negotiate with all the policers on its path, particularly those in the far access network. The IETF has already traveled that path with the Intserv architecture and found it constrains scalability [RFC2208].

Anyway, if bottleneck policers were ever widely deployed, they would be likely to be bypassed by determined attackers. They inherently have to police fairness per flow or per source-destination pair. Therefore they can easily be circumvented either by opening multiple flows (by varying the end-point port number); or by spoofing the source address but arranging with the receiver to hide the true return address at a higher layer.

6.1.3. Re-ECN Incentive Framework

The aim is to create an incentive environment that ensures optimal sharing of capacity despite everyone acting selfishly (including lying and cheating). Of course, the mechanisms put in place for this can lie dormant wherever co-operation is the norm.

Throughout this document we focus on path congestion. But some forms of fairness, particularly TCP's, also depend on round trip time. If TCP-fairness is required, we also propose to measure downstream path delay using re-feedback. We give a simple outline of how this could work in Appendix F. However, we do not expect this to be necessary, as researchers tend to agree that only congestion control dynamics need to depend on RTT, not the rate that the algorithm would converge on after a period of stability.

Figure 8 sketches the incentive framework that we will describe piece by piece throughout this section. We will do a first pass in overview, then return to each piece in detail. We re-use the earlier example of how downstream congestion is derived by subtracting upstream congestion from path congestion (Figure 1) but depict multiple trust boundaries to turn it into an internetwork. For clarity, only downstream congestion is shown (the difference between the two earlier plots). The graph displays downstream path congestion seen in a typical flow as it traverses an example path from sender S to receiver R, across networks N1, N2 & N4. Everyone is shown using re-ECN correctly, but we intend to show why everyone would /choose/ to use it correctly, and honestly.

Three main types of self-interest can be identified:

- Users want to transmit data across the network as fast as possible, paying as little as possible for the privilege. In this respect, there is no distinction between senders and receivers, but we must be wary of potential malice by one on the other;

- Network operators want to maximise revenues from the resources they invest in. They compete amongst themselves for the custom of users.

- Attackers (whether users or networks) want to use any opportunity to subvert the new re-ECN system for their own gain or to damage the service of their victims, whether targeted or random.

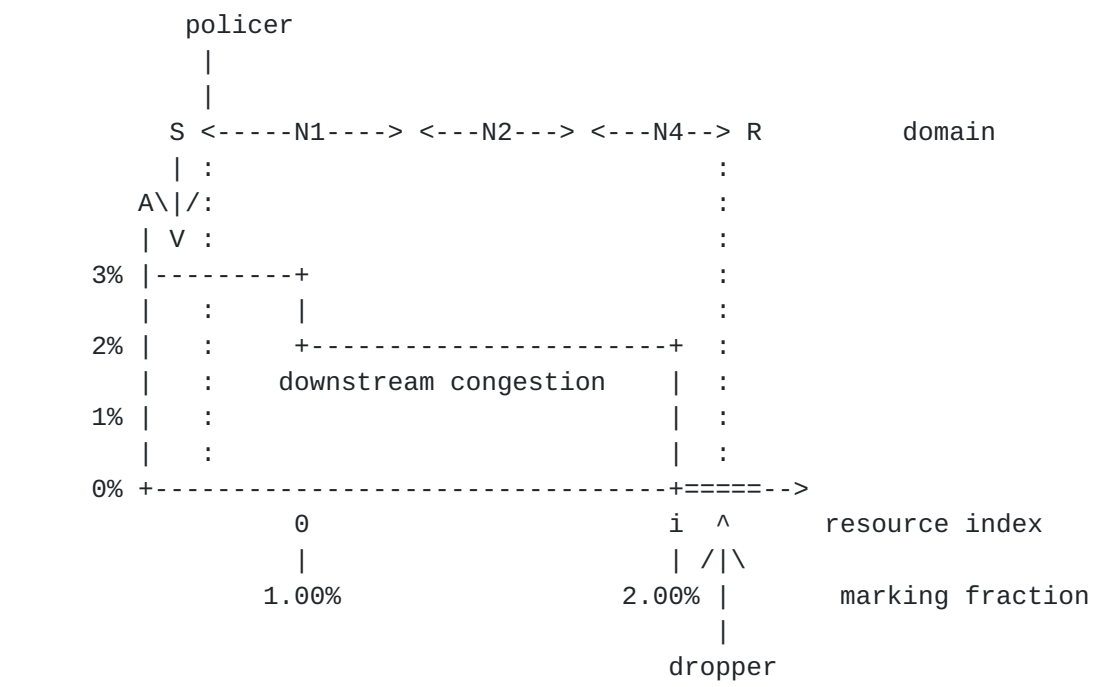

Figure 8: Incentive Framework, showing creation of opposing pressures to under-declare and over-declare downstream congestion, using a policer and a dropper

Source congestion control: We want to ensure that the sender will throttle its rate as downstream congestion increases. Whatever the agreed congestion response (whether TCP-compatible or some enhanced QoS), to some extent it will always be against the sender's interest to comply.

Ingress policing: But it is in all the network operators' interests to encourage fair congestion response, so that their investments are employed to satisfy the most valuable demand. The re-ECN protocol ensures packets carry the necessary information about their own expected downstream congestion so that N1 can deploy a policer at its ingress to check that S1 is complying with whatever congestion control it should be using (Section 6.1.5). If N1 is extremely conservative it could police each flow, but it is likely to just police the bulk amount of congestion each customer causes without regard to flows, or if it is extremely liberal it need not police congestion control at all. Whatever, it is always preferable to police traffic at the very first ingress into an internetwork, before non-compliant traffic can cause any damage.

Edge egress dropper: If the policer ensures the source has less right to a high rate the higher it declares downstream congestion, the source has a clear incentive to understate downstream congestion. But, if flows of packets are understated when they

enter the internetwork, they will have become negative by the time they leave. So, we introduce a dropper at the last network egress, which drops packets in flows that persistently declare negative downstream congestion (see Section 6.1.4 for details).

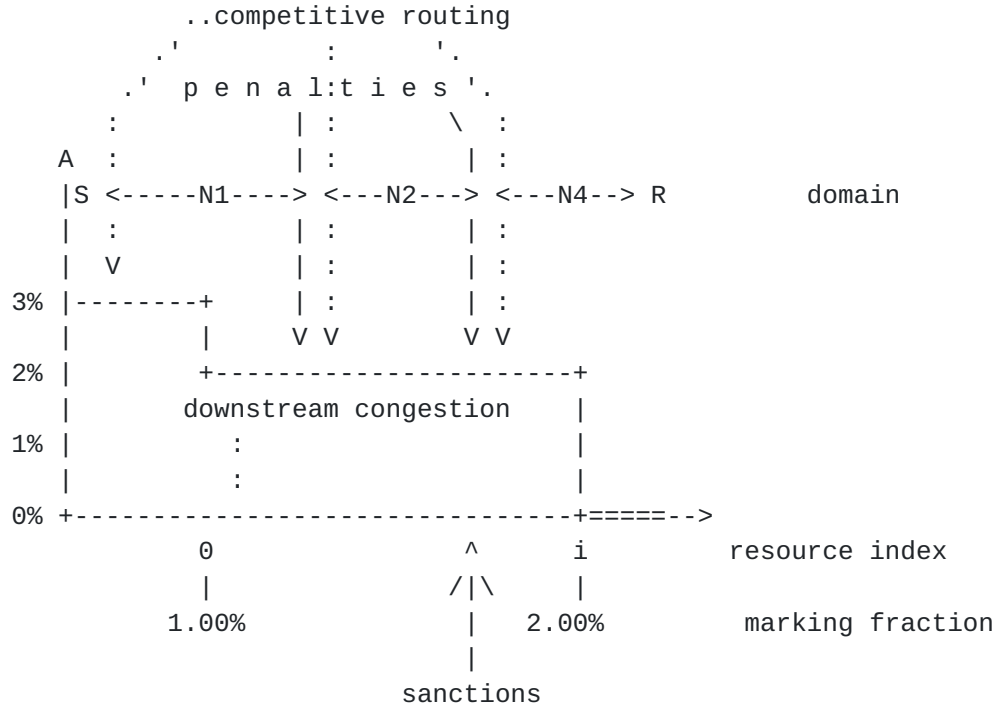

Figure 9: Incentives at Inter-domain Borders

Inter-domain traffic policing: But next we must ask, if congestion arises downstream (say in N4), what is the ingress network's (N1's) incentive to police its customers' response? If N1 turns a blind eye, its own customers benefit while other networks suffer. This is why all inter-domain QoS architectures (e.g. Intserv, Diffserv) police traffic each time it crosses a trust boundary. We have already shown that re-ECN gives a trustworthy measure of the expected downstream congestion that a flow will cause by subtracting negative volume from positive at any intermediate point on a path. N4 (say) can use this measure to police all the responses to congestion of all the sources beyond its upstream neighbour (N2), but in bulk with one very simple passive mechanism, rather than per flow, as we will now explain using Figure 9.

