AL-FEC and MBMS services

vincent.roca@inria.fr
Inria Rhône-Alpes

16 octobre 2018

- **Part 1**

3GPP-MBMS (Multimedia Broadcast/Multicast Services)

Outline

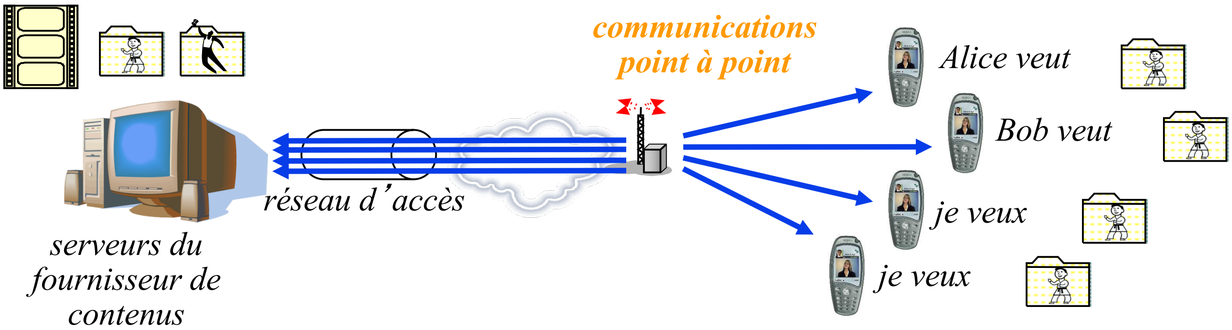
- some background on related standards…
  1. ***massively scalable delivery***
  2. IETF FLUTE/ALC standard for file massively scalable delivery
- and their use in 3GPP…
  1. MBMS

Massively scalable delivery

- comment transmettre un contenu très populaire ?
  - exemple : contenus enrichis à destination des spectateurs d'un d'évènements sportifs

Massively scalable delivery… (cont')

- comment transmettre un contenu très populaire ?
  - multi-vues en direct (streaming), contenus enrichis
    - **l'UMTS permet à chacun de surfer sur le Web…**
    - **…mais à quel prix ?**

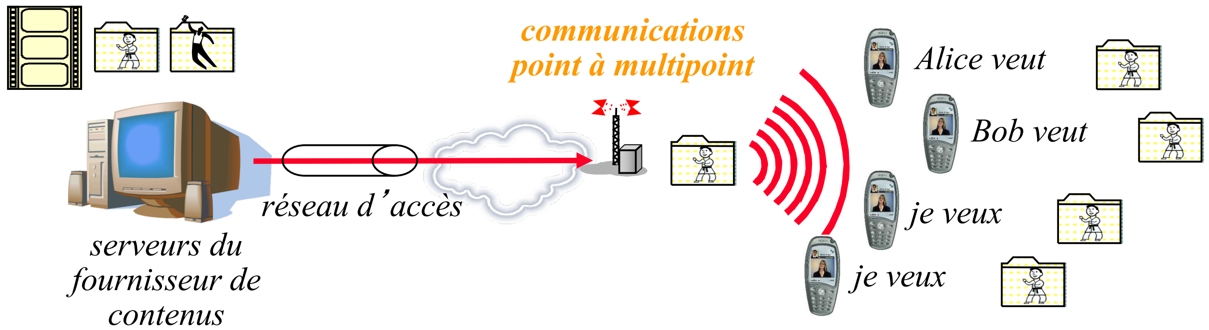


Massively scalable delivery… (cont')

- avec des communications point-à-point (ex. TCP), cela coince à plusieurs endroits
  - coté **serveur**
    - **milliers/millions de clients à traiter individuellement, à qui retransmettre les données manquantes, etc.**
  - coté **réseau d'accès** (liaison serveur-Internet)
    - **débit_par_client = débit_ligne / nb_clients**
    - **avec des millions de clients, ce n'est pas beaucoup et cela coûte très cher !**
  - coté **réseau de distribution** sans fil
    - **transmissions hertziennes ➔ médium partagé ➔ transmettre un très grand nombre de fois le même contenu est (1) un gâchis, et (2) impensable avec de gros contenus/grand nombre de clients**

Massively scalable delivery… (cont')

- d'où les services de diffusion fiables
  - 3GPP MBMS (Multimedia Broadcast/Multicast Services)
- fonctionne en mode point-à-multipoint
  - i.e. multicast ou diffusion
  - l'information traverse une seule fois un lien donné ☺
  - efficace qu'il y ait 1 utilisateur ou 1 000 000…



Massively scalable delivery… (cont')

- bénéfices
  - coté **serveur**
    - **passage à l'échelle massif si l'on utilise la bonne approche de transmission fiable (FLUTE et FEC)**
  - coté **réseau d'accès**
    - **débit_par_client = débit_ligne / nb_contenus_diffusés**
    - **avoir un millions de client ou un seul revient au même**
  - coté **réseau de distribution** sans fil
    - **exploite pleinement les caractéristiques naturelles de diffusion du réseau hertzien**

- les solutions reposent sur IP ➔ **protocole fédérateur qui permet une intégration triviale Internet/WiFi/LAN/UMTS/DVB-*/3GPP/…**

Outline

- some background on related standards…
  1. massively scalable delivery
  2. ***IETF FLUTE/ALC standard for file massively scalable delivery***

