Workgroup: QUIC
Internet-Draft: draft-michel-quic-fec-00
Published: 21 October 2022
Intended Status: Experimental
Expires: 24 April 2023
Authors: F. Michel O. Bonaventure
UCLouvain UCLouvain

Forward Erasure Correction for QUIC loss recovery

Abstract

This documents lays down the QUIC protocol design considerations needed for QUIC to apply Forward Erasure Correction on the data sent through the network.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://francoismichel.github.io/i-d-quic-fec/draft-michel-quic-fec.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-michel-quic-fec/.

Discussion of this document takes place on the QUIC Working Group mailing list (mailto:quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/. Subscribe at https://www.ietf.org/mailman/listinfo/quic/.

Source for this draft and an issue tracker can be found at https://github.com/francoismichel/i-d-quic-fec.

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 24 April 2023.

Copyright Notice

Table of Contents

1. Introduction

The QUIC protocol [QUICv1] relies on retransmissions to ensure the reliable delivery of stream data. Retransmitting the lost information requires the loss recovery mechanism to identify lost packets which may take up to several hundreds of milliseconds [QUIC-RECOVERY]. Depending on their delay-sensitivity, some applications using QUIC could not afford such a waiting time to ensure a good quality of experience to their users.

Works has already been done to consider the use of Forward Erasure Correction (FEC) for the QUIC protocol to ensure timely data delivery for delay-sensitive applications [QUIC-FEC] [FlEC] [rQUIC] [I-D.swett-nwcrg-coding-for-quic]. This document defines additions to the QUIC protocol to extend its loss recovery mechanism 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.

2.1. FEC-related definitions

Source symbol: piece of information exchanged by two endpoints. This document considers QUIC packets payloads as source symbols.

Repair symbol: redundant information constructed from the combination of several source symbols.

Erasure: loss of one or more symbols

Erasure correction code: algorithm generating repair symbols and reconstructing missing source symbols from a set of source and repair symbols.

Forward Erasure Correction: process of recovering erased symbols before the detection of their erasure.

FEC scheme: the conjunction of an erasure correction code and the specific protocol elements required to use it with the design described in this document.

Encoder: entity producing repair symbols using an erasure correction code. The encoder can be a library used by the protocol

implementation or a program running in a separate process or machine.

Decoder: entity reconstructing missing source symbols using an erasure correction code. The decoder can be a library used by the protocol implementation or a program running in a separate process or machine.

Coding window: window of source symbols that are required to decode a repair symbol.

3. The network channel and the coding channel

QUIC endpoints exchange information over a network channel. Adding Forward Erasure Correction to QUIC enables an endpoint to receive information over a coding channel. A coding channel can be seen as a communication channel between a QUIC receiver and a FEC decoder. The decoder often runs on the same machine as the QUIC receiver and can even be part of the protocol implementation itself. A source symbol is received through the coding channel when it is recovered by the FEC decoder instead of being explicitly received through the network. Only the data sent on the network channel is really transmitted on the wire between the two QUIC endpoints. Figure 1 illustrates how symbols can be received from both channels.

