RS + LDPC-Staircase Codes for the Erasure Channel: Standards, Usage and Performance

Vincent Roca, Mathieu Cunche, Cédric Thienot, Jonathan Detchart, Jérôme Lacan

RS + LDPC-Staircase Codes for the Erasure Channel: Standards, Usage and Performance

Vincent Roca* Mathieu Cunche** Cédric Thienot † Jonathan Detchart‡ Jérôme Lacan‡
*Inria, France * CITI - INSA-Lyon, France †Expway, France ‡Université de Toulouse - ISAE, France
{vincent.roca,mathieu.cunche}@inria.fr, cedric.thienot@expway.com, {jonathan.detchart,jerome.lacan}@isae.fr

***Abstract*—Application-Level Forward Erasure Correction (AL-FEC) codes are a key element of telecommunication systems. They are used to recover from packet losses when retransmission are not feasible and to optimize the large scale distribution of contents. In this paper we introduce Reed-Solomon/LDPC-Staircase codes, two complementary AL-FEC codes that have recently been recognized as superior to Raptor codes in the context of the 3GPP-eMBMS call for technology [1]. After a brief introduction to the codes, we explain how to design high performance codecs which is a key aspect when targeting embedded systems with limited CPU/battery capacity. Finally we present the performances of these codes in terms of erasure correction capabilities and encoding/decoding speed, taking advantage of the 3GPP-eMBMS results where they have been ranked first.**[1]

***Keywords*—Erasure channel, AL-FEC codes, FLUTE/ALC, FECFRAME, 3GPP-MBMS.**

I. Introduction

Erasure channels are characterized by the property that the transmitted data units are either received without error or are erased (lost). In telecommunications, Internet and many wireless systems can be regarded as erasure channels where losses can be caused by router congestions, non-recoverable corruptions due to poor wireless reception conditions, or intermittent connectivity. However, if a packets reaches the upper layers, its integrity has been verified several times (CRC and checksums) and it is assumed to be error-free. Reliability may be achieved thanks to retransmissions (e.g. with TCP), but not always: the return channel used for acknowledgement and retransmission requests may not exist (e.g. with a unidirectional broadcast network), the RTT (round trip time) delay generated by the retransmission may be too large for real-time applications, or scalability issues may prevent the use of the feedback channel (e.g. with a huge number of receivers). This is why FEC codes are so important: reliability is achieved thanks to *erasure codes* that add some redundancy to the transmitted information. Such codes are known as Application-Layer Forward Erasure Correction (AL-FEC) codes, as they are usually implemented in the upper layers, close to the application.

Encoding a source object requires this object to be divided into k pieces of equal size, called source symbols. These symbols are then FEC encoded into n encoding symbols, with $n \geq k$, and these encoding symbols are sent over the network in packets (e.g. using FLUTE/ALC [2] or RTP packets). AL-FEC codes considered here are all systematic, meaning that the k source symbols are part of the n encoding symbols. The $CR = k/n$ ratio is called code rate, and quantifies the amount of redundancy added by the AL-FEC code. At least, k distinct encoding symbols are required to decode the source object, and codes capable of achieving this bound are called Maximum Distance Separable (MDS) codes.

When designing AL-FEC codes, several aspects must be considered. First the receiver should be able to decode an object from a number of encoding symbol as small as possible, k being the optimum. Then, in order to be used on lightweight terminals (e.g. smartphones), the *encoding and decoding speeds* must be high (equivalently, the CPU load must be low) and the *maximum memory consumption* kept to a minimum. Sometimes other features are required: the *large-block* capability (i.e. encoding a very large object directly, without being obliged to split it into several blocks that are encoded separately), or the *small-rate* capability (producing a large number of encoding symbols), or at the extreme a quasi-infinite number of encoding symbols with rateless codes.

In this paper we introduce an AL-FEC solution based on LDPC-Staircase and Reed-Solomon codes. We detail the techniques used to design high speed decoders and we provide extensive performance analyses, essentially conducted in the context of the 3GPP-eMBMS call for technology [1]. Note that during the design of LDPC-Staircase codes, we refrained ourselves from using techniques known to be patented in order to maximize the probability of having an IPR free solution (although we recognize this is in practice impossible to prove). We therefore chose "old" techniques, for which prior art exists. In spite of this constraint we show that their performance is exceptionally high and often identical or sometimes superior to that of Raptor codes. This is our main contribution.