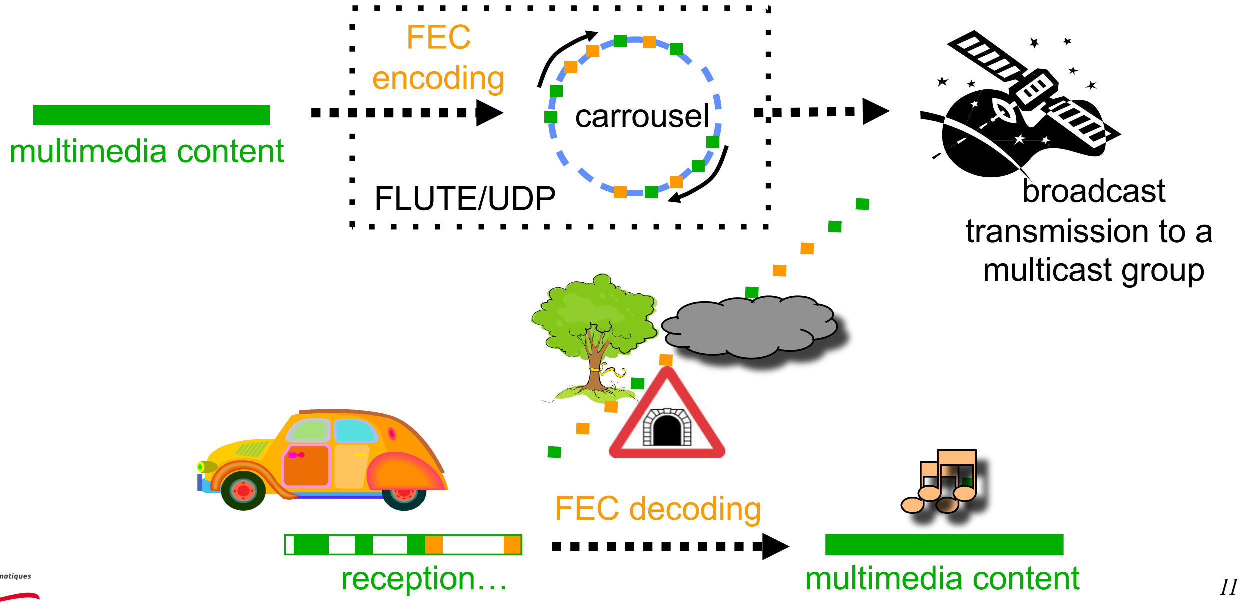
- and their use in 3GPP…
  1. MBMS

FLUTE/ALC

- diffusion de fichiers (au sens large)
  - service complémentaire aux transmissions point à point
  - repose sur de nouveaux protocoles/briques
    - **FLUTE** ➔ **application de transfert de fichiers**
    - **ALC** ➔ **protocole transport multicast fiable**
    - **AL-FEC** ➔ **codes correcteurs d'erreurs**
    - **le tout au dessus de UDP/IP**
  - l'objectif est une transmission ***totalement fiable***

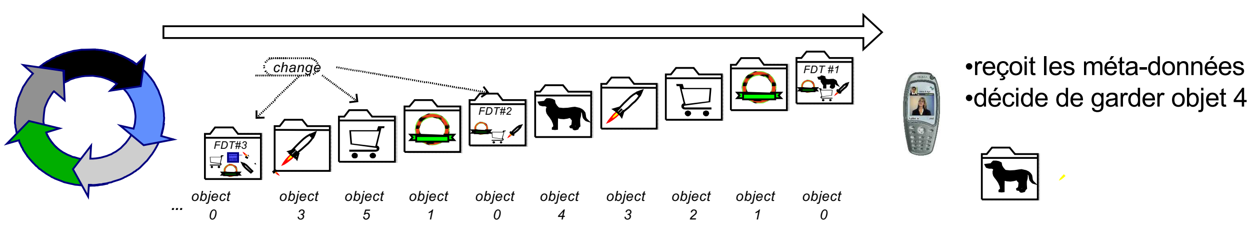
FLUTE/ALC… (cont')

- an example where AL-FEC improves both the reliability and the service
  - reliable distribution of digital content library to vehicles
  - relies on FLUTE/ALC for the file-casting service



FLUTE/ALC… (cont')

- ALC effectue la transmission (transport)
- approche de type « carrousel » (par ex.)
  - les contenus sont transmis en boucle, longtemps
    - ➔ **plusieurs cycles de transmission**
  - chaque client « pioche » ce qui l'intéresse dans la session, grâce aux méta-données qui décrivent le contenu disponible



carrousel (contenu dynamique)

FLUTE/ALC… (cont')

- FLUTE décrit le contenu transmis
  - les méta-données des fichiers de la session sont rassemblés dans la FDT (File Delivery Table)
  - exemple (XML/MIME) :

```
<?xml version="1.0" encoding="UTF-8"?>
<FDT-Instance xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:fl="http://www.example.com/flute"
         xsi:schemaLocation="http://www.example.com/flute-fdt.xsd"
         Expires="2890842807">
         <File    Content-Location="www.example.com/menu/tracklist.html"
                  TOI="1"
                  Content-Type="text/html"/>
         <File    Content-Location="www.example.com/tracks/track1.mp3"
                  TOI="2"
                  Content-Length="6100"
                  Content-Type="audio/mp3"
                  Content-Encoding="gzip"
                  Content-MD5="Eth76GlkJU45sghK"/>
</FDT-Instance>
```


FLUTE/ALC… (cont')

- le standard ne définit pas comment est transmis FDT Instance et contenus
  - **ex. 1 : FDT Instance transmise périodiquement + tous les paquets de tous les objets (source + redondance) transmis dans un ordre aléatoire**
  - **ex. 2 : FDT Instance transmise périodiquement + objets transmis en séquence suivi pour chacun des paquets de redondance associés**

Outline

- some background on related standards…
  1. massively scalable delivery
  2. IETF FLUTE/ALC standard for file massively scalable delivery

- and their use in 3GPP…
  1. ***MBMS***

The MBMS services

- 3GPP Multimedia Broadcast Multicast Service (MBMS)
  - **point to multipoint specifications for 3GPP cellular networks**
  - **heavily relies on IETF standards, in particular FLUTE/ALC**
  - **MBMS protocol stack has been reused (with modifications) by other standards**
    - DVB-H/SH
    - ISDB-Tmm
    - ATSC-M/H
    - OMA BCAST

3GPP TS 26.346 V15.2.0 (2018-09)

Technical Specification

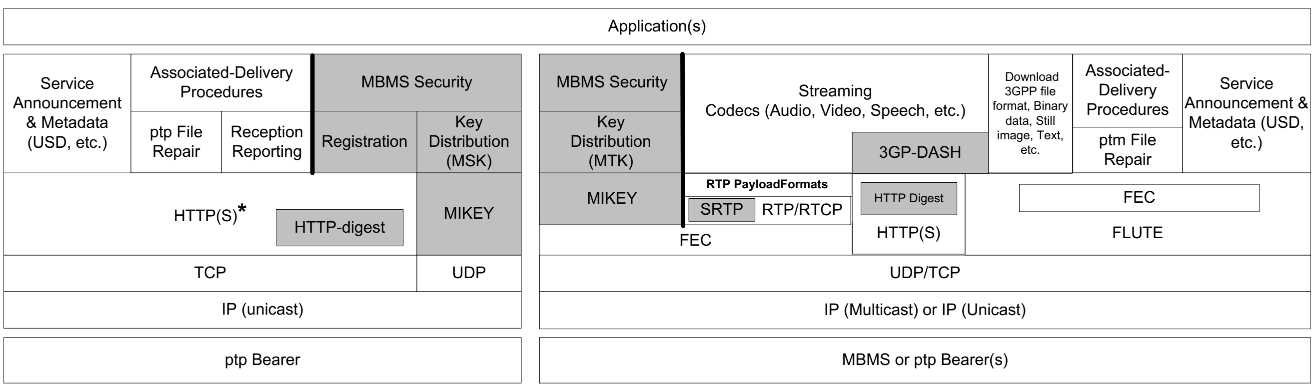
3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Multimedia Broadcast/Multicast Service (MBMS);
Protocols and codecs
(Release 15)