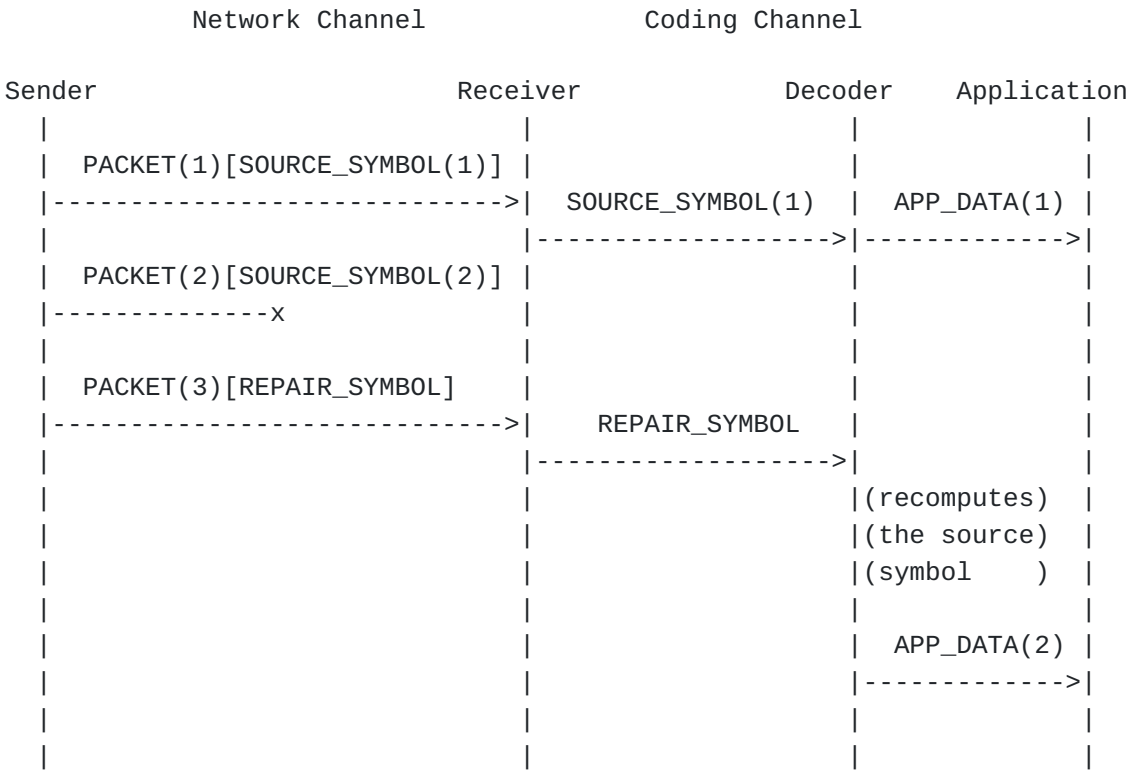

Figure 1: Receiving symbols through the network channel

In this illustration, the sender sends three QUIC packets through the network channel. Packets 1 and 2 carry one source symbol each and packet 3 carries one repair symbol protecting the two source symbols. The source symbols each carry application data (APP_DATA), e.g. through STREAM frames Packet 2 is lost due to network imperfection preventing SOURCE_SYMBOL(2) from being received through the network channel. The FEC decoder reconstructs SOURCE_SYMBOL(2) by combining SOURCE_SYMBOL(1) and REPAIR_SYMBOL. SOURCE_SYMBOL(2) is thus received through the coding channel. Note that SOURCE_SYMBOL(2) is not received as a packet since QUIC packets are only exchanged through the network channel. On the other hand, source symbols are carried by QUIC packets through the network channel and sent directly to the decoder through the coding channel..

4. FEC and the loss recovery mechanism

The FEC mechanism described in this document is an enhancement of the classical QUIC loss recovery mechanism [QUIC-RECOVERY]. It does not replace it by any means. A QUIC endpoint **MAY** ignore every received repair symbol and **MAY** not perform any symbol recovery at all. The FEC mechanism is only intended to allow a receiver recovering faster from packet losses on the network channel if it values the timeliness of data delivery.

5. Protocol requirements for protecting information through FEC

In this section, we list the points that must be defined by the protocol for allowing QUIC endpoints to protect information using FEC.

5.1. Defining the FEC-protected parts of a QUIC payload

There is no need to protect every piece information sent on the wire by QUIC. Some pieces of information are already sent redundantly (e.g. ACKs) and some data are not delay sensitive and can be retransmitted later with no harm (e.g. background download on a separate stream). Endpoints need to agree on which parts of the packets are part of the protected source symbols and how to compose a source symbol from what is sent on the wire.

Versions 01 and 02 of [I-D.swett-nwcrg-coding-for-quic] only protect streams payload. The idea is simple when using a single stream but becomes complicated and requires more signaling in a multi-stream scenario. It also cannot protect DATAGRAM frames. In this document, we propose to consider whole frames as part of the source symbols. Source symbols are thus the counterpart to QUIC packets for the coding channel. Packets carry frames through the network channel and source symbols allow receiving frames from the coding channel. In order to reduce signalling between the peers, a single source symbol

MUST NOT contain the frames of several QUIC packets at the same time.

5.2. Identifying the source symbols from QUIC packets

Upon reception of a QUIC packet, a receiver needs to identify the source symbols contained in the packet to forward to the FEC decoder. The decoder also needs to know which source symbols were lost. Since QUIC does not enforce sending contiguously increasing packet numbers, it is not possible for a receiver to distinguish a lost packet from a packet that has never been sent. Furthermore, a QUIC sender may not want to protect the payload of some packets if they do not carry latency-sensitive information. Source symbols are thus attributed a Symbol ID (SID). The SID of the first source symbol **MUST** be zero and the SIDs are contiguously increasing. When a FEC decoder notes gaps in the received SIDs, the missing SIDs correspond to lost source symbols that can be recovered using FEC.

