FEC and NC performance evaluation
<update from Sept. 2018 Interim meeting presentation>

vincent.roca@inria.fr, with inputs from Morten V. Pedersen
November 5th, 2018 IETF 103, Bangkok
Previous presentations

• Already mentioned in NWCRG as an interesting topic + a few hints
  ✓ “Performance and Feature Comparison of Erasure Correcting Coding Software Libraries”, Steinwurf

• Also, for a concrete use-case:
  ✓ “FECFRAME–extension: Adding convolutional FEC codes support to the FEC Framework”, Vincent Roca et al. (slides 13-23)
Several aspects

1. Codec parameters
   - how to initialize/control the codec?

2. Performance metrics
   - what?

3. Evaluation methodology
   - how?

4. Communication channels considered
   - which use-cases?

5. Tools
   - a hackathon’s project?  yes, but codec is #1 prio
Topic 1: defining the codec parameters

• **what** are the parameters?
• **how to derive** the parameters?
  ▪ depending on the flow features (real-time or not)
Topic 1: defining the codec parameters (2)

- **we often express windows in number of symbols**
  - e.g., if we assume CBR flows (e.g., before or after FEC encoding), with symbols of fixed size, time constraints of a real-time flow are easily translated into # symbols
  - e.g., an encoding window of maximum size 20 symbols

- **otherwise we can express windows in seconds**
  - e.g., an encoding window size of 0.2s means “as many symbols that can fit in a 0.2s encoding window”

- **symbol size (E) (in bytes)**
  - assumed constant
Topic 1: defining the codec parameters (3)

- at an encoder
  - code rate (cr)
  - current and max encoding window size (ew_size/ew_max_size),
    ✓ in symbols or seconds

- with a real-time flow: max. FEC-related latency budget (max_lat)
  ✓ in seconds
  ✓ max amount of time to devote to FEC encoding and decoding
  ✓ used to derive ew_max_size along with other parameters
Topic 1: defining the codec parameters (4)

- at a decoder
  - **linear system maximum size** ($ls_{\text{max\_size}}$)
    - in symbols
    - max. number of received or lost source symbols in the linear system
  - with a **real-time flow**: **decoding window max. size** ($dw_{\text{max\_size}}$)
    - in symbols or seconds
    - maximum number of received or lost source symbols that are still within their latency budget

```
l_{\text{max\_size}}
\---------------------------------------------------^-----------------------------\n| late source symbols                            | dw_{\text{max\_size}}           \\
| (pot. decoded but not delivered)               | \---------------------------------------------------^-----------------------------\nsrc0 src1 src2 src3 src4 src5 src6 src7 src8 src9 src10 src11 src12
```
Topic 1: defining the codec parameters (4)

- a possible, non exhaustive, answer: “RLC FEC Scheme”
  

- is there another (better?) way to derive these parameters?
Topic 2: Performance metrics

• usual metrics:
  ▪ erasure correction performance
    ✓ average overhead, decoding failure probability WRT number packets received, …
  ▪ codec oriented
    ✓ encoding and decoding speed, number of finite field operations, maximum amount of memory, …

• but also:
  ▪ required code rate (i.e., amount of redundancy) to achieve a certain quality (residual packet erasures below a threshold, e.g., $10^{-3}$), for a FEC-related latency
Topic 2: Performance metrics (2)

- **time is difficult to catch…**
  - e.g., to evaluate FEC-related latency on the whole path
    - ✓ requires to be reproducible
    - ✓ easier with a CBR source flow
Topic 3: Evaluation methodology

- 3 main approaches
  - theoretical analyzes
    - important but not addressed thereafter
  - simulated end-to-end transmission
    - fully controlled and reproducible
    - accuracy needs to be checked
  - real-world experiments
    - important, but not addressed thereafter
    - maximum accuracy (if done correctly)
    - complex, partial control, less reproducible

yes, our focus
Topic 3: Evaluation methodology (2)

- example of simulated transmission system: “eperftool”
  - (cf. http://openfec.org)

- a single process for everything
- no true transmission, it’s simulated
- true FEC encoding, true FEC decoding, with a fully operational codec
- packet losses are simulated (various loss models, possibility to use true loss traces)
Topic 3: Evaluation methodology (3)

- beware of codec design choices
  - how source and repair packets are transmitted (tx order) will impact results significantly…
  - example: block FEC code, CBR output

single output leaky bucket: sending repair packets takes time and delays source packets

two output leaky buckets: repair and source packets transmitted in parallel
Topic 3: Evaluation methodology (4)

- less an issue with sliding window FEC codes
  - there’s an incentive to encode and transmit regularly
  - unlike block codes, there’s usually no big bunch of repair packets to transmit
Topic 4: Communication channels considered

- we need channel models representative of a target use-case

- we’ve been using 3GPP SA4 official mobility traces
  - which packets are lost?
  - not universal (obtained under precise circumstances) but useful
  - freely available
    (see [https://hal.inria.fr/hal-01571609v1/en/](https://hal.inria.fr/hal-01571609v1/en/))

- do we have something else?

- vehicle passenger → losses are "evenly" spread
  - 4 different average loss rates (1%, 5%, 10%, 20%)
  - each "#" indicates a loss
    - 120 km/h vehicle passenger, 20% average loss rate

- pedestrian → loss bursts
  - 4 different average loss rates (1%, 5%, 10%, 20%)
    - 3 km/h vehicle passenger, 20% average loss rate
Wrap up

• perf. evaluation is essential but non trivial
• requires to also define codec parameters and their derivation

• outcomes:
  • a new I-D?
  • a new hackathon project?

• we need to share experience/tools/channels/good practices/…
  • who’s interested in addition to myself and Morten?

stay focused: codec is #1 prio

sure!