FECFRAME – extension
Adding sliding window codes support to FECFRAME

Vincent Roca, Inria, France
Ali Begen, Networked Media, Turkey

November 2017, IETF100, Singapore
Note well for FECFRAME-ext + RLC l-Ds

- we, authors, didn’t try to patent any of the material included in this presentation/I-D
- we, authors, are not reasonably aware of patents on the subject that may be applied for by our employer
- if you believe some aspects may infringe IPR you are aware of, then fill in an IPR disclosure and please, let us know
Reminder: this I-D is about…

- …an EXTENSION of the FEC Framework (or FECFRAME) / RFC 6363

- goal of FECFRAME is to add AL-FEC protection to real-time unicast or multicast flows

- goal of this FECFRAME-extension is to add support for Sliding Window AL-FEC codes
  - better suited to latency constrained data flows than block codes (e.g. Reed-Solomon, LDPC, Raptor(Q))
Situation and next step

- individual I-D -00: July 2016
- WG item I-D -00: July 2017

authors’ opinion:
- a small update needed to fix typos
- then ready for WG Last Call
Sliding Window Random Linear Codes (RLC) FEC Scheme
...for FECFRAME - extended

Vincent Roca, Belkacem Teibi, Inria, France

November 2017, IETF100, Singapore
Main changes: two FEC Schemes

- **CHANGE 1**: new co-author
  - added Belkacem who’s working with me on RLC…

- **CHANGE 2**: two FEC Schemes
  - **OLD**: a single FEC Scheme for GF(2), GF(2⁴) and GF(2⁸)
    - drawback: a compatible codec must support all three variants even if a single one is used
  - **NEW**: two separate FEC Schemes (for the moment)
    - one for GF(2): for very large encoding windows
    - one for GF(2⁸): for small encoding windows
    - simplifies a little bit a compatible codec by removing unrequired compliancy
Main changes: add density parameter

CHANG 3: add a density parameter

- **OLD**: a repair symbol is the sum of **all** source symbols
- **NEW**: a repair symbol is either the sum of **all** source symbols or a subset of them

```
repair_1 = α_1 * src_1 + α_2 * src_2 + α_3 * src_3 + α_4 * src_4 + α_5 * src_5 + α_6 * src_6

density 1
```

+ potentially sparse equations

```
repair_1 = α_1 * src_1 + 0 + 0 + α_4 * src_4 + α_5 * src_5 + 0

density 1/2
```
Main changes: add density parameter (2)

- motivation for change 3: increase applicability with larger window size support

  - **small encoding windows with maximum density**
    - algorithmic complexity is not an issue
    - dense equations enable maximum loss recovery performance

  - **larger encoding windows with reduced density**
    - reduces algorithmic complexity (limiting factor)
    - loss recovery performance remains good because larger encoding windows compensate
Main changes: add density parameter (3)

- made possible with a new parameter
  - Coding coefficients Density Threshold, DT (4-bit field)
    - 0 (density 1/16) up to 15 (density (15+1)/16=1)

- added to each FEC repair packet
  - DT value may change dynamically if loss conditions evolve

Figure 7: Repair FEC Payload ID Encoding Format

- an input parameter to the `generate_coding_coefficients()` function
Main changes since IETF’99 (4)

- does it work?
  - clear benefits at the encoder
    - density ½ → reduce complexity almost by half
  - at a decoder, interesting but we expect higher gains
    - work in progress…

<table>
<thead>
<tr>
<th>Encoding</th>
<th>window size=200 symbols, decoding window size=266, linear system size=532</th>
</tr>
</thead>
<tbody>
<tr>
<td>density 1(old)</td>
<td>- 120Kmh, 20% : encoding: 165 Mbps decoding: 341 Mbps</td>
</tr>
<tr>
<td>density ½ (new)</td>
<td>- 120Kmh, 20%: encoding: 291 Mbps decoding: 408 Mbps (+76.4% bitrate increase) (+19.6% bitrate increase)</td>
</tr>
</tbody>
</table>

MacBook Pro 15.4” Retina Core i7 quad-core, 2.5 GHz
Next steps

* finish to assess the benefits of the density threshold parameter
  - ready for IETF 101

* finalize document…
  - ready for IETF 101 (?)