The paper is organized as follows. Section II introduces the AL-FEC codes and their integration in standards is discussed in section III. Section IV discusses the implementation of high performance codecs. Finally thorough performance evaluations are presented in section V with regards to erasure recovery capabilities and in section VI with regards to decoding speeds and memory requirements.

II. AL-FEC codes presentation

A. LDPC-Staircase codes

LDPC-Staircase codes [3], [4] belong to the well-known class of Low-Density Parity Check (LDPC) codes that are characterized by a sparse parity check matrix. The parity check matrix, defines a linear system of $n-k$ equations involving the k source symbols (original data) and $n-k$ repair symbols (redundancy). Each column is associated with a symbol and

[1]This work was supported in part by the ANR-09-VERS-019-02 grant (ARSSO project).

each row represents an equation (A.K.A. constraint). The left-most part of the LDPC-Staircase's matrix, H_1, is a $(n-k) \times k$ sub-matrix whose columns correspond to source symbols, and such that there are N_1 '1' per column and at least two '1' per row. The rightmost part of the matrix, H_2, is a $(n-k) \times (n-k)$ matrix whose columns correspond to repair symbols, with a staircase (or double-diagonal) structure (see Figure 1).

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

Fig. 1. Example of LDPC-Staircase parity check matrix, with $k = 6$, $n = 12$, $N_1 = 3$ (NB: the example is too small to highlight its sparseness nature).

a) Encoding: Encoding consists in creating the encoding symbol from the source object. Thanks to the staircase structure of H_2, the repair symbol are created in sequence, with a linear complexity (section VI-A).

b) Decoding: Decoding consists in rebuilding the source object from the set of received encoding symbols, by solving the corresponding linear system. One approach is to use a Maximum Likelihood (ML) algorithm (e.g. Gaussian Elimination, GE). ML decoding guarantees optimal correction capabilities, at the cost of a theoretical cubic complexity with GE. But the sparse nature of LDPC codes enables an alternative approach, called Iterative (IT) decoding [5] (A.K.A. peeling decoder), that uses an iterative approach to solve one by one the equations where only one unknown variable remains. It features a linear complexity but sometimes fails to solve a system admitting a solution.

A good approach [6], [7] is to start with IT decoding: in good conditions the low complexity IT decoder is sufficient. Then, if needed, we switch to ML decoding, which will work on a simplified linear system. We discuss another improvement, mid-way between IT and ML decoding, in section IV.

B. Reed-Solomon codes

Reed-Solomon codes [8], [9] are an example of MDS codes. They rely on a Galois Field (GF) such as GF(2^m), that is a set of 2^m elements on which multiplication and addition operations are defined. In our case, RFC5510 [9] specifies systematic Reed-Solomon codes for the packet erasure channel, based on Vandermonde matrices. As for the LDPC codes, Reed-Solomon codes require the object to be divided into k symbols[2]. Each symbol is itself composed of S elements of GF(2^m), amounting to a total of $S * m$ bits per symbol. Encoding is done by multiplying the vector containing the source symbols by a $k \times n$ encoding matrix with elements in GF(2^m). This results in n encoding symbols of size S m-bit elements each, the first k symbols being source symbols (systematic code property). At the receiver, as soon as k distinct encoding symbols are received, the source object is decoded using a GE method (by design, the linear system is guaranteed not to be singular).

The larger the Galois field, the longer the code is (n), but also the slower the operations are. This is why, for complexity reasons, practical implementations use Galois Fields of size 2^8, limiting the code to a length of $n \leq 255$ encoding symbols.

[2] Splitting an object into blocks of similar size may also be needed if this object is too large W.R.T. the AL-FEC code limitations.

C. Raptor/RaptorQ AL-FEC codes

Raptor codes [10] form another family of AL-FEC codes. They belong to the class of *rate-less* codes as they can (in theory) produce an infinite number of encoding symbols. However they are often used as traditional block codes, producing a number of symbols that is predetermined by the application. Two versions exist: *Raptor* [11], the first one, and *RaptorQ* [12]. The main difference between them is that Raptor are binary codes (like LDPC), while RaptorQ are non-binary codes as they partially rely, like the Reed-Solomon codes, on operations over GF(2^8). Those are the two AL-FEC codes against which we compare our solution.

III. LDPC-Staircase/Reed-Solomon in Standards

A. IETF Specifications

Two IETF working groups are concerned. In the Reliable Multicast Transport (RMT) WG, which focuses on reliable file delivery, LDPC-Staircase is standardized as RFC5170 [4] and Reed-Solomon as RFC5510 [9]. These standards explain how to use them with FLUTE/ALC [2], to enable a reliable and scalable (no limit on the number of receivers) multicast/broadcast delivery of contents over unidirectional transport networks (i.e. without feedbacks). In the FEC Framework WG, which focuses on real-time delivery services, they are standardized as RFC6816 [13] and RFC6865 [14].