A GLOBAL INITIATIVE

www.3gpp.org/DynaReport/26346.htm

The MBMS services… (cont')

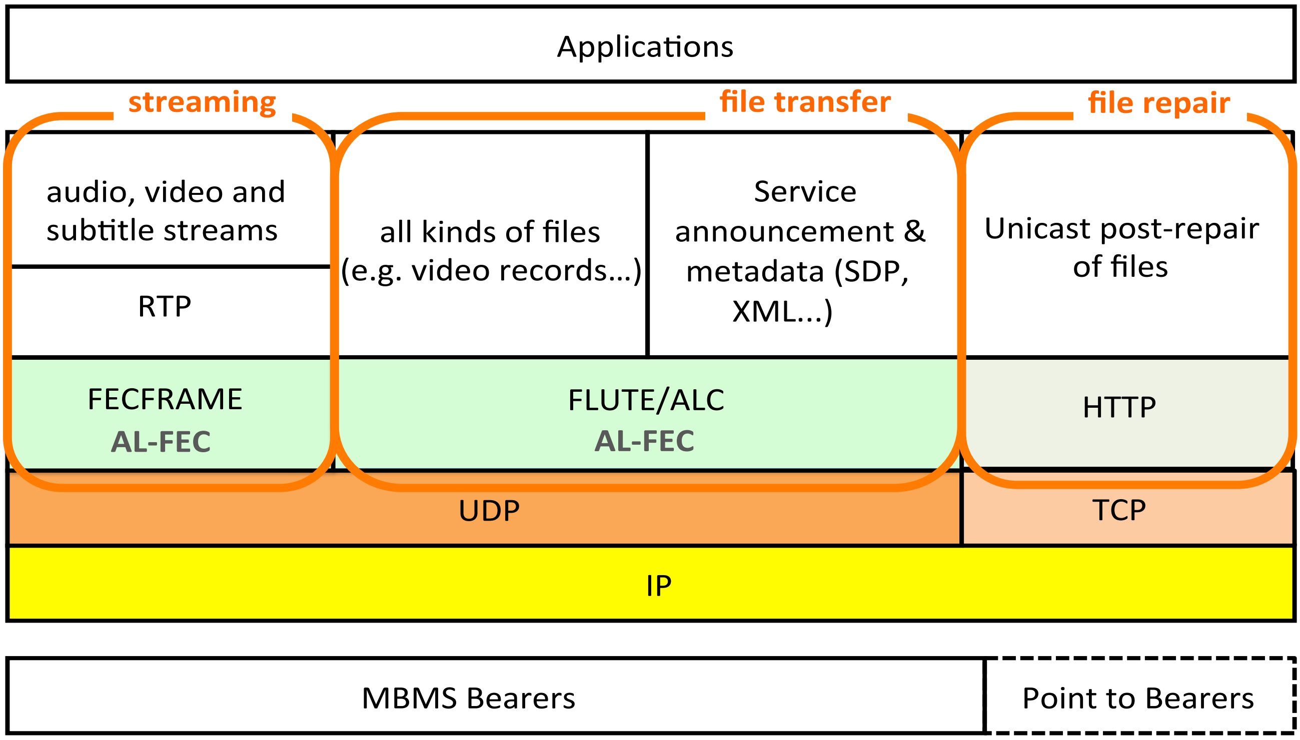
- the big view of MBMS protocols



point to point communications only   point to point + multicast communications

The MBMS services… (cont')

- simplified view of a subset of MBMS protocols



FLUTE usage in MBMS

- relies on FLUTE/ALC as the core transport protocol
  - **no change**
- relies on HTTP repair servers
  - **an object can be sent during a limited time span in a FLUTE/ALC session…**
  - **… so a receiver may not be able to get it**
  - **if that happens, this receiver can ask for missing source symbols using HTTP GET requests**
  - **he either asks for**
    - source symbols still missing after FEC decoding (no additional FEC decoding)
    - or a sufficient number of source symbols to finish FEC decoding

● Part 2

Application-Level FEC codes

http://openfec.org

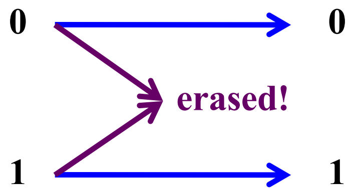
Outline

1. **the erasure channel**
2. AL-FEC codes
3. a few AL-FEC codes and their performances

The erasure channel

- erasure channel
  - definition:

    a symbol either arrives to the destination, without any error… … or is erased and never received



    ≠ BSC (binary symmetric) and AWGN channels…

  - the integrity assumption is a strong hypothesis
    - **a received symbol is 100% guaranteed error free**
  - largely simplifies decoding:
    - **belief propagation decoding** ⇒ **iterative decoding**
    - **maximum likelihood decoding** ⇒ **Gaussian elimination**
    - **more details to come…**

The erasure channel… (cont')

- where do we find erasure channels?
  - the **Internet** is intrinsically an erasure channel
    - **an IP datagram is either received or erased**
    - **usually caused by a router congestion, or a routing error**
  - because of Ethernet CRC or TCP/UDP/IP **checksum** verifications
    - **when the underlying layers failed to recover/identify errors, the "packet" is eventually discarded by higher layers checks**
    - **certain MAC layers can ask for retransmissions, but it's usually not the case**

The erasure channel… (cont’)

