Workgroup: QUIC Working Group
Internet-Draft:
draft-dmoskvitin-quic-short-message-fec-00
Published: 23 October 2023
Intended Status: Standards Track
Expires: 25 April 2024
Authors: D. Moskvitin Huawei
E. Onegin Huawei
R. Huang Huawei
H. Luo Huawei
Q. Chen Huawei

Forward Erasure Correction for Short-Message Delay-Sensitive QUIC Connections

Abstract

This document proposes a FEC scheme for single packet protection in accordance to the sender observed packet loss rate..

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 25 April 2024.

Table of Contents

1. Introduction

The QUIC protocol [QUICv1] is general purpose transport protocol, which supports both reliable and unreliable data transmission. Depending on application-specific use cases, one can choose either to rely on QUIC built-in loss recovery mechanism [QUIC-RECOVERY] or to use application layer retransmission mechanism. In both cases it takes more than one round-trip time to retransmit lost data, which may be crucial in delay-sensitive applications (e.g. RTC) if RTT is high enough. Forward erasure coding allows to avoid or minimize extra delay incurred by the retransmission of lost data, while redundancy adaptation minimizes data overhead generated by FEC..

There are several works considering the use of Forward Erasure Correction (FEC) in QUIC protocol:
[I-D.roca-nwcrg-rlc-fec-scheme-for-quic],
[I-D.swett-nwcrg-coding-for-quic] and [I-D.michel-quic-fec]. These works share the common idea of protecting a sequence of packets each containing protected data. But there is a broad area of applications there it’s only needed to send short messages of data (one or couple QUIC packets) in timely manner with relatively high time gap in-between, for example instant messaging, control channels, notifications, etc. In this case it’s more beneficial to protect every message independently of each over.

This document defines extensions to the QUIC protocol to add the ability to protect single packets independently and make it able to recover from packet losses prior to retransmission using FEC.

2. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Design Overview

In this document it's proposed to protect whole QUIC packet, motivated by the applications described in introduction. Therefore, each protected packet corresponds to several repair symbols, and each repair symbol correspond to one protected packet. Prior to encryption, serialized QUIC packet is divided into several parts, called shards, each shard acts as source symbol and sequence of shards corresponding to single QUIC packet act as message (see Figure 1).

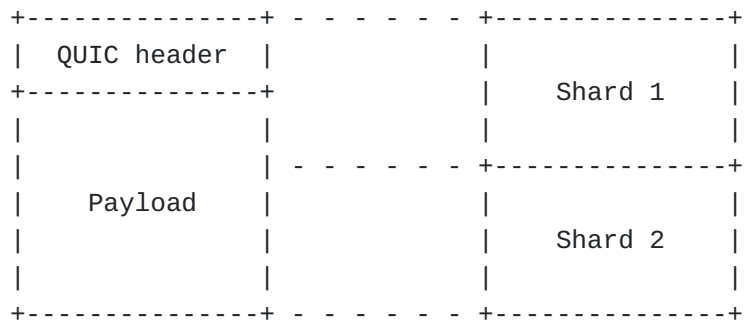

Figure 1: Example of packet division into two shards

Number of shards depend on encoding algorithm and parameters used for protection. If the size of protected packet is either too small or can't be evenly divided, then the protected packet may be padded with PADDING frame during serialization to achieve desirable size. It is noted that this method only protect the application data space.

After sending protected packet, sender **SHOULD** encode corresponding shards into repair symbols and send them afterwards. Several repair symbols **MAY** be sent in one QUIC packet, but it's **RECOMMENDED** to send repair symbols in separate packets. In general, distribution of repair symbols between packets depends on properties of concrete erasure code used. Each repair symbol **MUST** contain information about

corresponding protected packet number. It allows to avoid marking protected packets with some FEC-specific id.

Receiver **MAY** ignore repair symbols, particularly in the cases when corresponding protected packet is either already received or recovered. Repair symbols are accumulated on receiver either until recovery or the moment when recovery of corresponding lost packet is not possible anymore.

