



HAL
open science

Not so random RLC AL-FEC codes

Vincent Roca, Kazuhisa Matsuzono

► **To cite this version:**

Vincent Roca, Kazuhisa Matsuzono. Not so random RLC AL-FEC codes. IETF88 - NWCRG meeting, Nov 2013, Vancouver, Canada. 2013. hal-00879834

HAL Id: hal-00879834

<https://inria.hal.science/hal-00879834>

Submitted on 12 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Not so random RLC AL-FEC codes

Kazuhisa Matsuzono, **Vincent Roca**
Inria, France

IETF88, NWCRG meeting
Nov. 7th, 2013, Vancouver

Note well

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

Motivations and goals

Motivations

- RLC are naturally random
 - encoding vectors on a given Finite Field (FF) are random
 - it's easy, efficient, and enables coding *inside the net*
- but there are incentives to have “structured” codes
 - sparse codes are **faster** to encode/decode
 - an order of magnitude difference, because:
 - fewer XOR and/or FF symbol operations
 - fast Iterative (IT) decoding works better
 - certain **structures** are extremely **efficient**
 - e.g., LDPC-Staircase [RFC5170] [WiMob13]
 - e.g., irregular LDPC codes perform the best with IT decoding

[WiMob13] V. Roca, M. Cunche, C. Thienot, J. Detchart, J. Lacan, “**RS + LDPC-Staircase Codes for the Erasure Channel: Standards, Usage and Performance**”, IEEE 9th Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob), October 2013.

<http://hal.inria.fr/hal-00850118/en/>

Goals of this work

- design codes that:
 - can be used as **sliding/elastic encoding window** (A.K.A. convolutional) and **block** codes
 - there are use-cases for each approach
 - can be used with encoding window/block sizes in **2-10,000s symbols** range
 - very large sizes are beneficial to bulk file transfers while small values are useful for real-time contents
 - can be used as **small-rate** codes
 - can generate a large number of repair symbols
 - even if it's rarely useful (e.g. it was not a selection criteria for 3GPP-MBMS [WiMob13]), it also simplifies performance evaluations 😊

Goals of this work... (cont')

- have excellent erasure recovery **performance**
 - often a **complexity versus performance tradeoff**
 - it's good to be able to adjust it on a per use-case basis
- enable **fast** encoding and decoding
 - sender and/or receiver can be an embedded device
- enable **compact and robust signaling**
 - transmitting the full encoding vector does not scale
 - prefer the use of a function that, as a function of a key lists the symbols that are considered
 - can be a PRNG + seed
 - can be a table + index
 - the function is known to both ends and the (e.g., 32-bit) key is carried in the packet header

Goals of this work... (cont')

- focus **only** on use-cases that require **end-to-end encoding**
 - “end” means either “host” or “middlebox”, it’s the same
 - there’s a single point for **AL-FEC encoding/decoding**
 - because it simplifies signaling and code design
 - **intermediate node re-encoding requires having the symbols encoding vectors which does not scale**
 - sure, it’s a subset of NWCRG candidate use-cases
 - e.g., it’s well suited to Tetrys
 - <http://www.ietf.org/proceedings/86/slides/slides-86-nwcrg-1.pdf>
 - but also to **FLUTE/ALC, FCAST/ALC, FCAST/NORM, FECFRAME** protocols

Our proposal

Experimental results of this presentation...

- ...use our <http://openfec.org> open-source project
 - Uses a mixture of [CeCILL\(-C\)](#) (GPL and LGPL like), “BSD like” licenses

OpenFEC.org

because open, free AL-FEC codes and codecs matter

- for the moment we've integrated Kodo RLC lib...
 - ...but we'll get rid of it ASAP
 - because STEINWURF research license is not compatible with our goal of free, reusable software in any context, commercial or not
- all measurements are made in **block mode**
 - because it's the way our <http://openfec.org> tools work...
 - ... but we'll update it