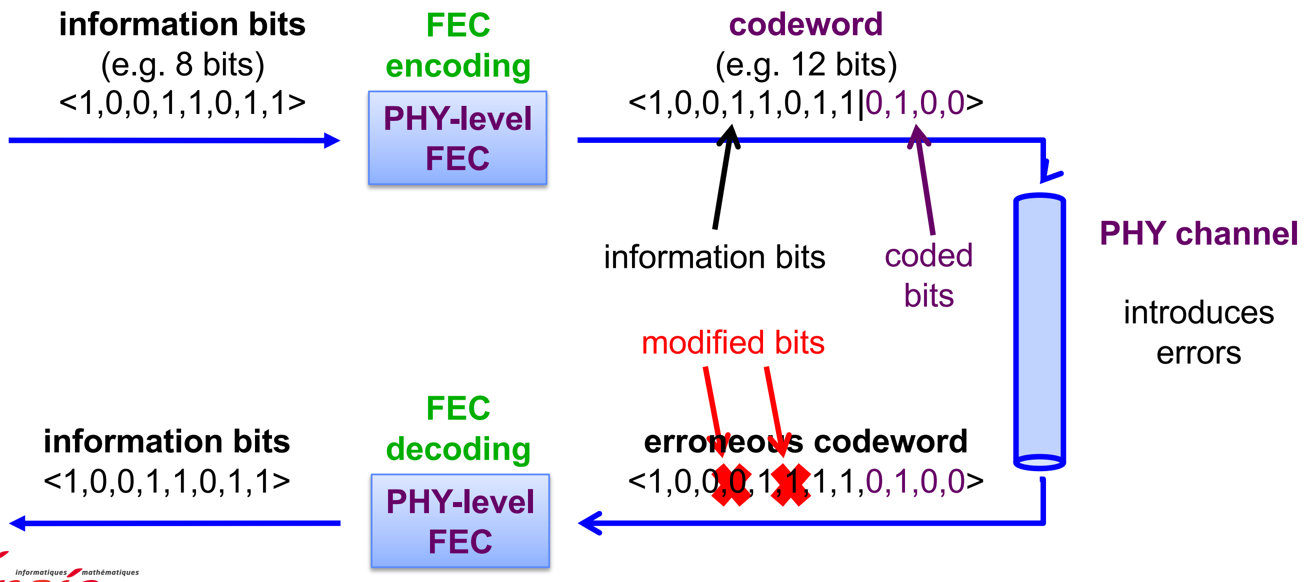
- where do we find erasure channels… (cont)
  - due to an **intermittent connection** model
    - **caused by the environment (obstacle in wireless comm.)**
      - a mobile terminals receives a subset of the packets sent
    - **caused by the application (e.g., if it crashes)**
      - packets sent during the off-period are lost
    - **these situations are easily recovered with point-to-point connections…**
      - TCP will do the job…
      - if a new connection is needed, it’s sufficient to remember where the download stopped and ask the sender to resume from that point
    - **…but it’s quite different if the content is broadcast/multicast**
      - the same content is sent to 100,000s of receivers!

The erasure channel… (cont’)

- where do we find erasure channels… (cont)
  - with **distributed storage** of a content (e.g., a file), when a storage point fails
    - **RAID disks**
    - **network-wide distributed storage, where a client selects a subset of the storage node, each of them having a chunk of the content (source or repair data)**

Quite different from PHY-layer FEC

- a PHY-layer FEC protects against bit errors, not packet losses!



Outline

1. the erasure channel
2. **AL-FEC codes**
3. a few AL-FEC codes and their performances

AL-FEC code definitions

- AL-FEC are codes for the **erasure channel**
  - only!
  - we always assume there's no error in what we receive

- **codes** and **codecs** are not the same

Be careful: codes ≠ codecs

specifies the way repair symbols are generated

implementation of the encoder and decoder

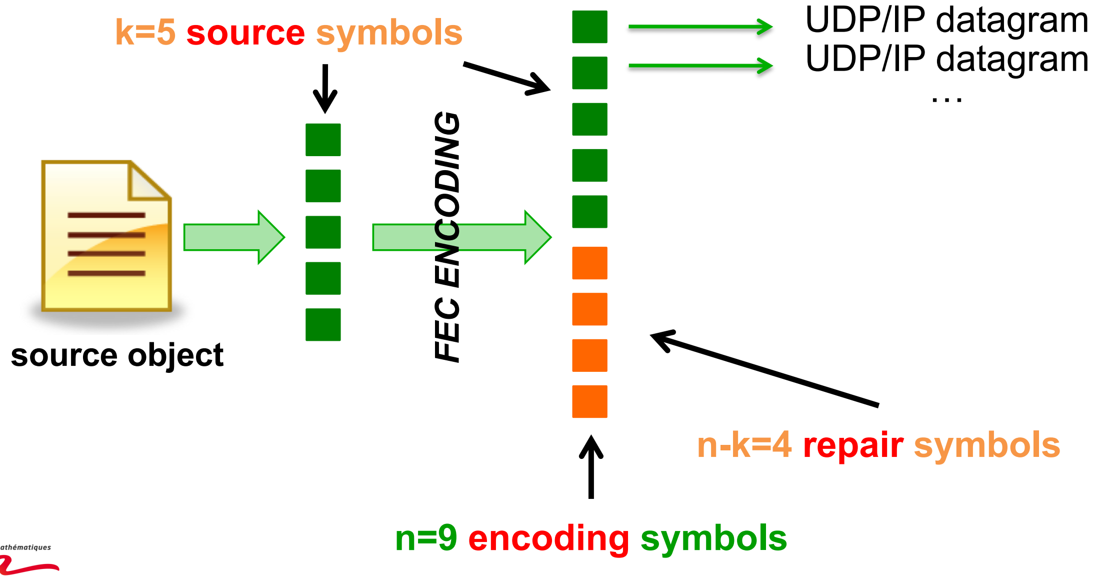
AL-FEC code definitions… (cont')

- the **symbol** is the data unit manipulated by the AL-FEC codec
  - usually a symbol is carried in a UDP/IP datagram
    - ⇒ **either received or erased**
    - **NB: sometimes there are several symbols per UDP/IP datagram**
      - goal is to artificially increase the # of symbols in an object
      - useful since LDPC/Raptor codes perform better when there are more than a few 1,000s of symbols
    - **let's keep it simple from now on…** ***one symbol per packet***

- initially **k source** symbols, **n encoding** symbols after FEC encoding

AL-FEC code definitions… (cont')

- example:
  - **source object, of size 5 kB, segmented into 5 source symbols of size 1 kB each, to which FEC encoding adds 4 repair symbols, also of size 1 kB. Source symbols are part of the 5+4 encoding symbols (systematic FEC code).**



AL-FEC code definitions… (cont')

- the **code rate** is a key parameter

$$code\_rate = \frac{k}{n} = \frac{before\_encoding}{after\_encoding}$$

- close to 1 ⇒ little/no redundancy
- close to 0 ⇒ high amount of redundancy

- **examples:**
  - code_rate = **0.5** means that there are as many redundancy symbols as source symbols
  - code_rate = **2/3** means that n=1.5*k, i.e. we add 50% of redundancy