Emulating policing with inter-domain congestion penalties: Between high-speed networks, we would rather avoid per-flow policing, and we would rather avoid holding back traffic while it is policed. Instead, once re-ECN has arranged headers to carry downstream congestion honestly, N2 can contract to pay N4 penalties in

proportion to a single bulk count of the congestion metrics crossing their mutual trust boundary (Section 6.1.6). In this way, N4 puts pressure on N2 to suppress downstream congestion, for every flow passing through the border interface, even though they will all start and end in different places, and even though they may all be allowed different responses to congestion. The figure depicts this downward pressure on N2 by the solid downward arrow at the egress of N2. Then N2 has an incentive either to police the congestion response of its own ingress traffic (from N1) or to emulate policing by applying penalties to N1 in turn on the basis of congestion counted at their mutual boundary. In this recursive way, the incentives for each flow to respond correctly to congestion trace back with each flow precisely to each source, despite the mechanism not recognising flows (see Section 6.2.2).

Inter-domain congestion charging diversity: Any two networks are free to agree any of a range of penalty regimes between themselves but they would only provide the right incentives if they were within the following reasonable constraints. N2 should expect to have to pay penalties to N4 where penalties monotonically increase with the volume of congestion and negative penalties are not allowed. For instance, they may agree an SLA with tiered congestion thresholds, where higher penalties apply the higher the threshold that is broken. But the most obvious (and useful) form of penalty is where N4 levies a charge on N2 proportional to the volume of downstream congestion N2 dumps into N4. In the explanation that follows, we assume this specific variant of volume charging between networks - charging proportionate to the volume of congestion.

We must make clear that we are not advocating that everyone should use this form of contract. We are well aware that the IETF tries to avoid standardising technology that depends on a particular business model. And we strongly share this desire to encourage diversity. But our aim is merely to show that border policing can at least work with this one model, then we can assume that operators might experiment with the metric in other models (see Section 6.1.6 for examples). Of course, operators are free to complement this usage element of their charges with traditional capacity charging, and we expect they will as predicted by economics.

No congestion charging to users: Bulk congestion penalties at trust boundaries are passive and extremely simple, and lose none of their per-packet precision from one boundary to the next (unlike Diffserv all-address traffic conditioning agreements, which dissipate their effectiveness across long topologies). But at any trust boundary, there is no imperative to use congestion charging.

Traditional traffic policing can be used, if the complexity and cost is preferred. In particular, at the boundary with end customers (e.g. between S and N1), traffic policing will most likely be more appropriate. Policer complexity is less of a concern at the edge of the network. And end-customers are known to be highly averse to the unpredictability of congestion charging.

NOTE WELL: This document neither advocates nor requires congestion charging for end customers and advocates but does not require inter-domain congestion charging.

Competitive discipline of inter-domain traffic engineering: With inter-domain congestion charging, a domain seems to have a perverse incentive to fake congestion; N2's profit depends on the difference between congestion at its ingress (its revenue) and at its egress (its cost). So, overstating internal congestion seems to increase profit. However, smart border routing [Smart_rtg] by N1 will bias its routing towards the least cost routes. So, N2 risks losing all its revenue to competitive routes if it overstates congestion (see Section 6.2.3). In other words, if N2 is the least congested route, its ability to raise excess profits is limited by the congestion on the next least congested route. This pressure on N2 to remain competitive is represented by the dotted downward arrow at the ingress to N2 in Figure 9.

Closing the loop: All the above elements conspire to trap everyone between two opposing pressures (the downward and upward arrows in Figure 8 & Figure 9), ensuring the downstream congestion metric arrives at the destination neither above nor below zero. So, we have arrived back where we started in our argument. The ingress edge network can rely on downstream congestion declared in the packet headers presented by the sender. So it can police the sender's congestion response accordingly.

Evolvability of congestion control: We have seen that re-ECN enables policing at the very first ingress. We have also seen that, as flows continue on their path through further networks downstream, re-ECN removes the need for further per-domain ingress policing of all the different congestion responses allowed to each different flow. This is why the evolvability of re-ECN policing is so superior to bottleneck policing or to any policing of different QoS for different flows. Even if all access networks choose to conservatively police congestion per flow, each will want to compete with the others to allow new responses to congestion for new types of application. With re-ECN, each can introduce new controls independently, without coordinating with other networks and without having to standardise anything. But, as we have just

seen, by making inter-domain penalties proportionate to bulk downtream congestion, downstream networks can be agnostic to the specific congestion response for each flow, but they can still apply more penalty the more liberal the ingress access network has been in the response to congestion it allowed for each flow.

6.1.3.1. The Case against Classic Feedback

A system that produces an optimal outcome as a result of everyone's selfish actions is extremely powerful. Especially one that enables evolvability of congestion control. But why do we have to change to re-ECN to achieve it? Can't classic congestion feedback (as used already by standard ECN) be arranged to provide similar incentives and similar evolvability? Superficially it can. Kelly's seminal work showed how we can allow everyone the freedom to evolve whatever congestion control behaviour is in their application's best interest but still optimise the whole system of networks and users by placing a price on congestion to ensure responsible use of this freedom [Evol_cc]). Kelly used ECN with its classic congestion feedback model as the mechanism to convey congestion price information. The mechanism could be thought of as volume charging; except only the volume of packets marked with congestion experienced (CE) was counted.

However, below we explain why relying on classic feedback /required/ congestion charging to be used, while re-ECN achieves the same powerful outcome (given it is built on Kelly's foundations), but does not /require/ congestion charging. In brief, the problem with classic feedback is that the incentives have to trace the indirect path back to the sender---the long way round the feedback loop. For example, if classic feedback were used in Figure 8, N2 would have had to influence N1 via all of N4, R & S rather than directly.

Inability to agree what is happening downstream: In order to police its upstream neighbour's congestion response, the neighbours should be able to agree on the congestion to be responded to. Whatever the feedback regime, as packets change hands at each trust boundary, any path metrics they carry are verifiable by both neighbours. But, with a classic path metric, they can only agree on the /upstream/ path congestion.

Inaccessible back-channel: The network needs a whole-path congestion metric if it wants to control the source. Classically, whole path congestion emerges at the destination, to be fed back from receiver to sender in a back-channel. But, in any data network, back-channels need not be visible to relays, as they are essentially communications between the end-points. They may be encrypted, asymmetrically routed or simply omitted, so no network

element can reliably intercept them. The congestion charging literature solves this problem by charging the receiver and assuming this will cause the receiver to refer the charges to the sender. But, of course, this creates unintended side-effects...

`Receiver pays' unacceptable: In connectionless datagram networks, receivers and receiving networks cannot prevent reception from malicious senders, so `receiver pays' opens them to `denial of funds' attacks.

End-user congestion charging unacceptable: Even if 'denial of funds' were not a problem, we know that end-users are highly averse to the unpredictability of congestion charging and anyway, we want to avoid restricting network operators to just one retail tariff. But with classic feedback only an upstream metric is available, so we cannot avoid having to wrap the `receiver pays' money flow around the feedback loop, necessarily forcing end-users to be subjected to congestion charging.

To summarise so far, with classic feedback, policing congestion response without losing evolvability /requires/ congestion charging of end-users and a `receiver pays' model, whereas, with re-ECN, it is still possible to influence incentives using congestion charging but using the safer `sender pays' model. However, congestion charging is only likely to be appropriate between domains. So, without losing evolvability, re-ECN enables technical policing mechanisms that are more appropriate for end users than congestion pricing.

We now take a second pass over the incentive framework, filling in the detail.

6.1.4. Egress Dropper

As traffic leaves the last network before the receiver (domain N4 in Figure 8), the fraction of positive octets in a flow should match the fraction of negative octets introduced by congestion marking, leaving a balance of zero. If it is less (a negative flow), it implies that the source is understating path congestion (which will reduce the penalties that N2 owes N4).