Idea 1: mix binary and non-binary

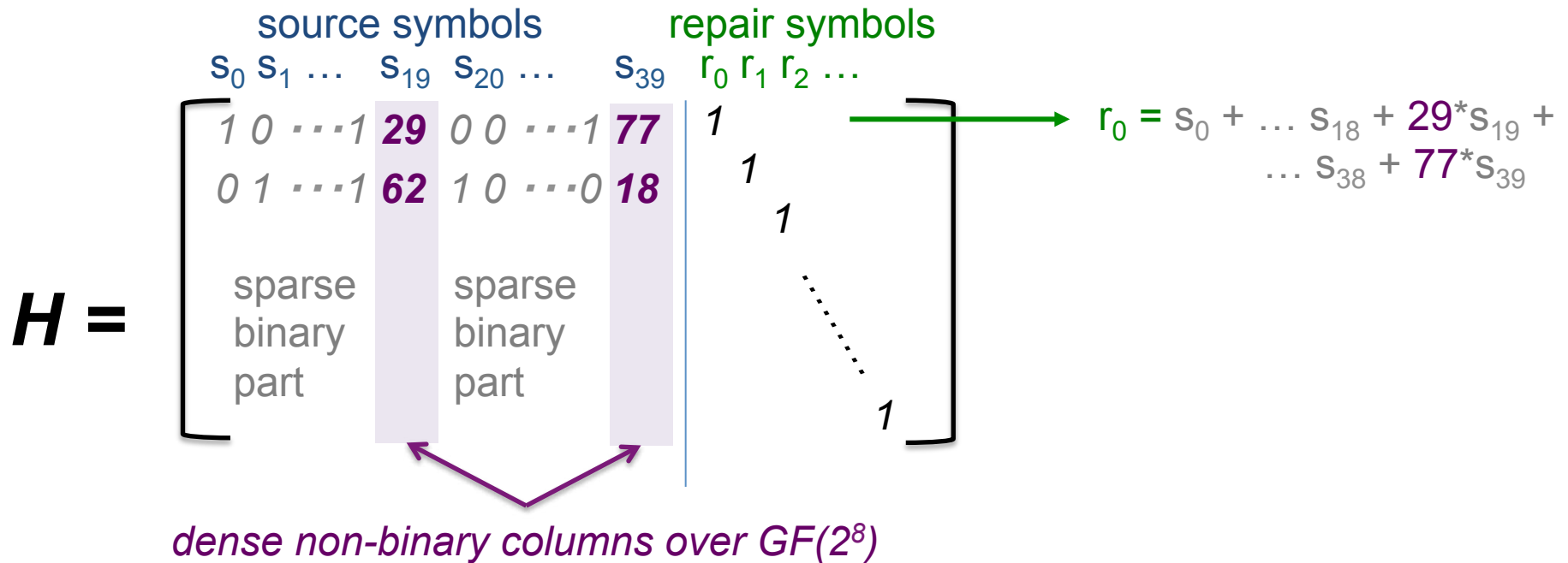
- mix binary and non binary
 - most equations are **sparse** and coefficients **binary**
 - a limited number of columns are **heavy**
 - dense binary columns ← not considered in the remaining
 - dense non-binary columns (e.g., with coeff. on $\text{GF}(2^8)$)
- there are good reasons for that:
 - sparseness is a key for high encoding/decoding speeds
 - density/non binary are good for recovery performances
 - gathering dense coefficients in columns (i.e. to certain symbols) is a key for high speed decoding [WiMob13]

[WiMob13] V. Roca, M. Cunche, C. Thienot, J. Detchart, J. Lacan, “**RS + LDPC-Staircase Codes for the Erasure Channel: Standards, Usage and Performance**”, IEEE 9th Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob), October 2013.

