Coding for QUIC – RLC for QUIC

draft-swett-nwcrg-coding-for-quic-01
draft-roca-nwcrg-rlc-fec-scheme-for-quic-00

Ian Swett, Marie-José Montpetit, Vincent Roca

IETF102, Montreal, July 19th, 2018

Status

- **most of the ideas and techniques should be in the two I-Ds**
  - general considerations
    - ✓ https://datatracker.ietf.org/doc/draft-swett-nwcrg-coding-for-quic/
  - application to RLC sliding window codes
    - ✓ https://datatracker.ietf.org/doc/draft-roca-nwcrg-rlc-fec-scheme-for-quic/
    - ✓ **RLC as the first example, others to add**

Main principles

1. FEC protection at the STREAM level
2. FEC negotiation
3. frame data to source symbol mapping
4. transmission in STREAM and REPAIR Frames

5. … FEC protection across several STREAMs

1- FEC protection at the STREAM level

- **key architectural principle**
  - FEC protection within a **single** STREAM of a QUIC session
    - ✓ **protect flow(s) that need it within the QUIC session**
    - ✓ **do not leverage on the QUIC "packet number" field**

- **open question**
  - FEC protection across **two or more** STREAMs of the same QUIC session?
    - ✓ **seems feasible… but it adds more complexity!**
    - ✓ **is it worth?**

2- FEC Scheme negotiation

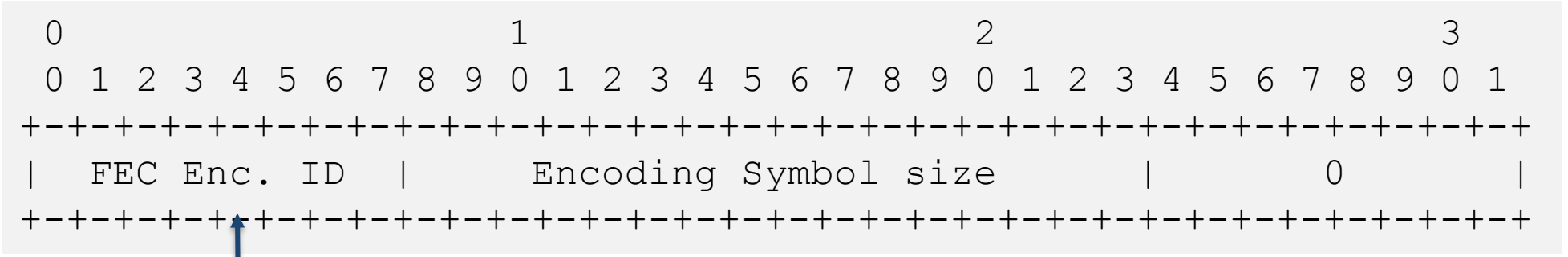
- **an endpoint initiates negotiation and lists supported FEC Schemes**
- **the other side selects the one preferred**
- **static parameters are always piggybacked**
  - meant to carry FEC Scheme configuration information (next slide)

```
QUIC sender                                          QUIC receiver
      < - - - - - - - - - - - - - - - - - - - - - - -
              supported_fec_scheme_32b{FEC_Encoding_ID1 | other}
              supported_fec_scheme_64b{FEC_Encoding_ID2 | other}

chooses FEC Scheme 1
      - - - - - - - - - - - - - - - - - - - - - - - >
  supported_fec_scheme_32b{FEC_Encoding_ID1 | other}
```

FEC Scheme negotiation: ex. of RLC (2)

- **RLC Configuration Information**
  - FEC Encoding ID (8 bits):
    - IANA registered identifier for RLC for QUIC
  - Encoding symbol size, E (in bytes) (16 bits):
    - size of any source/repair symbol



