Less latency and better protection with sliding window codes: a robust multimedia CBR broadcast case study

Vincent Roca, B. Teibi (Inria, FR)
C. Burdinat, T. Tran, C. Thienot (Expway, FR)

July 2017, IETF99, Prague

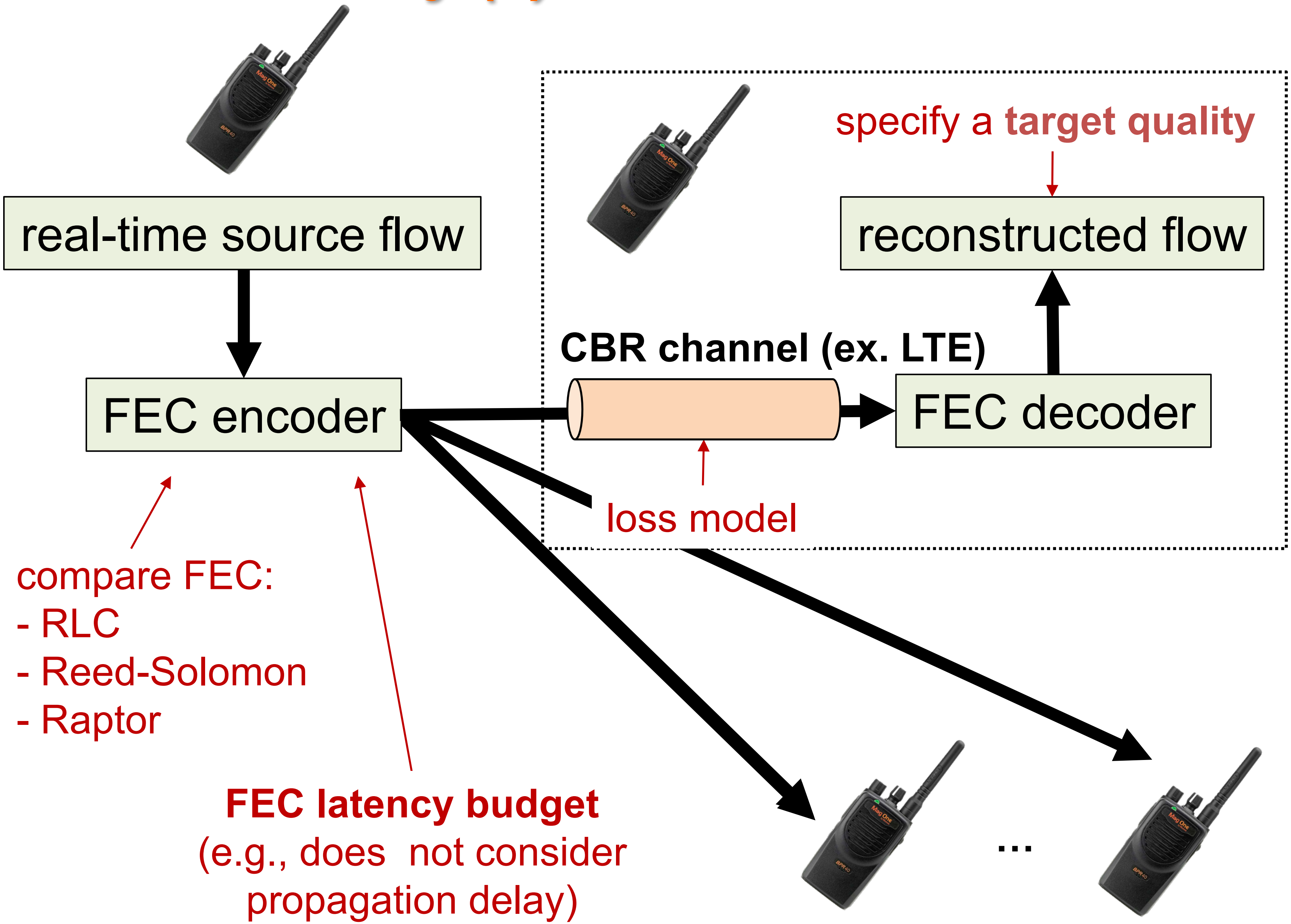
Note well

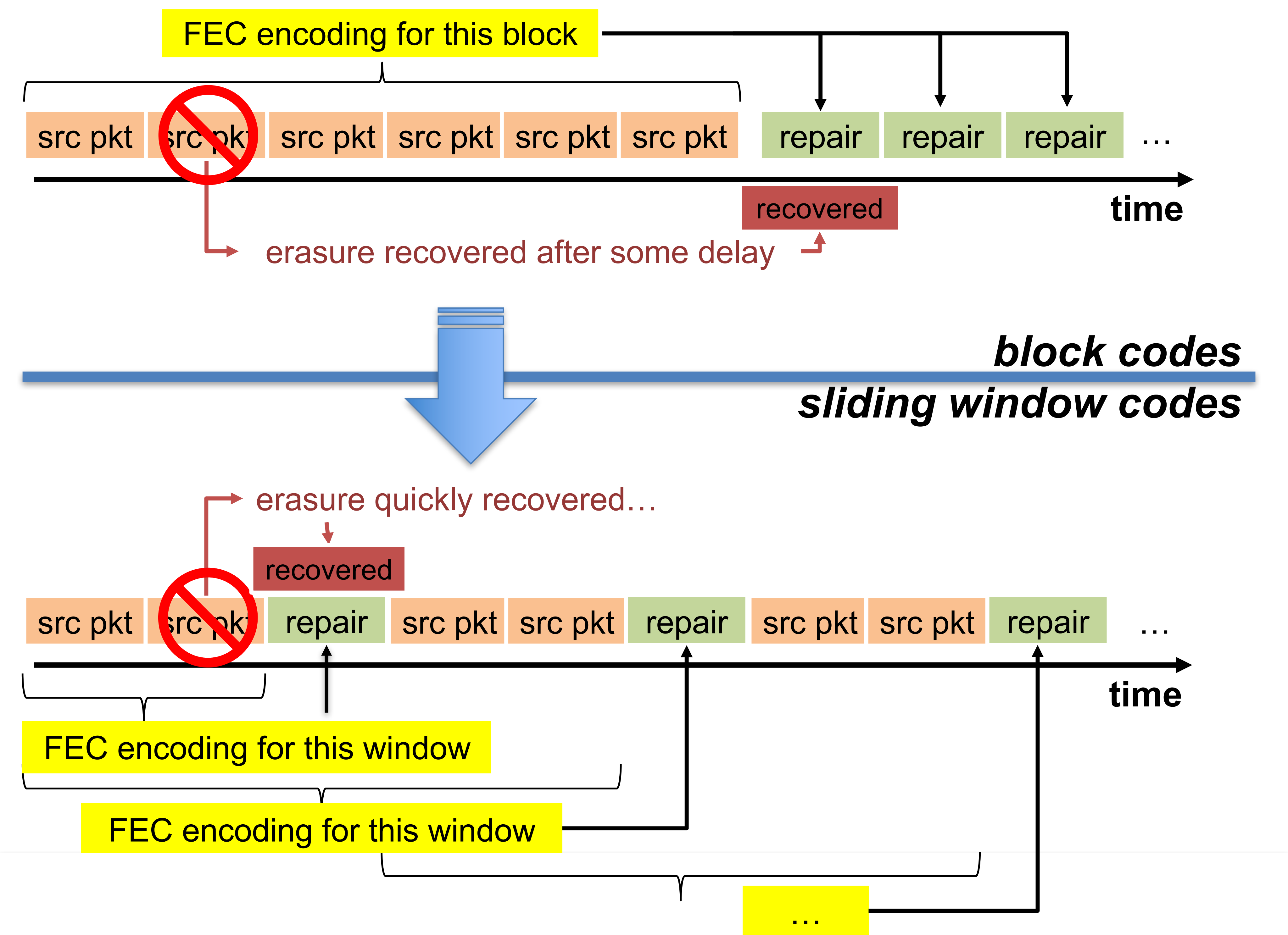
- **we, authors, didn't try to patent** any of the material included in this presentation
- **we, authors, are not reasonably aware** of patents on the subject that may be applied for by our employer
- if you believe some aspects may infringe IPR you are aware of, then fill in an IPR disclosure and please,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

Our case study

- (1) existing **3GPP Multimedia Broadcast/Multicast Service (MBMS)** and (2) future **3GPP Mission Critical Push-To-Talk (MCPTT)** standards
  - **everybody's interested by the same content at the same time at the same place**
    - audio ⇒ adhoc solution
    - files ⇒ FLUTE/ALC + block code
    - **video ⇒ ???**
  - **end-to-end latency DOES matter**

Our case study (2)




FEC encoding for this block
src pkt
src pkt
src pkt
src pkt
src pkt
src pkt
repair
repair
repair
…
recovered
time
erasure recovered after some delay
block codes
sliding window codes
erasure quickly recovered…
recovered
src pkt
src pkt
repair
src pkt
src pkt
repair
src pkt
src pkt
repair
…
time
FEC encoding for this window
FEC encoding for this window
…

The key question:

to what extent is the intuition true with more complex loss models?

Two types of benefits for sliding window

- reduced FEC related latency

  intuition:

  - repair packets are quickly produced and they quickly recover an isolated loss

- improved robustness for real-time flows

  intuition:

  - encoding windows overlap with one another which better protects against long loss bursts
  - because of reduced latency, encoding/decoding window sizes are larger than block sizes

Experimental setup

non-ideal block code (in 3GPP std)