B. Use in Other Standardized Systems

Since mid 2012, LDPC-Staircase codes are the AL-FEC scheme being used in the Japanese ISDB-Tmm standard [15]. This is a broadcast technology (based on ISDB-T), used for digital terrestrial TV services and suited to mobile environments. LDPC-Staircase codes are the core AL-FEC technology, and they are used along with FLUTE/ALC to improve the reliability and efficiency of push video services.

IV. Codec Design

A good AL-FEC solution requires both a good code and a good codec. Since AL-FEC codes work within or close to the application, they are typically implemented in software (C language in our case). Our http://openfec.org/ project is meant to promote open and free software AL-FEC codecs. An LDPC-Staircase codec is available, which features a hybrid IT/ML decoder: a preliminary IT decoder is followed, when needed, by a GE on the linear system defined by the parity check matrix. Another codec is for Reed-Solomon using Vandermonde matrices of $GF(2^8)$. From the LDPC-Staircase codec we have forked a second codec, now commercialized by Expway (http://www.expway.com/), that we intensively optimized in terms of speed and reduced memory consumption. The speed and memory measurements of section VI have all been made thanks to this commercial codec, whereas the erasure performance measurements of section V can be made indifferently by any codecs.

A. High Performance Codec Design Techniques

We now give some hints on two key techniques that significantly improved the LDPC-Staircase codec performance.

1) Structured Gaussian Elimination (SGE): In order to achieve high decoding speeds, it is essential to use the SGE approach introduced simultaneously by LaMacchia/Odlyzko [16] and Pomerance/Smith [17]. In our context, it enables to continue using the high speed IT decoding even when no weight one equation remains. This is made possible by declaring as "inactive" some of the remaining active columns of the matrix, until one or more rows of weight one appear. IT then resumes, and once blocked again, we loop and declare some of the remaining columns "inactive". Once no active column remains, the inactivated columns are gathered and this significantly smaller subsystem is solved by classic GE. [18] is a related work that elaborates on this technique, studying several methods to chose the columns to declare inactive.

Example: Let us consider $k = 8192$, $CR = 2/3$, $N1 = 7$. At the recovery limit, with an overhead of 7 symbols (i.e. $k+7$ symbols are received and 4089 symbols randomly chosen are erased), IT decoding recovers 51 symbols (of which 49 are source symbols). The linear system now involves $4089 - 51 = 4038$ variables (even if we are only interested in source symbols). With a standard GE, this is the size of the system to invert. With SGE, the remaining sub-system to solve with GE only involves 778 variables, i.e. 5.19 times less. GE having a complexity in $O(N^3)$ with a dense system, N being the number of variables, this is a significant saving.

2) Symbol XOR Functions: The second optimization concerns the function(s) that XOR symbols together (i.e. buffers of size usually a few 100s bytes or more). Given the huge number of XOR operations performed (642000 in the above example), they play a major role in the speed performance. Most recent processors have SIMD[3] processing units that help improving these functions: e.g. NEON SIMD extensions on most ARM/Cortex processors can perform XOR operations on 16 bytes (128-bit) data chunks at a time. Also, XOR'ing the same symbol to several destination symbols at the same time (and vice-versa) can help reducing the number of read/write operations performed compared to a solution where symbols are XORed two by two.

Concerning Reed-Solomon codes, we improved decoding performances as follows.

3) Coefficient-Symbol Multiplications over GF(2^8): Reed-Solomon decoding (and encoding) involves many operations of the form: $d = d \oplus c * s$ where d is a destination symbol (typically an erased source symbol), s is the current symbol (typically a received symbol) and c a coefficient over GF(2^8) (typically a coefficient of the inverted decoding matrix). To multiply two elements, the default optimization consists in using a pre-computed table that contains the result of the multiplications of all elements pairs over GF(2^8) (total of 256^2 entries). Each byte of s is processed independently, using these tables, and XORed to the corresponding byte of d. Therefore this method requires accessing each byte of s independently, which is a costly operation.

[3] The Single Instruction Multiple Data (SIMD) is a CPU extension capable of doing the same processing on large amounts of data at the same time.

