Network Working Group I. Johansson
Internet-Draft Ericsson AB
Intended status: Informational February 21, 2017
Expires: August 25, 2017

# ECN support in QUIC
## draft-johansson-quic-ecn-01

Abstract

This memo outlines the ECN support in QUIC. The intention is that most of the material ends up updating other new or existing QUIC protocol specifications, thus it may be possible that this draft does not warrant a working group status.

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 RFC 2119 [RFC2119].

Status of This Memo



Copyright Notice

Table of Contents



## 1. Introduction

ECN support in transport protocols is a fundamental feature that should be included in the QUIC specification as a mandatory element. The benefits of ECN is described in [I-D.ietf-aqm-ecn-benefits]. The ECN support should be implemented to support both present and future ECN, the latter is outlined in [I-D.ietf-tsvwg-ecn-experimentation], of particular interest is the ability to discriminate between classic ECN and L4S ECN by means of differentiation between the use of the ECT(0) and ECT(1) code points. This draft does however not delve into the details of the congestion control implementation.

## 2. Elements of ECN support

This draft covers the following aspects of ECN support:

- ECN negotiation

- ECN echo

- ECN bits in the IP header, semantics

- Fallback in case of ECN fault

o OS socket specifics, access to the ECN bits

o Monitoring

## 2.1. ECN negotialtion

ECN support in QUIC needs to be negotiated. The reasons is that network elements may not support ECN and may either clear the ECN bits or simply discard packets that have the ECN bits set. In addition, a QUIC implementation may not have access to the ECN bits in the IP header due to OS dependent restrictions, investigations (Piers O'Hanlon) have indicated that this is in certain cases an asymmetric property, for instance while it is possible to set the ECN bits it is not possible to read them.

It is also required that the ECN negotiation does not interfere with the connection setup, in other words a failed ECN negotation should not cause an extra roundtrip for the connection setup.

The suggested method in this draft is to add an ECN negotiation frame that is transmitted when connection setup is completed. Both peers MUST transmit the ECN negotation frame. The ECN negotiation frame is shown below.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type          |C|R|W|U U U|E E|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: ECN negotation frame

The 2nd byte contains the flags:

o C: Challenge bit, indicates that the transmitted ECN negotiation frame is a challenge, if bit is not set then it is a response.

o R: Possible to read ECN bits in IP header

o W: Possible to write ECN bits in IP header

o EE : Echo of ECN bits

o U: Unused

A peer transmits the ECN negotiation frame with the R,W and EE bits in the 2nd byte set to '0' and the C bit set to '1'. This frame is echoed back with the flags set according to the degree of ECN support

and with the ECN bits in the IP header of the received ECN negotiation frame copied to the EE field, the C bit is '0'. As both peers MUST transmit an ECN negotation frame there will be a total of 4 ECN negotiation frames transmitted, two challenges and two responses.

The IP header for the ECN negotiation frame should set the ECN bits to CE '11'. When the corresponding response is received then an EE pattern of '11' indicates that ECN is likely supported in the network. This does not give a full quarantee that ECN is supported in the network. Monitoring of the ECN field in the ACK-frame serves to give further indication of ECN support once ECN is turned on.

A peer is not allowed to set ECT on outgoing data packets until a ECN negotiation response that inticates that ECN is supported is received. In other words it is only the ECN negotiation frame that is allowed to set the ECN bits in the IP header.

A lack of an ECN negotiation response may indicate that the ECN challenge frame or the ECN response frame was lost or that a node in the network deliberately discards ECN-CE marked packets. The peer can transmit additonal ECN challeges with given time intervals to rule out accidentail packet loss. The detailed timing for this is T.B.D.

The mode mechanism in [RFC6679] can serve as in input to a solution for the support of ECN in the case that OS ECN support is asymmetric. It is however unclear how a QUIC implementation can determine asymmetric ECN support in the underlying OS. For instance the method to send ECN marked packets to the local host to determine OS support does not reveal if the OS ECN support is asymmetric.

