TCP modifications for Congestion Exposure
draft-ietf-conex-tcp-modifications-07

Abstract

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

   The needed markings to provide ConEx signaling are defined in the
   ConEx Destination Option (CDO) for IPv6 [draft-ietf-conex-destopt].
   Specifically, the use of four bits are defined: the X (ConEx-
   capable), the L (loss experienced), the E (ECN experienced) and C
   (credit) bit.

   ConEx signaling is based on loss or Explicit Congestion Notification
   (ECN) marks [RFC3168] as a congestion indication. This congestion
   information is retrieved by the sender based on existing feedback
   mechanisms from the receiver to the sender in TCP. No changes are
   needed at the receiver to implement ConEx signaling. Therefore no
   additional negotiation is needed to implement and use ConEx at the

sender. This document specifies actions needed by sender to provide
meaningful ConEx information to the network.

Section 2 provides an overview of the needed modifications for TCP
senders to implement ConEx. First congestion information have to be
extracted from loss or ECN feedback in TCP as described in section 3.
Section 4 details how to set the CDO marking based on the accounted
congestion information. Section 6 finally discusses timeliness of
the ConEx feedback signal as congestion is a temporary state.

This document describes congestion accounting for both TCP with and
without the Selective Acknowledgment (SACK) extension [RFC2018] in
section 3.1. However, ConEx benefits from more accurate information
about the number of packets dropped in the network. It is therefore
recommended to use the SACK extension when using TCP with ConEx. The
detailed mechanism to respectively set the L bit in response to loss-
based congestion feedback signal is given in section 4.1.

While loss-based congestion feedback should be minimized, ECN could
actually provide more fine-grained feedback information. ConEx-based
traffic measurement or management mechanisms would benefit from this.
Unfortunately, the current ECN feedback mechanism does not reflect
multiple congestion markings which occur within the same Round-Trip
Time (RTT). A more accurate feedback extension to ECN is proposed in
a separate document [draft-kuehlewind-tcpm-accurate-ecn], as this is
also useful for other mechanisms.

The congestion accounting for both, with the classic ECN feedback as
well as a more accurate ECN feedback are explained in detail in
section 3.2 while the setting of the E bit 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 that would like to use ConEx signaling.

A ConEx sender MUST negotiate for both SACK and ECN or the more
accurate ECN feedback in the TCP handshake if these TCP extension are
available at the sender. Therefore a ConEx sender SHOULD 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 accECN-ConEx (no SACK but accurate ECN feedback)

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

o SACK-ECN-ConEx (SACK and 'classic' instead of accurate 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 account congestion byte-wise (and not packet-wise). A sender
MUST mark subsequent packets (after the congestion notification) with
the respective ConEx bit 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 only 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 [draft-briscoe-tsvwg-byte-pkt-mark], 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 accounting in each operation mode
are described in the next section. Further handling of the IPv6 bits
itself if congestion was accounted is described in the subsequent
section afterwards.

3. Accounting congestion

A ConEx sender maintains two counters: one that accounts congestion
based on the information retrived by loss detection, and a second
that accounts for ECN based congestion feedback (in TCP). These
counters hold the number of outstanding bytes that should be ConEx
marked either with the E bit or the L bit in subsequent packets.

The outstanding bytes for congestion indications based on loss are
maintained in the loss exposure gauge (LEG) and the accounting is
explained in Section 3.1.

The outstanding bytes accounted based on ECN feedback information are maintained in the congestion exposure gauge (CEG). The accounting of these bytes from the ECN feedback is explained in more detail next in Section 3.2.

Furthermore, those counters will be reduced every time a ConEx capable packet with the E or L bit set is sent. This is explained for both counters in Section 4.1.

Usually all bytes of an IP packet must be accounted. Therefore the sender SHOULD take the headers into account, too. If equal sized packets, or at least equally distributed packet sizes can be assumed, the sender MAY only account the 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. This case is assumed for simplification in the following sections.

Otherwise if this is not the case and a sender sends different sized packets (with unequally distributed packet sizes), the sender needs to memorize or estimate the number of ECN-marked or lost packets. A sender might be able to reconstruct the number of packets and thus the header bytes if the packet sizes of all packets that were sent during the last RTT are known. Otherwise if no additional information is available the worst case number of packets and thus header bytes should be estimated in a conservative way based on a minimum packet size (of all packets sent in the last RTT). If the number of ConEx marked packets is smaller (or larger) than the estimated number of ECN-marked or lost packets, the additional header bytes should the added to (or can be subtracted from) the respective counter.