<http://hal.inria.fr/hal-00850118/en/>

Idea 1: mix bin and non-bin... (cont')

- block code example
 - (sparse + non-bin. columns) only



Idea 1: mix bin and non-bin... (cont')

- convolutional code example

- (sparse + non-bin. columns) only

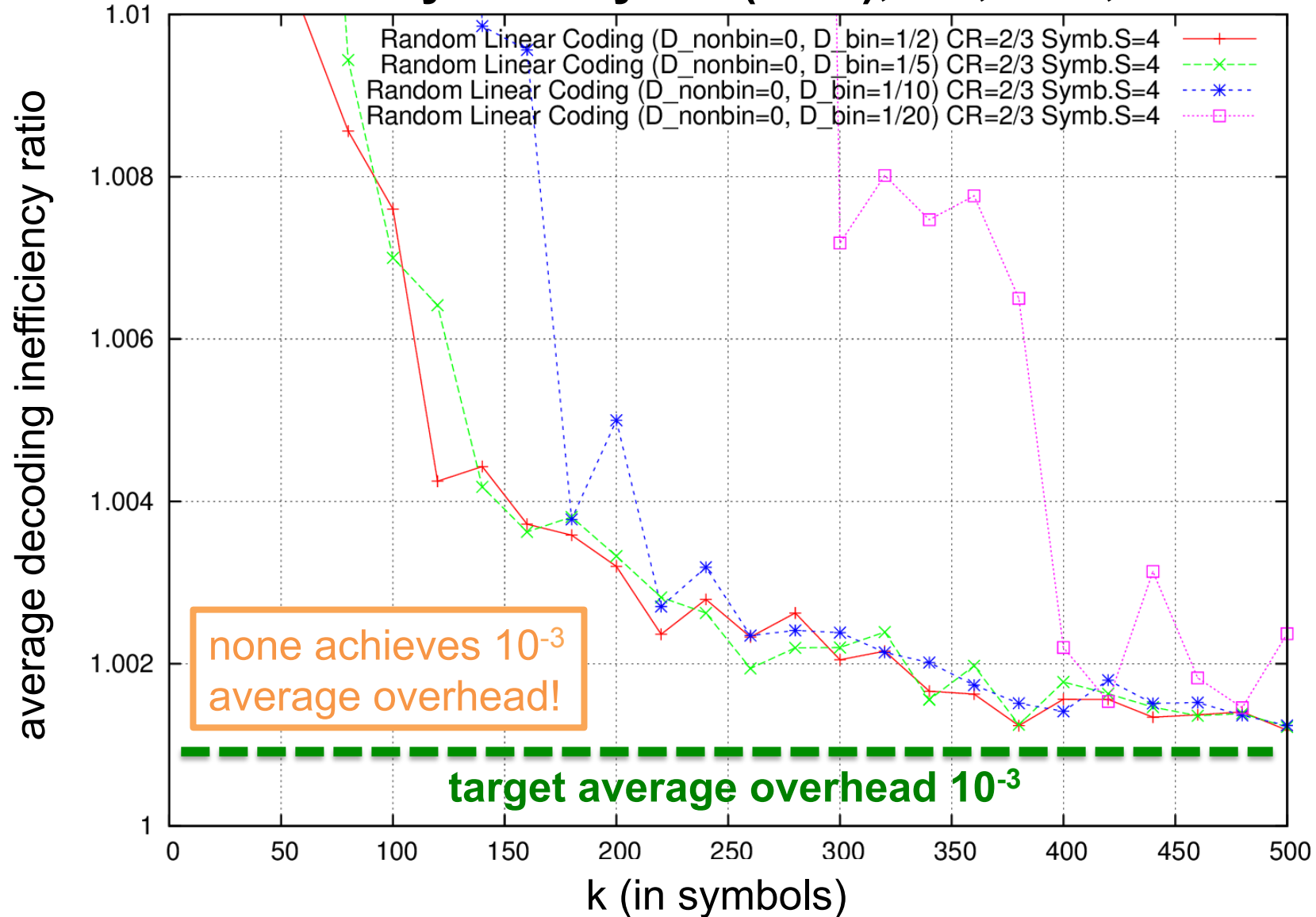
```
/* r points to repair symbol to build */
memset(r, 0, pkt_sz);
for (each source symbol s in current encoding window)
    if (s identifier % (1/D_nonbin) == 0)
        /* non binary column */
        choose a non-bin coefficient, c;
        r ^= c * s;
    else
        /* binary part */
        do r ^= s with probability D_bin
```