If flows are positive, N4 need take no action---this simply means its upstream neighbour is paying more penalties than it needs to, and the source is going slower than it needs to. But, to protect itself against persistently negative flows, N4 will need to install a dropper at its egress. Appendix E gives a suggested algorithm for this dropper. There is no intention that the dropper algorithm needs to be standardised, it is merely provided to show that an efficient, robust algorithm is possible. But whatever algorithm is used must

meet the criteria below:

- It SHOULD introduce minimal false positives for honest flows;

- It SHOULD quickly detect and sanction dishonest flows (minimal false negatives);

- It MUST be invulnerable to state exhaustion attacks from malicious sources. For instance, if the dropper uses flow-state, it should not be possible for a source to send numerous packets, each with a different flow ID, to force the dropper to exhaust its memory capacity;

- It MUST introduce sufficient loss in goodput so that malicious sources cannot play off losses in the egress dropper against higher allowed throughput. Salvatori [CLoop_pol] describes this attack, which involves the source understating path congestion then inserting forward error correction (FEC) packets to compensate expected losses.

Note that the dropper operates on flows but we would like it not to require per-flow state. This is why we have been careful to ensure that all flows MUST start with a packet marked with the FNE codepoint. If a flow does not start with the FNE codepoint, a dropper is likely to treat it unfavourably. This risk makes it worth setting the FNE codepoint at the start of a flow, even though there is a cost to the sender of setting FNE (positive `worth'). Indeed, with the FNE codepoint, the rate at which a sender can generate new flows can be limited (Appendix G). In this respect, the FNE codepoint works like Handley's state set-up bit [Steps_DoS].

Appendix E also gives an example dropper implementation that aggregates flow state. Dropper algorithms will often maintain a moving average across flows of the fraction of RE blanked packets. When maintaining an average across flows, a dropper SHOULD only allow flows into the average if they start with FNE, but it SHOULD NOT include packets with the FNE codepoint set in the average. A sender sets the FNE codepoint when it does not have the benefit of feedback from the receiver. So, counting packets with FNE cleared would be likely to make the average unnecessarily positive, providing headroom (or should we say footroom?) for dishonest (negative) traffic.

If the dropper detects a persistently negative flow, it SHOULD drop sufficient negative and neutral packets to force the flow to not be negative. Drops SHOULD be focused on just sufficient packets in misbehaving flows to remove the negative bias while doing minimal extra harm.

6.1.5. Policing

Access operators who wish to limit the congeston that a sender is able to cause can deploy policers at the very first ingress to the internetwork. Re-ECN has been designed to avoid the need for bottleneck policing so that we can avoid a future where a single rate adaptation policy is embedded throughout the network. Instead, re-ECN allows the particular rate adaptation policy to be solely agreed bilaterally between the sender and its ingress access provider (Section 5.5.2 discusses possible ways to signal between them), which allows congestion control to be policed, but maintains its evolvability, requiring only a single, local box to be updated.

Appendix G gives examples of per-user policing algorithms. But there is no implication that these algorithms are to be standardised, or that they are ideal. The ingress rate policer is the part of the re-ECN incentive framework that is intended to be the most flexible. Once endpoint protocol handlers for re-ECN and egress droppers are in place, operators can choose exactly which congestion response they want to police, and whether they want to do it per user, per flow or not at all.

The re-ECN protocol allows these ingress policers to easily perform bulk per-user policing (Appendix G.1). This is likely to provide sufficient incentive to the user to correctly respond to congestion without needing the policing function to be overly complex. If an access operator chose they could use per-flow policing according to the widely adopted TCP rate adaptation (Appendix G.2) or other alternatives, however this would introduce extra complexity to the system.

If a per-flow rate policer is used, it should use path (not downstream) congestion as the relevant metric, which is represented by the fraction of octets in packets with positive (Re-Echo and FNE) and canceled (CE(0)) markings. Of course, re-ECN provides all the information a policer needs directly in the packets being policed. So, even policing TCP's AIMD algorithm is relatively straightforward (Appendix G.2).

Note that we have included canceled packets in the measure of path congestion. Canceled packets arise when the sender re-echoes earlier congestion, but then this Re-Echo packet just happens to be congestion marked itself. One would not normally expect many canceled packets at the first ingress because one would not normally expect much congestion marking to have been necessary that soon in the path. However, a home network or campus network may well sit between the sending endpoint and the ingress policer, so some congestion may occur upstream of the policer. And if congestion does

occur upstream, some canceled packets should be visible, and should be taken into account in the measure of path congestion.

But a much more important reason for including canceled packets in the measure of path congestion at an ingress policer is that a sender might otherwise subvert the protocol by sending canceled packets instead of neutral (RECT) packets. Like neutral, canceled packets are worth zero, so the sender knows they won't be counted against any quota it might have been allowed. But unlike neutral packets, canceled packets are immune to congestion marking, because they have already been congestion marked. So, it is both correct and useful that canceled packets should be included in a policer's measure of path congestion, as this removes the incentive the sender would otherwise have to mark more packets as canceled than it should.

An ingress policer should also ensure that flows are not already negative when they enter the access network. As with canceled packets, the presence of negative packets will typically be unusual. Therefore it will be easy to detect negative flows at the ingress by just detecting negative packets then monitoring the flow they belong to.

Of course, even if the sender does operate its own network, it may arrange not to congestion mark traffic. Whether the sender does this or not is of no concern to anyone else except the sender. Such a sender will not be policed against its own network's contribution to congestion, but the only resulting problem would be overload in the sender's own network.

Finally, we must not forget that an easy way to circumvent re-ECN's defences is for the source to turn off re-ECN support, by setting the Not-RECT codepoint, implying legacy traffic. Therefore an ingress policer should put a general rate-limit on Not-RECT traffic, which SHOULD be lax during early, patchy deployment, but will have to become stricter as deployment widens. Similarly, flows starting without an FNE packet can be confined by a strict rate-limit used for the remainder of flows that haven't proved they are well-behaved by starting correctly (therefore they need not consume any flow state---they are just confined to the `misbehaving' bin if they carry an unrecognised flow ID).

6.1.6. Inter-domain Policing

One of the main design goals of re-ECN is for border security mechanisms to be as simple as possible, otherwise they will become the pinch-points that limit scalability of the whole internetwork. We want to avoid per-flow processing at borders and to keep to passive mechanisms that can monitor traffic in parallel to

forwarding, rather than having to filter traffic inline---in series with forwarding. Such passive, off-line mechanisms are essential for future high-speed all-optical border interconnection where packets cannot be buffered while they are checked for policy compliance.

So far, we have been able to keep the border mechanisms simple, despite having had to harden them against some subtle attacks on the re-ECN design. The mechanisms are still passive and avoid per-flow processing.

The basic accounting mechanism at each border interface simply involves accumulating the volume of packets with positive worth (Re-Echo and FNE), and subtracting the volume of those with negative worth: CE(-1). Even though this mechanism takes no regard of flows, over an accounting period (say a month) this subtraction will account for the downstream congestion caused by all the flows traversing the interface, wherever they come from, and wherever they go to. The two networks can agree to use this metric however they wish to determine some congestion-related penalty against the upstream network. Although the algorithm could hardly be simpler, it is spelled out using pseudo-code in Appendix H.1.

Various attempts to subvert the re-ECN design have been made. In all cases their root cause is persistently negative flows. But, after describing these attacks we will show that we don't actually have to get rid of all persistently negative flows in order to thwart the attacks.

In honest flows, downstream congestion is measured as positive minus negative volume. So if all flows are honest (i.e. not persistently negative), adding all positive volume and all negative volume without regard to flows will give an aggregate measure of downstream congestion. But such simple aggregation is only possible if no flows are persistently negative. Unless persistently negative flows are completely removed, they will reduce the aggregate measure of congestion. The aggregate may still be positive overall, but not as positive as it would have been had the negative flows been removed.

In Section 6.1.4 we discussed how to sanction traffic to remove, or at least to identify, persistently negative flows. But, even if the sanction for negative traffic is to discard it, unless it is discarded at the exact point it goes negative, it will wrongly subtract from aggregate downstream congestion, at least at any borders it crosses after it has gone negative but before it is discarded.

We rely on sanctions to deter dishonest understatement of congestion. But even the ultimate sanction of discard can only be effective if

the sender is bothered about the data getting through to its destination. A number of attacks have been identified where a sender gains from sending dummy traffic or it can attack someone or something using dummy traffic even though it isn't communicating any information to anyone:

- A host can send traffic with no positive markings towards its intended destination, aiming to transmit as much traffic as any dropper will allow [Bauer06]. It may add forward error correction (FEC) to repair as much drop as it experiences.

- A host can send dummy traffic into the network with no positive markings and with no intention of communicating with anyone, but merely to cause higher levels of congestion for others who do want to communicate (DoS). So, to ride over the extra congestion, everyone else has to spend more of whatever rights to cause congestion they have been allowed.

