                 TCP Modifications for Congestion Exposure (ConEx)

Abstract

   Congestion Exposure (ConEx) is a mechanism by which senders inform
   the network about expected congestion based on congestion feedback
   from previous packets in the same flow. This document describes the
   necessary modifications to use ConEx with the Transmission Control
   Protocol (TCP).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   Congestion Exposure (ConEx) is a mechanism by which senders inform
   the network about expected congestion based on congestion feedback
   from previous packets in the same flow. ConEx concepts and use cases
   are further explained in [RFC6789]. The abstract ConEx mechanism is
   explained in [RFC7713]. This document describes the necessary
   modifications to use ConEx with the Transmission Control Protocol
   (TCP).

   The markings for ConEx signaling are defined in the ConEx Destination
   Option (CDO) for IPv6 [RFC7837]. Specifically, the use of four flags
   is defined: X (ConEx-capable), L (loss experienced), E (ECN
   experienced), and C (credit).

   ConEx signaling is based on the use of either loss or Explicit
   Congestion Notification (ECN) marks [RFC3168] as congestion
   indication. The sender collects this congestion information based on
   existing TCP feedback mechanisms from the receiver to the sender. No
   changes are needed at the receiver side to implement ConEx signaling.
   Therefore, no additional negotiation is needed to implement and use
   ConEx at the sender side. This document specifies the sender's
   actions that are needed to provide meaningful ConEx information to
   the network.

   Section 2 provides an overview of the modifications needed for TCP
   senders to implement ConEx. First, congestion information has to be
   extracted from TCP's loss or ECN feedback as described in Section 3.
   Section 4 details how to set the CDO marking based on this congestion
   information. Section 5 discusses the loss of packets carrying ConEx
   information. Section 6 discusses the timeliness of the ConEx
   feedback signal, given that congestion is a temporary state.

   This document describes congestion accounting for TCP with and
   without the Selective Acknowledgement (SACK) extension [RFC2018] (in
   Section 3.1). However, ConEx benefits from the more accurate
   information that SACK provides about the number of bytes dropped in
   the network, and it is therefore preferable to use the SACK extension
   when using TCP with ConEx. The detailed mechanism to set the L flag
   in response to the loss-based congestion feedback signal is given in
   Section 4.1.

   While loss has to be minimized, ECN can provide more fine-grained
   feedback information. ConEx-based traffic measurement or management
   mechanisms could benefit from this. Unfortunately, the current ECN
   feedback mechanism does not reflect multiple congestion markings if
   they occur within the same Round-Trip Time (RTT). A more accurate

feedback extension to ECN (AccECN) is proposed in a separate document
[ACCURATE], as this is also useful for other mechanisms.

Congestion accounting for both classic ECN feedback and AccECN
feedback is explained in detail in Section 3.2. Setting the E flag
in response to ECN-based congestion feedback is again detailed in
Section 4.1.

1.1. Requirements Language

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].

2. Sender-Side Modifications

This section gives an overview of actions that need to be taken by a
TCP sender modified to use ConEx signaling.

In the TCP handshake, a ConEx sender MUST negotiate for SACK and ECN
preferably with AccECN feedback. Therefore, a ConEx sender MUST also
implement SACK and ECN. Depending on the capability of the receiver,
the following operation modes exist:

o SACK-accECN-ConEx (SACK and accurate ECN feedback)

o SACK-ECN-ConEx (SACK and classic instead of accurate ECN)

o accECN-ConEx (no SACK but accurate ECN feedback)

o ECN-ConEx (no SACK and no accurate ECN feedback, but classic ECN)

o SACK-ConEx (SACK but no ECN at all)

o Basic-ConEx (neither SACK nor ECN)

A ConEx sender MUST expose all congestion information to the network
according to the congestion information received by ECN or based on
loss information provided by the TCP feedback loop. A TCP sender
SHOULD count congestion byte-wise (rather than packet-wise; see next
paragraph). After any congestion notification, a sender MUST mark
subsequent packets with the appropriate ConEx flag in the IP header.
Furthermore, a ConEx sender must send enough credit to cover all
experienced congestion for the connection so far, as well as the risk
of congestion for the current transmission (see Section 4.2).