- NB:

- it's the same except that the encoding windows moves over the source symbol flow (convolutional mode) instead of being fixed (block mode)

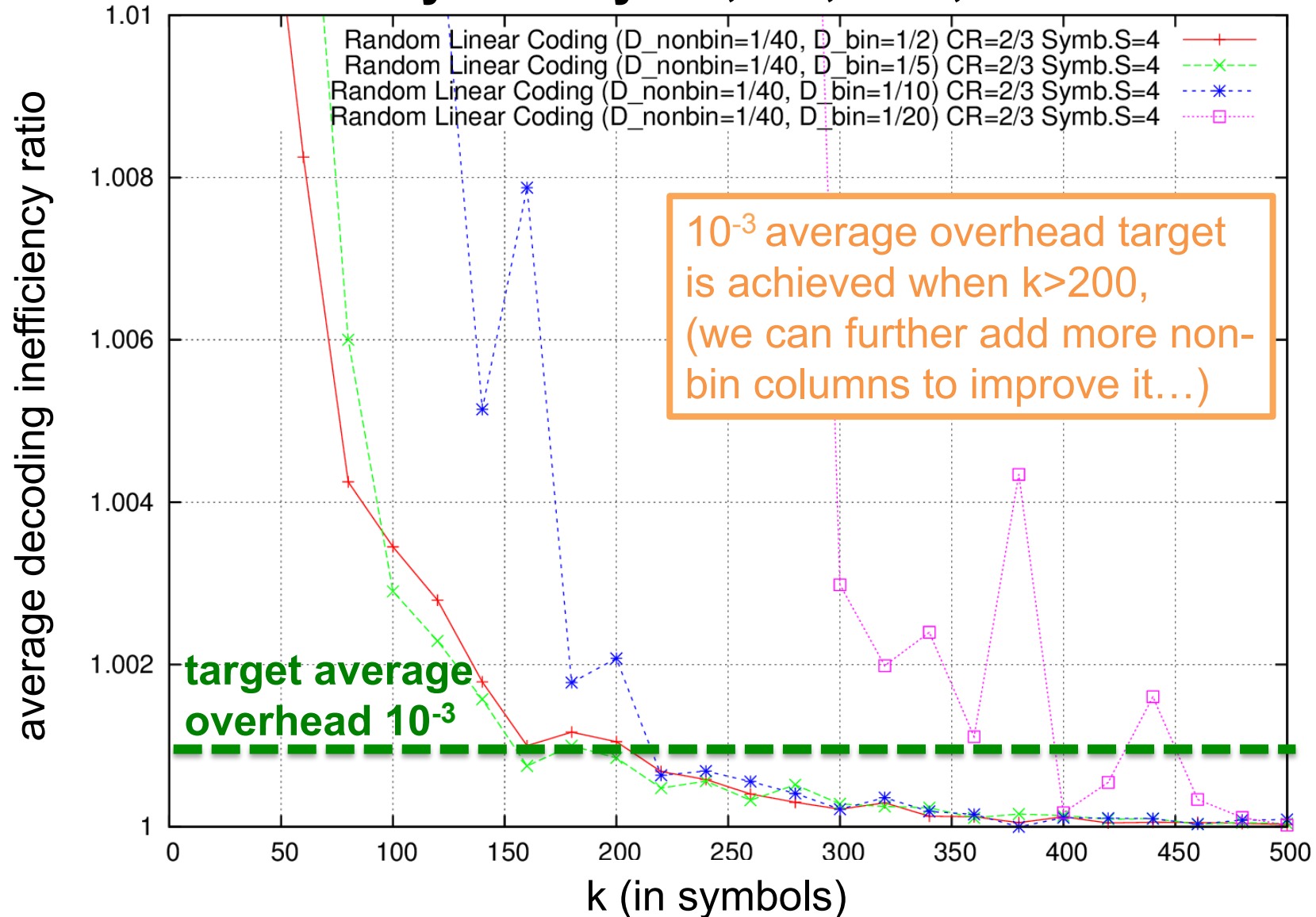
On the usefulness of non-bin columns

Test 1: **no** dense non-binary column
binary density 1/2 (RLC), 1/5, 1/10, 1/20



On the usefulness of non-bin cols... (cont')

Test 2: **with** dense non-binary column (1 every 40 cols)
binary density 1/2, 1/5, 1/10, 1/20



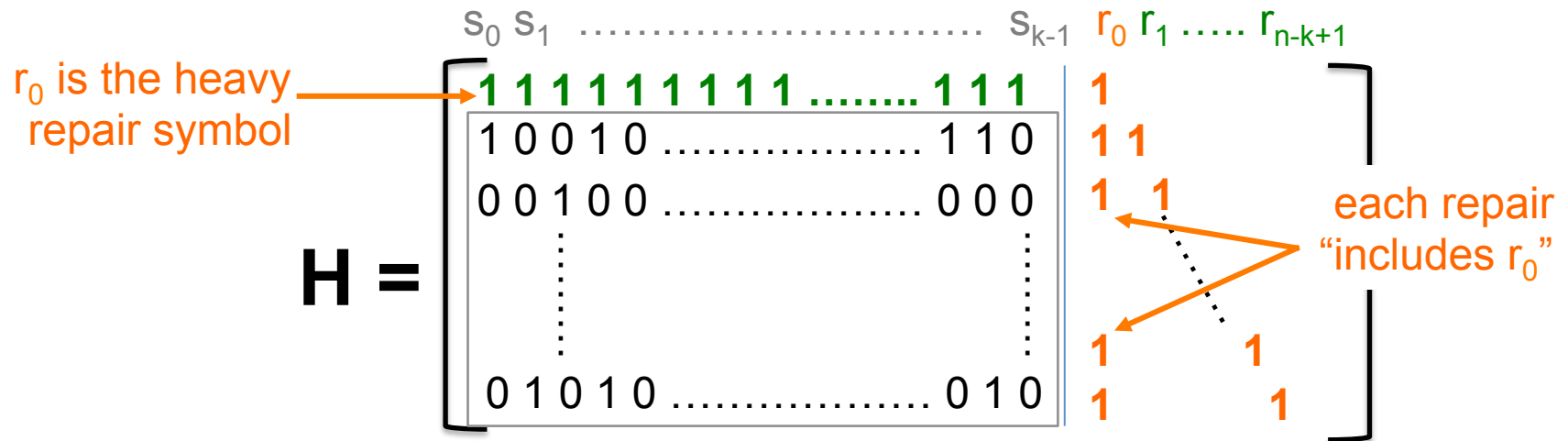
Idea 2: add a structure... (cont')

