FECFRAMEv2
Adding Sliding Encoding Window Capabilities to the FEC Framework: Problem Position

Vincent Roca, Inria, France
https://datatracker.ietf.org/doc/draft-roca-nwcrg-fecframev2-problem-position/

July 2015, IETF93, Prague
Note well

- we, authors, didn’t try to patent any of the material included in this presentation
- 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

http://irtf.org/ipr
What this I-D is about

- a follow-up of the “Forward Error Correction (FEC) Framework”, A.K.A. FECFRAME
  - RFC 6363, M. Watson, A. Begen, V. Roca, October 2011
    - produced by the FECFRAME IETF WG
    - goal of FECFRAME is to add AL-FEC protection to real-time unicast or multicast flows in a flexible way
  - also part of 3GPP MBMS standards
**FECFRAME (RFC 6363) principles**

- a shim layer to add reliability to real-time flows in a flexible way

---

**source data flow(s)**

- Application Data Unit (ADU)

---

**FECFRAME framework**

- source symbols
- repair symbols

---

**FEC source packet**

**FEC repair packet**

---

**transport protocol (e.g. UDP)**

---

**FEC scheme building block**

---

**one or several transport flows**
**FECFRAME principles... (2)**

- flexibility is the key
  - can be one or more repair flows that protect one or more source flows
    - many mappings possible
  - can work with unicast, multicast or broadcast flows
  - can be backward compatible in particular situations
  - can be deployed in end-hosts or in middle-boxes

- most of the details are in the FEC Schemes

- two constraints in RFC 6363
  - a **single** encoding and single decoding points
  - limited to **block AL-FEC codes**
    - e.g. Raptor(Q), Reed-Solomon, LDPC-Staircase, 2D XOR
Do the same with convolutional codes!

- we propose a **backward compatible** extension
  - add a sliding window encoding mode, using convolutional FEC codes
  - block codes can still be used whenever appropriate
    - e.g., with legacy receivers

**motivations**

- **with RFC 6363**, the block creation time at the source is the minimum decoding latency any receiver experiences in case of erasures 😞
  - no repair symbol for the current block can be received before
  - protection against long erasure bursts is an incentive to increase this delay

- this delay is avoided with convolutional codes **that can encode immediately** 😊
3 comments and 1 question

- no fundamental issue in updating RFC 6363
  - most changes will be in the new FEC schemes

- single/multi flows and intra/inter flows coding
  - everything is supported since the beginning (see RFC 6363)

- single versus multi-paths
  - supported since the beginning (see RFC 6363)

- should FECFRAMEv2 support in-network recoding?
  - not considered in FECFRAME use-cases and not possible with initial block codes
  - possible with convolutional codes (as with Tetrys). Is it worth?
  ⇒ To Be Decided
How does it compare to Tetrys?

• similar coding techniques
  ○ (elastic) sliding encoding window

• signaling is totally different
  ○ major differences for historical reasons
    • shim layer (FECFRAME) versus protocol instantiation (Tetrys)

• there’s no feedback at FECFRAME level…
  ○ …whereas Tetrys can use feedback in unicast or small multicast groups
    ○ e.g., to identify packets received/recovered
  ○ with large multicast groups, Tetrys does not use feedback either and both are pretty similar
Next steps...

● decide for the “in-network recording” capability

● work on RFC 6363 update
  ○ technically speaking not a big deal

● work on FEC scheme(s) in parallel
  ○ to identify potential issues…