With SACK the number of lost payload bytes is known, but not the
number of packets carrying these bytes. With classic ECN only an
indication is given that a marking occurred, but not the exact number
of payload bytes nor packets. As network congestion is usually byte-
congestion [RFC7141], the byte-size of a packet marked with a CDO
flag is defined to represent that number of bytes of congestion
signaling [RFC7837]. Therefore, the exact number of bytes should be
taken into account, if available, to make the ConEx Signal as exact
as possible.

Detailed mechanisms for congestion counting in each operation mode
are described in the next section.

3. Counting Congestion

A ConEx TCP sender maintains two counters: one that counts congestion
based on the information retrieved by loss detection, and a second
that accounts for ECN-based congestion feedback. These counters hold
the number of outstanding bytes that should be ConEx-Marked with,
respectively, the E flag or the L flag in subsequent packets.

The outstanding bytes for congestion indications based on loss are
maintained in the Loss Exposure Gauge (LEG), as explained in
Section 3.1.

The outstanding bytes counted based on ECN feedback information are
maintained in the Congestion Exposure Gauge (CEG), as explained in
Section 3.2.

When the sender sends a ConEx-capable packet with the E or L flag
set, it reduces the respective counter by the byte-size of the
packet. This is explained for both counters in Section 4.1.

Note that all bytes of an IP packet must be counted in the LEG or CEG
to capture the right number of bytes that should be marked.
Therefore, the sender SHOULD take the payload and headers into
account, up to and including the IP header. However, in TCP the
information regarding how large the headers of a lost or marked
packet were is usually not available, as only payload data will be
acknowledged.

If equal-sized packets, or at least equally distributed packet sizes,
can be assumed, the sender MAY only add and subtract TCP payload
bytes. In this case, there should be about the same number of ConEx-
Marked packets as the original packets that were causing the
congestion. Thus, both contain about the same number of header bytes
so they will cancel out. This case is assumed for simplicity in the
following sections.

Otherwise, if a sender sends different sized packets (with unequally
distributed packet sizes), the sender needs to memorize or estimate
the number of lost or ECN-marked packets. If the sender has
sufficient memory available, the most accurate way to reconstruct the
number of lost or marked packets is to remember the sequence number
of all sent but not acknowledged packets. In this case, a sender is
able to reconstruct the number of packets, and thus the header bytes
that were sent during the last RTT. Otherwise (e.g., if not enough
memory is available), the sender would need to estimate the packet
size. The average packet size can be estimated if the distribution
pattern of packet sizes in the last RTT is known; alternatively, the
minimum packet size seen in the last RTT can be used as the most
conservative estimate.

If the number of newly sent-out packets with the ConEx L or E flag
set is smaller (or larger) than this estimated number of lost/ECN-
marked packets, the additional header bytes should be added to (or
can be subtracted from) the respective gauge.

3.1. Loss Detection

This section applies whether or not SACK support is available. The
following subsection (Section 3.1.1) handles the case when SACK is
not available.

A TCP sender detects losses and subsequently retransmits the lost
data. Therefore, the ConEx sender can simply set the ConEx L flag on
all retransmissions in order to at least cover the amount of bytes
lost. If this approach is taken, no LEG is needed.

However, any retransmission may be spurious. In this case, more
bytes have been marked than necessary. To compensate for this
effect, a ConEx sender can maintain a local signed counter (the LEG)
that indicates the number of outstanding bytes to be sent with the
ConEx L flag and also can become negative.

Using the LEG, when a TCP sender decides that a data segment needs to
be retransmitted, it will increase the LEG by the size of the TCP
payload bytes in the retransmission (assuming equal sized segments
such that the retransmitted packet will have the same number of
header bytes as the original ones):

For each retransmission:

LEG += payload

Note how the LEG is reduced when the ConEx L marking is set as
described in Section 4.

Further, to accommodate spurious retransmissions, a ConEx sender
SHOULD make use of heuristics to detect such spurious retransmissions
(e.g., F-RTO [RFC5682], DSACK [RFC3708], and Eifel [RFC3522],
[RFC4015]), if already available in a given implementation. If no
mechanism for detecting spurious retransmissions is available, the
ConEx sender MAY chose to implement one of the mechanisms stated
above. However, given the inaccuracy that ConEx may have anyway and
the timeliness of ConEx information, a ConEx MAY also chose not to
compensate for spurious retransmission. In this case, if spurious
retransmissions occur, the ConEx sender has simply sent too many
ConEx Signals which, e.g., would decrease the congestion allowance in
a ConEx policer unnecessarily.