○ so we add a **single heavy row** and make **all repair symbols depend on it**

○ it's now quite simple, even when used in convolutional mode

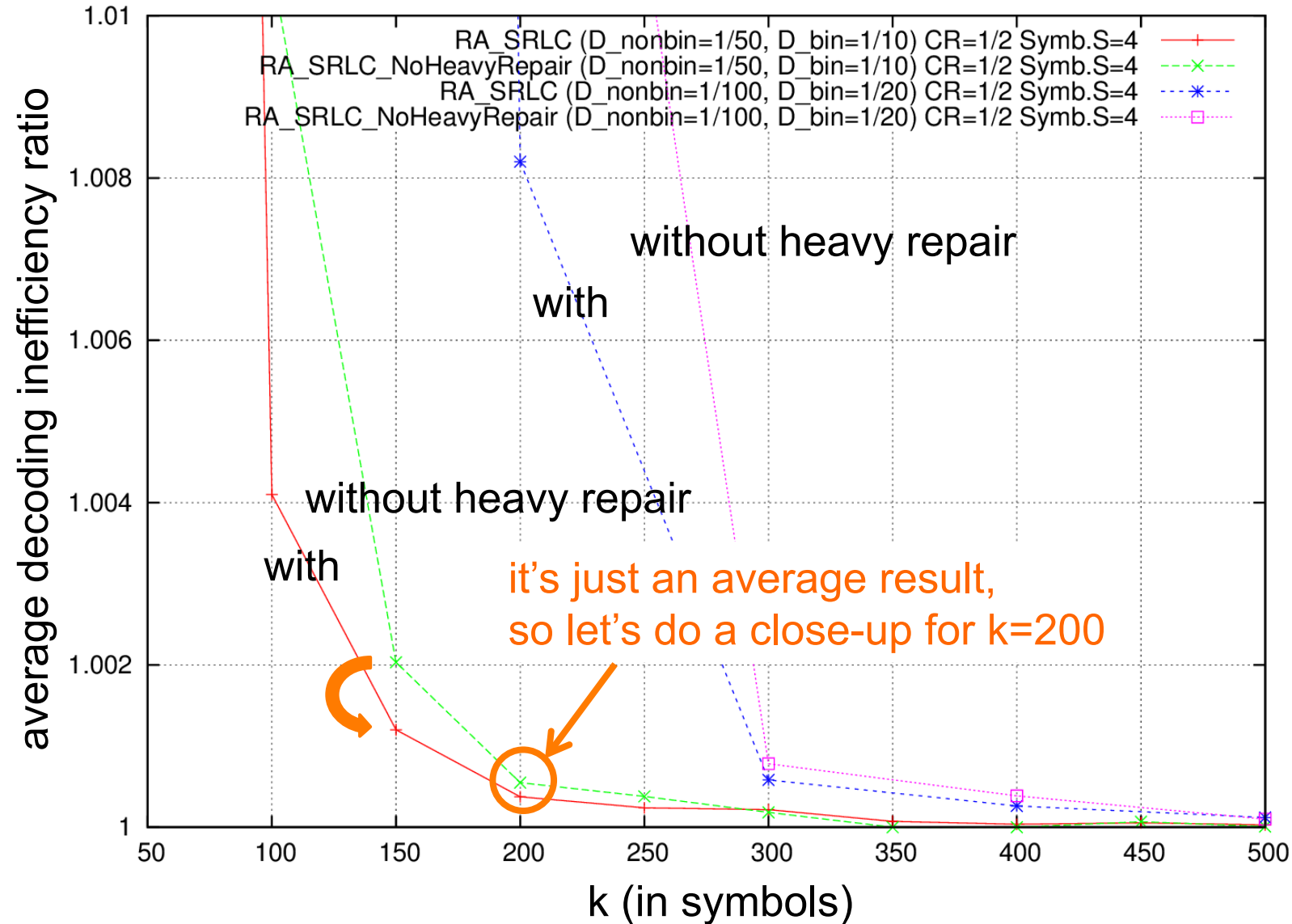
- several sums will be transmitted (e.g., periodically), and it is sufficient to identify the last symbol of the sum in the signaling header

○ it's efficient (see later), at the price of extra XOR operations



○ NB: other ways to define heavy rows are feasible (e.g., with random coefficients over $GF(2^8)$...)

On the usefulness of heavy repair symbols



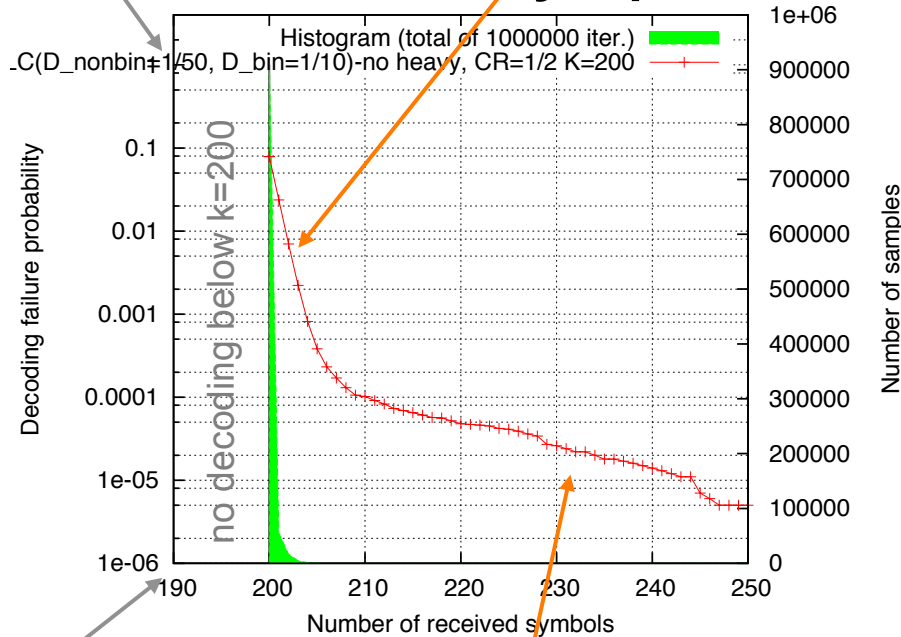
On the usefulness of heavy repair... (cont')