- compare RLC vs. Reed-Solomon vs. Raptor codes

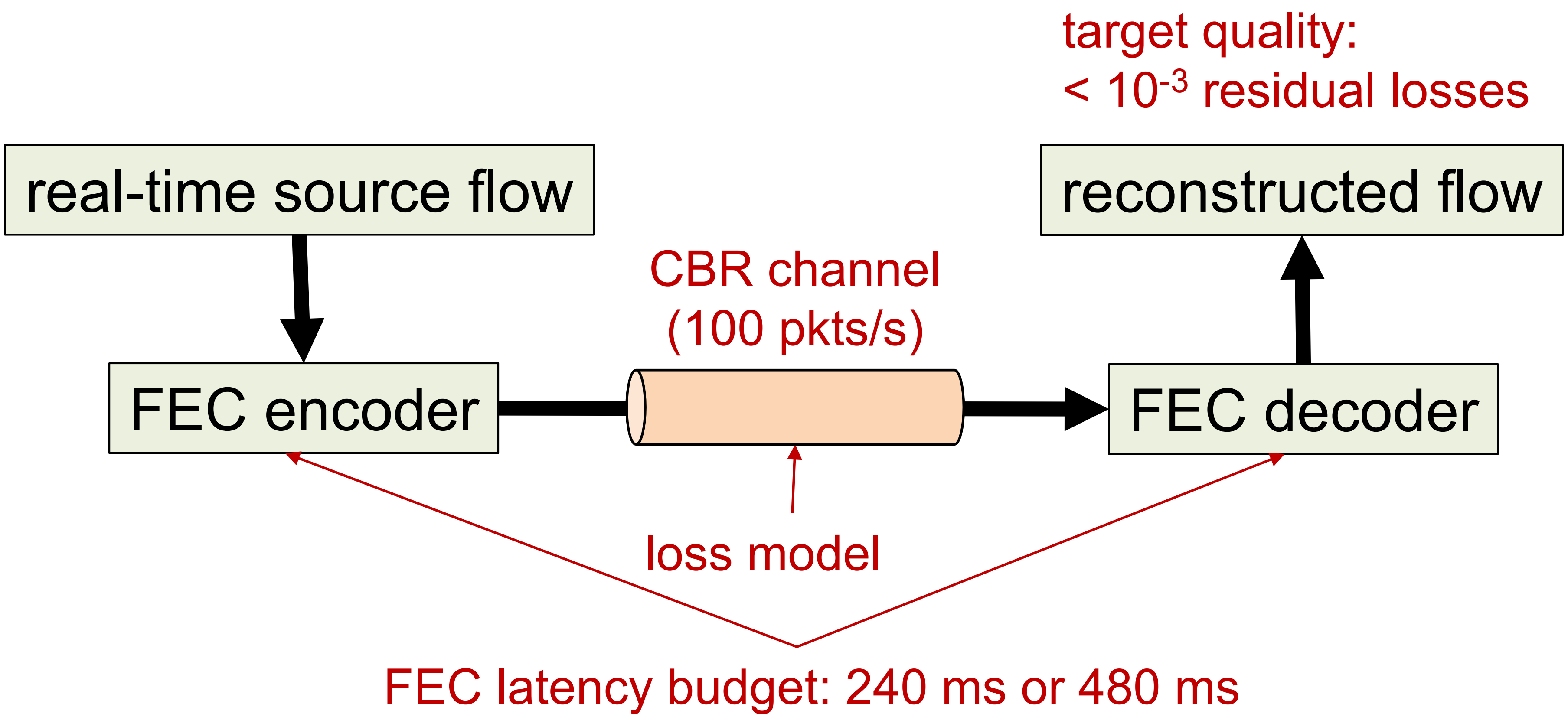
sliding window code

ideal block code (max. loss recovery performance!)

- **evaluation based on true C-language codecs, using an update of http://openfec.org**
  - only transmissions are simulated
- **assume CBR transmissions**
  - because 3GPP defines CBR channels
  - because we solely focus on FEC codes
- **use 3GPP loss scenarios representative of mobile use-cases(*)**

(*) ETSI, "Evaluation of MBMS FEC enhancements (final report)," Dec. 2015, 3GPP TR 26.947 version 13.0.0 Rel. 13

Experimental setup… (2)



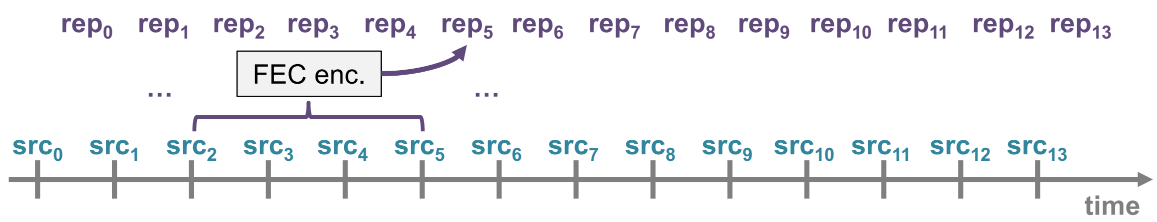
How much repair traffic to achieve the target quality?

In turn this parameter determines:

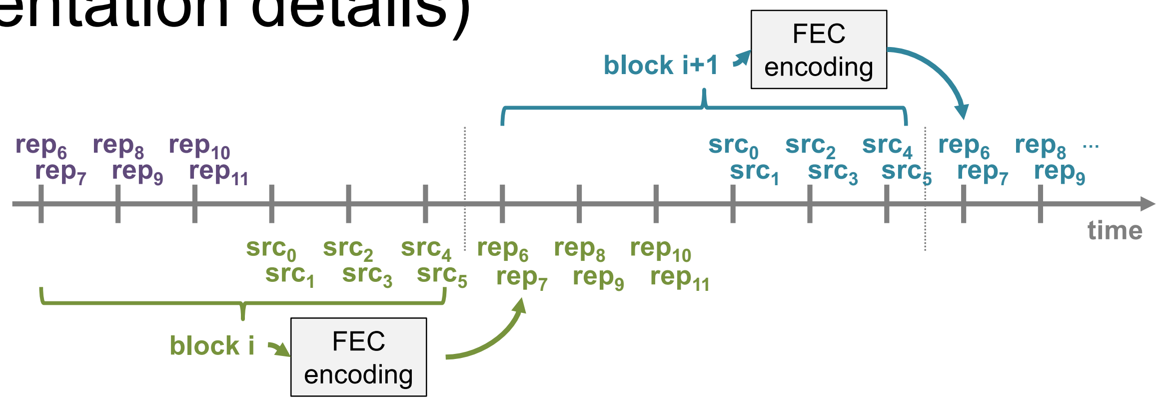
- block or en/decoding window sizes
- maximum source flow bitrate

Experimental setup… (3)