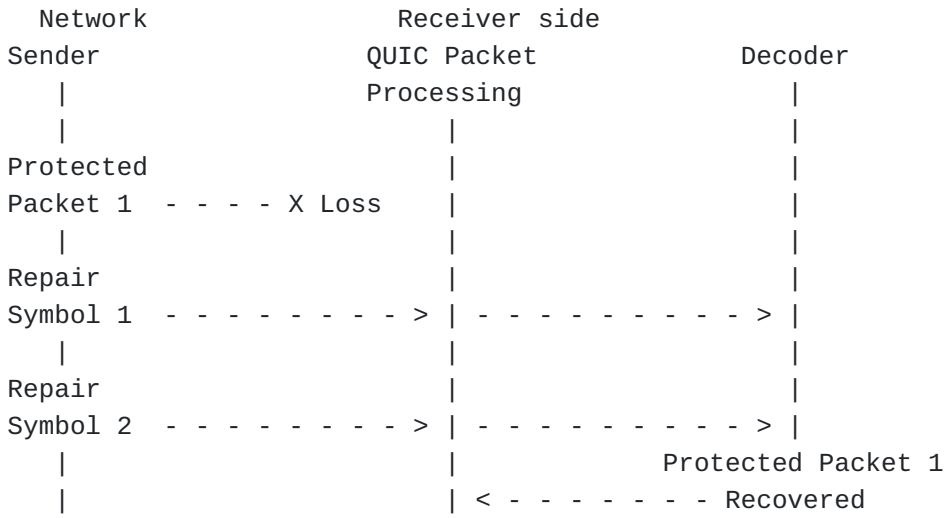

Figure 2: Example of protected packet recovery after two corresponding repair symbols received

4. FEC and Loss Recovery

The FEC extension described in this document **SHOULD** work as an addition to built-in QUIC loss recovery mechanism and intended only for faster recovery of data lost during transmission through network. Modules of QUIC, which rely on precise loss rate measurements (e.g. congestion control) during transmission, should be provided with actual packet loss rate from wire, rather than loss rate measured after recovery. Depending on repair symbol scheduling, packet recovery may be observed as packet reordering in wire, therefore should be properly accounted. Finally, FEC adaptive redundancy mechanism should consider possible losses incurred by congestion control overshooting and avoid positive feedback loop: recursively increase redundancy as a reaction to increased packet loss rate.

5. Protocol Extensions

QUIC protocol extensions are specified in this sectioin.

5.1. Handshake Negotiation and Transport Parameter

This extension defines a new transport parameter, used to negotiate the use of FEC extensions during the connection handshake, as specified in [QUICv1]. The new transport parameter is defined as follows:

enable_FEC: The enable_FEC transport parameter is included if the endpoint supports FEC mechanism and FEC extensions as defined in this draft. This parameter specify the the support FEC algorithm. How the FEC algorithm is encoded will be added in future versions.

5.2. Repair Symbol Frame

FEC-REPAIR Frame are used by a sender to contain the FEC symbol which will be used by the receiver to recover a lost packet. FEC_REPAIR frames are formatted as follow Figure 3.

```
FEC_REPAIR {
   Type (i) = TBD,
   FEC Version (i),
   Packet Number Length (2),
   Protected Packet Number (8..32),
   FEC Meta Data (..),
   Reserved (i) = 0,
   FEC Payload (..)
}
```

Figure 3: FEC_REPAIR frame format

It contains the following fields:

FEC Version: A variable-length integer which specifies the encoding algorithm used and the structure of the following FEC Meta Data field.

Packet Number Length: As defined in [QUICv1]

Protected Packet Number: The corresponding source packet number that the FEC frame protects. As defined in the packet number definition in [QUICv1].

FEC Meta Data: A structure, which is intended for additional information about repair symbol, such as identificator of repair symbol in the coding block, encoding parameters, FEC Payload size, etc. Contents of this structure depend on concrete implementation and encoding algorithm used.