- A network can simply create its own dummy traffic to congest another network, perhaps causing it to lose business at no cost to the attacking network. This is a form of denial of service perpetrated by one network on another. The preferential drop measures in Section 5.3 provide crude protection against such attacks, but we are not overly worried about more accurate prevention measures, because it is already possible for networks to DoS other networks on the general Internet, but they generally don't because of the grave consequences of being found out. We are only concerned if re-ECN increases the motivation for such an attack, as in the next example.

- A network can just generate negative traffic and send it over its border with a neighbour to reduce the overall penalties that it should pay to that neighbour. It could even initialise the TTL so it expired shortly after entering the neighbouring network, reducing the chance of detection further downstream. This attack need not be motivated by a desire to deny service and indeed need not cause denial of service. A network's main motivator would most likely be to reduce the penalties it pays to a neighbour. But, the prospect of financial gain might tempt the network into mounting a DoS attack on the other network as well, given the gain would offset some of the risk of being detected.

The first step towards a solution to all these problems with negative flows is to be able to estimate the contribution they make to downstream congestion at a border and to correct the measure accordingly. Although ideally we want to remove negative flows themselves, perhaps surprisingly, the most effective first step is to cancel out the polluting effect negative flows have on the measure of

downstream congestion at a border. It is more important to get an unbiased estimate of their effect, than to try to remove them all. A suggested algorithm to give an unbiased estimate of the contribution from negative flows to the downstream congestion measure is given in Appendix H.2.

Although making an accurate assessment of the contribution from negative flows may not be easy, just the single step of neutralising their polluting effect on congestion metrics removes all the gains networks could otherwise make from mounting dummy traffic attacks on each other. This puts all networks on the same side (only with respect to negative flows of course), rather than being pitched against each other. The network where this flow goes negative as well as all the networks downstream lose out from not being reimbursed for any congestion this flow causes. So they all have an interest in getting rid of these negative flows. Networks forwarding a flow before it goes negative aren't strictly on the same side, but they are disinterested bystanders---they don't care that the flow goes negative downstream, but at least they can't actively gain from making it go negative. The problem becomes localised so that once a flow goes negative, all the networks from where it happens and beyond downstream each have a small problem, each can detect it has a problem and each can get rid of the problem if it chooses to. But negative flows can no longer be used for any new attacks.

Once an unbiased estimate of the effect of negative flows can be made, the problem reduces to detecting and preferably removing flows that have gone negative as soon as possible. But importantly, complete eradication of negative flows is no longer critical---best endeavours will be sufficient.

For instance, let us consider the case where a source sends traffic with no positive markings at all, hoping to at least get as much traffic delivered as network-based droppers will allow. The flow is likely to go at least slightly negative in the first network on the path (N1 if we use the example network layout in Figure 9). If all networks use the algorithm in Appendix H.2 to inflate penalties at their border with an upstream network, they will remove the effect of negative flows. So, for instance, N2 will not be paying a penalty to N1 for this flow. Further, because the flow contributes no positive markings at all, a dropper at the egress will completely remove it.

The remaining problem is that every network is carrying a flow that is causing congestion to others but not being held to account for the congestion it is causing. Whenever the fail-safe border algorithm (Section 6.1.7) or the border algorithm to compensate for negative flows (Appendix H.2) detects a negative flow, it can instantiate a focused dropper for that flow locally. It may be some time before

the flow is detected, but the more strongly negative the flow is, the more quickly it will be detected by the fail-safe algorithm. But, in the meantime, it will not be distorting border incentives. Until it is detected, if it contributes to drop anywhere, its packets will tend to be dropped before others if routers use the preferential drop rules in Section 5.3, which discriminate against non-positive packets. All networks below the point where a flow goes negative (N1, N2 and N4 in this case) have an incentive to remove this flow, but the router where it first goes negative (in N1) can of course remove the problem for everyone downstream.

In the case of DDoS attacks, Section 6.2.1 describes how re-ECN mitigates their force.

6.1.7. Inter-domain Fail-safes

The mechanisms described so far create incentives for rational network operators to behave. That is, one operator aims to make another behave responsibly by applying penalties and expects a rational response (i.e. one that trades off costs against benefits). It is usually reasonable to assume that other network operators will behave rationally (policy routing can avoid those that might not). But this approach does not protect against the misconfigurations and accidents of other operators.

Therefore, we propose the following two mechanisms at a network's borders to provide "defence in depth". Both are similar:

Highly positive flows: A small sample of positive packets should be picked randomly as they cross a border interface. Then subsequent packets matching the same source and destination address and DSCP should be monitored. If the fraction of positive marking is well above a threshold (to be determined by operational practice), a management alarm SHOULD be raised, and the flow MAY be automatically subject to focused drop.

Persistently negative flows: A small sample of congestion marked (negative) packets should be picked randomly as they cross a border interface. Then subsequent packets matching the same source and destination address and DSCP should be monitored. If the balance of positive minus negative markings is persistently negative, a management alarm SHOULD be raised, and the flow MAY be automatically subject to focused drop.

Both these mechanisms rely on the fact that highly positive (or negative) flows will appear more quickly in the sample by selecting randomly solely from positive (or negative) packets.

6.1.8. Simulations

Simulations of policer and dropper performance done for the multi-bit version of re-feedback have been included in section 5 "Dropper Performance" of [Re-fb]. Simulations of policer and dropper for the re-ECN version described in this document are work in progress.

6.2. Other Applications

6.2.1. DDoS Mitigation

A flooding attack is inherently about congestion of a resource. Because re-ECN ensures the sources causing network congestion experience the cost of their own actions, it acts as a first line of defence against DDoS. As load focuses on a victim, upstream queues grow, requiring honest sources to pre-load packets with a higher fraction of positive packets. Once downstream routers are so congested that they are dropping traffic, they will be CE marking the traffic they do forward 100%. Honest sources will therefore be sending Re-Echo 100% (and therefore being severely rate-limited at the ingress).

Senders under malicious control can either do the same as honest sources, and be rate-limited at ingress, or they can understate congestion by sending more neutral RECT packets than they should. If sources understate congestion (i.e. do not re-echo sufficient positive packets) and the preferential drop ranking is implemented on routers (Section 5.3), these routers will preserve positive traffic until last. So, the neutral traffic from malicious sources will all be automatically dropped first. Either way, the malicious sources cannot send more than honest sources.

Further, hosts under malicious control will tend to be re-used for many different attacks. They will therefore build up a long term history of causing congestion. Therefore, as long as the population of potentially compromisable hosts around the Internet is limited, the per-user policing algorithms in Appendix G.1 will gradually throttle down zombies and other launchpads for attacks. Therefore, widespread deployment of re-ECN could considerably dampen the force of DDoS. Certainly, zombie armies could hold their fire for long enough to be able to build up enough credit in the per-user policers to launch an attack. But they would then still be limited to no more throughput than other, honest users.

Inter-domain traffic policing (see Section 6.1.6)ensures that any network that harbours compromised `zombie' hosts will have to bear the cost of the congestion caused by traffic from zombies in downstream networks. Such networks will be incentivised to deploy

per-user policers that rate-limit hosts that are unresponsive to congestion so they can only send very slowly into congested paths. As well as protecting other networks, the extremely poor performance at any sign of congestion will incentivise the zombie's owner to clean it up. However, the host should behave normally when using uncongested paths.

Uniquely, re-ECN handles DDoS traffic without relying on the validity of identifiers in packets. Certainly the egress dropper relies on uniqueness of flow identifiers, but not their validity. So if a source spoofs another address, re-ECN works just as well, as long as the attacker cannot imitate all the flow identifiers of another active flow passing through the same dropper (see Section 6.3). Similarly, the ingress policer relies on uniqueness of flow IDs, not their validity. Because a new flow will only be allowed any rate at all if it starts with FNE, and the more FNE packets there are starting new flows, the more they will be limited. Essentially a re-ECN policer limits the bulk of all congestion entering the network through a physical interface; limiting the congestion caused by each flow is merely an optional extra.

6.2.2. End-to-end QoS

{ToDo: (Section 3.3.2 of [Re-fb] entitled `Edge QoS' gives an outline of the text that will be added here).}

6.2.3. Traffic Engineering

{ToDo: }

6.2.4. Inter-Provider Service Monitoring

{ToDo: }

6.3. Limitations

The known limitations of the re-ECN approach are:

- We still cannot defend against the attack described in Section 10 where a malicious source sends negative traffic through the same egress dropper as another flow and imitates its flow identifiers, allowing a malicious source to cause an innocent flow to experience heavy drop.