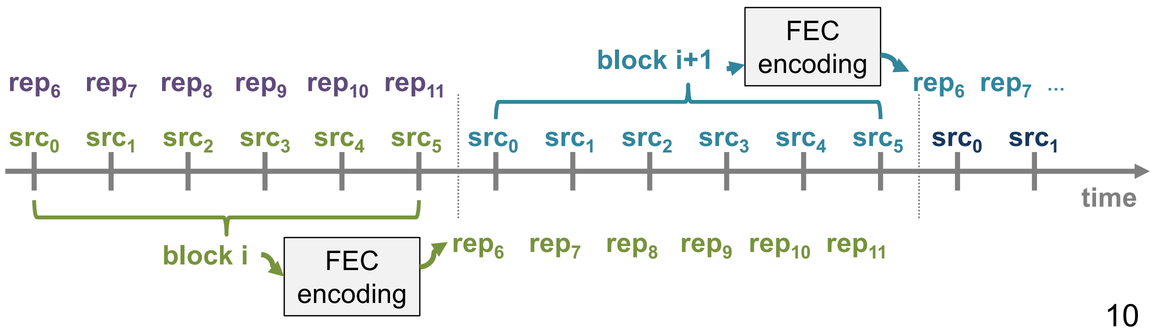
- take CBR packet scheduling into account
  - RLC



  - two possibilities with **Reed-Solomon** and **Raptor** (depends on implementation details)
    1. **block-BEGINNING**



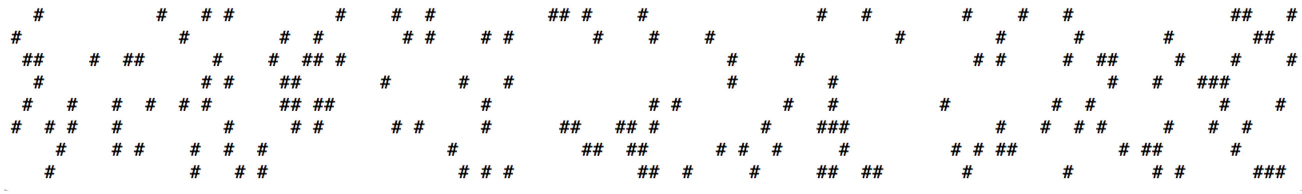
    2. **block-DURING**



Experimental setup… (4)

- take 3GPP mobility scenarios into account(*)
  - vehicle passenger ⇒ losses are "evenly" spread

    4 different average loss rates (1%, 5%, 10%, 20%)

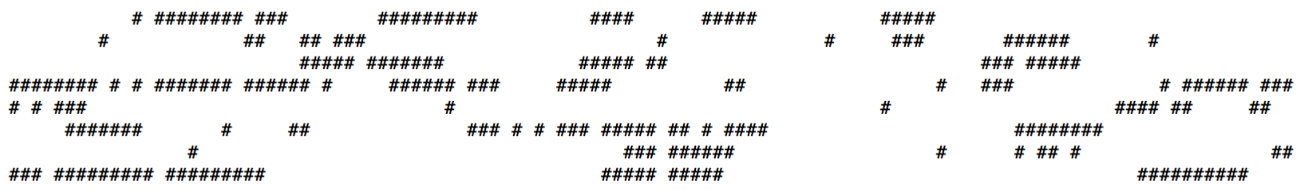


each "#" indicates a loss

120 km/h vehicle passenger, 20% average loss rate

  - pedestrian ⇒ loss bursts

    4 different average loss rates (1%, 5%, 10%, 20%)



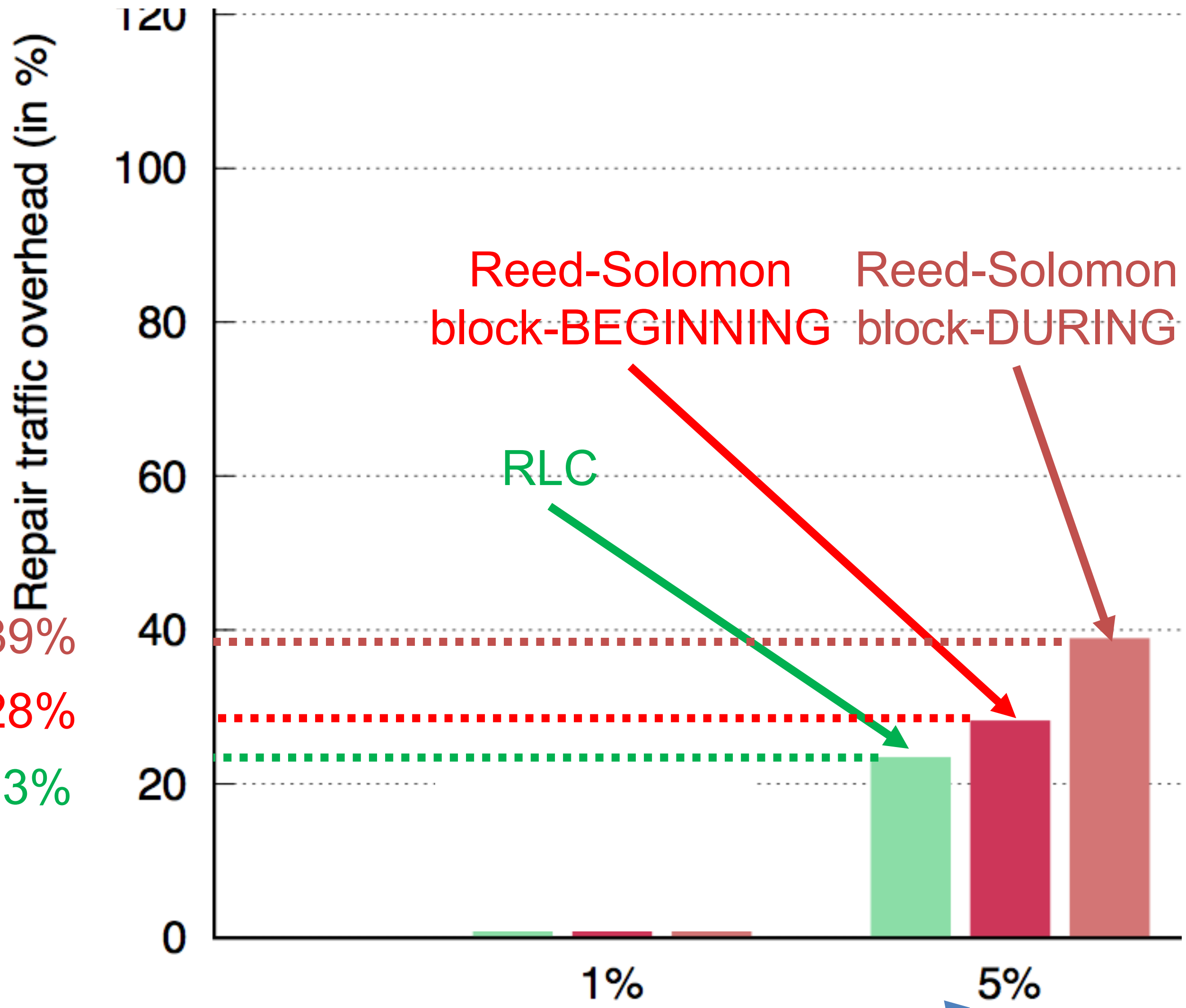
3 km/h vehicle passenger, 20% average loss rate