Reserved: A variable-length integer reserved for future use. Default value is 0. FEC Payload: The bytes generated by FEC encoder for corresponding repair symbol.

5.3. Acknowledgement of recovered packets

In order to inform sender about what packets have been recovered, acknowledgement of recovered packets **MUST** be provided. There are two alternatives to be consider:

5.3.1. Alternative 1: New FEC ACK Frame

A new frame is proposed. Receiver **SHOULD** send FEC_ACK frame either after each recovery or with the first ACK frame after recovery.

```
FEC_ACK {
   Type (i) = TBD,
   FEC Latest Restored Packet Number (i),
   FEC Restored Bytes (i),
   FEC Restored Packets (i),
   FEC Restored Packet List Size (i),
   FEC Restored Packet Number (i) ...
}
```

Figure 4: FEC_ACK frame format

FEC Latest Restored Packet Number: A variable-length integer which contains latest restored packet number.

FEC Restored Bytes: A variable-length integer which contains number of bytes restored on receiver side. This field is intended for statistics collection and analysis on sender.

FEC Restored Packets: A variable-length integer which contains number of packets restored on receiver side. This field is intended for statistics collection and analysis on sender.

FEC Restored Packet List Size: A variable-length integer which contains amount of packet numbers following after in that frame.

FEC Restored Packet Number: A list of recovered packet numbers listed in restoring order (newer first).

This alternative doesn't modify the existing QUIC mechanisms and can be easily as a add-on to the current QUIC implementation.

5.3.2. Alternative 2: Extended ACK Frame

This alternative is to extend ACK Frame to include optional FEC recovery information as defined in Section 5.3.1.

6. Loss Rate Calculation

Using FEC recovery acknowledgement mechanism it’s possible to measure packet loss rate in network, considering the diagram as follow.

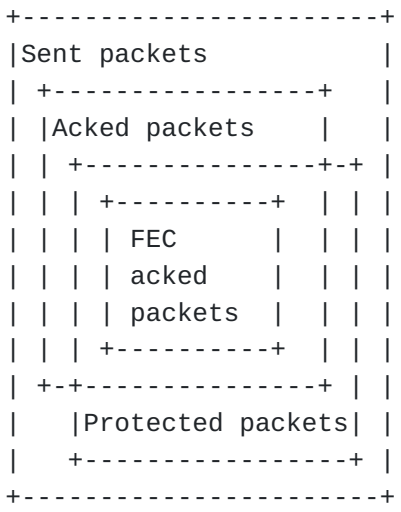

Figure 5: Relationship between acked and FEC acked protected packets

It is suggested that FEC_ACK is a subset of ACK. In another words, if some PN is FEC acked, then it’s also is acked.

To avoid the retransmssion of repaired lost packets, the receiver **SHOULD** include the recovered packets as the received packets in the ACK frames. When calculating the loss rate, the recovered packets **SHOULD** be considered as the lost packets as well.

7. Adaptive redundancy

In order to achieve maximum efficiency, sender should change encoding algorithm parameters to match network conditions. To be precise, from a set of algorithm parameters, which provide a certain protection level at current network packet loss rate (we provide example of relation between loss rate in BEC and performance of FEC scheme with fixed parameters as follow), choose one that have maximum coding rate. This way it will tend to reduce redundancy while providing certain level of protection.

