Main considerations

• inline with “Coding for QUIC” -03 I-D
  o “cross packet FRAMES” approach
  o being FRAME agnostic and compatible with DATAGRAM FRAMES is a must

• leverages on RFC 8681 (RLC) and RFC 8682 (TinyMT32) specifications
  o do not repeat code internals, focus on QUIC specific mapping and signaling
    ➔ keeps this I-D very compact

• potentially add similar I-Ds for other FEC codes and FEC schemes
  o different features, different use cases, for instance (to be tested):
    o real-time, a few tens of packets ➔ RLC on GF(2^8)
    o higher BDP, a few thousands of packets ➔ block code (Raptor(Q), LDPC)
QUIC packet to source symbols mapping

- to be used for FEC encoding/decoding, not transmitted per se
Source QUIC packets with their FPI

• add a **SRC_FPI frame** to the original QUIC packet, then send it
  - no other change to the original QUIC packet (no padding, no meta-data, we do not transmit source symbols per se)
  - the SRC_FPI frame is **ignored** by a QUIC receiver that does not support FEC

```
+-----------------+-----------------+-----------------+-----------------+
|                  |                  |                  |                  |
| 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 |
|                  |                  |                  |                  |
| ID of First Source Symbol in Packet (i)  ... |
```


Repair QUIC packets

- transmit repair symbols in dedicated **REPAIR frames**
  - it includes repair FPI signaling information
  - one or more REPAIR frames can be packed in the same QUIC packet (e.g., if E is small WRT the PMTU), for reduced overhead
  - a REPAIR frame is **ignored** by a QUIC receiver that does not support FEC

<table>
<thead>
<tr>
<th>ID of First Source Symbol in EW (i)</th>
<th>Repair_Key</th>
<th>NSS</th>
<th>Repair Symbols</th>
<th>DT</th>
<th>NRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+-----------------+-----------------+------------------+
| 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 | | |

**to be able to compute the frame length**
Situation

• François has a compliant open-source implementation
  o we know it can work as specified

• A few open questions:
  o (from ietf105 discussion:) considering a subset of the QUIC packet frames
  o (from ietf105 discussion:) choice of E
  o move the "source symbols mapping" to this I-D
    o it's FEC scheme dependent
  o QUIC frames are all idempotent:
    o “a valid frame does not cause undesirable side effects or errors when received more than once.”
      … except our SRC FPI frame that is attached to a given QUIC packet
    o not a major issue, it's just an exception
Thanks

• Additional slides, from ietf105 “Coding for QUIC” presentation. They still apply here…
QUIC packet to source symbols mapping (1)

- requirements:
  1. packets are of variable size, symbols are of fixed size \( E \), so **we need a mapping**
  2. the **symbol size**, \( E \), needs to be initialized wisely:
     - "small \( E \)" is fine when the QUIC packet sizes is largely variable (very small + a few very large packets), but has a cost
     - "large \( E \)" makes it simple (everything fits in a single symbol) but is suboptimal with a majority of small packets, and it may require to fragment the QUIC/UDP at IP level
     - \( E \) could be adjusted depending on the target use-case (if known)
  3. anticipate the potential need to **avoid exceeding the PMTU** (we add extra FEC related signaling)
     - choose \( E \) small enough
**QUIC packet to source symbols mapping (2)**

- **Step 1:** from QUIC packet payload to **chunks**
  - packet payload data is of **variable** size but source symbols are **fixed** size (E bytes)
  - use QUIC **zero padding FRAMES before** the payload to align payload size

  *to avoid problems with STREAM frames that do not encode their length*

(1) pad with initial padding FRAMES to align packet size

<table>
<thead>
<tr>
<th>pkt 0</th>
<th>Header</th>
<th>Packet Payload</th>
<th>chunks 0, 1, 2, 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>pkt 1</td>
<td>Header</td>
<td>0</td>
<td>Packet Payload</td>
</tr>
<tr>
<td>pkt 2</td>
<td>Header</td>
<td>0</td>
<td>Packet Payload</td>
</tr>
</tbody>
</table>

(2) segment padded payloads into E-1 byte long chunks
(E-5 for the 1st one to leave room for additional info)
QUIC packet to source symbols mapping (3)

- **Step 2:** from a packet chunks to **source symbols**
  - **1st chunk:** prepend a long signaling (5 bytes)

<meta-data + packet number + 1st chunk> constitute the source symbol
**QUIC packet to source symbols mapping (4)**

- **Step 2:** from a packet chunks to **source symbols**
  - 1\textsuperscript{st} chunk: prepend a long signaling (5 bytes)
  - **following chunks:** prepend a short header (1 byte)

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+----------+----------+----------+----------+
| metadata | metadata | metadata | metadata |
+----------+----------+----------+----------+
```

<meta-data + chunk> constitute the source symbol
QUIC packet to source symbols mapping (5)

- **Step 2:** from a packet chunks to **source symbols**

  - meta-data (all chunks):
    - N: packet Number is there
    - S (start): first chunk of a QUIC packet
    - E (end): last chunk of a QUIC packet

  - Packet Number (4 bytes) (first chunk only, optional):
    - when "N" (Packet Number) field is 1 in the meta-data
    - required at a receiver to determine the QUIC packet number associated after decoding all the symbols of the lost packet
Yes, we need this extra meta-data / packet number

| recvd or decoded | +-------------+ +-------------+ +-------------+ +-------------+ | recvd or decoded or lost... |
|-----------------|-----------------|------------------|----------------|------------------|----------------|
| | m | pn | chnk | | m | chunk | | m | chunk | | m | chunk | | m | chunk |
| +-------------+ | +-------------+ | +-------------+ | +-------------+ | +-------------+ |

S=1: we know the chunk contains one or more padding FRAMES plus original FRAMES
N=1: we can also recover the corresponding QUIC Packet Number

E=1: we know this chunk is the last one of packet

we have all chunks (S=1 | middle | middle | E=1) of the QUIC packet. Done 😊

decoding successful for those 4 source symbols