Network Working Group
Request for Comments: 3540
Category: Experimental

N. Spring
D. Wetherall
D. Ely
University of Washington

June 2003

Robust Explicit Congestion Notification (ECN) Signaling with Nonces

Status of this Memo

Copyright Notice

Abstract

This note describes the Explicit Congestion Notification (ECN)-nonce, an optional addition to ECN that protects against accidental or malicious concealment of marked packets from the TCP sender. It improves the robustness of congestion control by preventing receivers from exploiting ECN to gain an unfair share of network bandwidth. The ECN-nonce uses the two ECN-Capable Transport (ECT)codepoints in the ECN field of the IP header, and requires a flag in the TCP header. It is computationally efficient for both routers and hosts.

1. Introduction

Statement of Intent

This specification describes an optional addition to Explicit Congestion Notification [RFC3168] improving its robustness against malicious or accidental concealment of marked packets. It has not been deployed widely. One goal of publication as an Experimental RFC is to be prudent, and encourage use and deployment prior to publication in the standards track. Another consideration is to give time for firewall developers to recognize and accept the pattern presented by the nonce. It is the intent of the Transport Area Working Group to re-submit this specification as an IETF Proposed Standard in the future after more experience has been gained.

The correct operation of ECN requires the cooperation of the receiver to return Congestion Experienced signals to the sender, but the protocol lacks a mechanism to enforce this cooperation. This raises the possibility that an unscrupulous or poorly implemented receiver could always clear ECN-Echo and simply not return congestion signals to the sender. This would give the receiver a performance advantage at the expense of competing connections that behave properly. More generally, any device along the path (NAT box, firewall, QOS bandwidth shapers, and so forth) could remove congestion marks with impunity.

The above behaviors may or may not constitute a threat to the operation of congestion control in the Internet. However, given the central role of congestion control, it is prudent to design the ECN signaling loop to be robust against as many threats as possible. In this way, ECN can provide a clear incentive for improvement over the prior state-of-the-art without potential incentives for abuse. The ECN-nonce is a simple, efficient mechanism to eliminate the potential abuse of ECN.

The ECN-nonce enables the sender to verify the correct behavior of the ECN receiver and that there is no other interference that conceals marked (or dropped) packets in the signaling path. The ECN-nonce protects against both implementation errors and deliberate abuse. The ECN-nonce:

- catches a misbehaving receiver with a high probability, and never implicates an innocent receiver.

- does not change other aspects of ECN, nor does it reduce the benefits of ECN for behaving receivers.

- is cheap in both per-packet overhead (one TCP header flag) and processing requirements.

- is simple and, to the best of our knowledge, not prone to other attacks.

We also note that use of the ECN-nonce has two additional benefits, even when only drop-tail routers are used. First, packet drops cannot be concealed from the sender. Second, it prevents optimistic acknowledgements [Savage], in which TCP segments are acknowledged before they have been received. These benefits also serve to increase the robustness of congestion control from attacks. We do not elaborate on these benefits in this document.

The rest of this document describes the ECN-nonce. We present an overview followed by detailed behavior at senders and receivers.

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

2. Overview

The ECN-nonce builds on the existing ECN-Echo and Congestion Window Reduced (CWR) signaling mechanism. Familiarity with ECN [ECN] is assumed. For simplicity, we describe the ECN-nonce in one direction only, though it is run in both directions in parallel.

The ECN protocol for TCP remains unchanged, except for the definition of a new field in the TCP header. As in [RFC3168], ECT(0) or ECT(1) (ECN-Capable Transport) is set in the ECN field of the IP header on outgoing packets. Congested routers change this field to CE (Congestion Experienced). When TCP receivers notice CE, the ECE (ECN-Echo) flag is set in subsequent acknowledgements until receiving a CWR flag. The CWR flag is sent on new data whenever the sender reacts to congestion.

The ECN-nonce adds to this protocol, and enables the receiver to demonstrate to the sender that segments being acknowledged were received unmarked. A random one-bit value (a nonce) is encoded in the two ECT codepoints. The one-bit sum of these nonces is returned in a TCP header flag, the nonce sum (NS) bit. Packet marking erases the nonce value in the ECT codepoints because CE overwrites both ECN IP header bits. Since each nonce is required to calculate the sum, the correct nonce sum implies receipt of only unmarked packets. Not only are receivers prevented from concealing marked packets, middle-boxes along the network path cannot unmark a packet without successfully guessing the value of the original nonce.