(*) ETSI, "Evaluation of MBMS FEC enhancements (final report)," Dec. 2015, 3GPP TR 26.947 version 13.0.0 Rel. 13

Understanding the following figures

for given **loss model** and **latency budget**, what protection do we need to achieve a 10^{-3} residual loss quality

required repair traffic overhead (100% means that repair traffic has same bitrate as source traffic)



RS-DURING: 39%

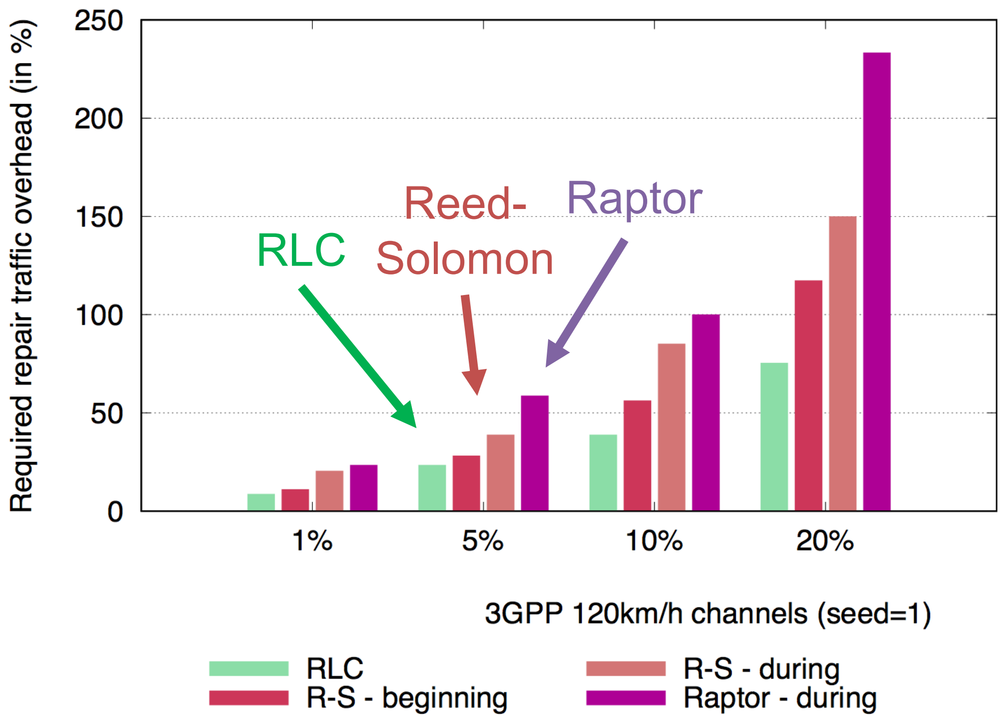
RS-BEGINNING: 28%