3.1. Loss Detection

A ConEx sender MUST maintain a loss exposure gauge (LEG), indicating the number of outstanding bytes that must be sent with the ConEx L bit. When a data segment is retransmitted, LEG will be increased by the size of the TCP payload bytes contained by the retransmission, assuming equal sized segments such that the retransmitted packet will have the same number of header bytes as the original ones.

Any retransmission may be spurious. To accommodate that, a ConEx sender SHOULD make use of heuristics to detect such spurious retransmissions (e.g. F-RTO [RFC5682], DSACK [RFC3708], and Eifel [RFC3522], [RFC4015]). When such a heuristic has determined, that a certain number of packets were retransmitted erroneously, the ConEx sender should subtract the payload size of these TCP packets from LEG.

[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 strongly but timeliness is important for ConEx.

For ConEx it is not important to know which data got lost but only how much. 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. Thus without SACK, the needed information for the ConEx feedback can be available with an additionally delay of one RTT by using the following estimation algorithm and 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 of time in the transmission, in the worst case, all packets in flight minus three that trigged 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 accounted.)
```

3.2. ECN

ECN [RFC3168] is an IP/TCP mechanism that allows network nodes to mark packets with the Congestion Experienced (CE) mark instead of (early) dropping them when congestion occurs. As soon as a CE mark is seen at the receiver, with classic ECN it will feed this information back to the sender by setting the Echo Congestion Experienced (ECE) bit in the TCP header of all subsequent ACKs until a packet with Congestion Window Reduced (CWR) bit in the TCP header is received to acknowledge the reception of the congestion notification. The sender sets the CWR bit in the TCP header once when the first ECE of a congestion notification is received.

A receiver can support 'classic' ECN, a more accurate ECN feedback scheme, or neither. In the case ECN is not supported at all, of course, no ECN marks will occur, thus the E bit will never be set. Otherwise, a ConEx sender must maintain a counter, the congestion exposure gauge (CEG), for the number of outstanding bytes that have to be ConEx marked with the E bit.

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 which 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.

Moreover with SACK, DeliveredData is increased by the number of bytes
provided by (new) SACK information (SACK_diff). Note, 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_byte), that has previously already
been accounted to DeliveredData based on SACK information.

Without SACK, DeliveredData is estimated to be 1 SMSS on duplicate
acknowledgements. For the subsequent partial or full ACK,
DeliveredData is estimated to be the newly acknowledged bytes, minus
one SMSS for each preceding duplicate ACK. Therefore 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.

The two cases, with and without more accurate ECN depending on the
receiver capability, are discussed in the following sections.

3.2.1. Accurate ECN feedback

With a more accurate ECN feedback scheme either the number of marked
packets/received CE marks or directly the number of marked bytes is
known. In the later case the CEG can directly be increased 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 max the number of newly acknowledged bytes:

CEG += min(SMSS*D, DeliveredData)

3.2.2. Classic ECN support

If the ConEx sender fully conforms to the semantics of the ECN
signaling as defined by [RFC5562], 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 much packets have
actually been CE marked during this RTT, the most conservative
assumption should 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 an Advanced Compatibility
Mode:

To extract more than one ECE indication per RTT, a ConEx sender could
set the CWR flag opportunistically to force the receiver to signal
only one ECE per CE mark. Unfortunately, the use of delayed ACKs
[RFC5681], as it is usually done today, will prevent a feedback of
every CE mark. If an CWR confirmation will be received before the
ECE can be sent out with the next ACK, ECN feedback information
information could get lost. Thus a sender should set CWR only on
those data segments, that will actually trigger a (delayed) ACK. The
sender would need an additional control loop to estimated which data
segment will trigger an ACK. But such a more sophisticated
heuristics could extract congestion notifications more timely. Still
the CEG need to be increased by DeliveredData, as one or more CE
