

Congestion Exposure (ConEx)
Internet-Draft
Intended status: Experimental
Expires: January 5, 2012

M. Kuehlewind, Ed.
University of Stuttgart
R. Scheffenegger
NetApp, Inc.
July 4, 2011

TCP modifications for Congestion Exposure
draft-kuehlewind-conex-tcp-modifications-00

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on January 5, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. Sender-side Modifications	3
3. Setting the ConEx IPv6 Bits	4
3.1. ECN	4
3.1.1. Accurate ECN feedback	4
3.1.2. Classic ECN support	5
3.2. Loss Detection with/without SACK	6
3.3. Credit Bits	7
3.4. Timeliness of the conex signals	8
4. Acknowledgements	9
5. IANA Considerations	9
6. Security Considerations	9
7. References	9
7.1. Normative References	9
7.2. Informative References	10
Appendix A. Spacing conex marks evenly	10
Authors' Addresses	12

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. This document describes the necessary modifications to use ConEx with the Transmission Control Protocol (TCP). The ConEx signal is based on loss or ECN marks [RFC3168] as a congestion indication.

With standard TCP without Selective Acknowledgments (SACK) [RFC2018] the actual number of losses is hard to detect, thus we recommend to enable SACK when using ConEx. However, we discuss both cases, with and without SACK support, later on.

Explicit Congestion Notification (ECN) is defined in such a way that only a single congestion signal is guaranteed to be delivered per Round-trip Time (RTT). For ConEx a more accurate feedback signal would be beneficial. Such an extension to ECN is defined in a separate document [draft-kuehlewind-conex-accurate-ecn], as it can also be useful for other mechanisms, as e.g. [DCTCP] or whenever the congestion control reaction should be proportional to the experienced congestion.

The current version of this draft is only a first collection of ConEx-based TCP modification and should not be regarded as feature-complete as the specification for the abstract ConEx mechanism is still under discussion. The next version will also go more precisely into implementation details.

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

A ConEx sender MUST negotiate for both SACK and the more accurate ECN feedback in the TCP handshake. Depending on the capability of the receiver, the following operation modes exist:

- o Full-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)

The handling of the IPv6 bits is described in the next section.

3. Setting the ConEx IPv6 Bits

ConEx is currently/will be defined as an destination option for IPv6. The use of four bits have been defined, namely the X (ConEx-capable), the L (loss experienced), the E (ECN experienced) and C (credit) bit.

By setting the X bit a packet is marked as ConEx-capable. It is not decided yet which or if any packets should not be ConEx capable. (e.g. control packets as pure ACKs or retransmits). It is not defined yet which bits (E, L, C) can be set at the same time in one (data) packet. It is assumed that ConEx marked packets are accounted by their respective IP size, as all the signals (Loss, ECN) are attributes of an IP packet, not a TCP segment or merely the TCP payload. Further discussion is needed here.

3.1. ECN

A receiver can support the accurate ECN feedback scheme, the 'classic' ECN or neither. In the case ECN is not supported at all, the transport is not ECN-capable and no ECN marks will occur, thus the E bit will never be set. In the other cases a ConEx sender has to maintain a gauge for the number of outstanding ConEx marks, the congestion exposure gauge (CEG).

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 sent by the IP layer (e.g. by MTU bytes for each MSS segment). Whenever a packet is sent with the E bit set, this gauge is decreased by the IP size of that packet. The two cases, depending on the receiver capability, are discussed in the following sections.

3.1.1. Accurate ECN feedback

When the accurate ECN feedback scheme is supported by the receiver, the receiver will maintain an echo congestion counter (ECC). The ECC will hold the number of CE marks received. A sender that is

understanding the accurate ECN feedback will be able to reconstruct this ECC value on the sender side by maintaining a counter ECC.r.

On the arrival of every ACK, the sender calculates the difference D between the local ECC.r counter, and the signaled value of the receiver side ECC counter. The value of ECC.r is increased by D, and D is assumed to be the number of CE marked packets that arrived at the receiver since it sent the previously received ACK.

Whenever the counter ECC.r is increased, the gauge CEG has to be increased by the amount of bytes sent on the IP layer which were marked:

```
CEG += acked_bytes + (IP.header+TCP.header)*D
```

3.1.2. Classic ECN support

A ConEx sender that communicates with a classic ECN receiver (conforming to [RFC3168] or [RFC5562]) MAY run in one of these modes:

- o Full compliance mode:

The ConEx sender fully conforms to all the semantics of the ECN signaling as defined by [RFC5562]. In this mode, only a single congestion indication can be signaled by the receiver per RTT. Whenever the ECE flag toggles from "0" to "1", the gauge CEG is increased by the SMSS plus headers.

Note that under severe congestion, a session adhering to these semantics may not provide enough ConEx marks. This may cause appropriate sanctions by an audit device in a ConEx enabled network.

