nwcrg
Internet-Draft
Intended status: Informational
Expires: September 10, 2020

I. Swett
Google
M-J. Montpetit
Triangle Video
V. Roca
INRIA
F. Michel
UCLouvain

March 9, 2020

Coding for QUIC
draft-swett-nwcrg-coding-for-quic-04

Abstract

This document focuses on the integration of FEC coding in the QUIC transport protocol, in order to recover from packet losses. This document does not specify any FEC code but defines mechanisms to negotiate and integrate FEC Schemes in QUIC. By using proactive loss recovery, it is expected to improve QUIC performance in sessions impacted by packet losses. More precisely it is expected to improve QUIC performance with real-time sessions (since FEC coding makes packet loss recovery insensitive to the round trip time), with short sessions (since FEC coding can help recovering from tail losses more rapidely than through retransmissions), with multicast sessions (since the same repair packet can recover several different losses at several receivers), and with multipath sessions (since repair packets add diversity and flexibility).

Table of Contents

1. Introduction

QUIC is a new transport that aims at improving network performance by enabling out of order delivery, partial reliability, and methods of recovery besides retransmission, while also improving security. This document specifies a framework to enable FEC codes to be used to recover from lost packets within a single QUIC stream or across several QUIC streams.

The ability to add FEC coding in QUIC may be beneficial in several situations:

- for a robust transmission of latency sensitive traffic, for instance real-time flows, since it enables to recover packet losses independently of the round trip time;
- for short sessions, in order to protect the last few packets sent, since it enables to recover from tail losses more rapidely than through retransmissions;
- for the transmission of contents to a large set of QUIC reception endpoints, since the same repair frame may help recovering several different packet losses at different receivers;
- for multipath communications, since repair traffic adds diversity and flexibility.

This framework does not mandate the use of any specific FEC code (i.e., how to encode and decode) nor FEC Scheme (i.e., that specifies both a FEC code and how to use it, in particular in terms of signaling). Instead it allows to negotiate the FEC Scheme to use at session startup, assuming that more than one solution could potentially be offered concurrently. Without loss of generality, we assume that the encoding operations compute a linear combination of QUIC packets, regardless of whether these codes are of block type (as with Reed-Solomon codes [RFC5510]) or sliding window type (as with RLC codes [RFC8681]).

2. Definitions and Abbreviations

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].

Terms and definitions that apply to coding are available in [nc-taxonomy]. More specifically, this document uses the following definitions:

Packet versus Symbol: a Packet is the unit of data that is exchanged over the network while a Symbol is the unit of data that is manipulated during the encoding and decoding operations

Source Symbol: a unit of data originating from the source that is used as input to encoding operations

Repair Symbol: a unit of data that is the result of a coding operation

This document uses the following abbreviations:

E: size of an encoding symbol (i.e., source or repair symbol), assumed fixed (in bytes)

3. General Design Considerations

This section lists a few general considerations that govern the framework for FEC coding support in QUIC.

3.1. FEC Code versus FEC Scheme, Block Codes versus Sliding Window Codes

A FEC code specifies the details of encoding and decoding operations. In addition to that, a FEC Scheme defines the additional protocol aspects required to use a particular FEC code [nc-taxonomy]. In particular the FEC Scheme defines signaling (e.g., information contained in Source and Repair Packet header or trailers) needed to synchronize encoders and decoders.

Block coding (e.g., Reed-Solomon [RFC5510]) and sliding window coding (e.g., RLC [RFC8681]) are two broad classes of FEC codes [nc-taxonomy]. In the first case, the input flow must be first segmented into a sequence of blocks, FEC encoding and decoding being performed independently on a per-block basis. In the second case rely, a sliding encoding window continuously slides over the input flow. It is envisioned that the two classes of codes could be used to bring FEC protection to QUIC, usually with an advantage for sliding window codes when it comes to low latency communications.

3.2. FEC Scheme Negotiation

There are multiple FEC Scheme candidates. Therefore a negotiation step is needed 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-transport], during the QUIC handshake.

Editor's notes:

* It is likely that FEC Scheme negotiation requires the use of a new dedicated Extension Frame Type. To Be Clarified and text updated.