AL-FEC code definitions… (cont')

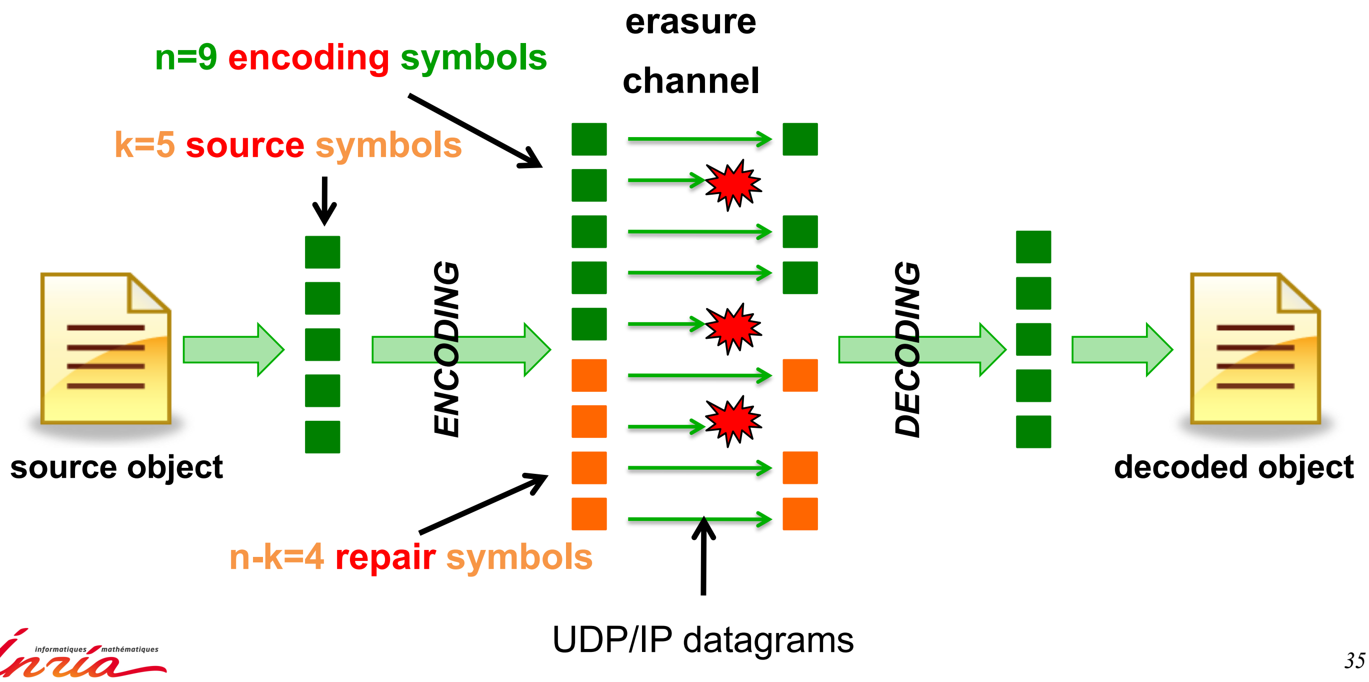
- **systematic** codes
  - codes for which the k source symbols are part of the n encoding symbols
    - **see previous example…**
  - preferred to non-systematic codes because:
    - **without loss, no decoding is needed**
      - save processing
    - **a receiver that does not implement the FEC code can still use source symbols (backward compatibility)**
      - of critical importance

AL-FEC code definitions… (cont')

- **ideal** or **MDS** (maximum distance separation) codes
  - a code for which decoding is always possible after receiving any set of k symbols among n possible
    - **example: Reed-Solomon codes over GF(2^m), with m=4, 8 or 16 for instance**
  - it's an optimum code in terms of erasure recovery…
    - **one cannot find something better**
  - … which does not mean it's necessarily the best possible code for a given use case
    - **there are constraints**
    - **example: RS over GF(2^8) have strict limits $k \leq n \leq 255$**