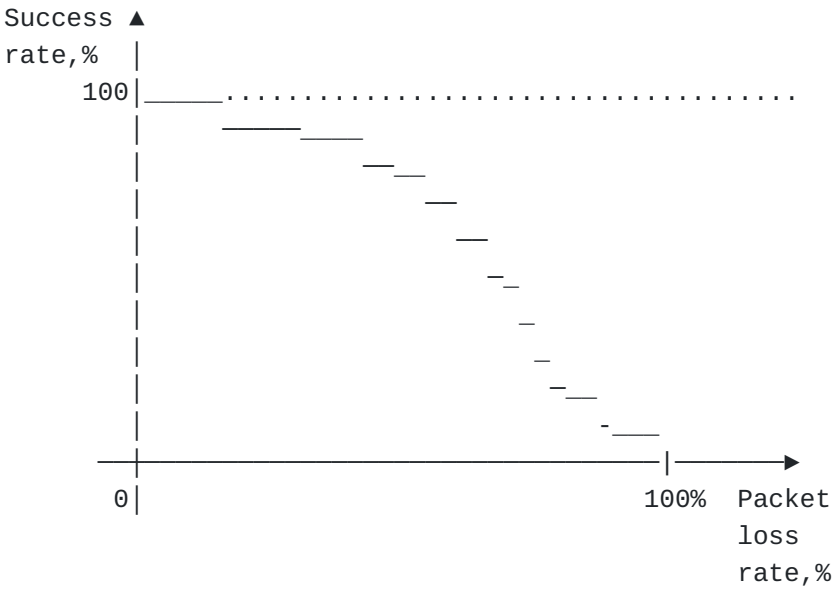

Figure 6: Example of performance profile of a given FEC scheme

8. Security Consideration

The FEC mechanism for QUIC only runs under 0-RTT and 1-RTT encryption levels and only operates inside the encrypted payload.

8.1. DoS due to difficult symbols recoveries

An attacker could try to cause a DoS of a receiver by selectively sending repair symbols to trigger intensive erasure correction operations on the receiver. A QUIC receiver is never forced to perform any erasure correction and may ignore any received repair symbol if it has doubts in its capabilities to decode it in a reasonable amount of time.

9. IANA Considerations

This document defines a new transport parameter for supporting FEC mechanisms, and possibly 2 new frame types.

Frame ID	Frame name	Specification
TBD	FEC Repair	{{fecrepair}}
TBD	FEC Ack	{{alternative1}}

Figure 7: List of new frames

10. References

10.1. Normative References

[QUIC-RECOVERY] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/rfc/rfc9002>.

[QUICv1] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

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

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

[rfc8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[rfc9265] Kuhn, N., Lochin, E., Michel, F., and M. Welzl, "Forward Erasure Correction (FEC) Coding and Congestion Control in Transport", RFC 9265, DOI 10.17487/RFC9265, July 2022, <https://www.rfc-editor.org/rfc/rfc9265>.

10.2. Informative References

[I-D.michel-quic-fec] Michel, F. and O. Bonaventure, "Forward Erasure Correction for QUIC loss recovery", Work in Progress, Internet-Draft, draft-michel-quic-fec-00, 21 October 2022, <https://datatracker.ietf.org/doc/html/draft-michel-quic-fec-00>.

[I-D.roca-nwcrg-rlc-fec-scheme-for-quic] Roca, V., Michel, F., Swett, I., and M. Montpetit, "Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for QUIC", Work in Progress, Internet-Draft, draft-roca-nwcrg-rlc-fec-scheme-for-quic-03, 9 March

2020, <https://datatracker.ietf.org/doc/html/draft-roca-nwcrg-rlc-fec-scheme-for-quic-03>.

[I-D.swett-nwcrg-coding-for-quic] Swett, I., Montpetit, M., Roca, V., and F. Michel, "Coding for QUIC", Work in Progress, Internet-Draft, draft-swett-nwcrg-coding-for-quic-04, 9 March 2020, <https://datatracker.ietf.org/doc/html/draft-swett-nwcrg-coding-for-quic-04>.

Authors' Addresses

Dmitry Moskvitin
Huawei

Email: dmitry.moskvitin@huawei.com

Evgeny Onegin
Huawei

Email: onegin.evgeny@huawei.com

Rachel Huang
Huawei

Email: rachel.huang@huawei.com

Hanlin Luo
Huawei

Email: luohanlin2@huawei.com

Qichang Chen
Huawei

Email: chenqichang1@huawei.com