If a heuristic method is used to detect spurious retransmission and
has determined that a certain number of packets were retransmitted
erroneously, the ConEx sender subtracts the payload size of these TCP
packets from LEG.

If a spurious retransmission is detected:

LEG -= payload

Note that LEG can become negative if too many L markings have already
been sent. This case is further discussed in Section 6.

3.1.1. Without SACK Support

If multiple losses occur within one RTT and SACK is not used, it may
take several RTTs until all lost data is retransmitted. With the
scheme described above, the ConEx information will be delayed
considerably, but timeliness is important for ConEx. For ConEx, it
is important to know how much data was lost; it is not important to
know what data is lost. During the first RTT after the initial loss
detection, the amount of received data, and thus also the amount of
lost data, can be estimated based on the number of received ACKs.

Therefore, a ConEx sender can use the following algorithm to
estimated the number of lost bytes with an additional delay of one
RTT using an additional Loss Estimation Counter (LEC):

        flight_bytes: current flight size in bytes
        retransmit_bytes: payload size of the retransmission

At the first retransmission in a congestion event, LEC is set:

        LEC = flight_bytes - 3*SMSS

        (At this point in the transmission, in the worst case,
        all packets in flight minus three that triggered the dupACks
        could have been lost.)

    Then, during the first RTT of the congestion event:

        For each retransmission:
           LEG += retransmit_bytes
           LEC -= retransmit_bytes

        For each ACK:
           LEC -= SMSS

    After one RTT:

        LEG += LEC

        (The LEC now estimates the number of outstanding bytes
        that should be ConEx L-marked.)

    After the first RTT for each following retransmissions:

        if (LEC > 0): LEC -= retransmit_bytes
        else if (LEC==0): LEG += retransmit_bytes

        if (LEC < 0): LEG += -LEC

        (The LEG is not increased for those bytes that were
        already counted.)

3.2. Explicit Congestion Notification (ECN)

   ECN [RFC3168] is an IP/TCP mechanism that allows network nodes to
   mark packets with the Congestion Experienced (CE) mark instead of
   dropping them when congestion occurs.

   A receiver might support classic ECN, the more accurate ECN feedback
   scheme (AccECN), or neither. In the case that ECN is not supported
   for a connection, of course no ECN marks will occur; thus, the sender
   will never set the E flag. Otherwise, a ConEx sender needs to
   maintain a signed counter, the Congestion Exposure Gauge (CEG), for
   the number of outstanding bytes that have to be ConEx-Marked with the
   E flag.

The CEG is increased when ECN information is received from an ECN-
capable receiver supporting the classic ECN scheme or the accurate
ECN feedback scheme. When the ConEx sender receives an ACK
indicating one or more segments were received with a CE mark, CEG is
increased by the appropriate number of bytes as described further
below.

Unfortunately, in case of duplicate acknowledgements, the number of
newly acknowledged bytes will be zero even though (CE-marked) data
has been received. Therefore, we increase the CEG by DeliveredData,
as defined below:

DeliveredData = acked_bytes + SACK_diff + (is_dup)*1SMSS -
(is_after_dup)*num_dup*1SMSS

DeliveredData covers the number of bytes that has been newly
delivered to the receiver. Therefore, on each arrival of an ACK,
DeliveredData will be increased by the newly acknowledged bytes
(acked_bytes) as indicated by the current ACK, relative to all past
ACKs. The formula depends on whether SACK is available: if SACK is
not available, SACK_diff is always zero, whereas if ACK information
is available, is_dup and is_after_dup are always zero.

With SACK, DeliveredData is increased by the number of bytes provided
by (new) SACK information (SACK_diff). Note that if less
unacknowledged bytes are announced in the new SACK information than
in the previous ACK, SACK_diff can be negative. In this case, data
is newly acknowledged (in acked_bytes) that was previously
accumulated into DeliveredData, based on SACK information.

Otherwise without SACK, DeliveredData is increased by 1 Sender
Maximum Segment Size (SMSS) on duplicate acknowledgements because
duplicate acknowledgements do not acknowledge any new data (and
acked_bytes will be zero). For the subsequent partial or full ACK,
acked_bytes cover all newly acknowledged bytes including those
already accounted for with the receipt of any duplicate
acknowledgement. Therefore, DeliveredData is reduced by one SMSS for
each preceding duplicate ACK. Consequently, is_dup is one if the
current ACK is a duplicated ACK without SACK, and zero otherwise.
is_after_dup is only one for the next full or partial ACK after a
number of duplicated ACKs without SACK and num_dup counts the number
of duplicated ACKs in a row (which usually is 3 or more).