- decoding failure probability = $f(\# \text{ symbols received})$
 - enables in-depth analysis, catching rare events

similar behaviors for very low overheads

1 \Rightarrow no decoding

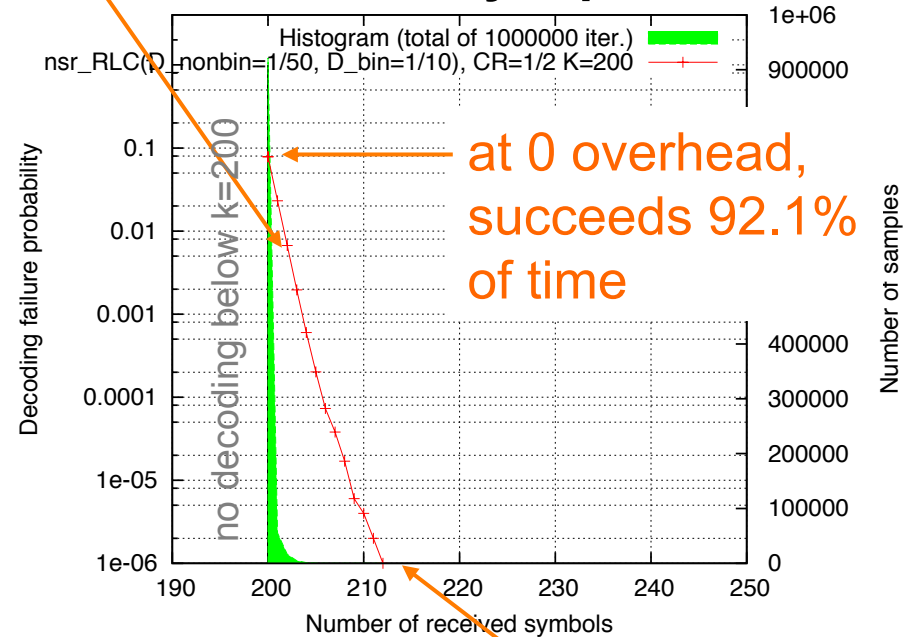
without heavy repair



small failure prob.

but significant number of tests fail for higher overheads ☹️

with heavy repair

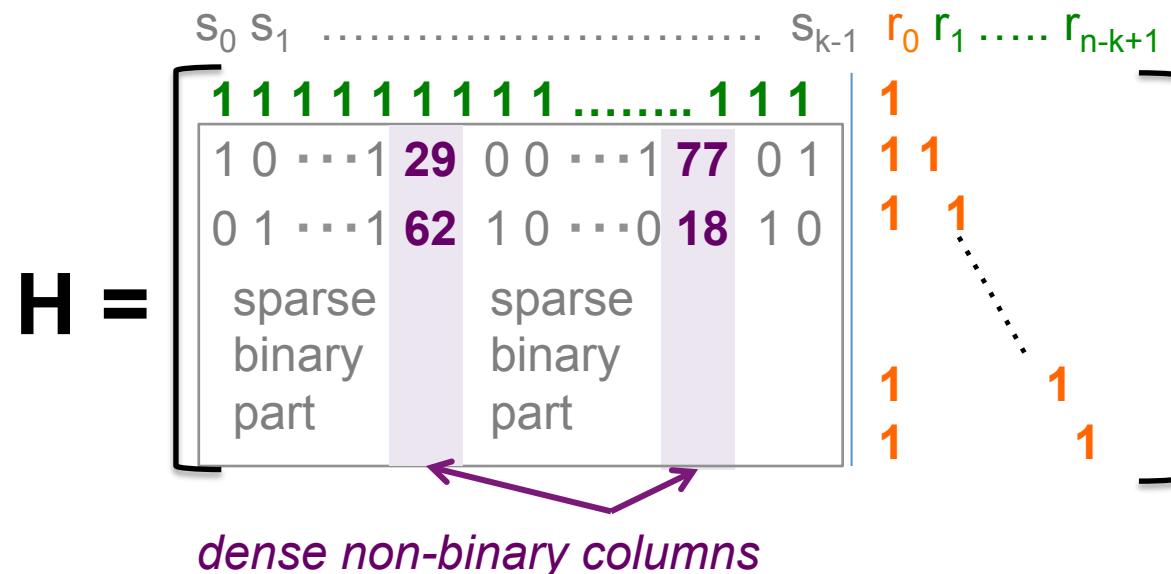


with a heavy repair symbol, there's no such behavior 😊

Let's put ideas 1 and 2 together

- 3 key parameters

- k : source block or current encoding window size
- D_{nonbin} : controls number of heavy non-binary columns
 - $D_{\text{nonbin}} = \text{nb_non-binary_coeffs} / k$
- D_{bin} : controls the density of the sparse sub-matrices
 - $D_{\text{bin}} = \text{nb_1_coeffs} / \text{total_nb_coeffs_in_binary_submatrix}$
- $\{D_{\text{nonbin}}, D_{\text{bin}}\}$ depend on k and target max. overhead



Finding the right (D_{nonbin} , D_{bin}) values

- set a target average overhead (e.g., 10^{-3})
- then:

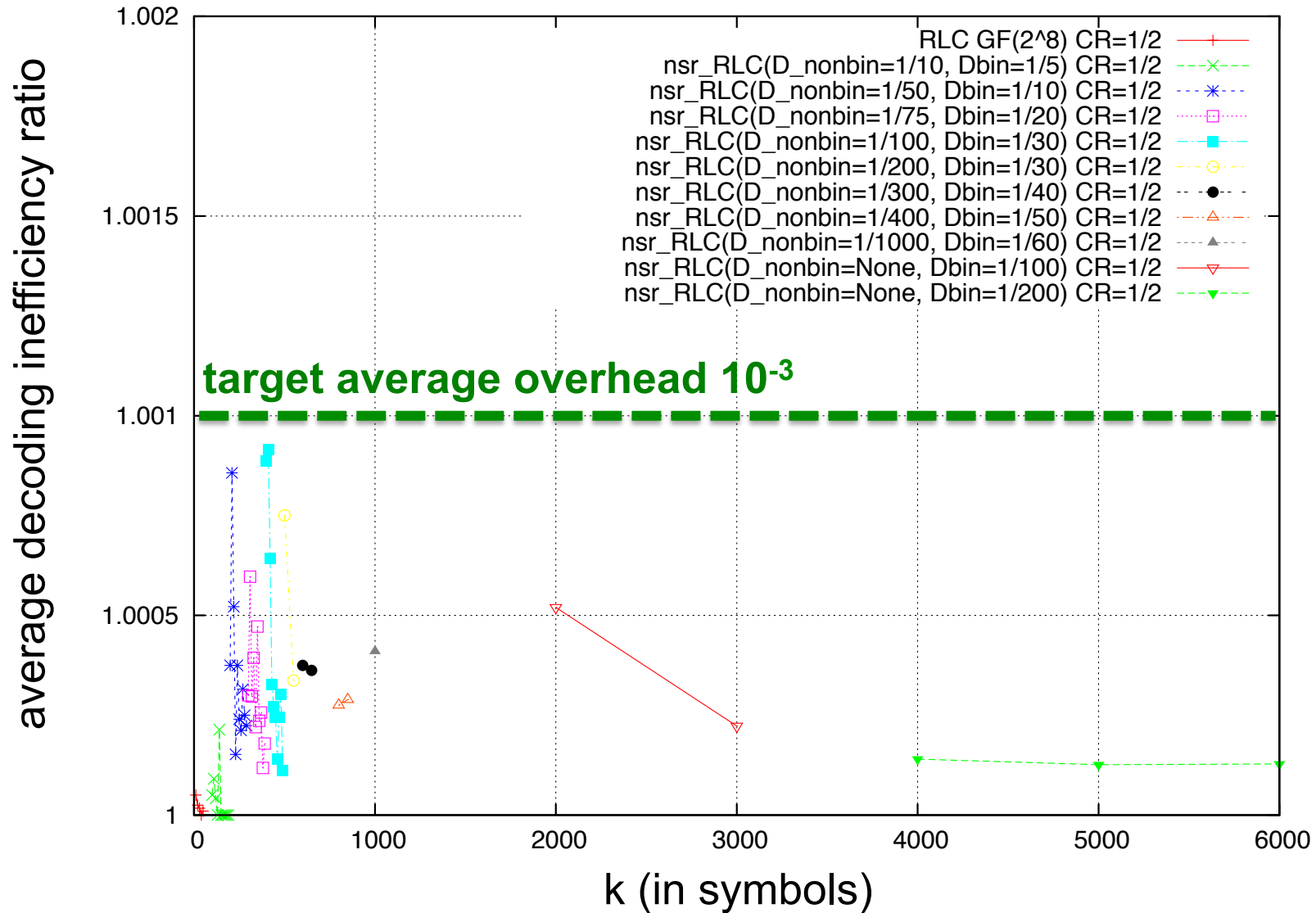
```
for (k in [2, 10 000])  
  carry out experiments with fixed  $D_{\text{bin}}=1/2$ ,  
  increasing  $D_{\text{nonbin}}$  until we achieve an average  
  overhead below  $\alpha \cdot 10^{-3}$ , where  $\alpha < 1$  is a  
  "security margin";  
  for this  $D_{\text{nonbin}}$ , carry out experiments by  
  reducing  $D_{\text{bin}}$  as much as possible while  
  remaining below target overhead  $10^{-3}$ 
```

- store all results in a table

- basically:

- Only non-bin columns for very small k
- Only bin columns for very high k
- A mixture of both in between...