Further optimizations are possible. Let $\{a^0, a^1, ..a^7\}$ be the vector composed of the successive powers of a, a root of the primitive polynomial of GF(2^8) (see [9]). This vector is a basis for GF(2^8) and $c = \sum_{i=0}^{7} c_i * a^i$, where each c_i coordinate belongs to GF(2). The idea consists in pre-calculating the multiplication of the current s symbol by the first eight powers of a (i.e. the elements of the basis). Then to calculate $d = d \oplus c * s$, it is sufficient to XOR all the pre-calculated $a^i * s$ that correspond to non-zero c_i coordinates, using the optimized symbol XOR function described above. Since a given received symbol s needs to be multiplied by as many coefficients c of the inverted decoding matrix as there are erased source symbols, the pre-calculations are easily amortized by decoding all erased source symbols in parallel.

V. Erasure Performance Analysis

Let us now study the erasure recovery performance, by determining the overhead (i.e. the number of symbols in addition to k) needed to reach a certain decoding failure probability, Pr_{fail}. All the tests are carried out with ML decoding and we demonstrate in section VI that ML decoding is fast when correctly implemented, even on a smartphone.

A. Raw Performance Results, with Medium to Large Blocks

Table I summarizes the erasure recovery performance results for various block sizes and code rates, both in terms of average overhead (i.e. $Pr_{fail} = 0.5$) and required overhead to achieve $Pr_{fail} \leq 10^{-4}$ as required in the 3GPP-eMBMS competition[4].

Figure 2(a) illustrates these results for the case where $CR = 2/3$. We see that the performance achieved is excellent as soon as the block size is larger than a few hundreds of symbols, and the overhead quickly drops below 1% above $k = 2000$, meaning that the results are within a 1% margin of ideal, MDS codes.

Figure 2(b) is a close-up that shows $P_{fail} = f(overhead)$ when $k = 1024$ and $CR = 2/3$. The green curve is the histogram of the overhead required for decoding to succeed, whereas the red curve shows the decoding failure probability. A total of 10^6 iterations are performed in order to reach the desired precision of 10^{-4}. We only plot the curve for values $\geq k$ since decoding is otherwise impossible. We see that the curve immediately collapses, meaning that a small overhead is sufficient for decoding to succeed in a significant number of cases, and the fact there is no visible error floor above 10^{-5} confirms the excellent results[5].

B. Raw Performance Results with Small Blocks

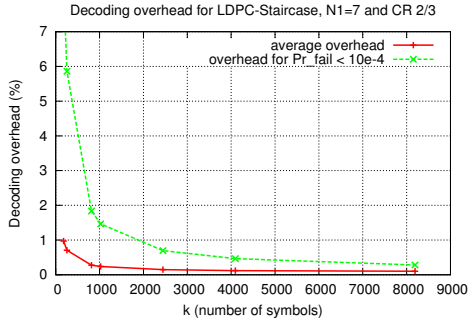
All LDPC codes are asymptotically good, and their erasure recovery performance decreases for small blocks (see figure 2(a)). The number of symbols can be increased by putting several smaller symbols per packet [4], and this is the role of the G parameter. E.g. if the packet payload size is such that by default 256 symbols should be considered, which

[4] Receiving that number of additional symbols guaranties that decoding will not fail more than one times out of 10^4.

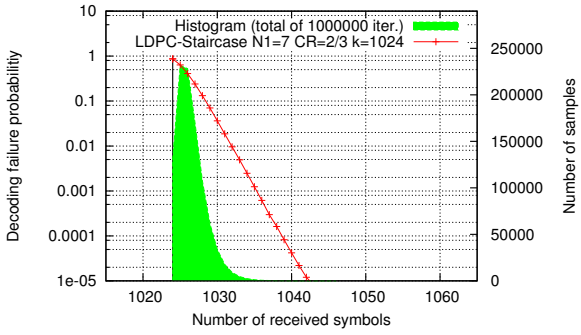
[5] An ideal, MDS, code would exhibit a failure probability equal to 1 as long as the number of received symbols is $< k$, and equal to 0 above k.