With classic ECN, one congestion-marked packet causes continuous
congestion feedback for a whole round trip, thus hiding the arrival
of any further congestion-marked packets during that round trip. A
more accurate ECN feedback scheme (AccECN) is needed to ensure that
feedback properly reflects the extent of congestion marking. The two

cases, with and without a receiver capable of AccECN, are discussed
in the following sections.

3.2.1. Accurate ECN Feedback

With a more accurate ECN feedback scheme (AccECN) that is supported
by the receiver, either the number of marked packets or the number of
marked bytes will be fed back from the receiver to the sender and,
therefore is known at the sender side. In the latter case, the CEG
can be increased directly by the number of marked bytes. Otherwise
if D is assumed to be the number of marks, the gauge (CEG) will be
conservatively increased by one SMSS for each marking or, at the
maximum, the number of newly acknowledged bytes:

CEG += min(SMSS*D, DeliveredData)

3.2.2. Classic ECN Support

With classic ECN, as soon as a CE mark is seen at the receiver side,
it will feed this information back to the sender by setting the Echo
Congestion Experienced (ECE) flag in the TCP header of subsequent
ACKs. Once the sender receives the first ECE of a congestion
notification, it sets the Congestion Window Reduced (CWR) flag in the
TCP header once. When this packet with the CWR flag in the TCP
header arrives at the receiver side acknowledging its first ECE
feedback, the receiver stops setting the ECE flag.

If the ConEx sender fully conforms to the semantics of ECN signaling
as defined by [RFC3168], it will receive one full RTT of ACKs with
the ECE flag set whenever at least one CE mark was received by the
receiver. As the sender cannot estimate how many packets have
actually been CE-marked during this RTT, the most conservative
assumption MAY be taken, namely assuming that all packets were
marked. This can be achieved by increasing the CEG by DeliveredData
for each ACK with the ECE flag:

CEG += DeliveredData

Optionally, a ConEx sender could implement the following technique
(that does not conform to [RFC3168]), called "advanced compatibility
mode", to considerably improve its estimate of the number of ECN-
marked packets:

To extract more than one ECE indication per RTT, a ConEx sender could
set the CWR flag continuously to force the receiver to signal only
one ECE per CE mark. Unfortunately, the use of delayed ACKs
[RFC5681] (which is common) will prevent feedback of every CE mark;
if a CWR confirmation is received before the ECE can be sent out on

the next ACK, ECN feedback information could get lost (depending on
the actual receiver implementation). Thus, a sender SHOULD set CWR
only on those data segments that will presumably trigger a (delayed)
ACK. The sender would need an additional control loop to estimate
which data segments will trigger an ACK in order to extract more
timely congestion notifications. Still, the CEG SHOULD be increased
by DeliveredData, as one or more CE-marked packets could be
acknowledged by one delayed ACK.

4. Setting the ConEx Flags

By setting the X flag, a packet is marked as ConEx-capable. All
packets carrying payload MUST be marked with the X flag set,
including retransmissions. Only if no congestion feedback
information is (currently) available, SHOULD the X flag be zero
(e.g., for control packets on a connection that has not sent any user
data for some time and, therefore is sending only pure ACKs that are
not carrying any payload).

4.1. Setting the E or the L Flag

As described in Section 3.1, the sender needs to maintain a CEG
counter and might also maintain a LEG counter. If no LEG is used,
all retransmission will be marked with the L flag.

Further, as long as the LEG or CEG counter is positive, the sender
marks each ConEx-capable packet with L or E respectively, and
decreases the LEG or CEG counter by the TCP payload bytes carried in
the marked packet (assuming headers are not being counted because
packet sizes are regular). No matter how small the value of LEG or
CEG, if the value is positive the sender MUST NOT defer packet
marking; this ensures that ConEx Signals are timely. Therefore, the
value of LEG and CEG will commonly be negative.

If both the LEG and CEG are positive, the sender MUST mark each
ConEx-capable packet with both L and E. If a credit signal is also
pending (see the next section), the C flag can be set as well.

4.2. Setting the Credit Flag

The ConEx abstract mechanism [RFC7713] requires that sufficient
credit MUST be signaled in advance to cover the expected congestion
during the feedback delay of one RTT.