### 2.2. ECN bits in the IP header, semantics

The ECN bits in the IP header should be set according to the recommendations in [I-D.ietf-tsvwg-ecn-experimentation]. This means that the meaning of ECT(0) and ECT(1) differ.

### 2.3. ECN echo

The ECN echo should prefferably go into the ACK frame [I-D.ietf-quic-transport], this is beneficial as the ECN information can then use some of the already existing data in the ACK frame for improved efficiency, this applies especially to alternatives 1 and 2 below. It is suggested that the 'U' bit in the ACK frame type is renamed 'E' to indicate the presence of an ECN field in the ACK frame, this makes it possible to omit the ECN information for the cases where ECN is not supported for the connection.

Currently there are three alternatives how to add ECN support to the ACK frames .

The first alternative inserts a one octet field that contains a 2 bit ECN echo, followed by the ACK block length. The ACK block length then dictates the number of received contiguous frames with the indicated ECN echo.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ECE (8)       | First Ack Block Length (8/16/32/48)         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  [Gap 1 (8)]  | ECE(8)        |[Ack Blk 1 L (8/16/32/48)]   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  [Gap 2 (8)]  | ECE(8)        |[Ack Blk 2 L (8/16/32/48)]   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  [Gap N (8)]  | ECE(8)        |[Ack Blk N L (8/16/32/48)]   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: ECN field in ACK frame ACK block, alt 1

The second alternative encodes a variable length field that contains the ECN echoes for the frames listed in the ACK blocks. The length of the field is inferred from the ACK block lengths. No ECN echoes are indicated for the gaps (it is, after all, impossible to indicate status of the ECN bits for lost packets). For instance if the ACK blocks list 10 frames, then the length of the ECN echo field becomes 2*10=20bits, with additional 4 bits of padding the ECN echo field will then become 3 octets long.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              First Ack Block Length (8/16/32/48)            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  [Gap 1 (8)]  |       [Ack Block 1 Length (8/16/32/48)]     ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  [Gap 2 (8)]  |       [Ack Block 2 Length (8/16/32/48)]     ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  [Gap N (8)]  |       [Ack Block N Length (8/16/32/48)]     ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|ECE|ECE|... variable length, padded to full octets           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: ECN field in ACK frame ACK block, alt 2

The third alternative encodes the number of bytes that are marked ECT(0), ECT(1) and CE with 32 bits each, the total extra overhead is thus 12 octets.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              First Ack Block Length (8/16/32/48)            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  [Gap 1 (8)]  |       [Ack Block 1 Length (8/16/32/48)]     ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  [Gap 2 (8)]  |       [Ack Block 2 Length (8/16/32/48)]     ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  [Gap N (8)]  |       [Ack Block N Length (8/16/32/48)]     ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   # ECT(0) bytes (32)                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   # ECT(1) bytes (32)                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   # ECN-CE bytes (32)                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: ECN field in ACK frame ACK block, alt 3

The fourth alternative use an extra byte to encode how many bits that encode each of the ECT/CE fields.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              First Ack Block Length (8/16/32/48)            ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  [Gap 1 (8)]  |       [Ack Block 1 Length (8/16/32/48)]     ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  [Gap 2 (8)]  |       [Ack Block 2 Length (8/16/32/48)]     ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                 ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  [Gap N (8)]  |       [Ack Block N Length (8/16/32/48)]     ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |R|R|E1 |E2 |CE | # ECT(0) bytes (0/16/32/48)                 ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  # ECT(1) bytes (0/16/32/48)                                ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  # ECN-CE bytes (0/16/32/48)                                ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: ECN field in ACK frame ACK block, alt 4

The E1,E2 and CE fields indicate the length of each encoding for the number of ECT(0), ECT(1) and ECN-CE marked bytes. This is encoded as:

- 00: 0 bits
- 01: 16 bits
- 10: 32 bits
- 11: 48bits

R indicates reserved bits.