As the QUIC packet number cannot be used to carry the SID, it must be transmitted using either a dedicated QUIC frame or a dedicated header field. The second solution being incompatible with [QUICv1], it is not discussed in this document. The source symbol payload can either be put inside a dedicated frame (Section 5.2.1) or infered when handling a specific frame (Section 5.2.2). Both alternatives lead to the exact same packet wire format and outcome.

5.2.1. Alternative 1: sending the source symbol inside a frame

This alternative is compatible with [QUICv1]. It defines a new SOURCE_SYMBOL frame as shown in Figure 2.

```
SOURCE_SYMBOL {
  SID (i),
  FEC Protected Payload (..)
}
```

Figure 2: SOURCE_SYMBOL frame format

The frame explicitly represents a source symbol. The FEC Protected Payload field is analogous to the payload of a QUIC packet: it contains a sequence of frames that are protected by FEC. The SOURCE_SYMBOL frame is idempotent and explicit: it exactly describes the frames inside the source symbol. The main drawback is that existing QUIC implementations are not used to write frames inside other frames which may increase the implementation cost of the approach. An example of the use of the SOURCE_SYMBOL frame is shown in Figure 3.

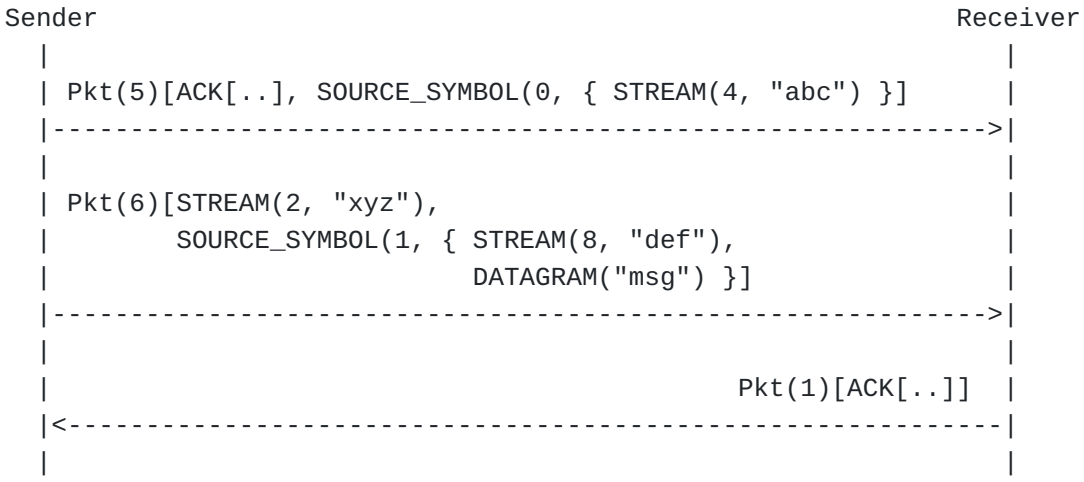

Figure 3: SID Alternative 1

The two SOURCE_SYMBOL frames contain the source symbols with SID 0 and 1. The first source symbol contains a frame for stream 4 while the second one contains a frame for stream 8 and a DATAGRAM frame. The ACK frame of packet 5 and the STREAM frame for stream 2 of packet 6 are not part of the source symbol and are thus not protected by FEC. In this scenario, every source symbol is correctly received and their reception can be deduced from the acknowledgements of packets 5 and 6.

5.2.2. Alternative 2: only sending the SID inside a frame

This alternative is compatible with [QUICv1]. It defines a new SID frame as shown in Figure 4.

```
SID {
  SID (i),
}
```

Figure 4: SID frame format

A QUIC packet carrying an SID frame means that the frames following the SID frames in the packet payload are part of a FEC source symbol. The SID of this symbol is represented by the only field of the SID frame. A packet **MUST NOT** contain more than one SID frame. A packet whose payload is FEC-protected **MUST** contain a SID frame whose SID field is the SID of the related source symbol. This alternative has one drawback: the SID frame is not idempotent since it is related to its containing packet. An example of the use of the SID frame is shown in Figure 5.