- Re-feedback for TTL (re-TTL) would also be desirable at the same time as re-ECN. Unfortunately this requires a further standards action for the mechanisms briefly described in Appendix F

- Traffic must be ECN-capable for re-ECN to be effective. The only defence against malicious users who turn off ECN capbility is that networks are expected to rate limit Not-ECT traffic and to apply higher drop preference to it during congestion. Although these are blunt instruments, they at least represent a feasible scenario for the future Internet where Not-ECT traffic co-exists with re-ECN traffic, but as a severely hobbled under-class. We recommend (Section 7.1) that while accommodating a smooth initial transition to re-ECN, policing policies should gradually be tightened to rate limit Not-ECT traffic more strictly in the longer term.

- When checking whether a flow is balancing positive markings with congestion marking, re-ECN can only account for congestion marking, not drops. So, whenever a sender experiences drop, it does not have to re-echo the congestion event. Nonetheless, it is hardly any advantage to be able to send faster than other flows only if your traffic is dropped and the other traffic isn't.

- We are considering the issue of whether it would be useful to truncate rather than drop packets that appear to be malicious, so that the feedback loop is not broken but useful data can be removed.

7. Incremental Deployment

7.1. Incremental Deployment Features

The design of the re-ECN protocol started from the fact that the current ECN marking behaviour of routers was sufficient and that re-feedback could be introduced around these routers 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 vanishly 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-RECT marking. Traffic from legacy 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 legacy 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 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 legacy ECN sources will gain by upgrading to re-ECN. Thus, towards the end of the voluntary incremental deployment period, legacy 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 3.3);

- No provision is necessary for a re-ECN capable source transport to use the ECN nonce (Section 4.1.2.1);

- 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 (Section 6);

- 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.

7.2. Incremental Deployment Incentives

It would only be worth standardising the re-ECN protocol if there existed a coherent story for how it might be incrementally deployed. In order for it to have a chance of deployment, everyone who needs to act must have a strong incentive to act, and the incentives must arise in the order that deployment would have to happen. Re-ECN works around unmodified ECN routers, but we can't just discuss why and how re-ECN deployment might build on ECN deployment, because there is precious little to build on in the first place. Instead, we aim to show that re-ECN deployment could carry ECN with it. We focus on commercial deployment incentives, although some of the arguments apply equally to academic or government sectors.

ECN deployment:

ECN is largely implemented in commercial routers, but generally not as a supported feature, and it has largely not been deployed by commercial network operators. It has been released in many Unix-based operating systems, but not in proprietary OSs like Windows or those in many mobile devices. For detailed deployment status, see [ECN-Deploy]. We believe the reason ECN deployment has not happened is twofold:

* ECN requires changes to both routers and hosts. If someone wanted to sell the improvement that ECN offers, they would have to co-ordinate deployment of their product with others. An ECN server only gives any improvement on an ECN network. An ECN network only gives any improvement if used by ECN devices. Deployment that requires co-ordination adds cost and delay and tends to dilute any competitive advantage that might be gained.

* ECN `only' gives a performance improvement. Making a product a bit faster (whether the product is a device or a network), isn't usually a sufficient selling point to be worth the cost of co-ordinating across the industry to deploy it. Network operators tend to avoid re-configuring a working network unless launching a new product.

ECN and re-ECN for Edge-to-edge Assured QoS:

We believe the proposal to provide assured QoS sessions using a form of ECN called pre-congestion notification (PCN) [PCN-arch] is most likely to break the deadlock in ECN deployment first. It only requires edge-to-edge deployment so it does not require endpoint support. It can be deployed in a single network, then grow incrementally to interconnected networks. And it provides a different `product' (internetworked assured QoS), rather than

merely making an existing product a bit faster.

Not only could this assured QoS application kick-start ECN deployment, it could also carry re-ECN deployment with it; because re-ECN can enable the assured QoS region to expand to a large internetwork where neighbouring networks do not trust each other. [Re-PCN] argues that re-ECN security should be built in to the QoS system from the start, explaining why and how.

If ECN and re-ECN were deployed edge-to-edge for assured QoS, operators would gain valuable experience. They would also clear away many technical obstacles such as firewall configurations that block all but the legacy settings of the ECN field and the RE flag.

ECN in Access Networks:

The next obstacle to ECN deployment would be extension to access and backhaul networks, where considerable link layer differences makes implementation non-trivial, particularly on congested wireless links. ECN and re-ECN work fine during partial deployment, but they will not be very useful if the most congested elements in networks are the last to support them. Access network support is one of the weakest parts of this deployment story. All we can hope is that, once the benefits of ECN are better understood by operators, they will push for the necessary link layer implementations as deployment proceeds.

Policing Unresponsive Flows:

Re-ECN allows a network to offer differentiated quality of service as explained in Section 6.2.2. But we do not believe this will motivate initial deployment of re-ECN, because the industry is already set on alternative ways of doing QoS. Despite being much more complicated and expensive, the alternative approaches are here and now.

But re-ECN is critical to QoS deployment in another respect. It can be used to prevent applications from taking whatever bandwidth they choose without asking.

Currently, applications that remain resolute in their lack of response to congestion are rewarded by other TCP applications. In other words, TCP is naively friendly, in that it reduces its rate in response to congestion whether it is competing with friends (other TCPs) or with enemies (unresponsive applications).

Therefore, those network owners that want to sell QoS will be keen to ensure that their users can't help themselves to QoS for free. Given the very large revenues at stake, we believe effective policing of congestion response will become highly sought after by network owners.

But this does not necessarily argue for re-ECN deployment. Network owners might choose to deploy bottleneck policers rather than re-ECN-based policing. However, under Related Work (Section 9) we argue that bottleneck policers are inherently vulnerable to circumvention.

Therefore we believe there will be a strong demand from network owners for re-ECN deployment so they can police flows that do not ask to be unresponsive to congestion, in order to protect their revenues from flows that do ask (QoS). In particular, we suspect that the operators of cellular networks will want to prevent VoIP and video applications being used freely on their networks as a more open market develops in GPRS and 3G devices.

Initial deployments are likely to be isolated to single cellular networks. Cellular operators would first place requirements on device manufacturers to include re-ECN in the standards for mobile devices. In parallel, they would put out tenders for ingress and egress policers. Then, after a while they would start to tighten rate limits on Not-ECT traffic from non-standard devices and they would start policing whatever non-accredited applications people might install on mobile devices with re-ECN support in the operating system. This would force even independent mobile device manufacturers to provide re-ECN support. Early standardisation across the cellular operators is likely, including interconnection agreements with penalties for excess downstream congestion.

We suspect some fixed broadband networks (whether cable or DSL) would follow a similar path. However, we also believe that larger parts of the fixed Internet would not choose to police on a per-flow basis. Some might choose to police congestion on a per-user basis in order to manage heavy peer-to-peer file-sharing, but it seems likely that a sizeable majority would not deploy any form of policing.

This hybrid situation begs the question, "How does re-ECN work for networks that choose to using policing if they connect with others that don't?" Traffic from non-ECN capable sources will arrive from other networks and cause congestion within the policed, ECN-capable networks. So networks that chose to police congestion would rate-limit Not-ECT traffic throughout their network, particularly at their borders. They would probably also set

higher usage prices in their interconnection contracts for incoming Not-ECT and Not-RECT traffic. We assume that interconnection contracts between networks in the same tier will include congestion penalties before contracts with provider backbones do.

A hybrid situation could remain for all time. As was explained in the introduction, we believe in healthy competition between policing and not policing, with no imperative to convert the whole world to the religion of policing. Networks that chose not to deploy egress droppers would leave themselves open to being congested by senders in other networks. But that would be their choice.

The important aspect of the egress dropper though is that it most protects the network that deploys it. If a network does not deploy an egress dropper, sources sending into it from other networks will be able to understate the congestion they are causing. Whereas, if a network deploys an egress dropper, it can know how much congestion other networks are dumping into it, and apply penalties or charges accordingly. So, whether or not a network polices its own sources at ingress, it is in its interests to deploy an egress dropper.

Host support:

In the above deployment scenario, host operating system support for re-ECN came about through the cellular operators demanding it in device standards (i.e. 3GPP). Of course, increasingly, mobile devices are being built to support multiple wireless technologies. So, if re-ECN were stipulated for cellular devices, it would automatically appear in those devices connected to the wireless fringes of fixed networks if they coupled cellular with WiFi or Bluetooth technology, for instance. Also, once implemented in the operating system of one mobile device, it would tend to be found in other devices using the same family of operating system.