There are pros an cons with the four alternatives:

- Alt 1: Is very compact in the case where the ECN bits are largely unchanged. However in the worst case where received frames flip forth and back between ECT and CE then each frame will require at least 3 octets overhead (ECE, ACK block length, Gap).

- Alt 2: Is quite compact as it only requires two bits encoding per frame. The additional overhead amounts to ceil(N*2/8) octets where the N is the sum of the ACK block lengths. On the downside is that it is a less efficient format for the case that the ECN

bits are unchanged. One uncertainty is if STOP_WAITING frames could make this encoding bulky.

- Alt 3: Has a fixed 12 octet overhead which may be beneficial as it gives a deterministic overhead. The possible drawback is that it is not possible to know exactly which frames have been remarked, something that can limit the ability to detect network ECN faults based on the method to transmit a pattern on ECT and CE marked packets.

- Alt 4: Is a variation to Alt 3 but has a variable length encoding that should consume less space, especially in the cases that one of the ECT code points is not used and for the case that packets are only sporadically ECN-CE marked. This alternative also makes it unnecessary to use a bit in the ACK frame type to indicate the precense of an ECN field as this can be indicated in a efficient way with the one byte header in this format. E0=E1=CE = 00 indicates that the following ECT and CE fields are encoded with zero bits.

Which of the three formats above (or something else) that is the best alternative is subject to discussion.

### 2.4. Fallback in case of ECN fault

ECN can be subject to issues in network equipment, such as remarking to Not-ECN, remarking from ECT(0) to ECT(1) and vice versa or constant remarking to ECN-CE. Furthermore ECT marked packets may be discarded in the network. While these problems seem to be rare, see for instance [McQuistin-Perkins], it is still necessary to safeguard against such problems.

A peer should disable ECN for its outgoing packets if ECN fault is detected, it is however still possible for the other peer to use ECN.

TODO add more information as regards to how to detect network ECN faults. [ECN-fallback](expired) gives a few examples for fault detection. Examples on how to detect ECN faults include for instance the method to set ECT and CE for outgoing packets according to a given pattern.

Fallback in case of ECN faults is not an issue only for QUIC, it is here suggested that mechanisms for this is described in a non QUIC related draft, for instance in TSVWG.

## 2.5. OS socket specifics, access to the ECN bits

ECN support in QUIC comes with the additional challenge that it is necessary to somehow access the ECN bits in the IP headers. In TCP this is provided without major concerns as TCP is generally implemented in OS kernel space. QUIC can however be implemented both in user space or kernel space and is layered on top of UDP, which means that access to the ECN bits is not a given, instead various tricks are needed.

The text below is copy-pasted from [OHanlon].

"To set ECN on Linux, BSD and OSX one can use IP_TOS socket option, with the setsockopt() call, to set the relevant ECN bits of the TOS byte. On Windows one can use a similar technique though firstly one has to enable TOS byte setting by enabling a particular Registry key ( DisableUserTOSSetting=0 (see https://msdn.microsoft.com/en-us/library/windows/desktop/dd874008%28v=vs.85%29.aspx One could also probably use the libpcap write functionality."

"To obtain the ECN bits from a packet one needs a mechanism to retrieve the ECN bits from each packet. On Linux, one needs to firstly set the IP_RECVTOS socket option on the receiving socket, and use the recvmsg() call to receive a packet, and then retrieve the TOS byte from the associated csmg structure returned by the recvmsg() call. This still works with linux-4.2.3. On OSX/BSD there are no suitable socket options to retrieve the ECN/TOS bits and one cannot use raw sockets as they do not function for UDP/TCP sockets (they do work with ICMP), so one has to use alternatives such the bpf interface, or a REDIRECT socket. Whilst on Windows it seems that the only way to retrieve the ECN bits is via a raw socket, or custom NDIS driver, though it's possible there's an API I'm missing."