identifies the FEC Scheme

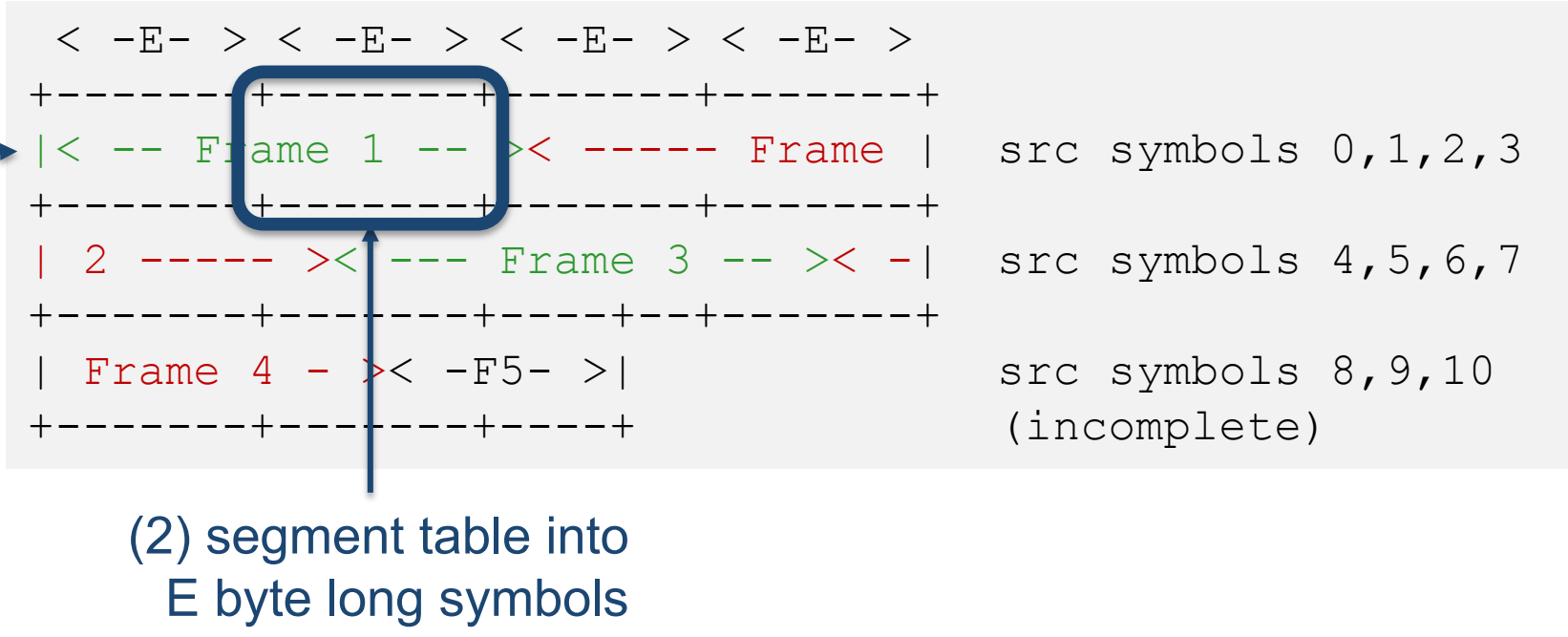
FEC Scheme negotiation (3)

- **open question**
  - is it worth selecting several FEC Schemes within the same QUIC session?
    - ✓ **FEC codes have different features: sliding window for real-time, block code for bulk non-real-time content**
    - ✓ **but adds complexity (e.g., need to further identify which FEC Scheme is used in each STREAM)**

3- From frame data to source symbols

- **mapping source symbols to frame data (input)**
  - application/frame data is of **variable** size but source symbols are **fixed** size
  - solution: mapping through a table

(1) fill in frame data in sequence (no gap)



(2) segment table into E byte long symbols

From application data to source symbols (2)

- **on the choice of E (i.e., the symbol size)**
  - any value possible, as long as a frame containing a repair symbol can fit into a QUIC packet
  - source symbol can straddle several STREAM data frames
    - ✓ **bad for reliability but almost unavoidable**
  - small source symbols reduce risk
    - ✓ **but increase complexity**
  - find an appropriate balance!