RLC: 23%

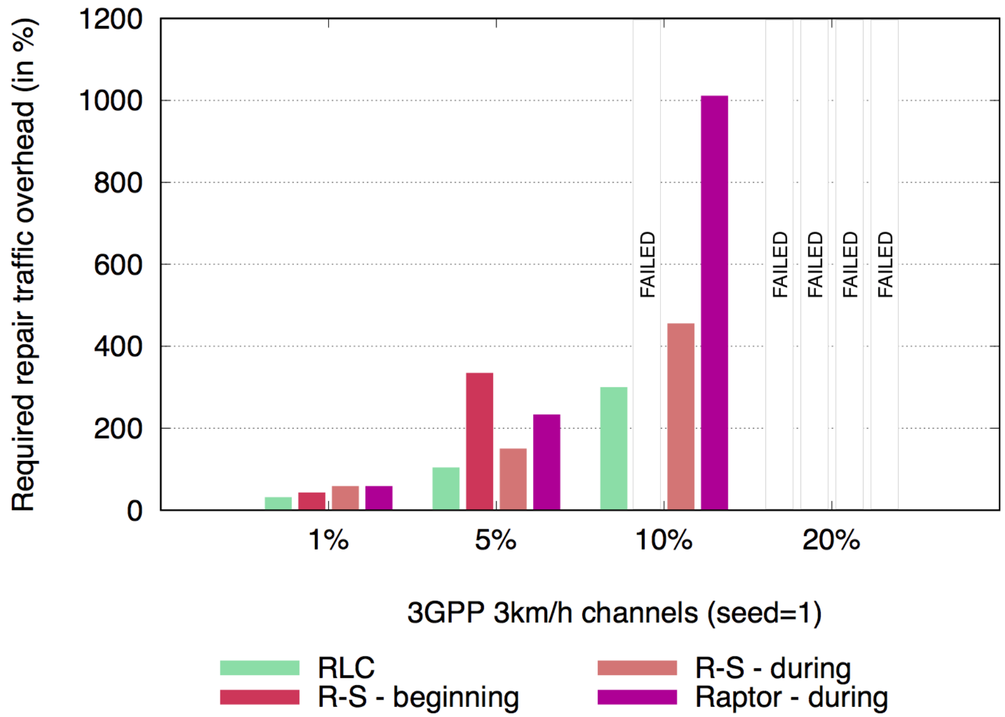
average loss rate for the channel

Results: min. FEC protection required…

240 ms latency budget for FEC



(a) 240 ms budget, 120 km/h channel

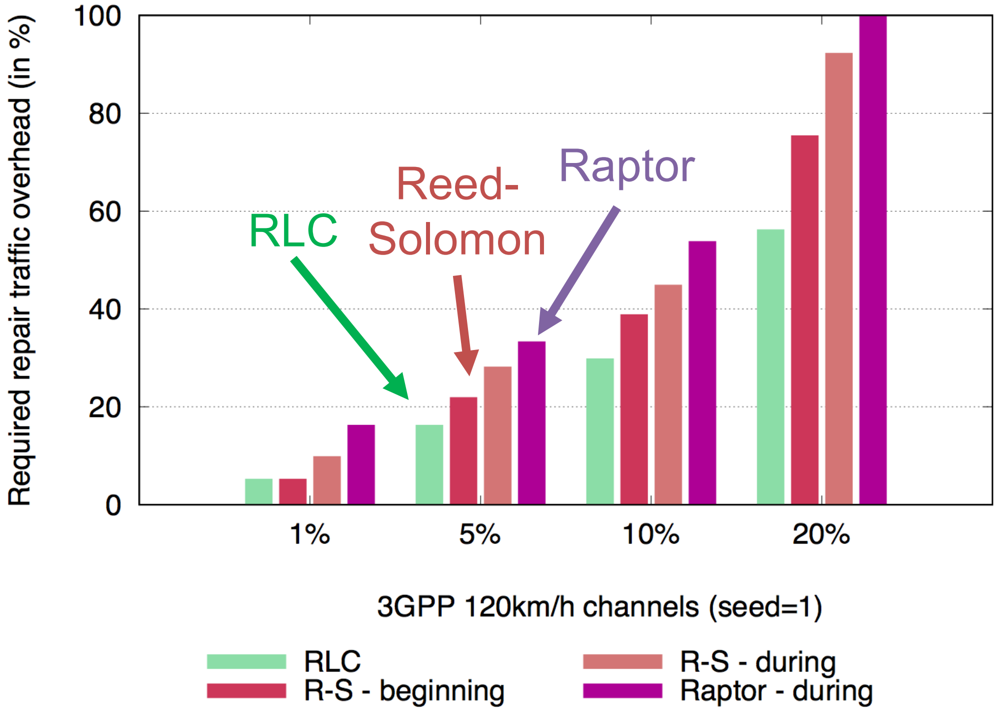


(b) 240 ms budget, 3 km/h channel

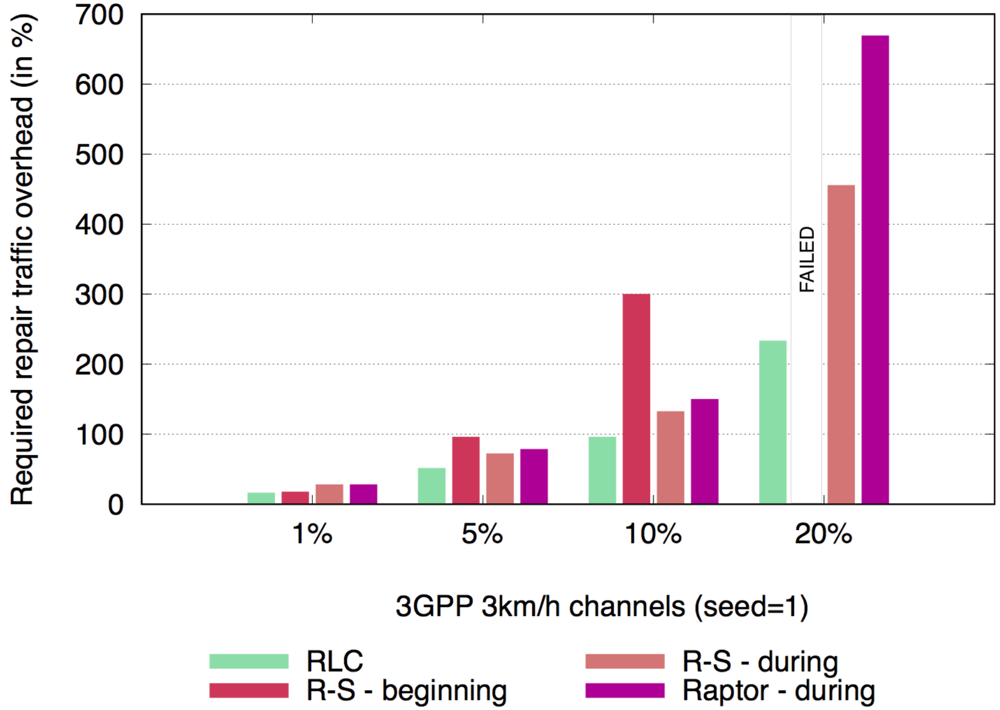
RLC is **always significantly better**, achieving the desired target quality with significantly less repair traffic!