To monitor the credit state at the audit, a ConEx sender needs to
maintain a Credit State Counter (CSC) in bytes. If congestion
occurs, credits will be consumed and the CSC is reduced by the number
of bytes that were lost or estimated to be ECN-marked. If the risk

of congestion was estimated wrongly, and thus too few credits were
sent, the CSC becomes zero but cannot go negative.

To be sure that the credit state in the audit never reaches zero, the
number of credits should always equal the number of bytes in flight
as all packets could potentially get lost or congestion-marked. In
this case, a ConEx sender also monitors the number of bytes in flight
F. If F ever becomes larger than the CSC, the ConEx sender sets the
C flag on each ConEx-capable packet and increases the CSC by the
payload size of each marked packet until the CSC is no less than F
again. However, a ConEx sender might also be less conservative and
send fewer credits if it, e.g., assumes that the congestion will be
low on a certain path based on previous experience.

Recall that the CSC will be decreased whenever congestion occurs;
therefore the CSC will need to be replenished as soon as the CSC
drops below F. Also recall that the sender can set the C flag on a
ConEx-capable packet whether or not the E or L flags are also set.

In TCP Slow Start, the congestion window might grow much larger than
during the rest of the transmission. Likely, a sender could consider
sending fewer than F credits but risking being penalized by an audit
function. However, the credits should at least cover the increase in
sending rate. Given the exponential increase as implemented in the
TCP Slow Start algorithm, which means that the sending rate doubles
every RTT, a ConEx sender should at least cover half the number of
packets in flight by credits.

Note that the number of losses or markings within one RTT does not
depend solely on the sender's actions. In general, the behavior of
the cross traffic, whether Active Queue Management (AQM) is used and
how it is parameterized influence how many packets might be dropped
or marked. As long as any AQM encountered is not overly aggressive
with ECN marking, sending half the flight size as credits should be
sufficient whether congestion is signaled by loss or ECN.

To maintain half of the packets in flight as credits, half of the
packet of the initial window must also be C-marked. In Slow Start
marking, every fourth packet introduces the correct amount of credit
as can be seen in Figure 1.

```
                                       in_flight  credits
            RTT1  |------XC------>|        1         1
                  |------X------->|        2         1
                  |------XC------>|        3         2
                  |              |
            RTT2  |------X------->|        3         2
                  |------X------->|        4         2
                  |------X------->|        4         2
                  |------XC------>|        5         3
                  |------X------->|        5         3
                  |------X------->|        6         3
                  |              |
            RTT3  |------X------->|        6         3
                  |------XC------>|        7         4
                  |------X------->|        7         4
                  |------X------->|        8         4
                  |------X------->|        8         4
                  |------XC------>|        9         5
                  |------X------->|        9         5
                  |------X------->|       10         5
                  |------X------->|       10         5
                  |------XC------>|       11         6
                  |------X------->|       11         6
                  |------X------->|       12         6
                  |      .       |
                  |      :       |
```

       Figure 1: Credits in Slow Start (with an initial window of 3)

   It is possible that a TCP flow will encounter an audit function
   without relevant flow state due to, e.g., rerouting or memory
   limitations. Therefore, the sender needs to detect this case and
   resend credits. A ConEx sender might reset the credit counter CSC to
   zero if losses occur in subsequent RTTs (assuming that the sending
   rate was correctly reduced based on the received congestion signal
   and using a conservatively large RTT estimation).

   This section proposes a concrete algorithm for determining how much
   credit to signal (with a separate approach used for Slow Start).
   However, experimentation in credit setting algorithms is expected and
   encouraged. The wider goal of ConEx is to reflect the "cost" of the
   risk of causing congestion on those that contribute most to it.
   Thus, experimentation is encouraged to improve or maintain
   performance while reducing the risk of causing congestion and,
   therefore potentially reducing the need to signal so much credit.

5. Loss of ConEx Information

   Packets carrying ConEx Signals could be discarded themselves. This
   will be a second order problem (e.g., if the loss probability is
   0.1%, the probability of losing a ConEx L signal will be 0.1% of 0.1%
   = 0.01%). Further, the penalty an audit induces should be
   proportional to the mismatch of expected ConEx marks and observed
   congestion, therefore the audit might only slightly increase the loss
   level of this flow. Therefore, an implementer MAY choose to ignore
   this problem, accepting instead the risk that an audit function might
   wrongly penalize a flow.