Therefore, whether or not a fixed network deployed ECN, or deployed re-ECN policers and droppers, many of its hosts might well be using re-ECN over it. Indeed, they would be at an advantage when communicating with hosts across Re-ECN policed networks that rate limited Not-RECT traffic.

Other possible scenarios:

The above is thankfully not the only plausible scenario we can think of. One of the many clubs of operators that meet regularly around the world might decide to act together to persuade a major operating system manufacturer to implement re-ECN. And they may agree between them on an interconnection model that includes congestion penalties.

Re-ECN provides an interesting opportunity for device manufacturers as well as network operators. Policers can be configured loosely when first deployed. Then as re-ECN take-up increases, they can be tightened up, so that a network with re-ECN deployed can gradually squeeze down the service provided to legacy devices that have not upgraded to re-ECN. Many device vendors rely on replacement sales. And operating system companies rely heavily on new release sales. Also support services would like to be able to force stragglers to upgrade. So, the ability to throttle service to legacy operating systems is quite valuable.

Also, policing unresponsive sources may not be the only or even the first application that drives deployment. It may be policing causes of heavy congestion (e.g. peer-to-peer file-sharing). Or it may be mitigation of denial of service. Or we may be wrong in thinking simpler QoS will not be the initial motivation for re-ECN deployment. Indeed, the combined pressure for all these may be the motivator, but it seems optimistic to expect such a level of joined-up thinking from today's communications industry. We believe a single application alone must be a sufficient motivator.

In short, everyone gains from adding accountability to TCP/IP, except the selfish or malicious. So, deployment incentives tend to be strong.

8. Architectural Rationale

In the Internet's technical community, the danger of not responding to congestion is well-understood, as well as its attendant risk of congestion collapse [RFC3714]. However, one side of the Internet's commercial community considers that the very essence of IP is to provide open access to the internetwork for all applications. They see congestion as a symptom of over-conservative investment, and rely on revising application designs to find novel ways to keep applications working despite congestion. They argue that the Internet was never intended to be solely for TCP-friendly applications. Meanwhile, another side of the Internet's commercial community believes that it is worthwhile providing a network for novel applications only if it has sufficient capacity, which can happen only if a greater share of application revenues can be

/assured/ for the infrastructure provider. Otherwise the major investments required would carry too much risk and wouldn't happen.

The lesson articulated in [Tussle] is that we shouldn't embed our view on these arguments into the Internet at design time. Instead we should design the Internet so that the outcome of these arguments can get decided at run-time. Re-ECN is designed in that spirit. Once the protocol is available, different network operators can choose how liberal they want to be in holding people accountable for the congestion they cause. Some might boldly invest in capacity and not police its use at all, hoping that novel applications will result. Others might use re-ECN for fine-grained flow policing, expecting to make money selling vertically integrated services. Yet others might sit somewhere half-way, perhaps doing coarse, per-user policing. All might change their minds later. But re-ECN always allows them to interconnect so that the careful ones can protect themselves from the liberal ones.

The incentive-based approach used for re-ECN is based on Gibbens and Kelly's arguments [Evol_cc] on allowing endpoints the freedom to evolve new congestion control algorithms for new applications. They ensured responsible behaviour despite everyone's self-interest by applying pricing to ECN marking, and Kelly had proved stability and optimality in an earlier paper.

Re-ECN keeps all the underlying economic incentives, but rearranges the feedback. The idea is to allow a network operator (if it chooses) to deploy engineering mechanisms like policers at the front of the network which can be designed to behave /as if/ they are responding to congestion prices. Rather than having to subject users to congestion pricing, networks can then use more traditional charging regimes (or novel ones). But the engineering can constrain the overall amount of congestion a user can cause. This provides a buffer against completely outrageous congestion control, but still makes it easy for novel applications to evolve if they need different congestion control to the norms. It also allows novel charging regimes to evolve.

Despite being achieved with a relatively minor protocol change, re-ECN is an architectural change. Previously, Internet congestion could only be controlled by the data sender, because it was the only one both in a position to control the load and in a position to see information on congestion. Re-ECN levels the playing field. It recognises that the network also has a role to play in moderating (policing) congestion control. But policing is only truly effective at the first ingress into an internetwork, whereas path congestion was previously only visible at the last egress. So, re-ECN democratises congestion information. Then the choice over who

actually controls congestion can be made at run-time, not design time---a bit like an aircraft with dual controls. And different operators can make different choices. We believe non-architectural approaches to this problem are unlikely to offer more than partial solutions (see Section 9).

Importantly, re-ECN does NOT REQUIRE assumptions about specific congestion responses to be embedded in any network elements, except at the first ingress to the internetwork if that level of control is desired by the ingress operator. But such tight policing will be a matter of agreement between the source and its access network operator. The ingress operator need not police congestion response at flow granularity; it can simply hold a source responsible for the aggregate congestion it causes, perhaps keeping it within a monthly congestion quota. Or if the ingress network trusts the source, it can do nothing.

Therefore, the aim of the re-ECN protocol is NOT solely to police TCP-friendliness. Re-ECN preserves IP as a generic network layer for all sorts of responses to congestion, for all sorts of transports. Re-ECN merely ensures truthful downstream congestion information is available in the network layer for all sorts of accountability applications.

The end to end design principle does not say that all functions should be moved out of the lower layers---only those functions that are not generic to all higher layers. Re-ECN adds a function to the network layer that is generic, but was omitted: accountability for causing congestion. Accountability is not something that an end-user can provide to themselves. We believe re-ECN adds no more than is sufficient to hold each flow accountable, even if it consists of a single datagram.

"Accountability" implies being able to identify who is responsible for causing congestion. However, at the network layer it would NOT be useful to identify the cause of congestion by adding individual or organisational identity information, NOR by using source IP addresses. Rather than bringing identity information to the point of congestion, we bring downstream congestion information to the point where the cause can be most easily identified and dealt with. That is, at any trust boundary congestion can be associated with the physically connected upstream neighbour that is directly responsible for causing it (whether intentionally or not). A trust boundary interface is exactly the place to police or throttle in order to directly mitigate congestion, rather than having to trace the (ir)responsible party in order to shut them down.

Some considered that ECN itself was a layering violation. The

reasoning went that the interface to a layer should provide a service to the higher layer and hide how the lower layer does it. However, ECN reveals the state of the network layer and below to the transport layer. A more positive way to describe ECN is that it is like the return value of a function call to the network layer. It explicitly returns the status of the request to deliver a packet, by returning a value representing the current risk that a packet will not be served. Re-ECN has similar semantics, except the transport layer must try to guess the return value, then it can use the actual return value from the network layer to modify the next guess.

The guiding principle behind all the discussion in Section 6.1.6 on Policing is that any gain from subverting the protocol should be precisely neutralised, rather than punished. If a gain is punished to a greater extent than is sufficient to neutralise it, it will most likely open up a new vulnerability, where the amplifying effect of the punishment mechanism can be turned on others.

For instance, if possible, flows should be removed as soon as they go negative, but we do NOT RECOMMEND any attempts to discard such flows further upstream while they are still positive. Such over-zealous push-back is unnecessary and potentially dangerous. These flows have paid their `fare' up to the point they go negative, so there is no harm in delivering them that far. If someone downstream asks for a flow to be dropped as near to the source as possible, because they say it is going to become negative later, an upstream node cannot test the truth of this assertion. Rather than have to authenticate such messages, re-ECN has been designed so that flows can be dropped solely based on locally measurable evidence. A message hinting that a flow should be watched closely to test for negativity is fine. But not a message that claims that a positive flow will go negative later, so it should be dropped. .

9. Related Work

{Due to lack of time, this section is incomplete. The reader is referred to the Related Work section of [Re-fb] for a brief selection of related ideas.}

9.1. Policing Rate Response to Congestion

ATM network elements send congestion back-pressure messages [ITU-T.I.371] along each connection, duplicating any end to end feedback because they don't trust it. On the other hand, re-ECN ensures information in forwarded packets can be used for congestion management without requiring a connection-oriented architecture and re-using the overhead of fields that are already set aside for end to

end congestion control (and routing loop detection in the case of re-TTL in Appendix F).