* It is not clear whether negotiation is meant to select a **single** FEC Scheme or **multiple** FEC Schemes. In the second case (multiple FEC) it is required to have a complementary mechanism to indicate which FEC Scheme is used in a given REPAIR frame (which could be done through as many REPAIR frame type values as potential FEC Scheme negotiated). Is it what we want to achieve? Not sure.

3.3. FEC Protection Within an Encrypted Channel

FEC encoding is applied before any QUIC encryption and authentication processing. Source symbols, that constitute the data units used by the FEC codec, contain cleartext data (application and/or QUIC data).

3.4. About Middleboxes

The coding approach described in this document does not allow on path elements (middleboxes) to take part in FEC protection. The traffic being encrypted end-to-end, the middleboxes are not in position to perform FEC decoding, nor to add any redundant traffic.

4. FEC Protection Principles

The present section explains how FEC encoding can be applied to QUIC. It defines the general ideas for mapping QUIC packet frames to source symbols, as well as the associated signaling. This section does not define the FEC Scheme specific details that need to be specified in a companion document.

4.1. Cross Packet Frames FEC Encoding

A QUIC packet payload consists in a set of QUIC frames. These frames either carry application data (e.g., in a STREAM or DATAGRAM frame) or control information (e.g., a MAX_DATA frame). Each packet is either entirely received or lost, and is uniquely identified by a monotonically increasing Packet Number.

Through the use of FEC encoding, application data can be protected proactively against packet losses, without requiring to go through packet retransmission. In addition to application data, QUIC transfers might benefit from protecting control frames having a potential impact on the transmission throughput, such as MAX_DATA or

MAX_STREAM_DATA frames. Therefore this document introduces an FEC protection across all -- or a subset of -- the frames of a given QUIC packet. This design choice impacts the QUIC packet to source symbols mapping, as well as signaling aspects, both of them being discussed hereafter.

4.2. Source Symbol Definition

The cross packet frames FEC encoding approach considers the sequence of frames (or a sub-sequence of them) transmitted within a given QUIC packet, seen as the QUIC packet payload. From this payload, it defines a mapping to source symbols (see Section 4.2.1 and Section 4.2.2). Source symbols are then used for encoding purposes, producing one or more repair symbols, the details of which depend on the FEC Scheme considered. However source symbols are never sent per se on the network. Instead the original QUIC packet, plus a dedicated signaling header, are sent and therefore implicitely carry those source symbols. The QUIC packets, containing one or more repair symbols, are sent on the network.

The only modification to the original QUIC packet is the addition of a dedicated FEC_SRC_FPI frame type, meant to carry source symbol signaling (known as Source FEC Payload Information, or FPI). On the opposite, frames that carry one or more repair symbols use a dedicated REPAIR frame type. In both cases, in order to facilitate experiments and enable backward compatibility, the FEC_SRC_FPI and REPAIR frame types are chosen within the type range dedicated to "Extension Frames". Thereby, a legacy receiver will automatically ignore these unknown frame types. As QUIC packets can be of different lengths, a special care must be taken to ensure having a fixed Source Symbol size to ease FEC Scheme implementations.

4.2.1. Packet Payload to Packet Chunk Mapping

This section defines a mechanism to segment a QUIC packet payload, composed of several frames, into fixed-size payload chunks, of size E-1 bytes or E-1-4 bytes for the first chunk when the QUIC Packet Number needs to be added ((Section 4.2.2). Depending on the relative value of E-1 (or E-1-4) and the QUIC packet payload size, a packet can potentially contain more than one chunks. This is a first step into producing source symbols. Figure 1 illustrates this process.