The sender can verify the nonce sum returned by the receiver to ensure that congestion indications in the form of marked (or dropped) packets are not being concealed. Because the nonce sum is only one bit long, senders have a 50-50 chance of catching a lying receiver whenever an acknowledgement conceals a mark. Because each acknowledgement is an independent trial, cheaters will be caught quickly if there are repeated congestion signals.

The following paragraphs describe aspects of the ECN-nonce protocol in greater detail.

Each acknowledgement carries a nonce sum, which is the one bit sum (exclusive-or, or parity) of nonces over the byte range represented by the acknowledgement. The sum is used because not every packet is acknowledged individually, nor are packets acknowledged reliably. If a sum were not used, the nonce in an unmarked packet could be echoed to prove to the sender that the individual packet arrived unmarked. However, since these acks are not reliably delivered, the sender could not distinguish a lost ACK from one that was never sent in order to conceal a marked packet. The nonce sum prevents the receiver from concealing individual marked packets by not acknowledging them. Because the nonce and nonce sum are both one bit quantities, the sum is no easier to guess than the individual nonces. We show the nonce sum calculation below in Figure 1.

```
 Sender              Receiver
                       initial sum = 1
   -- 1:4 ECT(0)  --> NS = 1 + 0(1:4) = 1(:4)
   <- ACK 4, NS=1 ---
   -- 4:8 ECT(1)  --> NS = 1(:4) + 1(4:8) = 0(:8)
   <- ACK 8, NS=0 ---
   -- 8:12 ECT(1)  -> NS = 0(:8) + 1(8:12) = 1(:12)
   <- ACK 12, NS=1 --
   -- 12:16 ECT(1) -> NS = 1(:12) + 1(12:16) = 0(:16)
   <- ACK 16, NS=0 --
```

Figure 1: The calculation of nonce sums at the receiver.

After congestion has occurred and packets have been marked or lost, resynchronization of the sender and receiver nonce sums is needed. When packets are marked, the nonce is cleared, and the sum of the nonces at the receiver will no longer match the sum at the sender. Once nonces have been lost, the difference between sender and receiver nonce sums is constant until there is further loss. This means that it is possible to resynchronize the sender and receiver after congestion by having the sender set its nonce sum to that of the receiver. Because congestion indications do not need to be conveyed more frequently than once per round trip, the sender suspends checking while the CWR signal is being delivered and resets its nonce sum to the receiver's when new data is acknowledged. This has the benefit that the receiver is not explicitly involved in the re-synchronization process. The resynchronization process is shown in Figure 2 below. Note that the nonce sum returned in ACK 12 (NS=0) differs from that in the previous example (NS=1), and it continues to differ for ACK 16.

```
 Sender              Receiver
                         initial sum = 1
   -- 1:4 ECT(0)       -> NS = 1 + 0(1:4) = 1(:4)
   <- ACK 4, NS=1      --
   -- 4:8 ECT(1) -> CE -> NS = 1(:4) + ?(4:8) = 1(:8)
   <- ACK 8, ECE NS=1  --
   -- 8:12 ECT(1), CWR -> NS = 1(:8) + 1(8:12) = 0(:12)
   <- ACK 12, NS=0     --
   -- 12:16 ECT(1)     -> NS = 0(:12) + 1(12:16) = 1(:16)
   <- ACK 16, NS=1     --
```

Figure 2: The calculation of nonce sums at the receiver when a packet (4:8) is marked. The receiver may calculate the wrong nonce sum when the original nonce information is lost after a packet is marked.

Third, we need to reconcile that nonces are sent with packets but acknowledgements cover byte ranges. Acknowledged byte boundaries need not match the transmitted boundaries, and information can be retransmitted in packets with different byte boundaries. We discuss the first issue, how a receiver sets a nonce when acknowledging part of a segment, in Section 6.1. The second question, what nonce to send when retransmitting smaller segments as a large segment, has a simple answer: ECN is disabled for retransmissions, so can carry no nonce. Because retransmissions are associated with congestion events, nonce checking is suspended until after CWR is acknowledged and the congestion event is over.

The next sections describe the detailed behavior of senders, routers and receivers, starting with sender transmit behavior, then around the ECN signaling loop, and finish with sender acknowledgement processing.

3. Sender Behavior (Transmit)

Senders manage CWR and ECN-Echo as before. In addition, they must place nonces on packets as they are transmitted and check the validity of the nonce sums in acknowledgments as they are received. This section describes the transmit process.