Results: min. FEC protection required…

480 ms latency budget for FEC ⇒ longer block/sliding window sizes



(c) 480 ms budget, 120 km/h channel



(d) 480 ms budget, 3 km/h channel

With a double "latency budget", RLC remains **significantly better**

Hey, we have a single output flow for all receivers!

- we're dealing with multicast/broadcast, so…
  - ❍ many receivers with different channels

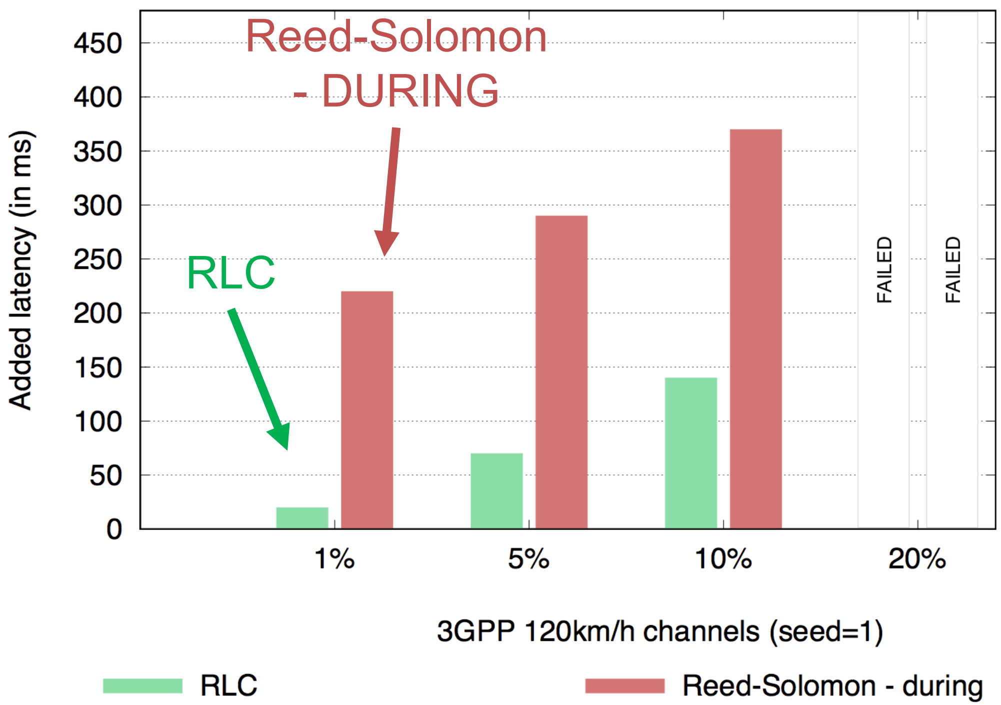
⇒ **decide the worst channel you want to support and/or the maximum repair traffic overhead we can "tolerate"**

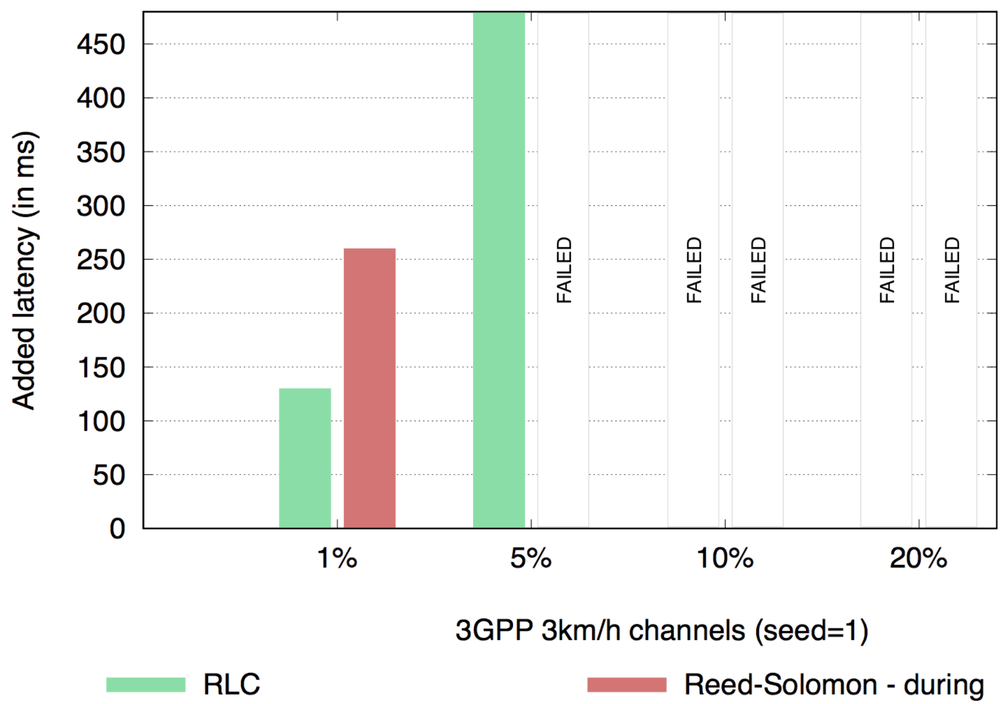
  - ❍ **the (single) multicast data flow will use this code rate**
  - ❍ **measure the experienced latency sufficient for a 10^{-3} residual loss rate for each supported channel**
  - ❍ **compare…**

And in terms of latency…

480 ms latency budget for FEC, and **fixed 50% repair traffic** (code rate=2/3)

NB: R-S Beginning and Raptor codes not considered here (poor perf.)