TODO: Write a more detailed description on how to implement ECN support in QUIC for different OS stacks.

## 2.6. Monitoring

A QUIC implementation should monitor the ECN functionality in order to provide input to e.g. service providers to improve ECN support in the networks. Items of interest are:

- Black holes, ECT or CE marked packets are discarded.

- Faulty remarking, e.g. ECT(0) is remarked to ECT(1) or Not-ECT.

- Continuous CE marking, possible indication of faulty on/off ECN marking, but can also be an effect of severe congestion.

- Degree of L4S support. L4S should generally give low queue latency. Estimation of one way queue delay for L4S enabled QUIC connections can be used to determine if there are congested nodes along the path that are not L4S capable.

## 3. IANA Considerations

T.B.D.

## 4. Open questions

A list of open questions:

- Is it sufficient that one peer sends an ECN negotiation challenge frame?.

- Should the ECN field in the ACK frame be mandatory ? (in which case it is not necessary to indicate its presence)

- Should all packets be ECT or should there be special patterns to improve fault detection.

- Write up a more detailed description on how to implement ECN support in QUIC for different OS stacks.

- Determine which ECN echo encoding in the ACK frame is the best alternative.

- Is a completely new ACK frame an alternative ?

- How do STOP_WAITING frames affect the ECN echo overhead.

- Outline possible connection migration actions

- Are there any security implications with the smalle ECN negotiation frame ?

## 5. Security Considerations

T.B.D

## 6. Acknowledgements

The following persons have contributed with comments and suggestions for improvements: Mirja Kuehlewind, Koen De Schepper, Piers O'Hanlon, Michael Welzl, Marcelo Bagnulo Braun, Martin Duke

# 7. References

## 7.1. Normative References

[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>.

## 7.2. Informative References

[Bagnulo] "Adding Explicit Congestion Notification (ECN) to TCP control packets and TCP retransmissions", <https://tools.ietf.org/id/draft-bagnulo-tcpm-generalized-ecn-00.txt>.

[ECN-fallback] "A Mechanism for ECN Path Probing and Fallback", <https://www.ietf.org/archive/id/draft-kuehlewind-tcpm-ecn-fallback-01.txt>.

[I-D.ietf-aqm-ecn-benefits] Fairhurst, G. and M. Welzl, "The Benefits of using Explicit Congestion Notification (ECN)", draft-ietf-aqm-ecn-benefits-08 (work in progress), November 2015.

[I-D.ietf-quic-transport] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-01 (work in progress), January 2017.

[I-D.ietf-tsvwg-ecn-experimentation] Black, D., "Explicit Congestion Notification (ECN) Experimentation", draft-ietf-tsvwg-ecn-experimentation-00 (work in progress), December 2016.

[McQuistin-Perkins] ""Is Explicit Congestion Notification usable with UDP?", Proceedings of the ACM Internet Measurement Conference, Tokyo, Japan, October 2015. DOI:10.1145/2815675.2815716", <https://csperkins.org/publications/2015/10/mcquistin2015ecn-udp.pdf>.

[OHanlon] "ECN support in different OS stacks", <https://mailarchive.ietf.org/arch/msg/rmcat/rRKF3PVmFL2zHCp1bOPKimqSsbM>.

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <http://www.rfc-editor.org/info/rfc6679>.

[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>.

[RFC7560] Kuehlewind, M., Ed., Scheffenegger, R., and B. Briscoe, "Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback", RFC 7560, DOI 10.17487/RFC7560, August 2015, <http://www.rfc-editor.org/info/rfc7560>.

Author's Address

Ingemar Johansson
Ericsson AB
Laboratoriegraend 11
Luleaa  977 53
Sweden

Phone: +46 730783289
Email: ingemar.s.johansson@ericsson.com