To place a one bit nonce value on every ECN-capable IP packet, the sender uses the two ECT codepoints: ECT(0) represents a nonce of 0, and ECT(1) a nonce of 1. As in ECN, retransmissions are not ECN-capable, so carry no nonce.

The sender maintains a mapping from each packet's end sequence number to the expected nonce sum (not the nonce placed in the original transmission) in the acknowledgement bearing that sequence number.

4. Router Behavior

Routers behave as specified in [RFC3168]. By marking packets to signal congestion, the original value of the nonce, in ECT(0) or ECT(1), is removed. Neither the receiver nor any other party can unmark the packet without successfully guessing the value of the original nonce.

5. Receiver Behavior (Receive and Transmit)

ECN-nonce receivers maintain the nonce sum as in-order packets arrive and return the current nonce sum in each acknowledgement. Receiver behavior is otherwise unchanged from [RFC3168]. Returning the nonce sum is optional, but recommended, as senders are allowed to discontinue sending ECN-capable packets to receivers that do not support the ECN-nonce.

As packets are removed from the queue of out-of-order packets to be acknowledged, the nonce is recovered from the IP header. The nonce is added to the current nonce sum as the acknowledgement sequence number is advanced for the recent packet.

In the case of marked packets, one or more nonce values may be unknown to the receiver. In this case the missing nonce values are ignored when calculating the sum (or equivalently a value of zero is assumed) and ECN-Echo will be set to signal congestion to the sender.

Returning the nonce sum corresponding to a given acknowledgement is straightforward. It is carried in a single "NS" (Nonce Sum) bit in the TCP header. This bit is adjacent to the CWR and ECN-Echo bits, set as Bit 7 in byte 13 of the TCP header, as shown below:

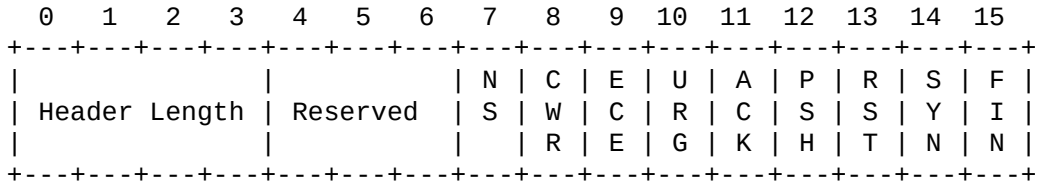


Figure 3: The new definition of bytes 13 and 14 of the TCP Header.

The initial nonce sum is 1, and is included in the SYN/ACK and ACK of the three way TCP handshake. This allows the other endpoint to infer nonce support, but is not a negotiation, in that the receiver of the SYN/ACK need not check if NS is set to decide whether to set NS in the subsequent ACK.

6. Sender Behavior (Receive)

This section completes the description of sender behavior by describing how senders check the validity of the nonce sums.

The nonce sum is checked when an acknowledgement of new data is received, except during congestion recovery when additional ECN-Echo signals would be ignored. Checking consists of comparing the correct nonce sum stored in a buffer to that carried in the acknowledgement, with a correction described in the following subsection.

If ECN-Echo is not set, the receiver claims to have received no marked packets, and can therefore compute and return the correct nonce sum. To conceal a mark, the receiver must successfully guess the sum of the nonces that it did not receive, because at least one packet was marked and the corresponding nonce was erased. Provided the individual nonces are equally likely to be 0 or 1, their sum is equally likely to be 0 or 1. In other words, any guess is equally likely to be wrong and has a 50-50 chance of being caught by the sender. Because each new acknowledgement is an independent trial, a cheating receiver is likely to be caught after a small number of lies.

If ECN-Echo is set, the receiver is sending a congestion signal and it is not necessary to check the nonce sum. The congestion window will be halved, CWR will be set on the next packet with new data sent, and ECN-Echo will be cleared once the CWR signal is received, as in [RFC3168]. During this recovery process, the sum may be incorrect because one or more nonces were not received. This does not matter during recovery, because TCP invokes congestion mechanisms at most once per RTT, whether there are one or more losses during that period.

6.1. Resynchronization After Loss or Mark

After recovery, it is necessary to re-synchronize the sender and receiver nonce sums so that further acknowledgments can be checked. When the receiver's sum is incorrect, it will remain incorrect until further loss.

