Coding for QUIC
draft-swett-nwcrg-coding-for-quic-00

Abstract

This document introduces means of integrating loss recovery coding in the proposed QUIC transport protocol. While no specific code is specified, the document defines how to integrate recent coding research to recover packets lost in QUIC sessions. This research targets the codes themselves as well as how to use them in real world protocols. Loss recovery should improve the QUIC performance in sessions impacted by loss.

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 https://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 September 4, 2018.

Copyright Notice

Copyright (c) 2018 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 (https://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 ............................................. 2
   1.1. Requirements Language ............................... 2
2. Design Considerations ...................................... 3
   2.1. Framing ............................................. 3
   2.2. Coding Symbol ....................................... 3
   2.3. Negotiation ......................................... 4
3. References .................................................. 4
   3.1. Normative References ............................... 4
   3.2. Informative References ............................. 4
Appendix A. Appendix: Reference Algorithms .................. 5
Appendix B. Appendix: Participating Middleboxes ............... 5
Appendix C. Appendix: APIS .................................. 5
Authors’ Addresses ............................................. 5

1. Introduction

QUIC is a new transport that wants to improve network performance by enabling out of order delivery, partial reliability, and methods of recovery besides retransmission while also improving security. This document specifies a design to enable error correcting codes to be used to recover lost data in QUIC. Error correcting codes have the ability to recover packet losses in less than 1 round trip at the cost of more total data transmission and decoding delay. The design does not specify a code but allows to negotiate it hence assumes that more than one code could be offered concurrently as well as leaving open the possibility of new codes in the future. Without loss of generality in the document we consider that the encoding operations compute a linear combination of QUIC packets. Terms and definitions that apply to coding are available in RFC xxxx [nc-taxonomy]

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 RFC 2119 [RFC2119].
2. Design Considerations

2.1. Framing

A new QUIC frame type is defined within the current framework [quic-basics] to add the error correction feature. This new frame type contains inputs to the negotiated loss recovery algorithm as specified by the specific algorithm and coded information. This frame payload begins with arguments that point to the negotiated function that generates the coding coefficients to be applied to the remaining payload. It also indicates the identity of the first packet in the coding window and its size as well as other necessary input. There are currently a number of options for identifying the coded packets. Firstly we can assign a new coded sequence number. If all packets are encoded in a session, then we can specify which packet is the first and with the window size it will be easy to recover the packets. If some packets are encoded and some are not then there could be gaps in the numbering that is not due to loss and we may need to specify which packets are inside the window using a form of run-length-encoding. We can specify a fixed-length field in order to identify the packets. Secondly, we could use the QUIC packet numbering which would reduce the overhead. This has the same functionality as using a sequence number except that in that case the gaps in numbering could be due to path migration and not losses. Note: a reference will be provided in a later version.

2.2. Coding Symbol

One of the the design consideration is the definition of the code symbol. In order to minimize the impact on the QUIC design, the QUIC loss recovery (QLR) will be applied inside the QUIC encrypted packet payloads. Hence raw packets are used as symbols, the units of recovery are what the coding coefficients are applied to. Any packet payloads smaller than the coded payload will be implicitly padded with zeros as to prevent the detection of coding on any path. To allow for the coded packets to have more encrypted payload than other packets, any QUIC PADDING frames(type 0x00) [quic-basics] will be removed from the payload before applying the algorithm. This new frame type contains inputs to the negotiated loss recovery algorithm as specified by the specific algorithm and coded information. We want to keep the coding implementation simple, provide for code negotiation and stay independent of any encryption mechanism. It is thus proposed to apply loss recovery code before the encryption, hence to clear data. This allows the encryption operation to be unencumbered by coding. Hence raw packets will be used as coding symbols. The downside of this approach is that it does not enable non-participating middleboxes to add or remove encoding from packets but this is considered insignificant compared to the complexities of
interacting with security mechanisms. In addition, it is assumed that the entire packet content will constitute a single source symbol. This choice is motivated by the desire to simplify the implementation. Coding should be applied to all QUIC packets except the 0RTT payloads. 0RTT payloads are sent prior to negotiation, and the QUIC negotiation mechanism does not allow sending extension frames prior to handshake completion.

2.3. Negotiation

There are already multiple candidates for the QUIC FEC and we assume that others may become available in the future as research continues. In order to stay as generic as possible and enable QUIC network operators to select their coding technology, it is assumed that a coding negotiation will be implemented to select one or more codes to be used over a QUIC session. This will be implemented using the one step negotiation of the new QUIC negotiation mechanism [quic-basics]. Each available coding algorithm should use the standard FEC frame [RFC3452] but reserve a different codepoint. We want to ensure that coding negotiation occurs during the QUIC handshake and can be used in all short header QUIC packets. Finally, to make the code selection as generic as possible, each algorithm should specify a sequence of coding coefficients or a function to generate them, window sizes as necessary as well as meta information to ensure the conformant performance of the coding and decoding operations.

3. References

3.1. Normative References


3.2. Informative References

[nc-taxonomy]

[quic-basics]

Appendix A. Appendix: Reference Algorithms

This ID does not mandate nor depends on any coding scheme. However, in order to have an initial implementation with good performance and not encumbered by intellectual property and proprietary implementations, it is suggested to use the Raptor (RFC 5053) as a reference algorithm. However, since the Raptor code perform badly with small blocks, depending on the application, another alternative is to use Reed-Solomon (RFC 5510) codes. It is assumed that other candidates that are free of IPR may become candidates in the future.

Appendix B. Appendix: Participating Middleboxes

The coding approach described in this document does allow on path elements that have the ephemeral keys to decrypt packets and add or remove FEC packets.

Appendix C. Appendix: APIS

It is planned that the QUIC coding mechanism will conform to any common API defined in the research group.

Authors’ Addresses

Ian Swett
Google
Cambridge, MA
US

Email: ianswett@google.com
Marie-Jose Montpetit
Triangle Video
Boston, MA
US

Email: marie@mjmontpetit.com

Vincent Roca
INRIA
Grenoble, France
US

Email: vincent.roca@inria.fr