marked packets could be acknowledged by one delayed ACK.

4. Setting the ConEx Bits

By setting the X bit a packet is marked as ConEx-capable. All
packets carrying payload MUST be marked with the X bit set including
retransmissions. No congestion feedback information are available
about control packets such as pure ACKs which are not carrying any
payload. Thus these packets should not be taken into account when
determining ConEx information. These packet MUST carry a ConEx
Destination Option with the X bit unset.

4.1. Setting the E and the L Bit

As long as the CEG or LEG counter is positive, ConEx-capable packets
SHOULD be marked with E or L respectively, and the CEG or LEG counter
is decreased by the TCP payload bytes carried in this packet. If the
CEG or LEG counter is negative, the respective counter SHOULD be
reset to zero within one RTT after it was decreased the last time or
one RTT after recovery if no further congestion occurred.

If SACK information is not available spurious retransmission are more
likely. In this case it might be valuable to slightly delay the
ConEx loss feedback until a spurious retransmission might be
detected. But the ConEx signal MUST NOT be delayed more than one RTT
if as long as data packets are sent out.

4.2. Credit Bits

The ConEx abstract mechanism requires that sufficient credit must be
signaled in advance to cover the expected congestion during the
feedback delay of one RTT. A ConEx sender should maintain a counter
of the sent credits c in bytes. If congestion occurs, credits will
be consumed and the c counter should be reduced by the number of
bytes that where lost or estimated to be ECN-marked. If the risk of

congestion was estimated wrongly and thus too few credits were sent, the c counter becomes zero but can not get negative.

The number of credits sent should always equal the number of bytes in flight, as all packets could potentially get lost or congestion marked. Thus a ConEx sender should monitor the number of bytes in flight f. If f ever becomes larger than c, the ConEx sender SHOULD send new credits. Remember that c will be decreased if congestion occurs.

In TCP Slow Start, the congestion window might grow much larger than during the rest of the transmission. Thus a sender could consider to sent fewer than f credits but risking potential penalization by an audit. In any case the credits should at least cover the increase in sending rate. As the sending rate increases exponentially in Slow Start, thus double 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 only depend actions taken by the sender. In general, the behavior of the cross traffic, and if Active Queue Management (AQM) is used, the respective parameterization influence how many packets get dropped or marked. But if the used AQM is not overly aggressive with ECN marking, sending halve the flight size as credits should be sufficient for both, congestion signaled by loss or ECN. Marking every fourth packet will allow the respective number of credits in Slow Start as it can be seen in Figure Figure 1.
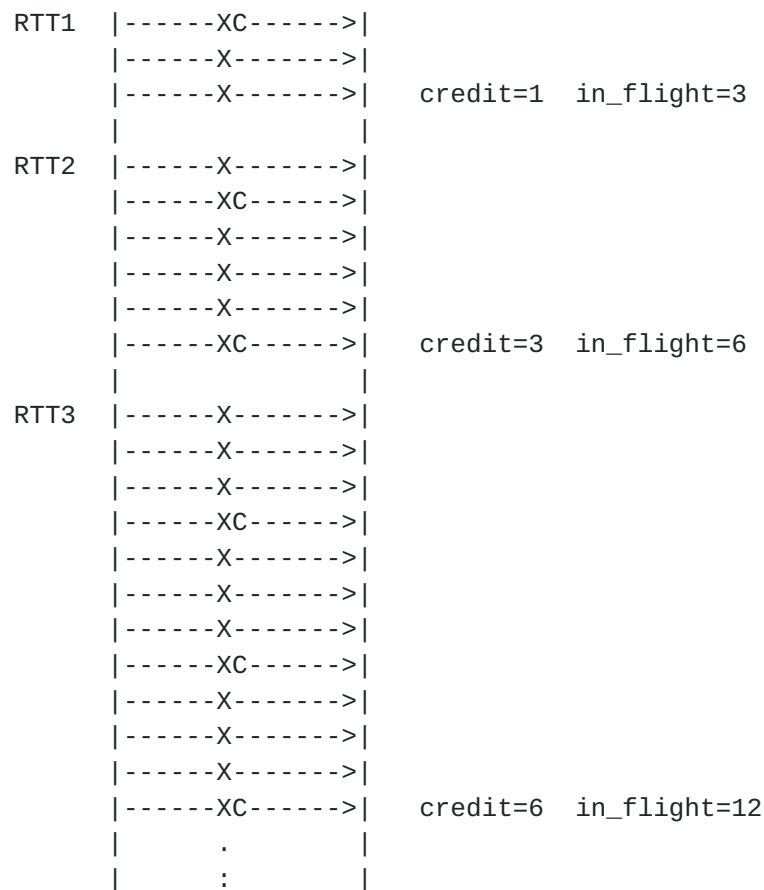
```
            RTT1  |------XC------>|
                  |------X------->|
                  |------X------->|   credit=1  in_flight=3
                  |              |
            RTT2  |------X------->|
                  |------XC------>|
                  |------X------->|
                  |------X------->|
                  |------X------->|
                  |------XC------>|   credit=3  in_flight=6
                  |              |
            RTT3  |------X------->|
                  |------X------->|
                  |------X------->|
                  |------XC------>|
                  |------X------->|
                  |------X------->|
                  |------X------->|
                  |------XC------>|
                  |------X------->|
                  |------X------->|
                  |------X------->|
                  |------XC------>|   credit=6  in_flight=12
                  |      .       |
                  |      :       |
```

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

   It is possible that the audit looses state due to e.g. rerouting or
   memory limitations. Therefore, the sender needs to detect this case
   and resend credits. Thus a ConEx sender should reset the credit
   count c to zero if losses occur in two subsequent RTTs (assuming that
   the sending rate was correctly reduced based on the received
   congestion signal).

5. Loss of ConEx information

   Packets carrying ConEx can also get lost. A ConEx sender must
   remember which packet was marked with either the L, the E or the C
   bit. If one of these packets is detected to be lost, the should
   increase the respective gauge, LEG or CEG, by the number of lost