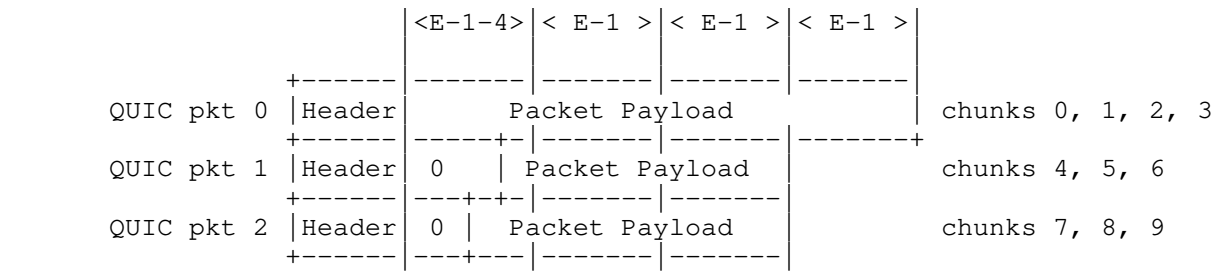

Figure 1: Example of QUIC packet to chunk mapping, when the E-1 value is relatively small, with prepended zero padding when needed (here packets 1 and 2), and assuming the first chunk contains the QUIC Packet Number in 4 bytes compressed version.

4.2.2. Packet Chunk to Source Symbol Mapping

The second step consists in producing the source symbols. A source symbols is the concatenation of a single byte of metadata, potentially followed by the Packet Number of the associated source, plus a packet chunk. Figure 2 illustrates the situation where a compressed QUIC packet number is added (in general for the first chunk of a QUIC packet). Figure 3 illustrates the situation where there is no QUIC packet number (in general for the following chunk(s) of a QUIC packet). When the QUIC packet number is present, this identifier can be recovered by a receiver after successful FEC decoding. It means that a RECOVERED frame can be generated to the sender to indicated that this packet (identified by the QUIC packet number) has been recovered. Each source symbol is of fixed-size E bytes. These source symbols are only used during encoding and decoding and are not sent as-is on the network.

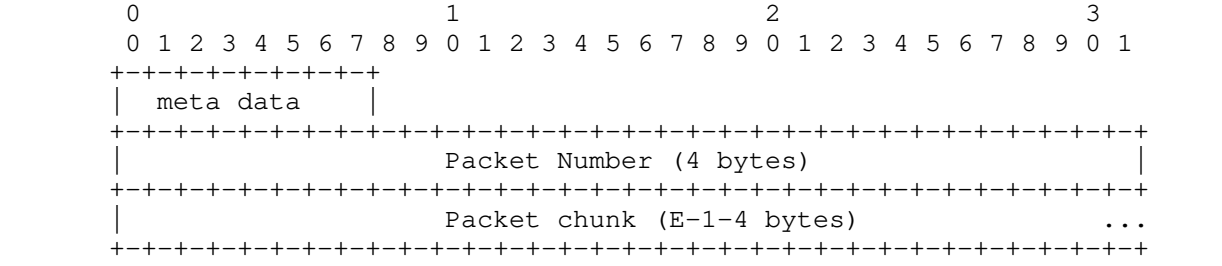

Figure 2: Source symbol format with Packet Number information (e.g., first packet chunk).

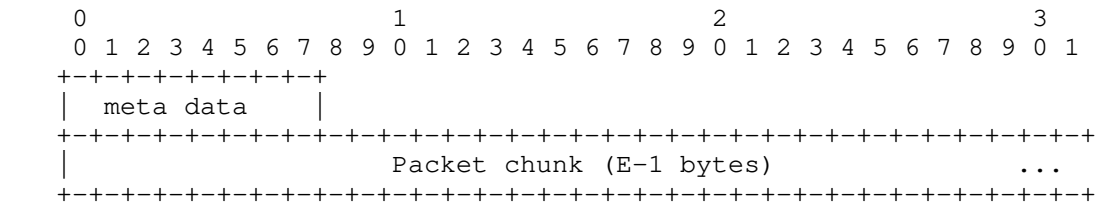

Figure 3: Source symbol format without Packet Number information (e.g., packet chunks except the first one).

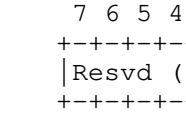

Figure 4: Source symbol metadata format.

Figure 4 shows the format of the 1 byte metadata. The fields are the following:

Reserved field (5 bits): for this specification, this field MUST be equal to zero.

Packet Number (N) field (1 bit): this field indicates that the following 4 bytes contain the Packet Number (short 32-bit representation) of the associated QUIC packet ([QUIC-transport] section 17.1., Packet Number Encoding and Decoding).