   Nonetheless, a ConEx sender is responsible for always signaling
   sufficient congestion feedback, and therefore SHOULD remember which
   packet was marked with either the L, the E, or the C flag. If one of
   these packets is detected as lost, the sender SHOULD increase the
   respective gauge(s), LEG or CEG, by the number of lost payload bytes
   in addition to increasing LEG for the loss.

6. Timeliness of the ConEx Signals

   ConEx Signals will only be useful to a network node within a time
   delay of about one RTT after the congestion occurred. To avoid
   further delays, a ConEx sender SHOULD send the ConEx signaling on the
   next available packet.

   Any or all of the ConEx flags can be used in the same packet, which
   allows delays to be minimized when multiple signals are pending. The
   need to set multiple ConEx flags at the same time can occur if, e.g,
   an ACK is received by the sender that simultaneously indicates that
   at least one ECN mark was received, and that one or more segments
   were lost. This may happen during excessive congestion, if the
   queues overflow even though ECN was used and currently all forwarded
   packets are marked, while others have to be dropped. Another case
   when this might happen is when ACKs are lost, so that a subsequent
   ACK carries summary information not previously available to the
   sender.

   If a flow becomes application-limited, there could be insufficient
   bytes to send to reduce the gauges to zero or below. In such cases,
   the sender cannot help but delay ConEx Signals. Nonetheless, as long
   as the sender is marking all outgoing packets, an audit function is
   unlikely to penalize ConEx-Marked packets. Therefore, no matter how
   long a gauge has been positive, a sender MUST NOT reduce the gauge by
   more than the ConEx-Marked bytes it has sent.

If the CEG or LEG counter is negative, the respective counter MAY be
reset to zero within one RTT after it was decreased the last time, or
one RTT after recovery if no further congestion occurred.

7. Open Areas for Experimentation

All proposed mechanisms in this document are experimental, and
therefore further large-scale experimentation on the Internet is
required to evaluate if the signaling provided by these mechanisms is
accurate and timely enough to produce value for ConEx-based (traffic
management or other) mechanisms.

The current ConEx specifications assume that congestion is counted in
the number of bytes (including the IP header that directly
encapsulates the CDO and everything that the IP header encapsulates)
[RFC7837]. This decision was taken because most network devices
today experience byte-congestion where the memory is filled exactly
with the number of bytes a packet carries [RFC7141]. However, there
are also devices that may allocate a certain amount of memory per
packet, no matter how large a packet is. These devices get congested
based on the number of packets in their memory and therefore, in this
case, congestion is determined by the number of packets that have
been lost or marked. Furthermore, a transport-layer endpoint such as
a TCP sender or receiver, might not know the exact number of bytes
that a lower layer was carrying. Therefore, a TCP endpoint may only
be able to estimate the exact number of congested bytes (assuming
that all lower-layer headers have the same length). If this
estimation is sufficient to work with, the ConEx Signal needs to be
further evaluated in tests on the Internet together with different
auditor implementations.

Further, the proposed marking schemes in this document are designed
under the assumption that all TCP packets of a ConEx-capable flow are
of equal size or that flows have a constant mean packet size over a
rather small time frame, like one RTT or less. In most
implementations, this assumption might be taken as well and is
probably true for most of the traffic flows. If this proposed scheme
is used, it is necessary to evaluate how much accuracy degrades if
this precondition is not met. Evaluating with real traffic from
different applications is especially important in making the decision
regarding whether the proposed schemes are sufficient or whether a
more complex scheme is needed.

In this context, the proposed scheme to set credit markings in Slow
Start runs the risk of providing an insufficient number of markings,
which can cause an audit function to penalize this flow. Both the
proposed credit scheme for Slow Start as well as the scheme in
Congestion Avoidance must be evaluated together with one or more

specific implementations of a ConEx auditor to ensure that both
algorithms, in the sender and in the auditor, work properly together
with a low risk of false positives (which would lead to penalization
of an honest sender). However, if a sender is wrongly assumed to
cheat, the penalization of the audit should be adequate and should
allow an honest sender using a congestion control scheme that is
commonly used today to recover quickly.