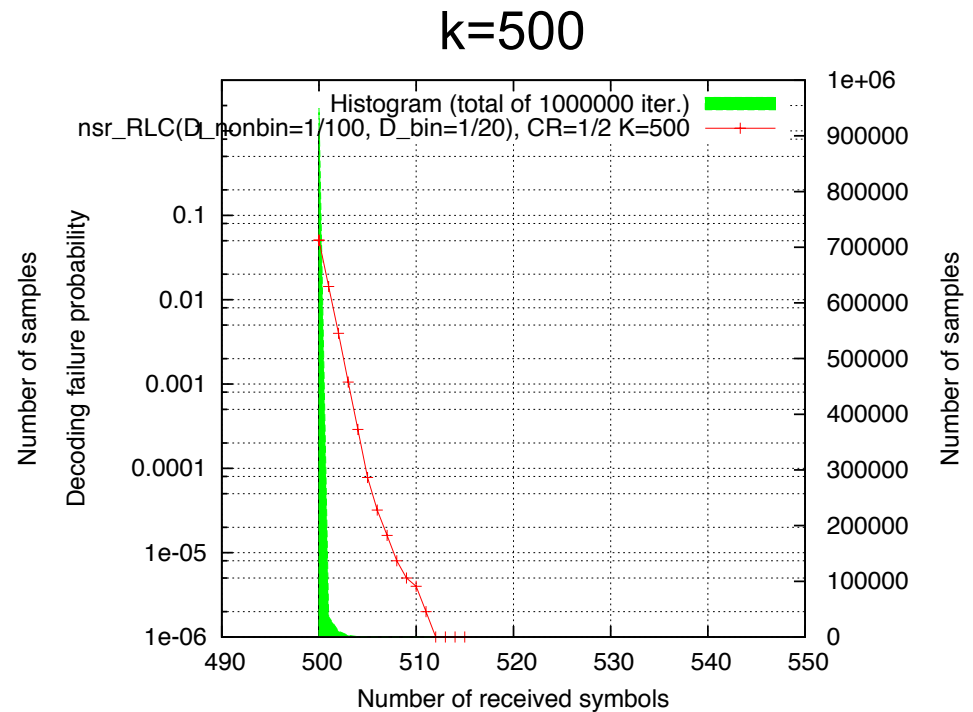
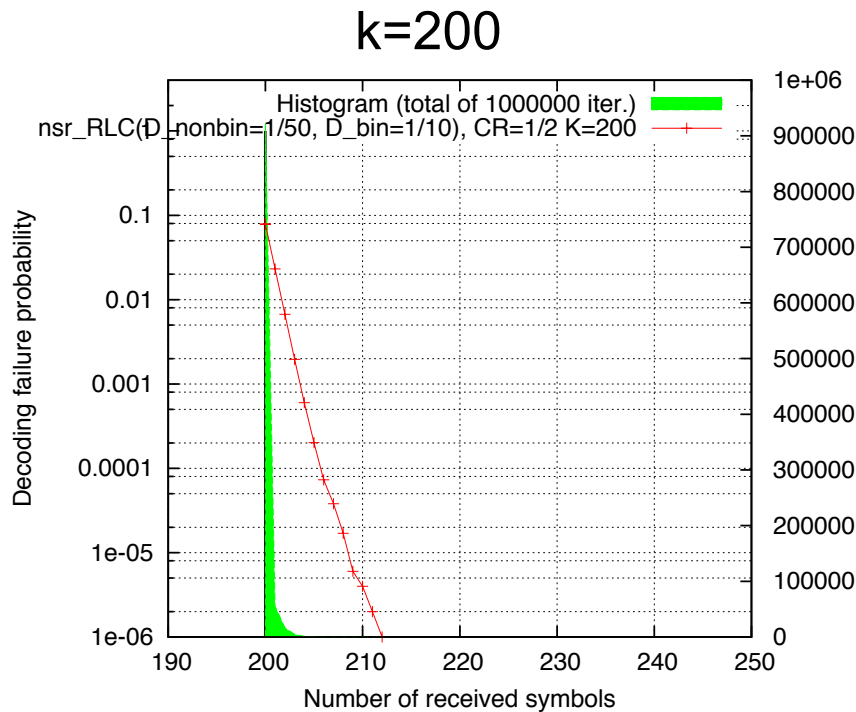
Preliminary results



NB: results are presented here as the concatenation of small curves...
 In practice it will be a single curve for a single code

Two close-ups

- decoding failure probability curves for $k=200, 500$
 - no visible error floor at 10^{-6} failure probability, which is excellent 😊



Conclusions and Future Works

Conclusions

- our proposal tries to take the best of RLC
 - **Use the right technique (bin vs. non-bin) at the right time, in the right way**
 - find balance between erasure recovery perf. and complexity
- our proposal tries to fill in the gap between sliding/elastic encoding window and block codes
 - **Side question: what about ALC and FECFRAME versions capable of using convolutional codes**
 - instead of being stuck to block AL-FEC?
- our proposal has a more limited scope than RLC
 - **but it is suited to concrete use-cases**
 - in IRTF/NWCRG (e.g., Tetrys)
 - in IETF/RMT and FECFRAME

Conclusions... (cont')

- many key questions remain
 - what are the **performances** when used in sliding or elastic encoding window?
 - e.g. with Tetrys
 - how **fast** is it?
 - e.g., compared to our optimized LDPC-Staircase/RS codecs
 - how does it **scale** with k ?
 - e.g., compared to our optimized LDPC-Staircase codec
 - define **signaling** aspects
 - FEC Payload ID (in each packet sent)
 - FEC Object Transmission Information (per object/session)