Start (S) bit (1 bit): this field, when set to 1, indicates that this source symbol contains the first chunk of the packet payload.

End bit (E) (1 bit): this field, when set to 1, indicates that this source symbol contains the last chunk of the packet payload.

Note that with a QUIC packet containing a single chunk, the associated metadata will contain S=E=1. On the opposite, a source symbol containing a intermediate chunk (i.e., neither the first nor the last chunk of the QUIC packet), the associated metadata will contain S=E=0.

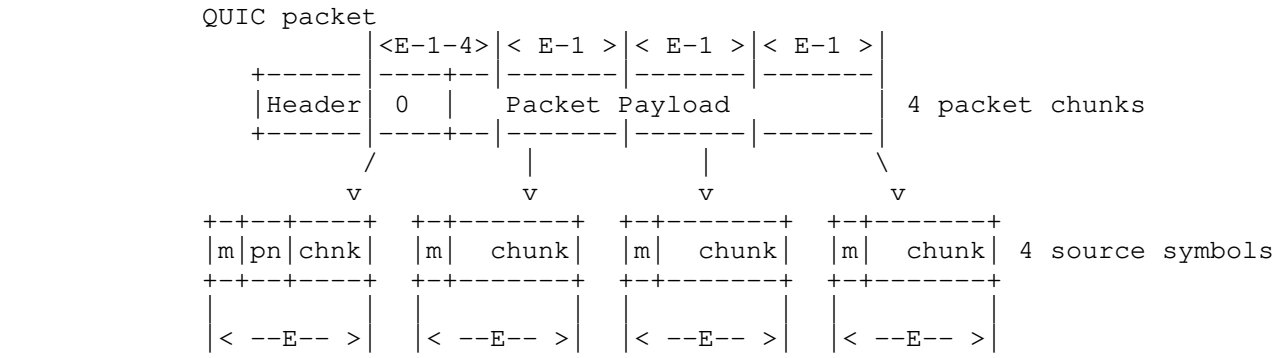

Figure 5: Example of packet chunk to source symbol mapping, when the E value is relatively small, in presence of the QUIC Packet Number for the first chunk.

Figure 5 shows an example where the 4 source symbols are created from the payload of a given QUIC packet. The first chunk may contain zero padding at the beginning in order to align the protected packet payload size to a multiple of E-1, and the first source symbol may contain the QUIC Packet Number.

Each source symbol is uniquely identified allowing to determine unambiguously its position in the source symbol flow. What information to associate to a source symbol to uniquely identify it is FEC Scheme dependent. Section 4.3 gives insight on this topic.

4.2.2.1. Open questions: Content of Source Symbols Metadata? Removing certain frames from FEC protection?

NB: section to remove once fixed.

During the FEC encoding phase, additional data can be added to the source symbol. These data are only added during the encoding and MUST NOT be transmitted on the network. The encoder and decoder MUST agree on the addition of these data to the source symbol in order to avoid decoding errors. Here are some examples of data that can be added to a source symbol during encoding and that will be decoded upon a source symbol recovery:

- The packet number: adding the packet number allows the decoder to know which packet has been recovered and potentially send a feedback of which packet has been recovered to the QUIC sender.

- Additional QUIC frames: the FEC encoder can for example add PADDING frames to a source symbol before proceeding to encoding. Adding PADDING frames to source symbols before encoding allows

protecting packets of different sizes. The smaller packet payload will be added PADDING frames to reach a size that is a multiple of E-1.

Note: Maybe the decision of adding data such as padding in the source symbols should be left to the underlying FEC Scheme.

Besides adding data to source symbols before encoding, some frames can be removed from the source symbol if their protection is not crucial for the transmission in order to reduce the size of the source symbol. For example, ACK frames can be systematically stripped out of the source symbols. Stream frames of non-delay-sensitive streams could also be removed from the source symbol. The encoder and decoder MUST agree on which frames must be stripped out of packet payloads. This information might for example be encoded in the Source Symbol ID by the FEC encoder.