Parameters	*Average overhead (i.e.* $Pr_{fail} = 0.5$*)*	*Overhead for* $Pr_{fail} \leq 10^{-4}$
CR = 0.9		
k=180	0.902%, i.e. 1.623 add. symbols	6.667%, i.e. 12 add. symbols, $Pr_{fail} = 3.8 * 10^{-5}$
k=256	0.638%, i.e. 1.633 add. symbols	5.469%, i.e. 14 add. symbols, $Pr_{fail} = 6.2 * 10^{-5}$
k=1024	0.168%, i.e. 1.720 add. symbols	1.367%, i.e. 14 add. symbols, $Pr_{fail} = 7.9 * 10^{-5}$
k=4096	0.051%, i.e. 2.089 add. symbols	0.366%, i.e. 15 add. symbols, $Pr_{fail} = 6.4 * 10^{-5}$
k=8192	0.030%, i.e. 2.474 add. symbols	0,183%, i.e. 15 add. symbols, $Pr_{fail} = 8.4 * 10^{-5}$
CR = 2/3		
k=180	0.971%, i.e. 1.748 add. symbols	8.333%, i.e. 15 add. symbols, $Pr_{fail} = 7.6 * 10^{-5}$
k=256	0.706%, i.e. 1.807 add. symbols	5.859%, i.e. 15 add. symbols, $Pr_{fail} = 5.9 * 10^{-5}$
k=1024	0.238%, i.e. 1.026 add. symbols	1.465%, i.e. 15 add. symbols, $Pr_{fail} = 8.2 * 10^{-5}$
k=4096	0.120%, i.e. 4.915 add. symbols	0.464%, i.e. 19 add. symbols, $Pr_{fail} = 7.1 * 10^{-5}$
k=8192	0.101%, i.e. 8.258 add. symbols	0,281%, i.e. 23 add. symbols, $Pr_{fail} = 9,3 * 10^{-5}$

TABLE I. LDPC-STAIRCASE ERASURE RECOVERY PERFORMANCE, ML DECODING, N1=7 AND G=1. NOTE THAT USE-CASES OF 3GPP TESTS INVOLVING SMALL BLOCKS ARE ADDRESSED EITHER WITH REED-SOLOMON CODES OR WITH LDPC-STAIRCASE CODES WITH $G > 1$.



(a) Overhead to reach $P_{fail} < 10^{-4}$ as a function of k



(b) $P_{fail} = f(overhead)$, $k = 1024$

Fig. 2. LDPC-Staircase performance, with $CR = 2/3$, $N1 = 7$, $G = 1$.

means a 5.8% overhead is needed to reach $Pr_{fail} < 10^{-4}$ (Table I), then using by $G = 4$ times more symbols, each of them of size $1/4$ the packet payload size, this overhead is reduced to 1.4%. It is an efficient way to better use these codes, at the price of a slightly increased processing load (section VI shows decoding remains fast with such values for k).

However this technique has limits, and our 3GPP-eMBMS proposal relies on Reed-Solomon over GF(2^8) codes for small blocks (e.g. when $k \leq 170$ in case of a code rate $2/3$) and LDPC-Staircase above. Reed-Solomon being MDS, we achieve a zero overhead whenever they are used.

This is not the case for Raptor which is supposed to be used in all situations. Therefore Raptor suffers from bad recovery capabilities with very small blocks, even if values up to $G = 10$ are used to mitigate the problem.

C. LDPC-Staircase used in a Non-Systematic Way

LDPC-Staircase codes can also be used in a non systematic way, by generating a sufficiently large number of repair symbols (if $CR < 1/2$, the number of repair symbols is higher than that of source symbols) and by sending only repair symbols. Erasure recovery performance under ML decoding is in that case excellent. With $k = 1024$, $CR = 0.4821$, $N1 = 7$ and $G = 1$, an overhead of 14 symbols is sufficient to achieve $Pr_{fail} < 10^{-4}$, which is even slightly better than the same code used in a systematic way.

D. Comparison with Raptor/RaptorQ codes and 3GPP-eMBMS Performance Results

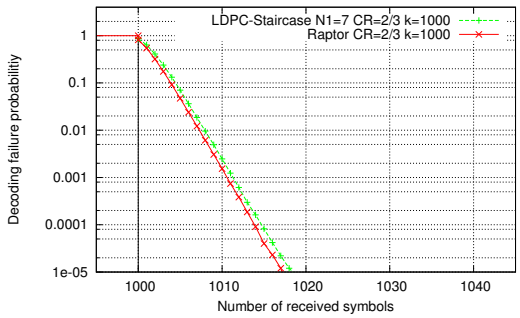


Fig. 3. LDPC-Staircase vs. Raptor perf., $k = 1024$, $CR = 2/3$ and $G = 1$.

We have compared these results with that of Raptor, i.e. a binary code that only involves `XOR` symbol operations (like LDPC-Staircase). Figure 3 shows that both codes, when used classically (i.e. $G = 1$), provide the same level of performance for $k = 1024$, $CR = 2/3$, $G = 1$. The difference (e.g. 1 additional symbol to reach the same Pr_{fail} value) is unnoticeable to the end user.