We borrowed ideas from policers in the literature [pBox],[XCHOKe], AFD etc. for our rate equation policer. However, without the benefit of re-ECN they don't police the correct rate for the condition of their path. They detect unusually high /absolute/ rates, but only while the policer itself is congested, because they work by detecting prevalent flows in the discards from the local RED queue. These policers must sit at every potential bottleneck, whereas our policer need only be located at each ingress to the internetwork. As Floyd & Fall explain [pBox], the limitation of their approach is that a high sending rate might be perfectly legitimate, if the rest of the path is uncongested or the round trip time is short. Commercially available rate policers cap the rate of any one flow. Or they enforce monthly volume caps in an attempt to control high volume file-sharing. They limit the value a customer derives. They might also limit the congestion customers can cause, but only as an accidental side-effect. They actually punish traffic that fills troughs as much as traffic that causes peaks in utilisation. In practice network operators need to be able to allocate service by cost during congestion, and by value at other times.

9.2. 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].

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.

9.3. Identifying Upstream and Downstream Congestion

Purple [Purple] proposes that routers should use the CWR flag in the TCP header of ECN-capable flows to work out path congestion and therefore downstream congestion in a similar way to re-ECN. However, because CWR is in the transport layer, it is not always visible to network layer routers and policers. Purple's motivation was to improve AQM, not policing. But, of course, nodes trying to avoid a policer would not be expected to allow CWR to be visible.

10. Security Considerations

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. Section 5.5.1 discusses one of these.

- 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.

11. 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 3.2)
- The new extension header for IPv6 (Section 5.2);
- The new combinations of flags in the TCP header for capability negotiation (Section 4.1.3);
- The new ICMP message type (Section 5.5.1).

12. Conclusions

{ToDo:}

13. 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), and comments from participants in the CRN/CFP Broadband and DoS-resistant Internet working groups.

14. Comments Solicited

Comments and questions are encouraged and very welcome. They can be addressed to the IETF Transport Area working group's mailing list <tsvwg@ietf.org>, and/or to the authors.

15. References

15.1. Normative References

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

[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.

[RFC2581] Allman, M., Paxson, V., and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[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.

[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.

15.2. 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.

[Bauer06] Bauer, S., Faratin, P., and R. Beverly, "Assessing the assumptions underlying mechanism design for the Internet", Proc. Workshop on the Economics of Networked Systems (NetEcon06) , June 2006, <http://www.cs.duke.edu/nicl/netecon06/papers/ne06-assessing.pdf>.

[CLoop_pol] Salvatori, A., "Closed Loop Traffic Policing", Politecnico Torino and Institut Eurecom Masters Thesis , September 2005.

[ECN-Deploy] Floyd, S., "ECN (Explicit Congestion Notification) in TCP/IP; Implementation and Deployment of ECN", Web-page , May 2004, <http://www.icir.org/floyd/ecn.html#implementations>.

[ECN-MPLS] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", draft-ietf-tsvwg-ecn-mpls-01 (work in progress), June 2007.

[ECN-tunnel] Briscoe, B., "Layered Encapsulation of Congestion Notification", draft-briscoe-tsvwg-ecn-tunnel-00 (work in progress), June 2007.

[Evol_cc] Gibbens, R. and F. Kelly, "Resource pricing and the evolution of congestion control", Automatica 35(12)1969--1985, December 1999,

<http://www.statslab.cam.ac.uk/~frank/evol.html>.

[I-D.ietf-tcpm-ecnsyn]
Kuzmanovic, A., "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets", draft-ietf-tcpm-ecnsyn-03 (work in progress), November 2007.

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

[ITU-T.I.371]
ITU-T, "Traffic Control and Congestion Control in {B-ISDN}", ITU-T Rec. I.371 (03/04), March 2004.

[Jiang02] Jiang, H. and D. Dovrolis, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", ACM SIGCOMM CCR 32(3)75-88, July 2002, <http://doi.acm.org/10.1145/571697.571725>.

[Mathis97]
Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", ACM SIGCOMM CCR 27(3)67--82, July 1997, <http://doi.acm.org/10.1145/263932.264023>.

[PCN-arch]
Eardley, P., Babiarz, J., Chan, K., Charny, A., Geib, R., Karagiannis, G., Menth, M., and T. Tsou, "Pre-Congestion Notification Architecture", draft-eardley-pcn-architecture-00 (work in progress), June 2007.

[Purple] Pletka, R., Waldvogel, M., and S. Mannal, "PURPLE: Predictive Active Queue Management Utilizing Congestion Information", Proc. Local Computer Networks (LCN 2003) , October 2003.

[RFC2208] Mankin, A., Baker, F., Braden, B., Bradner, S., O'Dell, M., Romanow, A., Weinrib, A., and L. Zhang, "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement Some Guidelines on Deployment", RFC 2208, September 1997.

[RFC2402] Kent, S. and R. Atkinson, "IP Authentication Header",

RFC 2402, November 1998.

[RFC2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

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

[RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[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.

[RFC3714] Floyd, S. and J. Kempf, "IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet", RFC 3714, March 2004.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[Re-PCN] Briscoe, B., "Emulating Border Flow Policing using Re-ECN on Bulk Data", draft-briscoe-re-pcn-border-cheat-00 (work in progress), July 2007.

[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>.

[Smart_rtg] Goldenberg, D., Qiu, L., Xie, H., Yang, Y., and Y. Zhang,

"Optimizing Cost and Performance for Multihoming", ACM SIGCOMM CCR 34(4)79--92, October 2004, <http://citeseer.ist.psu.edu/698472.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.

[Tussle] Clark, D., Sollins, K., Wroclawski, J., and R. Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet", ACM SIGCOMM CCR 32(4)347--356, October 2002, <http://www.acm.org/sigcomm/sigcomm2002/papers/tussle.pdf>.

[XCHOKe] Chhabra, P., Chuig, S., Goel, A., John, A., Kumar, A., Saran, H., and R. Shorey, "XCHOKe: Malicious Source Control for Congestion Avoidance at Internet Gateways", Proceedings of IEEE International Conference on Network Protocols (ICNP-02) , November 2002, <http://www.cc.gatech.edu/~akumar/xchoke.pdf>.

[pBox] Floyd, S. and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", IEEE/ACM Transactions on Networking 7(4) 458--472, August 1999, <http://www.aciri.org/floyd/end2end-paper.html>.

Appendix A. Precise Re-ECN Protocol Operation

{ToDo: fix this}

The protocol operation in the middle described in Section 3.3 was an approximation. In fact, standard ECN router marking combines 1% and 2% marking into slightly less than 3% whole-path marking, because routers deliberately mark CE whether or not it has already been marked by another router 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 router to n-1 at the last.

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

- u_j represents congestion *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 router 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 the upstream congestion (CE marking fraction) fed back from the destination. 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, if p = 1 - (1 - u_j)(1 - v_j), then