- o Simple compatibility mode:

The sender will set the CWR permanently to force the receiver to signal only one ECE per CE mark. Unfortunately, in a high congestion situation where all packets are CE marked over a certain period of time, the use of delayed ACKs, as it is usually done today, will prevent a feedback of every CE mark. With an ACK rate of m, about $m-1/m$ CE indications will not be signaled back by the receiver (e.g. 50% with $M=2$). Thus, in this mode the ConEx sender MUST increase CEG by a count of

```
M*(SMSS+IP.header+TCP.header)
```

for each received ECE signal.

- o Advanced compatibility mode:

More sophisticated heuristics, such as a phase locked loop, to set CWR only on those data segments, that will actually trigger an (delayed) ACK, could extract congestion notifications more timely. A ConEx sender MAY choose to implement such a heuristic. In addition, further heuristics SHOULD be implemented, to determine the value of each ECE notification. E.g. for each consecutive ACK received with the ECE flag set, D should be increased by one, and CEG increased by

```
CEG += min((SMSS+IP.header+TCP.header)*D, ((acked_bytes/
SMSS)+1)*(IP+TCP Header) + acked_bytes)
```

If an ACK is received with the ECE flag cleared, D must be set to zero. This heuristic is conservative during more serious congestion, and more relaxed at low congestion levels.

3.2. Loss Detection with/without SACK

For all the data segments that are determined by a ConEx sender as lost, an identical number of IP bytes MUST be sent with the ConEx L bit set. Loss detection typically happens by use of duplicate ACKs, or the firing of the retransmission timer. A ConEx sender MUST maintain a loss exposure gauge (LEG), indicating the number of outstanding IP bytes that must be sent with the ConEx L bit. When a data segment is retransmitted, LEG should be increased by the size of the IP packet containing the retransmission. When sending subsequent segments (including TCP control segments), the ConEx L bit is set as long as LEG is positive, and LEG is decreased by the size of the sent IP packet with the ConEx L bit set.

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 size of these IP packets from LEG. As ConEx credits have only a limited lifetime, whenever the gauge becomes negative, it should be drained at a low rate (e.g. 1 count per sent packet).

Note that the above heuristics delays the conex signal by one segment, and also decouples them from the retransmissions themselves, as some control packets (e.g. pure ACKs, window probes, or window updates) may be sent in between data segment retransmissions. A simpler approach would be to set the ConEx signal for each retransmitted data segment. However, it is important to remember,

that a ConEx signal and TCP segments do not natively belong together.

3.3. Credit Bits

The ConEx abstract mechanism requires that the transport SHOULD signal sufficient credit in advance to cover any reasonably expected congestion during its feedback delay. To be very conservative the number of credits would need to equal the number of packets in flight, as every packet could get lost or congestion marked. With a more moderate view, only an increase in the sending rate should cause congestion.

For TCP sender using the [RFC5681] congestion control algorithm, we recommend to only send credit in Slow Start, as in Congestion Avoidance an increase of one segment per RTT should only cause a minor amount of congestion marks. If an more aggressive congestion control is used, a sufficient amount of credits need to be set.

In TCP Slow Start the sending rate will increase exponentially and that means double every RTT. Thus the number of credits should equal half the number of packets in flight in every RTT. Under the assumption that all marks will not get invalid for the whole Slow Start phase, marks of a previous RTT have to be summed up. Thus the marking of every fourth packet will allow sufficient credits in Slow Start.

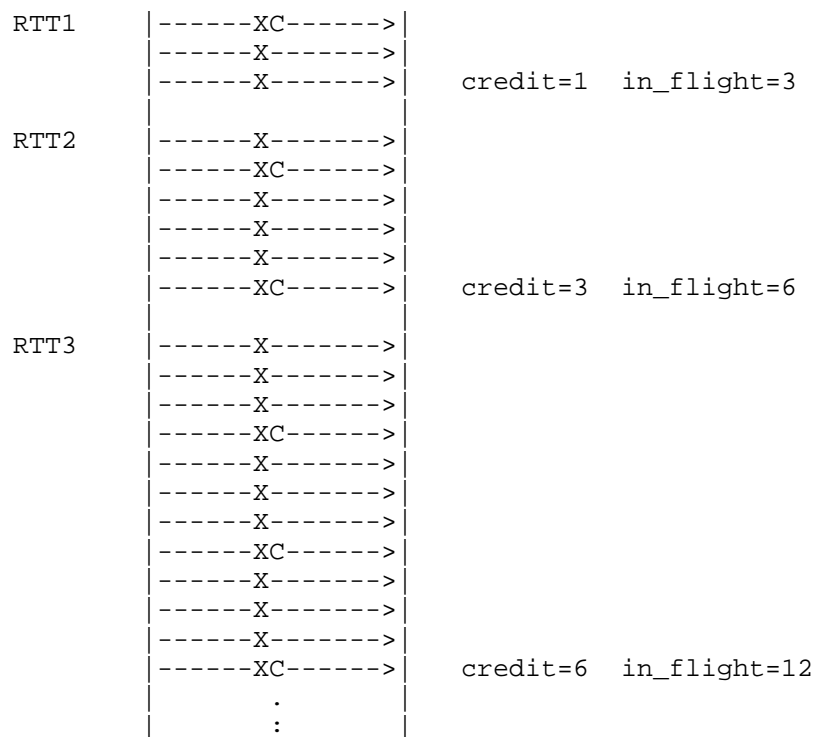


Figure 1: Credits in Slow Start (with an initial window of 3)

If a ConEx sender detects an increasing number of losses even though the sender reduced the sending rate, the sender SHOULD assume that those losses are incorporated by an audit device and thus should send further credits. Upto now its not clear if the credits say valid as long as the connection is established or if an expiration of the credits need to be assumed by the sender.