Another open issue is the accuracy of the ECN feedback signal. At
the time of this document's publication, there is no AccECN mechanism
specified yet, and further AccECN will also take some time to be
widely deployed. This document proposes an advanced compatibility
mode for classic ECN. The proposed mechanism can provide more
accurate feedback by utilizing the way classic ECN is specified but
has a higher risk of losing information. To figure out how high this
risk is in a real deployment scenario, further experimental
evaluation is needed. The following argument is intended to prove
that suppressing repetitions of ECE, however, is still safe against
possible congestion collapse due to lost congestion feedback and
should be further proven in experimentation:

Repetition of ECE in classic ECN is intended to ensure reliable
delivery of congestion feedback. However, with advanced
compatibility mode, it is possible to miss congestion notifications.
This can happen in some implementations if delayed acknowledgements
are used. Further, an ACK containing ECE can simply get lost. If
only a few CE marks are received within one congestion event (e.g.,
only one), the loss of one acknowledgement due to (heavy) congestion
on the reverse path can prevent that any congestion notification is
received by the sender.

However, if loss of feedback exacerbates congestion on the forward
path, more forward packets will be CE-marked, increasing the
likelihood that feedback from at least one CE will get through per
RTT. As long as one ECE reaches the sender per RTT, the sender's
congestion response will be the same as if CWR were not continuous.
The only way that heavy congestion on the forward path could be
completely hidden would be if all ACKs on the reverse path were lost.
If total ACK loss persisted, the sender would time out and do a
congestion response anyway. Therefore, the problem seems confined to
potential suppression of a congestion response during light
congestion.

Furthermore, even if loss of all ECN feedback leads to no congestion
response, the worst that could happen would be loss instead of ECN-
signaled congestion on the forward path. Given that compatibility
mode does not affect loss feedback, there would be no risk of
congestion collapse.

8. Security Considerations

   General ConEx security considerations are covered extensively in the
   ConEx abstract mechanism [RFC7713]. This section covers TCP-specific
   concerns that may occur with the addition of ConEx to TCP (while not
   discussing generally well-known attacks against TCP). It is assumed
   that any altering of ConEx information can be detected by protection
   mechanisms in the IP layer and is, therefore, not discussed here but
   in [RFC7837]. Further, [RFC7837] describes how to use ConEx to
   mitigate flooding attacks by using preferential drop where the use of
   ConEx can even increase security.

   The ConEx modifications to TCP provide no mechanism for a receiver to
   force a sender not to use ConEx. A receiver can degrade the accuracy
   of ConEx by claiming that it does not support SACK, AccECN, or ECN,
   but the sender will never have to turn ConEx off. Further, the
   receiver cannot force the sender to have to mark ConEx more
   conservatively, in order to cover the risk of any inaccuracy.
   Instead, it is always the sender's choice to either mark very
   conservatively, which ensures that the audit always sees enough
   markings to not penalize the flow, or estimate the needed number of
   markings more tightly. This second case can lead to inaccurate
   marking, and therefore increases the likelihood of loss at an audit
   function that will only harm the receiver itself.

   Assuming the sender is limited in some way by a congestion allowance
   or quota, a receiver could spoof more loss or ECN congestion feedback
   than it actually experiences, in an attempt to make the sender draw
   down its allowance faster than necessary. However, over-declaring
   congestion simply makes the sender slow down. If the receiver is
   interested in the content, it will not want to harm its own
   performance.

   However, if the receiver is solely interested in making the sender
   draw down its allowance, the net effect will depend on the sender's
   congestion control algorithm as permanently adding more and more
   additional congestion would cause the sender to more and more reduce
   its sending rate. Therefore, a receiver can only maintain a certain
   congestion level that is corresponding to a certain sending rate.
   With NewReno [RFC6582], doubling congestion feedback causes the
   sender to reduce its sending rate such that it would only consume
   sqrt(2) = 1.4 times more congestion allowance. However, to improve
   scaling, congestion control algorithms are tending towards less
   responsive algorithms like Cubic or Compound TCP, and ultimately to
   linear algorithms like Data Center TCP (DCTCP) [DCTCP] that aim to
   maintain the same congestion level independent of the current sending
   rate and always reduce its sending window if the signaled congestion
   feedback is higher. In each case, if the receiver doubles congestion

feedback, it causes the sender to respectively consume more allowance
by a factor of 1.2, 1.15, or 1, where 1 implies the attack has become
completely ineffective as no further congestion allowance is consumed
but the flow will decrease its sending rate to a minimum instead.

