Internet Engineering Task Force
INTERNET DRAFT
draft-ietf-tsvwg-tcp-nonce-00.txt

David Wetherall
David Ely
Neil Spring
University of Washington

January, 2001
Expires: July, 2001

Robust ECN Signaling with Nonces

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This note describes a simple modification to ECN that protects against accidental or malicious concealment of marked packets from the TCP sender. This is valuable because it improves the robustness of congestion control by preventing receivers from exploiting ECN to gaining an unfair share of the bandwidth. The mechanism uses a slightly different encoding than the existing two ECN bits in the IP header, and also requires one additional bit in the TCP header. It is computationally efficient for both routers and hosts.

1. Introduction

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 in turn would give the receivers 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 of the central role of congestion control, we feel 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. In this note, we show how this can be achieved while at the same time keeping the protocol simple and efficient.

Our signaling mechanism uses random one bit quantities to allow the sender to verify that the receiver has implemented ECN signaling correctly and that there is no other interference that conceals marked (or dropped) packets in the signaling path. This provides protection against both implementation errors and deliberate abuse. In developing the nonce signaling mechanism we met the following goals:

- It provides an additional check but does not change other aspects of ECN, and nor does it reduce the benefits of ECN for behaving receivers.

- It catches a misbehaving receiver with a high probability, and never convicts an innocent receiver

- It is cheap in terms of per-packet overhead (one TCP header bit more than the existing ECN mechanism) and processing requirements.

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

The rest of this note describes the mechanism, first with an overview and then in terms of more detailed behavior at senders, routers and receivers.

2. Overview of Solution

Our scheme builds on the existing ECN-Echo and CWR signaling mechanism. Familiarity with ECN is assumed in this note. Also, for simplicity, we describe our scheme in one direction only, though it is run in both directions in parallel.

In a nutshell, our approach is to detect misbehavior by attaching a random one bit nonce value to packets at the sender and having acknowledgments from the receiver echo the nonce information that is received. At routers, packet marking becomes the process of erasing the nonce value. Therefore, once a packet has been marked by a router, it cannot be unmarked by any party without successfully guessing the value of the erased nonce. Thus receipt of correct nonce information from the receiver provides the sender with a probabilistic proof-of- receipt check for unmarked packets. The check is used by the TCP sender to verify that the ECN-Echo bit is being set correctly and that congestion indications in the form of marked (or dropped) packets are not being concealed. Because one bit of information is returned with each acknowledgement, senders have a 50-50 chance of catching a lying receiver every time they perform a check. Because the check for each acknowledgement is an independent trial it is highly likely that cheaters will be caught quickly if there are repeated packet marks.

There are several areas of detail missing from the preceding high-level description. We mention those areas to complete the overview.

First, the nonce values are echoed in the form of nonce sums. Each nonce sum is carried in an acknowledgement, and represents the one bit sum (XOR or parity) of nonces over the byte range represented by the acknowledgement. To understand why the sum is used, rather than individual echoes, consider the following argument. If every packet were reliably ACKed, then the nonce carried in the unmarked packet could simply be echoed. This would probabilistically prove to the sender that the receiver received the packet and the packet was unmarked. However, ACKs are not carried across the network reliably, and not every packet is ACKed. In this case, the sender cannot distinguish a lost ACK from one that was never sent in order to conceal a marked packet. It would require additional mechanism, beyond that used in TCP, to convey the nonce bits reliably. Instead, we send the nonce sum that corresponds to the cumulative ACK. This sum prevents individual marked packets from being concealed by not acknowledging them. Note that because they are both one bit quantities, the sum is no easier to guess than the individual nonces.

Second, resynchronization of the sender and receiver sums is needed after congestion has occurred and packets have been marked. When

packets are marked, the nonce is cleared, and the sum of the nonces at the receiver will no longer reflect the sum at the sender. While this is conceptually fixed by having the receiver send a series of partial sums for the ranges of unmarked packets that it has received, this solution is clumsy because the required range information is not already being sent. Fortunately, there is a simple solution that does not require range information because ECN congestion indications do not need to indicate the particular packets that were marked. We observe that once nonces have been lost, the difference between sender and receiver nonce sums will be fixed 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, we suspend checking while the CWR signal is being delivered and acknowledged by the receiver. We reset the sender nonce sum to the receiver sum when new data is acknowledged. This scheme is simple and has the benefit that the receiver is not explicitly involved in the re-synchronization process.

Third, we need to reconcile the nonces that are sent with packet with acknowledgements that cover byte ranges. Acknowledged byte boundaries need not match the transmitted boundaries, and during retransmissions information can be resent with different byte boundaries. To handle these factors, we compute nonces and nonce sums using an underlying mapping of byte ranges to nonce values. Both sender and receiver understand this mapping, and can convert to and from the nonces carried on individual packets.

The next sections describe the detailed behavior of senders, routers and receivers. We start with sender transmit behavior, and work our way around the ECN signaling loop until we arrive back at sender receive behavior. Comments in parenthesis highlight the changes between the nonce mechanism and the existing ECN specification.

3. Sender Behavior (Transmit)

Senders in our scheme 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 on packets as they are received. This section describes the transmit process.

To place a one bit nonce value on every IP packet requires first of all a way to encode these bits in IP packets. We use the following encoding of the ECN bits to identify different packet states. This encoding must be understood by all ECN capable senders, routers, and receivers in our scheme. (The second state below is currently unused

in the existing ECN specification, while the other states retain their existing meanings.)