3.4. Timeliness of the conex signals

Conex signals will anyway be evaluated with a slight time delay of about one RTT by a network node. Therefore, it is not necessary to immediately signal ConEx bits when they become known (e.g. L and E bits), but a sender SHOULD NOT delay ConEx signaling excessively.

Multiple ConEx bits may become available for signaling at the same time, for example when an ACK is received by the sender, that indicates that at least one segment has been lost, and that one or more ECN marks were received at the same time. This may happen during excessive congestion, where buffer queues overflow and some 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.

It may be preferable to signal only one ConEx bit per segment, and to space out the signaling of multiple bits across a (short) period of time - or number of segments. However, that delay should not be excessive, and ideally also shorter than the RTT of the affected TCP session. The heuristic sketched in Appendix A uses a maximum delay of 10 packets or 1/4 of the congestion window, whatever is smaller to minimize delay.

It is important to remember, that ConEx bits and TCP retransmissions do not interact with each other. However, a retransmission should be accompanied by one ConEx L bit in close proximity nevertheless. This does not mean, that TCP retransmissions may never contain ConEx marks. In a typical scenario using SACK, the first retransmission would not carry a ConEx L bit, while subsequent retransmissions in the same recovery episode, would be marked with the ConEx L bit. Spreading the ConEx bits over a small number of segments increases the likelihood that most devices along the path will see some ConEx marks even during heavy congestion.

4. Acknowledgements

5. IANA Considerations

6. Security Considerations

7. References

7.1. 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.
- [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.

7.2. 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.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 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.
- [draft-kuehlewind-conex-accurate-ecn] Kuehlewind, M. and R. Scheffenegger, "Accurate ECN Feedback in TCP", draft-kuehlewind-conex-accurate-ecn-00 (work in progress), Jun 2011.

Appendix A. Spacing conex marks evenly

Under certain circumstances, very high marking conex marking rates may need to be signaled. However, as conex maintains running averages, it may be beneficial to send these marks more evenly

spaced, than in bursts of consecutive segments, all with the conex bits set.

If only a single conex mark needs to be sent, it should be sent immediately to maintain optimal timeliness. Any subsequent conex marks may be delayed slightly, to disentangle retransmissions of the transport protocol from packets carrying conex marks.

The following algorithm will provide such a method. When very high marking rates are required, it will automatically set a conex mark with every sent packet.

When an ACK is received, the sender determines the number of bytes which need to be sent with a conex mark, and the next segment to be sent should carry at least part of the conex signal:

```
CEF .. congestion.experienced.flag CEG ..
congestion.experienced.gauge CEC .. congestion.experienced.counter
LEF .. loss.experienced.flag LEG .. loss.experienced.gauge LEC ..
loss.experienced.counter
```

```
if (marks.received > 0) {
    CEF = 1;           # for the immediate mark
    CEG += bytes.received.with.CE; # for the delayed mark
}
if (lost segment retransmitted) {
    LEF = 1;
    LEG += IP.size.of.retransmitted.segment;
}
```

When sending a segment, the following algorithm is run to determine if the segment should carry a conex mark. Note that the counter is initialized to at most $10 \times (\text{MTU of the path})$. This spaces two consecutive received marks at most 10 full sized data segments apart.

For each of the L (loss) and E (ECN) conex bits, a similar algorithm needs to run.

```
if ((LEF == 0) and (LEG > 0)) {
  if ((LEC <= 0) or (LEG >= LEC)) {
    LEG -= IP.size.of.segment.to.be.sent;
    LEC = min(10*(SMSS+IP.header+TCP.header), Flightsize/4);
    LEF = 1;
  } else {
    LEC -= LEG;
  }
}
if ((CEF == 0) and (CEG > 0)) {
  if ((CEC <= 0) or (CEG >= CEC)) {
    CEG -= IP.size.of.segment.to.be.sent;
    CEC = min(10*(SMSS+IP.header+TCP.header), Flightsize/4);
    CEF = 1;
  } else {
    CEC -= CEG;
  }
}

if (LEF == 1) {
  LEF = 0;
  SendSegment(conex.L.bit);
} else {
  if (CEF == 1) {
    CEF = 0;
    SendSegment(conex.E.bit);
  } else
    SendSegment;
}
if (CEG < 0) {
  CEG += 1; # slowly reduce credit
}
if (LEG < 0) {
  LEG += 1;
}
}
```

Authors' Addresses

Mirja Kuehlewind (editor)
University of Stuttgart
Pfaffenwaldring 47
Stuttgart 70569
Germany

Email: mirja.kuehlewind@ikr.uni-stuttgart.de

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

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