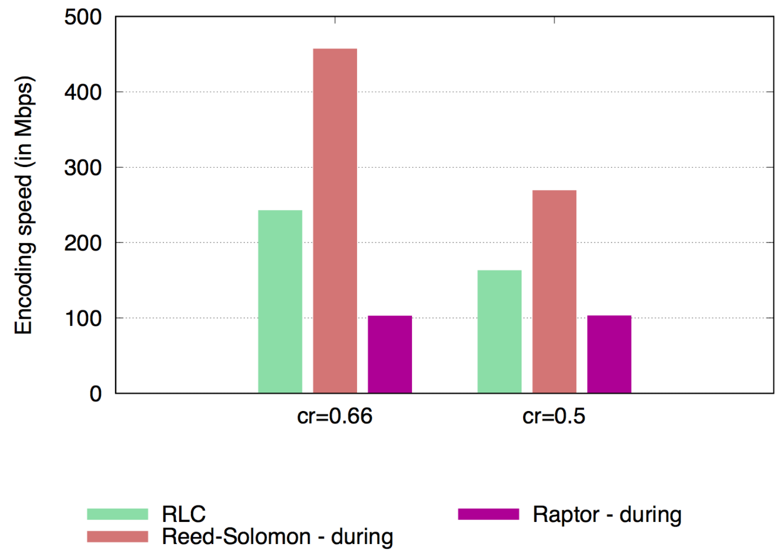


more channels are supported by RLC, and **the added latency to good receivers is far below the maximum 480 ms latency budget**

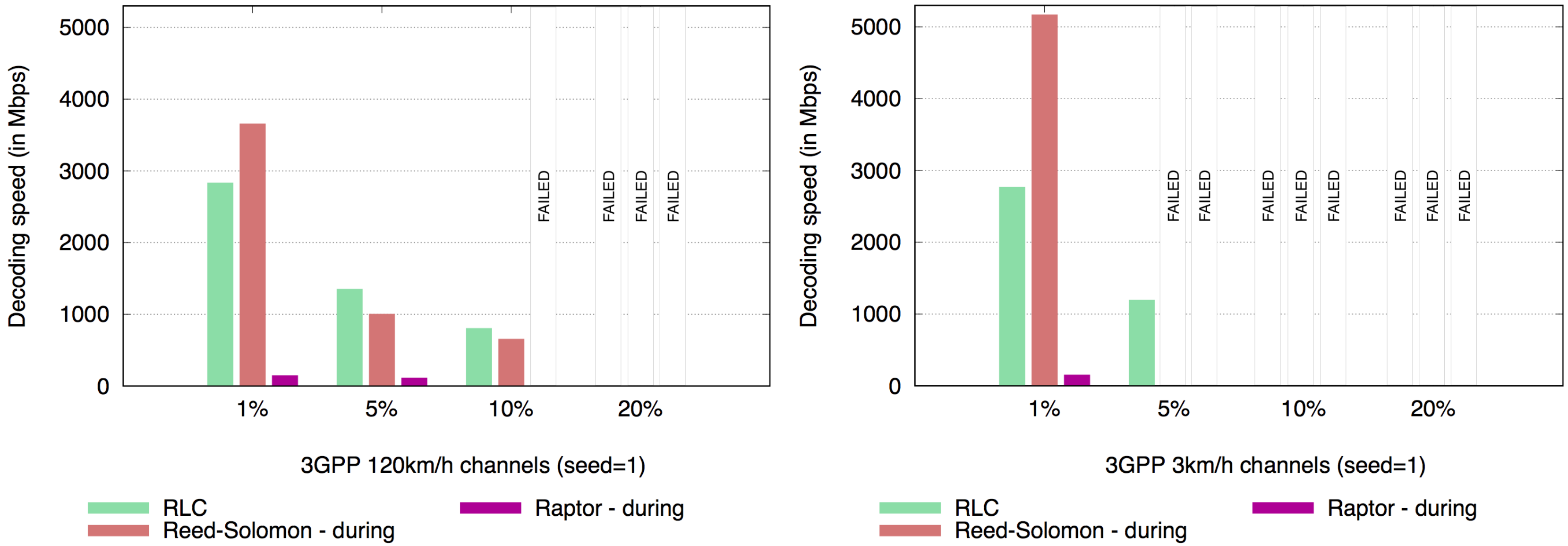
How fast is it?

- sufficiently with RLC (ARM Cortex-A15@1.5GHz, 480ms latency budget)

Encoding speed



Decoding speed (CR=0.66)



(a) cr=0.66, 120 km/h channel

(b) cr=0.66, 3 km/h channel

Conclusions

- sliding window codes really make a difference…
  - …when trying to minimize FEC related latency
  - significant robustness improvement (due to larger windows that overlap)
  - less latency to achieve a certain target quality
  - extremely fast (we're dealing with very small window sizes)
- we focused on broadcast/multicast communications
  - … but make sense with unicast communications as well

Conclusions (2)

- Related IETF activity:
  - "Forward Error Correction (FEC) Framework Extension to Sliding Window Codes"
    - draft-ietf-tsvwg-fecframe-ext-00
  - "Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Scheme for FECFRAME"
    - draft-ietf-tsvwg-rlc-fec-scheme-00
- A question? vincent.roca@inria.fr