This leads to a simple re-synchronization mechanism where the sender resets its nonce sum to that of the receiver when it receives an acknowledgment for new data sent after the congestion window was reduced. When responding to explicit congestion signals, this will be the first acknowledgement without the ECN-Echo flag set: the acknowledgement of the packet containing the CWR flag.

```
 Sender              Receiver
                          initial sum = 1
   -- 1:4 ECT(0)       -> NS = 1 + 0(1:4) = 1(:4)
   <- ACK 4, NS=1      --
   -- 4:8 ECT(1) -> LOST
   -- 8:12 ECT(1)      -> nonce sum calculation deferred
                            until in-order data received
   <- ACK 4, NS=0      --
   -- 12:16 ECT(1)     -> nonce sum calculation deferred
   <- ACK 4, NS=0      --
   -- 4:8 retransmit   -> NS = 1(:4) + ?(4:8) +
                               1(8:12) + 1(12:16) = 1(:16)
   <- ACK 16, NS=1     --
   -- 16:20 ECT(1) CWR ->
   <- ACK 20, NS=0     -- NS = 1(:16) + 1(16:20) = 0(:20)
```

Figure 4: The calculation of nonce sums at the receiver when a packet is lost, and resynchronization after loss. The nonce sum is not changed until the cumulative acknowledgement is advanced.

In practice, resynchronization can be accomplished by storing a bit that has the value one if the expected nonce sum stored by the sender and the received nonce sum in the acknowledgement of CWR differ, and zero otherwise. This synchronization offset bit can then be used in the comparison between expected nonce sum and received nonce sum.

The sender should ignore the nonce sum returned on any acknowledgements bearing the ECN-echo flag.

When an acknowledgment covers only a portion of a segment, such as when a middlebox resegments at the TCP layer instead of fragmenting IP packets, the sender should accept the nonce sum expected at the next segment boundary. In other words, an acknowledgement covering part of an original segment will include the nonce sum expected when the entire segment is acknowledged.

Finally, in ECN, senders can choose not to indicate ECN capability on some packets for any reason. An ECN-nonce sender must resynchronize after sending such ECN-incapable packets, as though a CWR had been sent with the first new data after the ECN-incapable packets. The sender loses protection for any unacknowledged packets until resynchronization occurs.

6.2. Sender Behavior - Incorrect Nonce Received

The sender's response to an incorrect nonce is a matter of policy. It is separate from the checking mechanism and does not need to be handled uniformly by senders. Further, checking received nonce sums at all is optional, and may be disabled.

If the receiver has never sent a non-zero nonce sum, the sender can infer that the receiver does not understand the nonce, and rate limit the connection, place it in a lower-priority queue, or cease setting ECT in outgoing segments.

If the received nonce sum has been set in a previous acknowledgement, the sender might infer that a network device has interfered with correct ECN signaling between ECN-nonce supporting endpoints. The minimum response to an incorrect nonce is the same as the response to a received ECE. However, to compensate for hidden congestion signals, the sender might reduce the congestion window to one segment and cease setting ECT in outgoing segments. An incorrect nonce sum is a sign of misbehavior or error between ECN-nonce supporting endpoints.

6.2.1. Using the ECN-nonce to Protect Against Other Misbehaviors

The ECN-nonce can provide robustness beyond checking that marked packets are signaled to the sender. It also ensures that dropped packets cannot be concealed from the sender (because their nonces have been lost). Drops could potentially be concealed by a faulty TCP implementation, certain attacks, or even a hypothetical TCP accelerator. Such an accelerator could gamble that it can either successfully "fast start" to a preset bandwidth quickly, retry with another connection, or provide reliability at the application level. If robustness against these faults is also desired, then the ECN-nonce should not be disabled. Instead, reducing the congestion window to one, or using a low-priority queue, would penalize faulty operation while providing continued checking.

The ECN-nonce can also detect misbehavior in Eifel [Eifel], a recently proposed mechanism for removing the retransmission ambiguity to improve TCP performance. A misbehaving receiver might claim to have received only original transmissions to convince the sender to undo congestion actions. Since retransmissions are sent without ECT, and thus no nonce, returning the correct nonce sum confirms that only original transmissions were received.

7. Interactions

7.1. Path MTU Discovery

As described in RFC3168, use of the Don't Fragment bit with ECN is recommended. Receivers that receive unmarked fragments can reconstruct the original nonce to conceal a marked fragment. The ECN-nonce cannot protect against misbehaving receivers that conceal marked fragments, so some protection is lost in situations where Path MTU discovery is disabled.