Note: We might want to propose standard ways/algorithms to add/remove data before the encoding ?

TODO: Add a mechanism to add QUIC packet identifier to the metadata. It's useful.

4.2.3. Source Symbol Size (E) Considerations

The source symbol size, E, MUST be strictly greater than zero bytes and strictly smaller than the minimum PMTU value allowed by QUIC. The packet header is not part of the FEC-protected data. When the packet payload size is not a multiple of E-1, zero-padding MUST be added at the beginning of the first chunk of the packet payload. This is equivalent to inserting PADDING frames at the beginning of the payload. This zero-padding, only used for FEC encoding, SHOULD NOT be sent on the wire.

The choice of an appropriate value for E may depends on the use case (in particular on the nature of application data). A reasonably small value reduces the expected value of the added padding needed to align the payload size with a multiple of E-1, which can be a good approach when dealing with QUIC packets whose size significantly vary. However an overly small value also increases processing complexity (FEC encoding and decoding are performed over a larger linear system since there are more source symbols), so there is an incentive to use a larger value. An appropriate balance should be found, and this choice is considered as out of scope for this document. Since a repair symbol will transit through a frame, the E value must take this into account to avoid having REPAIR frames that do not fit into a single QUIC packet.

4.3. Source Symbol Signaling

An explicit signaling is needed by a decoder to identify the source symbols and their position in the block (i.e., for block codes) or coding window (i.e., for sliding window codes). While the QUIC packet number increases monotonically, it cannot be used to identify the position of a packet in the coding window as the packet number is not needed to increase by 1 for each new packet. There is thus an ambiguity on the decoder-side between lost packets and packets that do not exist. Similarly to FECFRAME, we propose to assign a identifier to source symbols to avoid this ambiguity. This identifier is opaque to the protocol and will be defined by the underlying FEC schemes. This is out of the scope of this document. An example of identifier could be an integer increasing by 1 for each new source symbol

In order to announce the source symbol identifier to the FEC decoder, we propose to add a new frame, the FEC_SRC_FPI frame to packets whose payload will contain one or more source symbols from the FEC decoder point of view. The FEC_SRC_FPI frame is part of the packet payload itself. Any packet containing a FEC_SRC_FPI frame MUST see its payload considered as one or more source symbol(s).

The FEC_SRC_FPI frame format is FEC Scheme specific and MUST be specified in the associated document.

4.4. Repair Symbol Signaling

An explicit signaling is needed by a decoder for each repair symbol received through a REPAIR frame. The goals are manyfold: identifying the repair symbols and their position in the block (i.e., for block codes) or coding window (i.e., for sliding window codes); carrying information on the way this repair symbol has been produced (e.g., with sliding window codes, it can indicate the encoding window composition).

One or more repair symbols can be present in a given QUIC packet. When there are multiple symbols, they SHOULD be concatenated in the same REPAIR frame. How to achieve this goal is FEC Scheme specific and therefore must be defined in the document describing this FEC Scheme.

4.5. Signaling a Symbol Recovery

When all the source symbols of a given QUIC packet have been lost but are recovered during FEC decoding, a QUIC receiver SHOULD advertise it to the sender in order to avoid the retransmission of already available data. However, the QUIC receiver MUST NOT acknowledge this

recovered packet through a regular acknowledement, as it would interfere with the behaviour of loss-based congestion controls such as [Cubic]. Therefore this document introduces a dedicated RECOVERED frame, that enables a receiver to indicate that a specific QUIC packet has been recovered through FEC decoding.

The RECOVERED frame works at the packet level. It is therefore required to be able to identify to which packet the recovered source symbols belong to. This is made possible by the QUIC packet identifier field added to the meta data prior to FEC encoding (Section 4.2.2).

4.6. About Gaps in the Set of Source Symbols Considered During Encoding

A given FEC Scheme MAY support or not the presence of gaps in the set of source symbols that constitute a block (for Block codes) or an encoding window (for Sliding Window codes). A potential cause for non contiguous sets of source symbols is the acknowledgment of one of them. When this happens, the QUIC sending endpoint may want to remove this source symbol from further FEC encodings. This is particularly true with Sliding Window codes because of their flexibility during FEC encoding (i.e., the encoding window can change between two consecutive FEC encodings).