00 = ECN incapable

10 = ECN capable, unmarked (nonce = 0)

01 = ECN capable, unmarked (nonce = 1)

11 = ECN capable, marked (nonce lost)

Next, we require a simple way to map nonces to transmitted TCP packets in a manner that is compatible with checking the nonce sum on received TCP acknowledgements. This is complicated by several factors. Nonces are sent per packet but acknowledgements cover byte ranges that do not necessarily correspond to the original packet ranges; this can depend on implementation buffering strategies. In the case of retransmissions, the boundary of retransmitted packets need not correspond to the original transmissions either (because of path MTU changes, retransmission batching, and so forth). Finally, there is ambiguity at the sender as to whether the original or retransmitted packet was received. It is important that our implementation behave correctly even in these rare cases so that a receiver is never incorrectly labeled as misbehaving.

We associate nonce values with byte ranges instead of individual packets to avoid these difficulties. Starting from the initial sequence number, each block of SMSS bytes (the maximum segment size) in the TCP byte stream is associated with a single pseudorandom nonce bit. The byte range of the packet determines what nonce value it will carry. If the packet, either original or a retransmission, spans multiple blocks, we use the block in which the final byte of the packet resides to determine which nonce value to transmit with the packet. A series of small packets will carry the same nonce value until an entire block's worth of SMSS bytes has been transmitted. This is a useful tradeoff because sending partial packets makes a flow less likely to cause congestion. Since no packet can carry more than SMSS bytes, each block's nonce bit will be carried in at least one packet.

The following table provides an example of how we decompose a byte stream into blocks and how we assign nonce values to each block (assuming a block size of 1460 bytes):

Bytes:	1...1460	1461...2920	2921 ... 4380
Nonce:	1	1	0
Nonce Sum:	1	0	1

4. Router Behavior

Routers must be able to identify packets sent by ECN capable transports and mark them if indicated by active queue management. To mark packets, routers change either of the unmarked states to the single marked state. (The operation of marking has changed only in that routers now need to recognize two states as meaning not marked and so is still straightforward.) This erases the state of the original nonce carried with the packet, which is key to our scheme. Neither the receiver nor any other party can now unmark the packet without successfully guessing the value of the original nonce.

5. Receiver Behavior (Receive and Transmit)

In addition to distinguishing marked and unmarked packets and setting the ECN- Echo flag as before, receivers in our scheme maintain a nonce sum as packets arrive, and return the sum that corresponds to a particular acknowledgment with the acknowledgment.

To maintain the nonce sum, receivers use the same mapping as the sender to convert the nonces carried in unmarked packets to the nonces of the underlying blocks. These nonce values are summed over the byte range covered by the acknowledgement. Computing this sum correctly when packets of size SMSS are sent requires that all packets up to the one acknowledged be received. New sums are computed by taking the old value and XORing it with a new nonce. That is, the sum is also a one bit quantity, and old nonce state does not need to be maintained.

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 bit in the TCP header. (This bit is in addition the CWR and ECN-Echo bits and would require one of the reserved bits to be allocated.)

These nonce sums are checked for validity at the sender, as described below.

6. Sender Behavior (Receive)

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

Checking is straightforward and is performed every time an acknowledgement is received, except during congestion recovery. Given the byte range covered by an acknowledgement and the mapping between bytes and nonces, the sender is able to compute the correct nonce sum. Minimal sender state is needed to do this because old nonce values can be discarded as acknowledgments and the sum advance. Checking consists of simply comparing the correct nonce sum and that carried in the acknowledgement.

If ECN-Echo is not set, the receiver claims to have received no marked packets, and can therefore compute the correct nonce sum. To cheat, 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 acknowledgement (that covers a new block) is an independent trial, a cheating receiver is highly 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. 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. However, after recovery, it is necessary to re-synchronize the sender and receiver nonce sums so that further acknowledgments can be checked. If might be possible to send the missing nonces to the receiver, but this would be cumbersome because TCP lacks the mechanism to do so conveniently. Instead, we observe that if there are no more marked packets, the sender and receiver sums should differ by a constant amount. 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. In most instances, this will be the first acknowledgement

without the ECN-Echo flag set.

A separate issue is the penalty for misbehavior that is caught by checking. During normal operation, both with and without packet marks and drops, no misbehavior will be uncovered unless some party after the marking router is behaving incorrectly. A simple remedy in this case would be to disable ECN at the sender, that is, not mark packets as ECN capable. This simultaneously deprives the receiver of the benefits of ECN and relieves the sender of the need to monitor the receiver. However, an additional consideration is that the nonce checking mechanism provides 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 a TCP accelerator willing to 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 considered valuable (as opposed to simply detecting a faulty ECN implementation) then it is not clear that the nonce mechanism should be turned off. Instead, a penalty such as reducing the congestion window by a factor of 4 may be preferable. This would provide continued checking while punishing faulty operation. Luckily, this issue is separate from the checking mechanism and does not need to be handled uniformly by senders.

7. Conclusion

We have described a simple modification to the ECN signaling mechanism that improves its 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 is retains the character and simplicity of existing ECN signaling. It is also practical for deployment in the Internet. It requires two bits in the IP header (ECT and CE with a slightly different encoding) and one additional bit in the TCP header (as well as CWR and ECN-Echo) and has simple processing rules.

Acknowledgements

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

Authors' Addresses

David Wetherall
Email: djw@cs.washington.edu
Phone +1 (206) 616 4367

David Ely
Email: ely@cs.washington.edu

Neil Spring
Email: nspring@cs.washington.edu

Computer Science and Engineering, 352350
University of Washington
Seattle, WA 98195-2350

Send comments by electronic mail to all three authors.

This draft was created in January 2001.
It expires July 2001.