When responding to a small path MTU, the sender will retransmit a smaller frame in place of a larger one. Since these smaller packets are retransmissions, they will be ECN-incapable and bear no nonce. The sender should resynchronize on the first newly transmitted packet.

7.2. SACK

Selective acknowledgements allow receivers to acknowledge out of order segments as an optimization. It is not necessary to modify the selective acknowledgment option to fit per-range nonce sums, because SACKs cannot be used by a receiver to hide a congestion signal. The nonce sum corresponds only to the data acknowledged by the cumulative acknowledgement.

7.3. IPv6

Although the IPv4 header is protected by a checksum, this is not the case with IPv6, making undetected bit errors in the IPv6 header more likely. Bit errors that compromise the integrity of the congestion notification fields may cause an incorrect nonce to be received, and an incorrect nonce sum to be returned.

8. Security Considerations

The random one-bit nonces need not be from a cryptographic-quality pseudo-random number generator. A strong random number generator would compromise performance. Consequently, the sequence of random nonces should not be used for any other purpose.

Conversely, the pseudo-random bit sequence should not be generated by a linear feedback shift register [Schneier], or similar scheme that would allow an adversary who has seen several previous bits to infer the generation function and thus its future output.

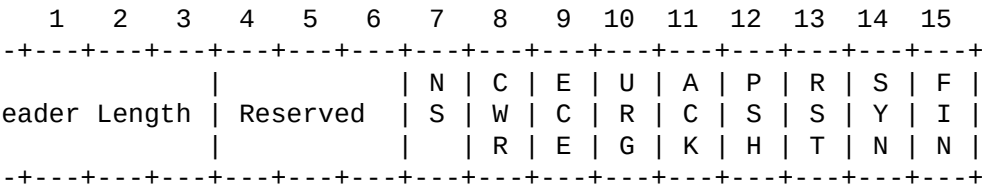
Although the ECN-nonce protects against concealment of congestion signals and optimistic acknowledgement, it provides no additional protection for the integrity of the connection.

9. IANA Considerations

The Nonce Sum (NS) is carried in a reserved TCP header bit that must be allocated. This document describes the use of Bit 7, adjacent to the other header bits used by ECN.

The codepoint for the NS flag in the TCP header is specified by the Standards Action of this RFC, as is required by RFC 2780. The IANA has added the following to the registry for "TCP Header Flags":

RFC 3540 defines bit 7 from the Reserved field to be used for the Nonce Sum, as follows:



TCP Header Flags

Bit	Name	Reference
7	NS (Nonce Sum)	[RFC 3540]

10. Conclusion

The ECN-nonce is a simple modification to the ECN signaling mechanism that improves ECN's robustness by preventing receivers from concealing marked (or dropped) packets. The intent of this work is to help improve the robustness of congestion control in the Internet. The modification retains the character and simplicity of existing ECN signaling. It is also practical for deployment in the Internet. It uses the ECT(0) and ECT(1) codepoints and one TCP header flag (as well as CWR and ECN-Echo) and has simple processing rules.

11. References

[ECN] "The ECN Web Page", URL "http://www.icir.org/floyd/ecn.html".

[RFC3168] Ramakrishnan, K., Floyd, S. and D. Black, "The addition of explicit congestion notification (ECN) to IP", RFC 3168, September 2001.

[Eifel] R. Ludwig and R. Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. Computer Communications Review, January, 2000.

[B97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[Savage] S. Savage, N. Cardwell, D. Wetherall, T. Anderson. TCP congestion control with a misbehaving receiver. SIGCOMM CCR, October 1999.

[Schneier] Bruce Schneier. Applied Cryptography, 2nd ed., 1996

12. Acknowledgements

This note grew out of research done by Stefan Savage, David Ely, David Wetherall, Tom Anderson and Neil Spring. We are very grateful for feedback and assistance from Sally Floyd.

13. Authors' Addresses

Neil Spring
EMail: nspring@cs.washington.edu

David Wetherall
Department of Computer Science and Engineering, Box 352350
University of Washington
Seattle WA 98195-2350
EMail: djw@cs.washington.edu

David Ely
Computer Science and Engineering, 352350
University of Washington
Seattle, WA 98195-2350
EMail: ely@cs.washington.edu

14. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.