No need for a separate Encoding Symbol ID (ESI)

- **ESI are traditionally symbol sequence numbers**
  - e.g., to identify symbols within the encoding window or block

- **useless here because:**
  - source data
    - ✓ **QUIC Offset field always enable to identify frame data position within the frame/symbol mapping table**
  - repair data
    - ✓ **do not need anything**

- **seems anecdotic but in practice it's a key point!**

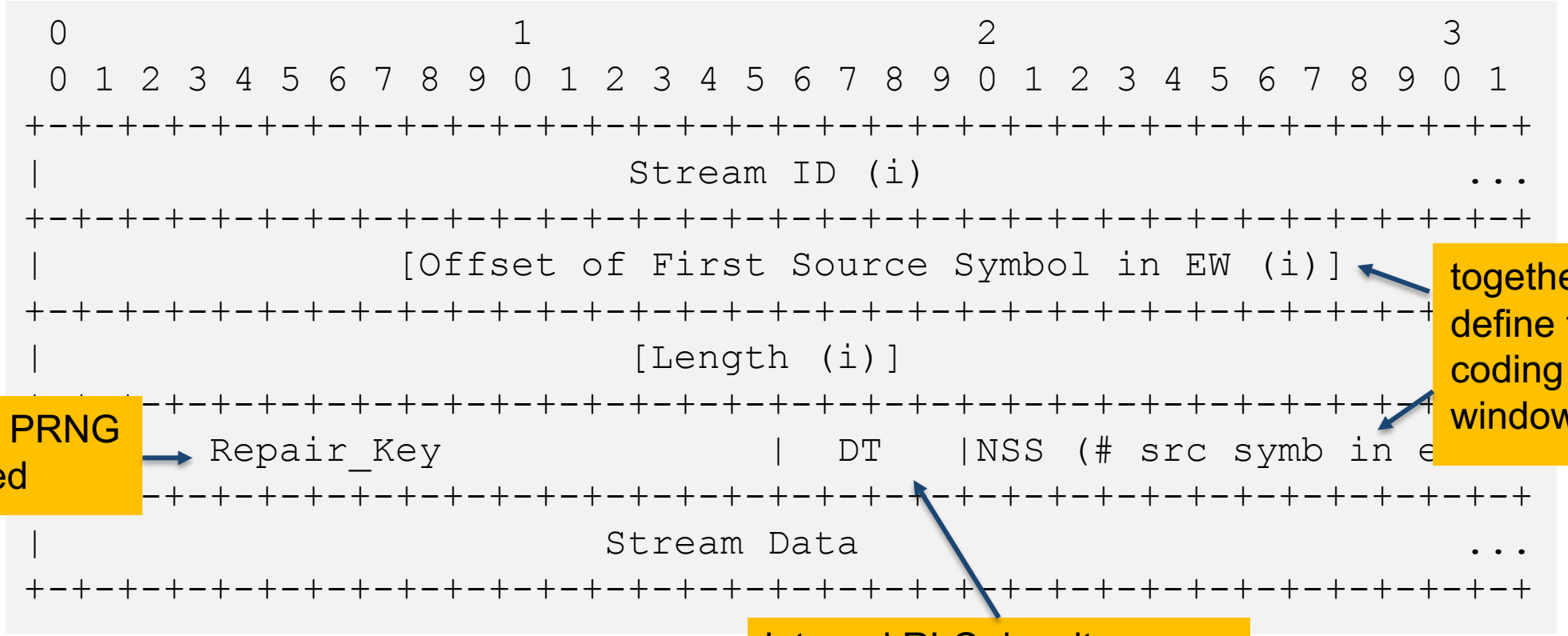
4- Transmission in STREAM and REPAIR Frames

- **no change for source data flow ☺**
  - fully **backward compatible**
    - ✓ **no need for a new frame type**
    - ✓ **any legacy QUIC receiver can process source data**

- **carried in dedicated REPAIR Frames**
  - defined as an **”extension frame”**
  - reuse the same REPAIR frame type for all FEC Schemes, even if the format changes
  - reuses the same STREAM ID (it’s for the same data flow)

Tx in STREAM and REPAIR Frames: ex. of RLC (2)

- **REPAIR format with RLC**



Management of silent periods and end of stream

- **classical difficulty**
  - last source symbol may not be filled in case of a silence!
- **potential solution(?)**
  - timer based
  - upon time-out, fall back to the alternative retransmission based loss recovery mechanism for the bytes of the last incomplete source symbol
  - … needs more thoughts/experiments