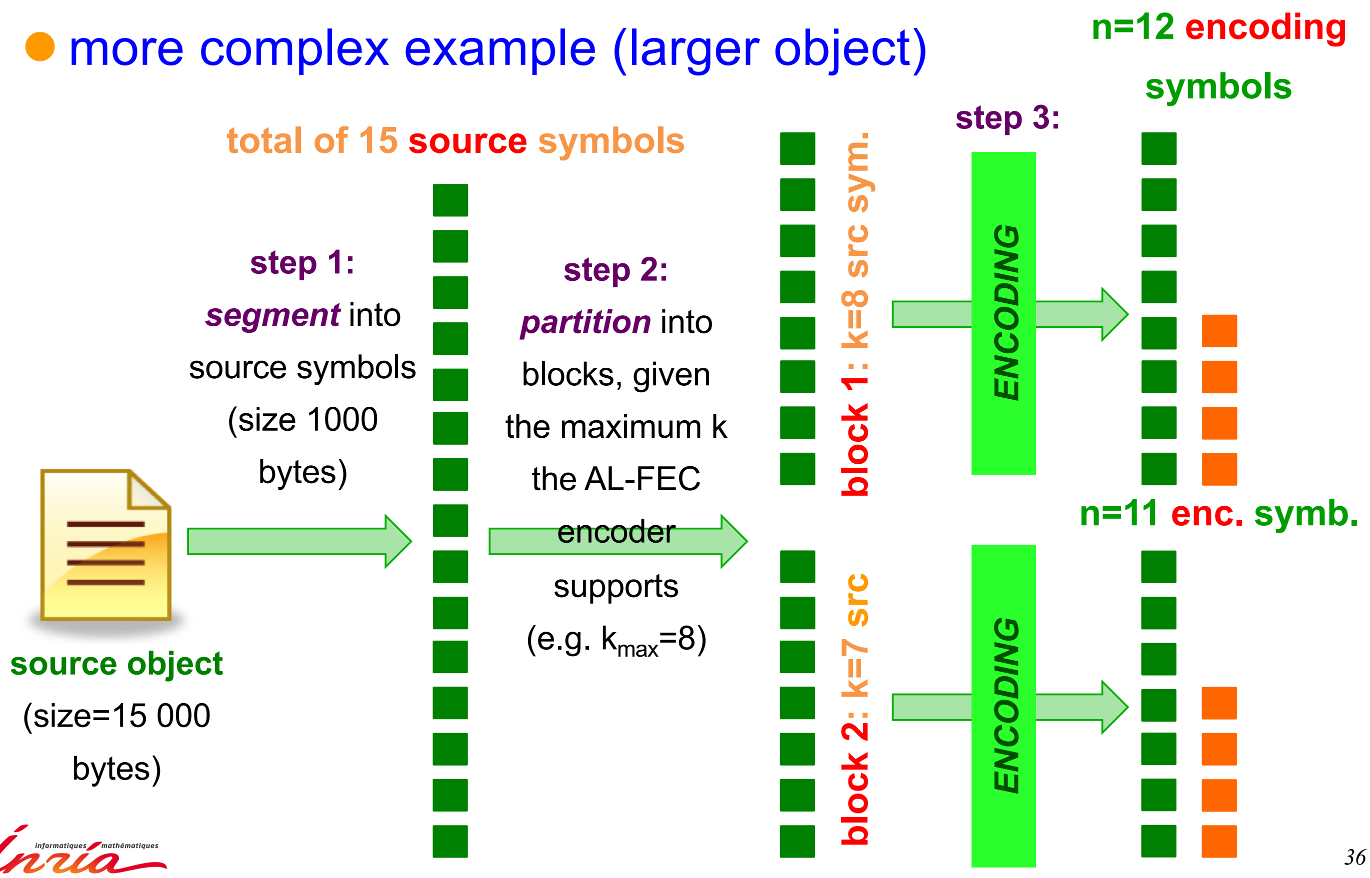
An example of use

- trivial example with a (9; 5) code



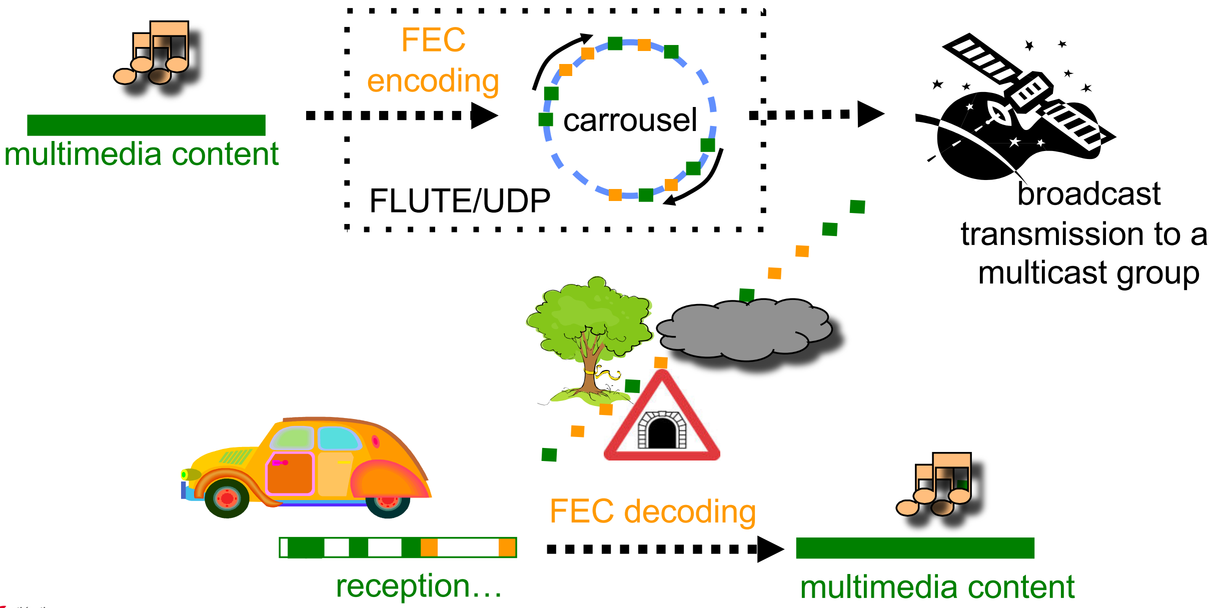
An example of use… (cont')

- more complex example (larger object)



An example of use… (cont')

- broadcast of a digital library to vehicles, using FLUTE/ALC in carousel mode, over a long period



AL-FEC differ from PHY-layer FEC

- they are usually implemented in higher comm. layers
  - rather than in the PHY layer
  - for instance:
    - **within the application**
    - **within the transport system (between RTP and UDP for streaming flows, in FLUTE/ALC for filecasting applications)**
    - **within the MAC layer (e.g., in DVB-H/MPE-FEC, or in DVB-SH/MPE-IFEC)**
  - hence their name "Application Level-FEC" (AL-FEC)

Performance metrics

- how can we define good AL-FEC codes?
  - ⇒ define performance metrics

Performance metrics

- metric 1: **decoding overhead** as a measure of the erasure recovery capabilities
  - **major performance criteria since many AL-FEC are not MDS**
  - **measured as the *overhead ratio*:**

$$decoding\_overhead\_ratio = \frac{\#\_of\_symbols\_required\_for\_decoding}{k} - 1$$

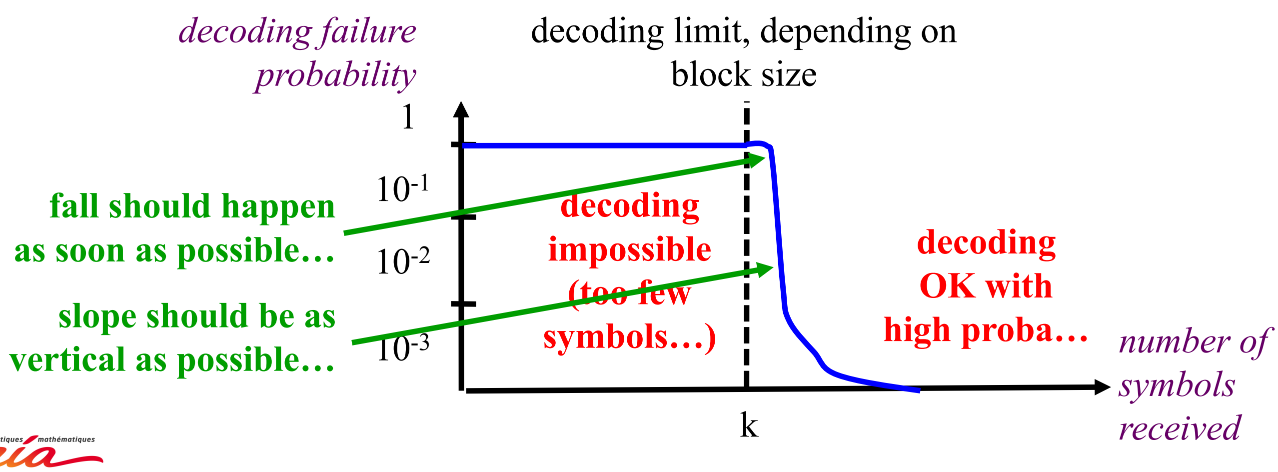
    - e.g. overhead=0.63% means that 0.63% of symbols in addition to k are needed for decoding to succeed
  - **or as the *raw overhead* (number of extra symbols):**

$$decoding\_overhead = \#\_of\_symbols\_required\_for\_decoding - k$$

  - **derivatives: average overhead, 99% margin above (resp. below) the average, etc.**