   payload bytes.

6. Timeliness of the ConEx Signals

   ConEx signals can only be evaluated by a network node with a time
   delay of about one RTT after the congestion occured. To avoid
   further delays, a ConEx sender SHOULD sent the ConEx signaling with

the next available packet. In cases where it is preferable to slightly delay the ConEx signal, the sender MUST NOT delay the ConEx signal more than one RTT.

Multiple ConEx bits may become available for signaling at the same time, for example when an ACK is received by the sender, that indicates at the same time that at least one segment has been lost, and that one or more ECN marks were received. This may happen during excessive congestion, where the queues overflow even though ECN was used and currently all packets are marked, while others have to be dropped nevertheless. Another possibility when this may happen are lost ACKs, so that a subsequent ACK carries summary information not previously available to the sender. As ConEx-capable packet can carry different ConEx marks at the same time, these information do not need to be distributed over several packets and thus can be sent without further delay.

7. Acknowledgements

The authors would like to thank Bob Briscoe who contributed with this initial ideas and valuable feedback. Moreover, thanks to Jana Iyengar who provided valuable feedback.

8. IANA Considerations

This document does not have any requests to IANA.

9. Security Considerations

With some of the advanced ECN compatibility modes it is possible to miss congestion notifications. Thus a sender will not decrease its sending rate. If the congestion is persistent, the likelihood to receive a congestion notification increases. In the worst case the sender will still react correctly to loss. This will prevent a congestion collapse.

10. References

10.1. Normative References

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

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

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

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

   [draft-ietf-conex-abstract-mech]
               Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx)
               Concepts and Abstract Mechanism", draft-ietf-conex-
               abstract-mech-06 (work in progress), October 2012.

   [draft-ietf-conex-destopt]
               Krishnan, S., Kuehlewind, M., and C. Ucendo, "IPv6
               Destination Option for ConEx", draft-ietf-conex-destopt-04
               (work in progress), March 2013.

10.2. Informative References

   [DCTCP]   Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel,
               P., Prabhakar, B., Sengupta, S., and M. Sridharan, "DCTCP:
               Efficient Packet Transport for the Commoditized Data
               Center", Jan 2010.

   [I-D.briscoe-tsvwg-re-ecn-tcp]
               Briscoe, B., Jacquet, A., Moncaster, T., and A. Smith,
               "Re-ECN: Adding Accountability for Causing Congestion to
               TCP/IP", draft-briscoe-tsvwg-re-ecn-tcp-09 (work in
               progress), October 2010.

   [RFC3522]   Ludwig, R. and M. Meyer, "The Eifel Detection Algorithm
               for TCP", RFC 3522, April 2003.

   [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,
               February 2004.

   [RFC4015]   Ludwig, R. and A. Gurtov, "The Eifel Response Algorithm
               for TCP", RFC 4015, February 2005.

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

[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,
            September 2009.

[RFC6789]   Briscoe, B., Woundy, R., and A. Cooper, "Congestion
            Exposure (ConEx) Concepts and Use Cases", RFC 6789,
            December 2012.

[draft-briscoe-tsvwg-byte-pkt-mark]
            Briscoe, B. and J. Manner, "Byte and Packet Congestion
            Notification", draft-briscoe-tsvwg-byte-pkt-mark-010 (work
            in progress), May 2013.

[draft-kuehlewind-tcpm-accurate-ecn]
            Kuehlewind, M. and R. Scheffenegger, "More Accurate ECN
            Feedback in TCP", draft-kuehlewind-tcpm-accurate-ecn-02
            (work in progress), Jun 2013.

Appendix A. Revision history

RFC Editior: This section is to be removed before RFC publication.

00 ... initial draft, early submission to meet deadline.

01 ... refined draft, updated LEG "drain" from per-packet to RTT-
based.

02 ... added Section 5 and expanded discussion about ECN interaction.

03 ... expanded the discussion around credit bits.

04 ... review comments of Jana addressed. (Change in full compliance
mode.)

05 ... changes on Loss Detection without SACK, support of classic ECN
and credit handling.

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

   Phone: +43 1 3676811 3146
   Email: rs@netapp.com