Supporting gaps can be motivated by the desire to reduce encoding and decoding complexity since there are fewer variables. However this choice has major consequences in terms of signaling. Indeed each repair symbol transmitted MUST be accompanied by enough information for the QUIC decoding endpoint to unambiguously identify the exact composition of the block or encoding window. Without any gap, the identity of the first source symbol plus the number of symbols in the block or encoding window is sufficient. With gaps, a more complex encoding needs to be used, perhaps similar to the encoding used for selective acknowledgments.

Whether gaps are supported MUST be clarified in each FEC Scheme.

5. FEC Scheme Negotiation in QUIC

FEC Scheme negotiation has two goals:

- Selecting a FEC Scheme (or FEC Schemes) that can be used by the QUIC transmission and reception endpoints. This process requires an exchange between them;

- Communicating a certain number of parameters, the "Configuration Information", that are not expected to change over the session lifetime. For instance, this is the case of the symbol size

parameter, E (in bytes), that needs either to be agreed between the endpoints, or chosen by the sender and communicated to the receiver(s);

Editor's notes:

* It is likely that FEC Scheme negotiation requires the use of a new dedicated Extension Frame Type. The details remain TBD.

* The Negotiation Frame Type format remains TBD.

* How to communicate the parameters remains TBD.

* The present document only provides high level principles, the details are of course the responsibility of the FEC Scheme.

* In case negotiation is different when protecting a single versus several streams, this section may be moved to the respective sections.

* How does it work in case of a multicast session?

* Do we negotiate here a FEC Scheme on a per-Stream basis (or group of Streams to be protected jointly)? Or do we negotiate a FEC Scheme on a QUIC session basis, therefore to be used for all the Streams that need FEC protection?

5.1. FEC Scheme Negotiation

Before defining the transport parameters, we define two structures, encoder_fec_scheme_t and decoder_fec_scheme_t, in Figure 6. The config field is an opaque field allowing the decoder to define supported configuration information for the associated FEC Scheme. A FEC Scheme specification MUST define the set of valid configurations for the FEC Scheme.

```
struct {
    varint fec_scheme_id;
} encoder_fec_scheme_t

struct {
    varint   fec_scheme_id;
    uint16_t config_length;
    uint8_t  config[config_length];
} decoder_fec_scheme_t
```

Figure 6: encoder_fec_scheme_t and decoder_fec_scheme_t structures.

The following three transport parameters are used by the QUIC endpoints to negotiate the FEC Scheme used during the connection.

- supported_encoder_fec_schemes: list of supported FEC schemes for the encoding part listed from the most to the least preferred. The value of this parameter consists in a list of encoder_fec_scheme_t. When announcing a FEC Scheme, the encoder MUST be able handle every FEC Scheme configuration considered valid.

- supported_decoder_fec_schemes: list of supported FEC schemes for the decoding part listed from the most to the least preferred. The value of this parameter consists in a list of decoder_fec_scheme_t, each one representing the ID of a supported FEC Scheme.

- receiving_symbol_size: the size in bytes of the symbols the peer is willing to receive and recover. The value is a 16-bits integer.

Since communications can be bidirectional, each QUIC endpoint can provide the three parameters. Conversely, providing an empty list indicates this endpoint does not support FEC for the associated communication path (e.g., an empty supported_decoder_fec_schemes list indicates this endpoint cannot perform FEC decoding).

The decoding FEC Scheme of a QUIC endpoint is set to the first FEC Scheme listed in its own supported_decoder_fec_schemes that also appears in the peer's supported_encoder_fec_schemes. The encoding FEC Scheme of a QUIC endpoint is set to the first FEC Scheme listed in the peer's supported_decoder_fec_schemes that also appears in its own supported_encoder_fec_schemes. The encoder-side symbol size (E) of a QUIC endpoint is set to the value announced by the peer's receiving_symbol_size transport parameter. The decoder-side symbol

size of a QUIC endpoint is set to the value announced in its own receiving_symbol_size transport parameter.