Performance metrics… (cont')

- metric 3: decoding **failure probability** = f(number of symbols received)
  - ❍ **similar to metric 2 but as a function of the number of symbols actually received**
    - the curve is an average over a large number of experiments, for a given {code rate, loss rate}, since each experiment impacts the identity of the symbols actually received



Performance metrics… (cont')

- metric 5: encoding and decoding **speed**
  - ❍ **to appreciate the algorithmic complexity**
    - often more reliable than theoretical algorithmic complexity analyzes
    - theoretical analyzes often neglect such important aspects as memory access/cache behavior/implementation details/…
  - ❍ **don't forget that the decoding algorithm impacts the erasure recovery metric…**
    - …usually there is an appropriate balance to find
    - some algorithms are faster but lead to lower erasure recovery capabilities (see later)
  - ❍ **sometimes decoding complexity is the key**
    - e.g., lightweight portable device
  - ❍ **sometimes encoding complexity is the key**
    - e.g., deep space (autonomous) probe

Performance metrics… (cont')

- metric 6: required **memory** during encoding and decoding
  - **even if RAM is used (rather than chipset memory), it should be used with caution**
    - e.g., if data can fit in the CPU L2 cache, it's great
  - **high impact of the decoding algorithm**

Performance metrics… (cont')

- performances depend on many parameters:
  - block size
    - **impacts the decoding overhead**
    - **some codes (e.g., LDPC and Raptor) are good asymptotically but sub-optimal with "small blocks"**
  - symbol size
    - **impacts speed and memory requirements**
  - decoding algorithm
    - **e.g., iterative decoding is fast, but leads to sub-optimal overhead results**

good AL-FEC = good code + good codec
(+ good lobbying ;-))

Outline

1. the erasure channel
2. AL-FEC codes
3. **a example of AL-FEC code**

What do AL-FEC codes look like?

- we'll, it's not so different from PHY codes
  - we find the same linear block codes
    - **e.g., Reed-Solomon and LDPC**
  - plus some **specialized small-rate**/rate-less codes
    - **e.g., Raptor(Q)™, GLDPC, LDPC (if used correctly)**
  - decoding techniques change, but not necessarily the inner coding techniques

- in the remaining, I'll quickly introduce LDPC-staircase
  - NB: we won't discuss Raptor(Q) ™ codes even if largely used in MBMS systems
    - **far too complex!**

LDPC codes

- in short
  - "Low Density Parity Check" (LDPC)
    - **linear block codes**
    - **discovered by Gallager in the 60's**
    - **re-discovered in mid-90s**
  - original codes were extremely costly to encode
    - **generator matrix (G) creation requires a matrix inversion**
    - **but we use high performance variants**
  - in the remaining we only consider **binary** codes

LDPC-Staircase codes

- LDPC-staircase codes (RFC 5170)
  - **a particular class of binary LDPC codes**
  - **A.K.A. "Repeat Accumulate" codes**
  - **a simple structure that defines a set of linear equations**
  - **IETF RFC 5170** http://datatracker.ietf.org/doc/rfc5170/

parity check matrix (H)

$$\begin{array}{c} \begin{array}{cccccccccc} \multicolumn{5}{c}{\text{source symbols}} & \multicolumn{5}{c}{\text{parity symbols}} \\ S_1 & S_2 & S_3 & S_4 & S_5 & P_1 & P_2 & P_3 & P_4 & P_5 \end{array} \\ \left(\begin{array}{ccccc|ccccc} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right) \end{array}$$

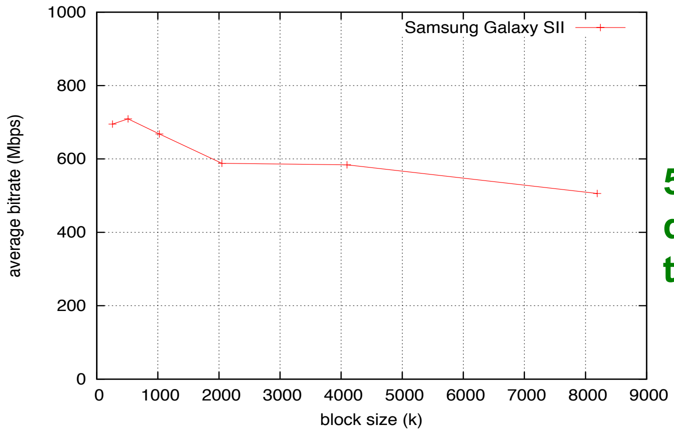
N_1 "1"s per column (e.g. 3)

linear system

$$\left\{\begin{array}{l} S_3 \oplus S_4 \oplus P_1 = 0 \\ S_1 \oplus S_4 \oplus S_5 \oplus P_1 \oplus P_2 = 0 \\ S_1 \oplus S_2 \oplus S_3 \oplus P_2 \oplus P_3 = 0 \\ S_2 \oplus S_4 \oplus S_5 \oplus P_3 \oplus P_4 = 0 \\ S_1 \oplus S_2 \oplus S_3 \oplus S_5 \oplus P_4 \oplus P_5 = 0 \end{array}\right.$$

LDPC-Staircase encoding

- encoding
  - it's **trivial** ☺
    - **produce repair symbols in sequence: P_1, then P_2, then P_3…**
  - guaranties a high encoding speed
    - **example: CR=2/3, N1=7, symbol size=1024 bytes**
    - **Samsung Galaxy S2 smartphone, ARM Cortex A9, 1.2GHz**



500-700 Mbps ☺ quelle que soit la taille du bloc

LDPC-Staircase decoding

- decoding (much more complex)
  - solve a system of binary linear equations
  - what are the equations?
  - what are the variables?
  - 3 solutions
    1. **use classic Gaussian Elimination?**
    2. **use a trivial iterative decoding (IT)?**
    3. **do something in-between?**
  - in practice, we often start with (2) and finish with (1) or (3) in bad reception conditions