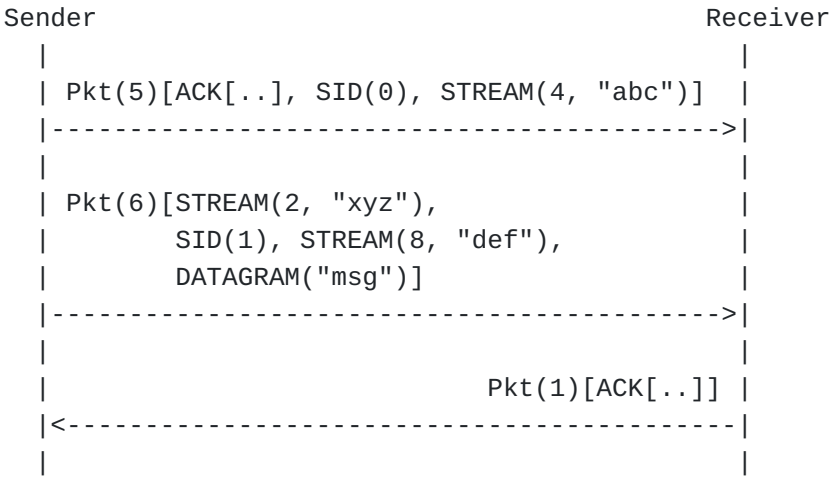

Figure 5: SID Alternative 2

The source symbols carried by the packets in this example are the same as for Figure 3 and the outcome and wire format are the same.

5.3. Sending the repair symbols

The repair symbols and the metadata attached to them are transferred using the REPAIR frame shown in Figure 6.

```
REPAIR {
  FEC Scheme Specific Repair Payload (1..),
}
```

Figure 6: REPAIR frame format

The payload of the REPAIR frame is specific to the underlying FEC scheme. In addition to the repair symbol itself, it may contain any metadata needed by the erasure-correcting code (e.g. identifying the source symbols protected by the repair symbol carried by the frame). Depending on the FEC scheme, the REPAIR frame **MAY** contain only a part of a repair symbol.

5.4. Announcing the coding window size

The receiver needs to store the received symbols in order to recover the lost source symbols. The FEC_WINDOW frame is sent by the receiver to announce the number of symbols that can be stored simultaneously by the receiver at a given point of time. The format of the FEC_WINDOW frame is described in Figure 7.

```
FEC_WINDOW {
  FEC Window Epoch (i),
  FEC Window Size (i),
}
```

Figure 7: FEC_WINDOW frame format

The FEC Window Epoch field is a unique identifier for the announced window. The first epoch is set to 0. Each time a new FEC_WINDOW frame is sent, the FEC Window Epoch field is increased by exactly one.

The Window Size field indicates the number of symbols that can be stored simultaneously by the receiver. The Window Size value overrides the window sizes received for smaller window epochs.

5.5. Announcing the recovered symbols

The FEC receiver **MAY** advertise the source symbols that have been received either through the network or coding channels to avoid the sender retransmitting the data of the recovered source symbols. This can be done using the SYMBOL_ACK frame as shown in Figure 8.

```
SYMBOL_ACK {
  Largest Acknowledged (i),
  ACK Range Count (i),
  First ACK Range (i),
  ACK Range (..) ...,
}
```

Figure 8: SYMBOL_ACK frame format

The frame has a similar format as the ACK frame, announcing the reception of SIDs instead of packet numbers. In addition to symbols recovered by FEC, this frame **MAY** also announce symbols received regularly through the network to avoid gaps in the ACK ranges and reduce the frame size. There is no obligation for a FEC receiver to send SYMBOL_ACK frames. The FEC receiver **MAY** decide to only advertize a subset of the received source symbols. The SYMBOL_ACK frame **MUST NOT** be used to infer any congestion state on the network (see Section 6).

6. Coding channel and congestion control

The coding and network channels being unrelated, receiving symbols through the coding channel **MUST NOT** be used to infer any congestion state on the network channel. More specifically, receiving a symbol through the coding channel **MUST NOT** be used to hide the network loss

event of the corresponding packet to the congestion control, applying Recommandation 1 of [RFC9265].

7. Negociating the FEC extension using transport parameters

This section defines the new transport parameters used to negociate and parametrize the FEC extension described in this document.

7.1. enable_fec

The use of the FEC extension is negociated using the enable_fec transport parameter defined in Table 1 :

Option	Definition
0x0	don't support FEC
0x1	supports FEC as defined in this document

Table 1: Values for enable_fec

When the enable_fec value is 0 or is not advertized by the peer, the QUIC endpoint **MUST NOT** use any frame or mechanism described in this document.