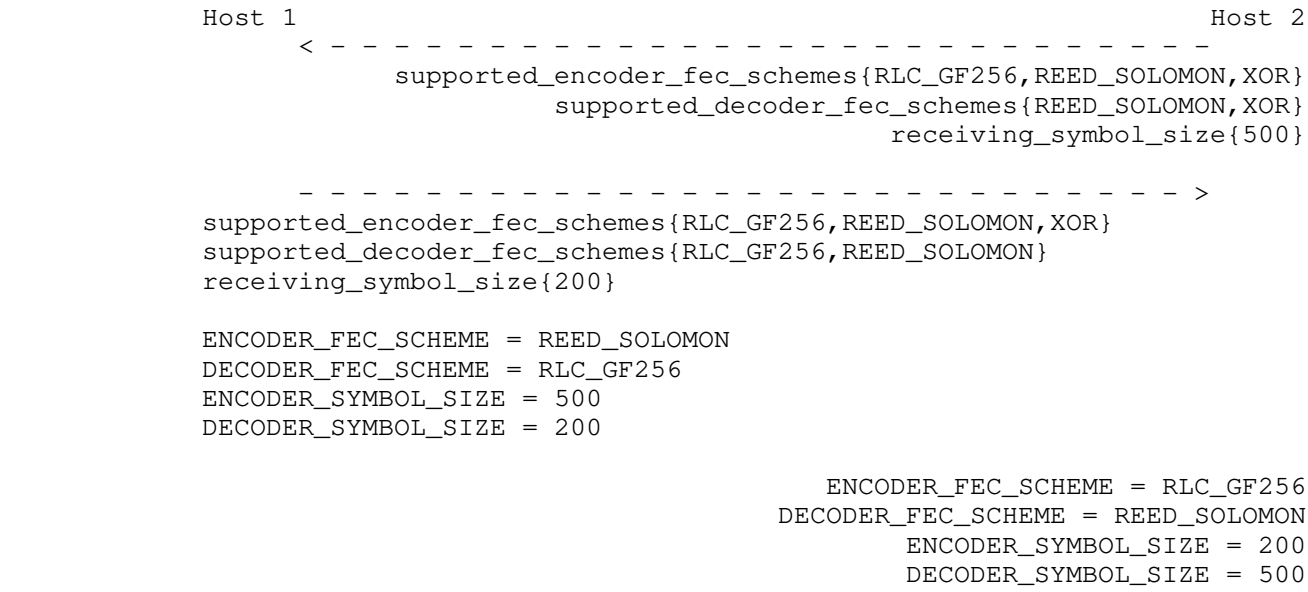

Figure 7: Example FEC Schemes negotiation during the QUIC handshake.

It is possible that the QUIC endpoint that receives one or more FEC Scheme proposals from the initiator cannot select any of them. In that case the negotiation process fails and no FEC protection is used.

6. Security Considerations

TBD

7. IANA Considerations

TBD

8. Acknowledgments

TBD

9. References

9.1. Normative References

[Cubic] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", RFC 8312, DOI 10.17487/RFC8312, February 2018, <https://www.rfc-editor.org/info/rfc8312>.

[QUIC-transport] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport (Work in Progress) (work in progress), January 2019, <https://datatracker.ietf.org/doc/draft-ietf-quic-transport/>.

[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/info/rfc2119>.

9.2. Informative References

[nc-taxonomy] Roca (Ed.) et al., V., "Taxonomy of Coding Techniques for Efficient Network Communications", Request For Comments RFC 8406, June 2018, <https://datatracker.ietf.org/doc/draft-irtf-nwcrg-network-coding-taxonomy/>.

[RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes", RFC 5510, DOI 10.17487/RFC5510, April 2009, <https://www.rfc-editor.org/info/rfc5510>.

[RFC8681] Roca, V. and B. Teibi, "Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for FECFRAME", RFC 8681, DOI 10.17487/RFC8681, January 2020, <https://www.rfc-editor.org/info/rfc8681>.

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
Univ. Grenoble Alpes
France

Email: vincent.roca@inria.fr

Francois Michel
UCLouvain
Louvain-la-Neuve
Belgium

Email: francois.michel@uclouvain.be