LDPC-Staircase decoding… (cont')

- the trivial "ITerative decoding"
  - ❍search equations with a single variable left
  - ❍then you know the variable value (constant term)
  - ❍re-inject in all equations where this variable is present
  - ❍re-iterate…

original linear system

$$\begin{cases} S_3 \oplus S_4 \oplus P_1 = 0 \\ S_1 \oplus S_4 \oplus S_5 \oplus P_1 \oplus P_2 = 0 \\ S_1 \oplus S_2 \oplus S_3 \oplus P_2 \oplus P_3 = 0 \\ S_2 \oplus S_4 \oplus S_5 \oplus P_3 \oplus P_4 = 0 \\ S_1 \oplus S_2 \oplus S_3 \oplus S_5 \oplus P_4 \oplus P_5 = 0 \end{cases}$$

simplified linear system

S1, S2, S5, P1, P2 already received or decoded

$$\begin{cases} S_3 \oplus S_4 = P_1 \\ S_4 = S_1 \oplus S_5 \oplus P_1 \oplus P_2 \\ S_3 \oplus P_3 = S_1 \oplus S_2 \oplus P_2 \\ S_4 \oplus P_3 \oplus P_4 = S_2 \oplus S_5 \\ S_3 \oplus P_4 \oplus P_5 = S_1 \oplus S_2 \oplus S_5 \end{cases}$$

  - ❍does it work here?

LDPC-Staircase decoding… (cont')

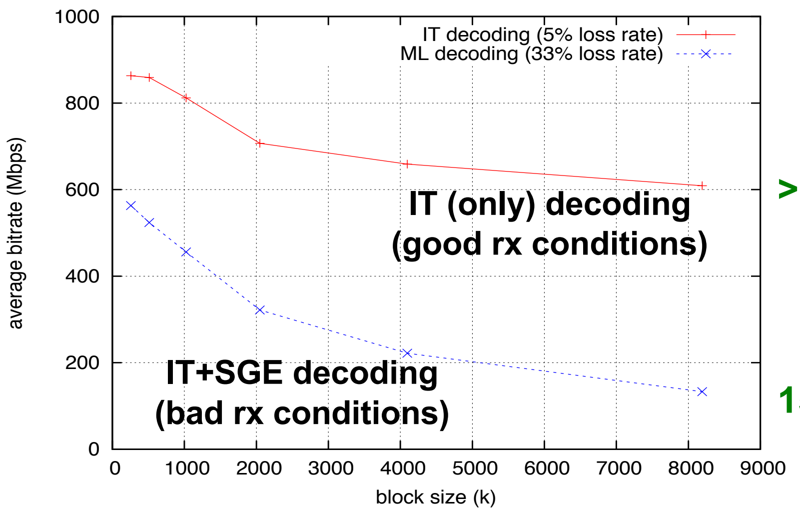
- the more complex **Structured Gaussian Elimination** (SGE)
  - ❍SGE is an old optimized way of using GE over sparse systems, midway between IT and regular GE

[16] B. A. LaMacchia and A. M. Odlyzko, "Solving large sparse linear systems over finite fields," in *Advances in Cryptology (Crypto'90), LNCS 537, Springer-Verlag*, 1991.

[17] C. Pomerance and J. W. Smith, "Reduction of huge, sparse matrices over finite fields via created catastrophes," *Experimental Mathematics, Vol. 1, No. 2*, 1992.

LDPC-Staircase codes… (cont')

- decoding: speed
  - ❍CR=2/3, N1=7, symbol size=1024 bytes
  - ❍Samsung Galaxy S2 smartphone, ARM Cortex A9, 1.2GHz



> 600 Mbps ☺

150 - 550 Mbps ☺

LDPC-Staircase codes… (cont')

- decoding: erasure recovery performance
  - ❍k=1024, code rate=2/3

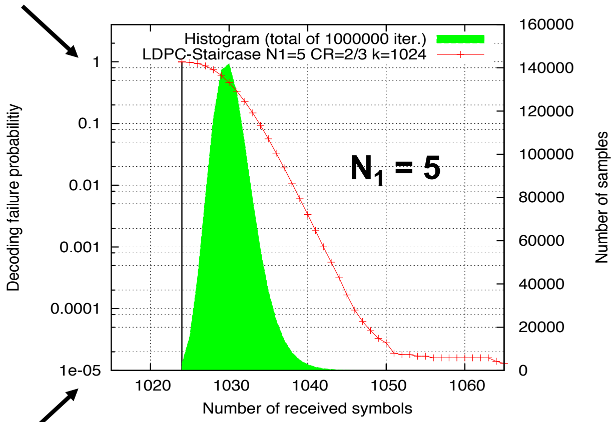
parameters	average overhead	overhead for a failure probability ≤ 10^{-4}
k=1024, N_1=5	0.636%	with 1046 symbols received (i.e. 2.1% or 22 symbols overhead): $Pr_{fail}=5.9*10^{-5}$
k=1024, N_1=7	0.238%	with 1039 symbols received (i.e. 1.5% or 15 symbols overhead): $Pr_{fail}= 8.2*10^{-5}$

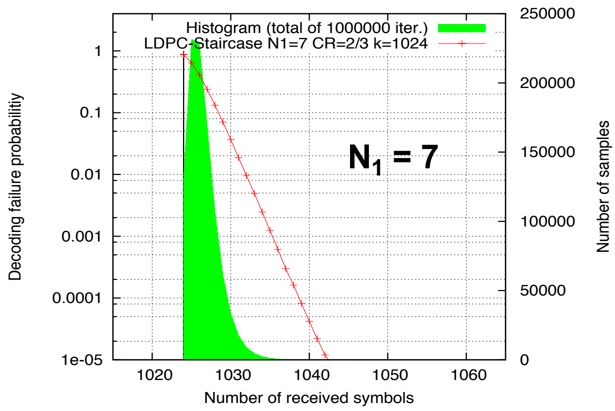
- we often choose **N1=7** for almost ideal recovery performance

LDPC-Staircase codes… (cont')

- decoding: erasure recovery performance
  - ❍ **k=1024, code rate=2/3**
  - ❍ **these curve give the decoding failure probability as a function of the number of symbols received (i.e. overhead)**
    - minimum number of symbols is 1024
    - we see the positive impacts of increasing N_1

Pr = 1, cannot be decoded





Pr << 1, high decoding probability