7.2. decoder_fec_scheme

Each QUIC endpoint uses the decoder_fec_scheme transport parameter to define the FEC scheme used to decode the received repair symbols. The QUIC sender **MUST** use the specified FEC scheme to generate repair symbols. The decoder_fec_scheme parameter is an integer value representing the identifier of the desired FEC scheme. For instance, a FEC scheme using Reed Solomon could be identified by the ID 0x0 and a FEC scheme using LDPC could be identified by 0x1.

This document does not specify nor identify any FEC scheme yet. Several FEC schemes have been proposed [I-D.roca-nwcrg-rlc-fec-scheme-for-quic] [RFC6865] [I-D.irtf-nwcrg-tetrys]. The next version of this document will detail how some of these schemes can be directly integrated in QUIC.

Future versions of this document will provide ways to format FEC scheme-specific payload for REPAIR frames. When the decoder_fec_scheme parameter is not advertized by the peer, the QUIC sender **MUST NOT** send any repair symbol.

7.3. initial_coding_window

Each QUIC endpoint uses the initial_coding_window transport parameter to define the initial coding window size it uses to store source and repair symbols (see Section 5.4). When the initial_coding_window parameter is not advertized by the peer, the

QUIC sender **MUST** consider a default value of 0 and **MUST NOT** send any repair symbol.

8. Security Considerations

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 source and 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

Disclaimer: the IDs defined in this section are present for experimental purposes only. They are not requested codepoints and are subject to change in the next versions of this document.

This document defines three new transport parameters and five new frames. The SID and SOURCE_SYMBOL frames serve the same purpose. One of them will be removed in next versions of this document. The values present in the tables are used for experiments.

9.1. New transport parameters

Parameter ID	Parameter name	Specification
0x238ffeceXX	enable_fec	Section 7.1
0x238ffecd	decoder_fec_scheme	Section 7.2
0x238ffecc	initial_coding_window	Section 7.3

Table 2: New transport parameters

The XX in 0x238ffeceXX are to be replaced by the version of this document that is implemented by the QUIC endpoint (e.g. the parameter ID for the version 00 of this document is 0x238ffece00).

9.2. New frames

Frame ID	Frame name	Specification
0x32a80fec	REPAIR	Section 5.3
0x32a80fec55	SOURCE_SYMBOL	Section 5.2.1
0x32a80fec1d	SID	Section 5.2.2
0x32a80fecac	SYMBOL_ACK	Section 5.5

Frame ID	Frame name	Specification
0x32a80fecc0	FEC_WINDOW	Section 5.4

Table 3: New frames

Acknowledgments

Maxime Piraux, Olivier Bonaventure and all the authors of [I-D.swett-nwcrg-coding-for-quic].

References

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>.

[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>.

Informative References

[FlEC] Michel, F., Cohen, A., Malak, D., Coninck, Q., Medard, M., and O. Bonaventure, "FlEC: Enhancing QUIC With Application-Tailored Reliability Mechanisms", IEEE/ACM Transactions on Networking pp. 1-14, DOI 10.1109/tnet.2022.3195611, 2022, <https://doi.org/10.1109/tnet.2022.3195611>.

[I-D.irtf-nwcrg-tetrys] Detchart, J., Lochin, E., Lacan, J., and V. Roca, "Tetrys, an On-the-Fly Network Coding Protocol", Work in Progress, Internet-Draft, draft-irtf-nwcrg-tetrys-03, 6 September 2022, <https://

datatracker.ietf.org/doc/html/draft-irtf-nwcrg-tetrys-03>.

[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>.

[QUIC-FEC] Michel, F., De Coninck, Q., and O. Bonaventure, "QUIC-FEC: Bringing the benefits of Forward Erasure Correction to QUIC", 2019 IFIP Networking Conference (IFIP Networking), DOI 10.23919/ifipnetworking.2019.8816838, May 2019, <https://doi.org/10.23919/ifipnetworking.2019.8816838>.

[RFC6865] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono, "Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6865, DOI 10.17487/RFC6865, February 2013, <https://www.rfc-editor.org/rfc/rfc6865>.

[rQUIC] Garrido, P., Sanchez, I., Ferlin, S., Aguero, R., and O. Alay, "rQUIC: Integrating FEC with QUIC for Robust Wireless Communications", 2019 IEEE Global Communications Conference (GLOBECOM), DOI 10.1109/globecom38437.2019.9013401, December 2019, <https://doi.org/10.1109/globecom38437.2019.9013401>.

Authors' Addresses

François Michel
UCLouvain

Email: francois.michel@uclouvain.be

Olivier Bonaventure
UCLouvain

Email: olivier.bonaventure@uclouvain.be