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

    ~= p - u_j;                    if u_j << 100%
```

So, between the two routers in the example in Section 3.3, 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 (the orthogonal scheme forms the main square of four codepoints in Figure 2). 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 legacy 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 4.1.3 is as follows.

Choice of SYN flags: A re-ECN sender can work with vanilla 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 vanilla 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 vanilla 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.

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 roundtrip 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 roundtrip), 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. Example Egress Dropper Algorithm

{ToDo: Write up the basic algorithm with flow state, then the aggregated one.}

Appendix F. Re-TTL

This Appendix gives an overview of a proposal to be able to overload the TTL field in the IP header to monitor downstream propagation delay. This is included to show that it would be possible to take account of RTT if it was deemed desirable.

Delay re-feedback can be achieved by overloading the TTL field, without changing IP or router TTL processing. A target value for TTL at the destination would need standardising, say 16. If the path hop count increased by more than 16 during a routing change, it would temporarily be mistaken for a routing loop, so this target would need to be chosen to exceed typical hop count increases. The TCP wire protocol and handlers would need modifying to feed back the destination TTL and initialise it. It would be necessary to standardise the unit of TTL in terms of real time (as was the original intent in the early days of the Internet).

In the longer term, precision could be improved if routers decremented TTL to represent exact propagation delay to the next router. That is, for a router to decrement TTL by, say, 1.8 time units it would alternate the decrement of every packet between 1 & 2 at a ratio of 1:4. Although this might sometimes require a seemingly dangerous null decrement, a packet in a loop would still decrement to zero after 255 time units on average. As more routers were upgraded to this more accurate TTL decrement, path delay estimates would become increasingly accurate despite the presence of some legacy routers that continued to always decrement the TTL by 1.

Appendix G. Policer Designs to ensure Congestion Responsiveness

G.1. Per-user Policing

User policing requires a policer on the ingress interface of the access router associated with the user. At that point, the traffic of the user hasn't diverged on different routes yet; nor has it mixed with traffic from other sources.

In order to ensure that a user doesn't generate more congestion in the network than her due share, a modified bulk token-bucket is maintained with the following parameter:

- b_0 the initial token level
- r the filling rate
- b_max the bucket depth

The same token bucket algorithm is used as in many areas of networking, but how it is used is very different:

- all traffic from a user over the lifetime of their subscription is policed in the same token bucket.
- only positive and canceled packets (Re-Echo, FNE and CE(0)) consume tokens

Such a policer will allow network operators to throttle the contribution of their users to network congestion. This will require the appropriate contractual terms to be in place between operators and users. For instance: a condition for a user to subscribe to a given network service may be that she should not cause more than a volume C_user of congestion over a reference period T_user, although she may carry forward up to N_user times her allowance at the end of each period. These terms directly set the parameter of the user

policer:

- b_0 = C_user
- r = C_user/T_user
- b_max = b_0 * (N_user +1)

Besides the congestion budget policer above, another user policer may be necessary to further rate-limit FNE packets, if they are to be marked rather than dropped (see discussion in Section 5.3.). Rate-limiting FNE packets will prevent high bursts of new flow arrivals, which is a very useful feature in DoS prevention. A condition to subscribe to a given network service would have to be that a user should not generate more than C_FNE FNE packets, over a reference period T_FNE, with no option to carry forward any of the allowance at the end of each period. These terms directly set the parameters of the FNE policer:

- b_0 = C_FNE
- r = C_FNE/T_FNE
- b_max = b_0

T_FNE should be a much shorter period than T_user: for instance T_FNE could be in the order of minutes while T_user could be in order of weeks.

G.2. Per-flow Rate Policing

Whilst we believe that simple per-user policing would be sufficient to ensure senders comply with congestion control, some operators may wish to police the rate response of each flow to congestion as well. Although we do not believe this will be neceesary, we include this section to show how one could perform per-flow policing using enforcement of TCP-fairness as an example. Per-flow policing aims to enforce congestion responsiveness on the shortest information timescale on a network path: packet roundtrips.

This again requires that the appropriate terms be agreed between a network operator and its users, where a congestion responsiveness policy might be required for the use of a given network service (perhaps unless the user specifically requests otherwise).

As an example, we describe below how a rate adaptation policer can be designed when the applicable rate adaptation policy is TCP-compliance. In that context, the average throughput of a flow will

be expected to be bounded by the value of the TCP throughput during congestion avoidance, given in Mathis' formula [Mathis97]

$$x\_TCP = k * s / (T * sqrt(m))$$

where:

- x_TCP is the throughput of the TCP flow in packets per second,
- k is a constant upper-bounded by sqrt(3/2),
- s is the average packet size of the flow,
- T is the roundtrip time of the flow,
- m is the congestion level experienced by the flow.

We define the marking period N=1/m which represents the average number of packets between two positive or canceled packets. Mathis' formula can be re-written as:

$$x\_TCP = k*s*sqrt(N)/T$$

We can then get the average inter-mark time in a compliant TCP flow, dt_TCP, by solving (x_TCP/s)*dt_TCP = N which gives

$$dt\_TCP = sqrt(N)*T/k$$

We rely on this equation for the design of a rate-adaptation policer as a variation of a token bucket. In that case a policer has to be set up for each policed flow. This may be triggered by FNE packets, with the remainder of flows being all rate limited together if they do not start with an FNE packet.

Where maintaining per flow state is not a problem, for instance on some access routers, systematic per-flow policing may be considered. Should per-flow state be more constrained, rate adaptation policing could be limited to a random sample of flows exhibiting positive or canceled packets.

As in the case of user policing, only positive or canceled packets will consume tokens, however the amount of tokens consumed will depend on the congestion signal.

When a new rate adaptation policer is set up for flow j, the following state is created:

- a token bucket b_j of depth b_max starting at level b_0
- a timestamp t_j = timenow()
- a counter N_j = 0
- a roundtrip estimate T_j
- a filling rate r

When the policing node forwards a packet of flow j with no Re-Echo:

- . the counter is incremented: N_j += 1

When the policing node forwards a packet of flow j carrying a congestion mark (CE):

- the counter is incremented: N_j += 1
- the token level is adjusted: b_j += r*(timenow()-t_j) - sqrt(N_j)* T_j/k
- the counter is reset: N_j = 0
- the timer is reset: t_j = timenow()

An implementation example will be given in a later draft that avoids having to extract the square root.

Analysis: For a TCP flow, for r= 1 token/sec, on average,

r*(timenow()-t_j)-sqrt(N_j)* T_j/k = dt_TCP - sqrt(N)*T/k = 0

This means that the token level will fluctuate around its initial level. The depth b_max of the bucket sets the timescale on which the rate adaptation policy is performed while the filling rate r sets the trade-off between responsiveness and robustness:

- the higher b_max, the longer it will take to catch greedy flows
- the higher r, the fewer false positives (greedy verdict on compliant flows) but the more false negatives (compliant verdict on greedy flows)

This rate adaptation policer requires the availability of a roundtrip estimate which may be obtained for instance from the application of re-feedback to the downstream delay Appendix F or passive estimation [Jiang02].

When the bucket of a policer located at the access router (whether it is a per-user policer or a per-flow policer) becomes empty, the access router SHOULD drop at least all packets causing the token level to become negative. The network operator MAY take further sanctions if the token level of the per-flow policers associated with a user becomes negative.

Appendix H. Downstream Congestion Metering Algorithms

H.1. Bulk Downstream Congestion Metering Algorithm

To meter the bulk amount of downstream congestion in traffic crossing an inter-domain border an algorithm is needed that accumulates the size of positive packets and subtracts the size of negative packets. We maintain two counters:

V_b: accumulated congestion volume

B: total data volume (in case it is needed)

A suitable pseudo-code algorithm for a border router is as follows:

```
=====================================================================
V_b = 0
B   = 0
for each re-ECN-capable packet {
    b = readLength(packet)      /* set b to packet size          */
    B += b                      /* accumulate total volume       */
    if readEECN(packet) == (Re-Echo || FNE) {
        V_b += b                /* increment...                  */
    } elseif readEECN(packet) == CE(-1) {
        V_b -= b                /* ...or decrement V_b...        */
    }                           /*...depending on EECN field     */
}
=====================================================================
```

At the end of an accounting period this counter V_b represents the congestion volume that penalties could be applied to, as described in Section 6.1.6.

For instance, accumulated volume of congestion through a border interface over a month might be V_b = 5PB (petabyte = 10^15 byte). This might have resulted from an average downstream congestion level of 1% on an accumulated total data volume of B = 500PB.

H.2. Inflation Factor for Persistently Negative Flows

The following process is suggested to complement the simple algorithm above in order to protect against the various attacks from persistently negative flows described in Section 6.1.6. As explained in that section, the most important and first step is to estimate the contribution of persistently negative flows to the bulk volume of downstream pre-congestion and to inflate this bulk volume as if these flows weren't there. The process below has been designed to give an unbiased estimate, but it may be possible to define other processes that achieve similar ends.

While the above simple metering algorithm is counting the bulk of traffic over an accounting period, the meter should also select a subset of the whole flow ID space that is small enough to be able to realistically measure but large enough to give a realistic sample. Many different samples of different subsets of the ID space should be taken at different times during the accounting period, preferably covering the whole ID space. During each sample, the meter should count the volume of positive packets and subtract the volume of negative, maintaining a separate account for each flow in the sample. It should run a lot longer than the large majority of flows, to avoid a bias from missing the starts and ends of flows, which tend to be positive and negative respectively.

Once the accounting period finishes, the meter should calculate the total of the accounts V_{bI} for the subset of flows I in the sample, and the total of the accounts V_{fI} excluding flows with a negative account from the subset I. Then the weighted mean of all these samples should be taken $a_S = \sum_{\forall I} V_{fI} / \sum_{\forall I} V_{bI}$.

If V_b is the result of the bulk accounting algorithm over the accounting period (Appendix H.1) it can be inflated by this factor a_S to get a good unbiased estimate of the volume of downstream congestion over the accounting period $a_S.V_b$, without being polluted by the effect of persistently negative flows.

Appendix I. 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.

Authors' Addresses

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

Phone: +44 1473 645196
Email: bob.briscoe@bt.com
URI: http://www.cs.ucl.ac.uk/staff/B.Briscoe/

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

Phone: +44 1473 648734
Email: toby.moncaster@bt.com

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

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

Full Copyright Statement

Intellectual Property

Acknowledgments

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA). This document was produced using xml2rfc v1.32 (of http://xml.resource.org/) from a source in RFC-2629 XML format.