9. References

9.1. Normative References

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

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

   [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition
              of Explicit Congestion Notification (ECN) to IP",
              RFC 3168, DOI 10.17487/RFC3168, September 2001,
              <http://www.rfc-editor.org/info/rfc3168>.

   [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion
              Control", RFC 5681, DOI 10.17487/RFC5681, September 2009,
              <http://www.rfc-editor.org/info/rfc5681>.

   [RFC7713] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx)
              Concepts, Abstract Mechanism, and Requirements", RFC 7713,
              DOI 10.17487/RFC7713, December 2015,
              <http://www.rfc-editor.org/info/rfc7713>.

   [RFC7837] Krishnan, S., Kuehlewind, M., Briscoe, B., and C. Ralli,
              "IPv6 Destination Option for Congestion Exposure (ConEx)",
              RFC 7837, DOI 10.17487/RFC7837, May 2016,
              <http://www.rfc-editor.org/info/rfc7837>.

9.2. Informative References

   [ACCURATE] Briscoe, B., Kuehlewind, M., and R. Scheffenegger, "More
              Accurate ECN Feedback in TCP", Work in Progress,
              draft-ietf-tcpm-accurate-ecn-00, December 2015.

   [DCTCP]   Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel,
              P., Prabhakar, B., Sengupta, S., and M. Sridharan, "Data
              Center TCP (DCTCP)", ACM SIGCOMM Computer Communication
              Review, Volume 40, Issue 4, pages 63-74,
              DOI 10.1145/1851182.1851192, October 2010,
              <http://portal.acm.org/citation.cfm?id=1851192>.

   [ECNTCP]   Briscoe, B., Jacquet, A., Moncaster, T., and A. Smith,
              "Re-ECN: Adding Accountability for Causing Congestion to
              TCP/IP", Work in Progress, draft-briscoe-conex-re-ecn-
              tcp-04, July 2014.

   [RFC3522] Ludwig, R. and M. Meyer, "The Eifel Detection Algorithm
              for TCP", RFC 3522, DOI 10.17487/RFC3522, April 2003,
              <http://www.rfc-editor.org/info/rfc3522>.

   [RFC3708] Blanton, E. and M. Allman, "Using TCP Duplicate Selective
              Acknowledgement (DSACKs) and Stream Control Transmission
              Protocol (SCTP) Duplicate Transmission Sequence Numbers
              (TSNs) to Detect Spurious Retransmissions", RFC 3708,
              DOI 10.17487/RFC3708, February 2004,
              <http://www.rfc-editor.org/info/rfc3708>.

   [RFC4015] Ludwig, R. and A. Gurtov, "The Eifel Response Algorithm
              for TCP", RFC 4015, DOI 10.17487/RFC4015, February 2005,
              <http://www.rfc-editor.org/info/rfc4015>.

   [RFC5682] Sarolahti, P., Kojo, M., Yamamoto, K., and M. Hata,
              "Forward RTO-Recovery (F-RTO): An Algorithm for Detecting
              Spurious Retransmission Timeouts with TCP", RFC 5682,
              DOI 10.17487/RFC5682, September 2009,
              <http://www.rfc-editor.org/info/rfc5682>.

   [RFC6582] Henderson, T., Floyd, S., Gurtov, A., and Y. Nishida, "The
              NewReno Modification to TCP's Fast Recovery Algorithm",
              RFC 6582, DOI 10.17487/RFC6582, April 2012,
              <http://www.rfc-editor.org/info/rfc6582>.

   [RFC6789] Briscoe, B., Ed., Woundy, R., Ed., and A. Cooper, Ed.,
              "Congestion Exposure (ConEx) Concepts and Use Cases",
              RFC 6789, DOI 10.17487/RFC6789, December 2012,
              <http://www.rfc-editor.org/info/rfc6789>.

   [RFC7141]   Briscoe, B. and J. Manner, "Byte and Packet Congestion
               Notification", BCP 41, RFC 7141, DOI 10.17487/RFC7141,
               February 2014, <http://www.rfc-editor.org/info/rfc7141>.

Authors' Addresses

   Mirja Kuehlewind (editor)
   ETH Zurich
   Switzerland

   Email: mirja.kuehlewind@tik.ee.ethz.ch

   Richard Scheffenegger
   NetApp, Inc.
   Am Euro Platz 2
   Vienna 1120
   Austria

   Email: rs.ietf@gmx.at