However with small blocks, Raptor is largely penalized. The artificial increase of the number of symbols by choosing a smaller symbol size and putting several symbols per packet (i.e. $G > 1$) is not sufficient. Our use of Reed-Solomon codes (section V-B) solves this issue as shown in [19].

The comparison with RaptorQ, a non-binary code, is less in favor of LDPC codes. This is caused by the use of operations in the GF(2^8) finite field into the encoding/decoding process. It adds some complexity (well managed however, the majority of symbol operations remaining `XOR` sums) but improves the erasure recovery performance by an order of magnitude.

However both RS+LDPC and RaptorQ codes are close enough of MDS codes and the difference, although real, is not significant for the end user. More precisely, the erasure recovery performance are ideal whenever Reed-Solomon codes are used, and within a 1% margin of MDS codes, when following the guidelines of section V-B. It means that the difference is not significant for the end user. Additionally, point-to-point HTTP based repair techniques of 3GPP-eMBMS services can easily recover from these erasures. Therefore it was decided during the 3GPP-eMBMS competition, after the May 2012 meeting, to base the selection on other metrics than the erasure recovery performance.

VI. Encoding/Decoding Speed and Maximum Memory Consumption

We now consider the speed and maximum memory requirements metrics. In order to be representative of 3GPP use-cases, we performed measurements on a Samsung Galaxy SII (GTI9100P) smartphone, featuring a 1.2GHz ARM Cortex-A9 CPU and running Android 2.3.4 (section VI-A) or 4.0.1 (section VI-B). Tests are carried out with Android in "performance" mode, on a single CPU core running at 1.2 GHz. Two types of experiments are done:

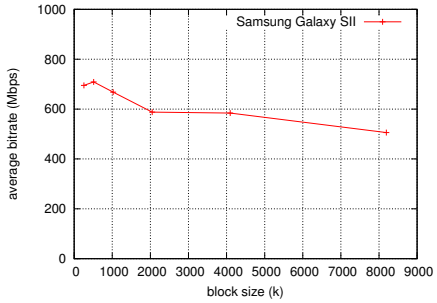
- raw tests where `eperftool` (see http://openfec.org) is used to measure speeds during Reed-Solomon and LDPC-Staircase encoding or decoding. The Raptor/RaptorQ codecs could not be tested this way;
- 3GPP-eMBMS tests, where the various codecs are integrated in FLUTE/ALC applications and globally benchmarked, in several well defined use-cases.

A. Raw Performance Results

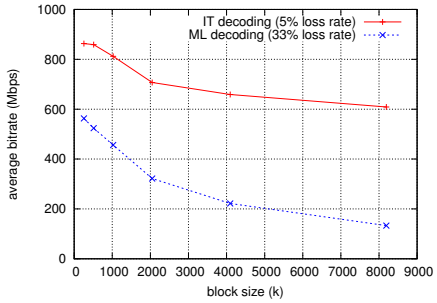
a) LDPC-Staircase: Let us consider LDPC-Staircase codes, with 1024 byte symbols, $CR = 2/3$, $N1 = 7$ and $G = 1$. Encoding speed is never an issue, as shown in figure 4(a), since a smartphone can do LDPC-Staircase encoding at speeds over 500 Mbps, no matter the block size. This can be very useful in situations where the terminal is the source of a multicast/broadcast flow.

Figure 4(b) focuses on the decoding speed, either in good reception conditions (i.e. 5% loss rate) where IT decoding is sufficient, or in bad reception conditions (33% loss rate, close to the 33.33% theoretical limit allowed by $CR = 2/3$) where ML decoding is needed. Although there is a progressive decrease of the decoding speed as k increases, we see that the worst case speed remains good with large blocks: with $k = 8192$, decoding is still performed at 134 Mbps, which is higher than the maximum reception speed over the network. With $k = 2048$, this speed amounts to 322 Mbps, a comfortable speed which also means that decoding does not consume too much CPU and battery resources.

b) Reed-Solomon: Table II shows that Reed-Solomon over GF(2^8) can be decoded at reasonable speeds (or high speed in case of very small blocks), although this is lower than the speeds achieved with LDPC-Staircase codes.



(a) Encoding speed = f(k)



(b) Decoding speed = f(k), good or bad conditions

Fig. 4. LDPC-Staircase encoding and decoding speeds on a smartphone, when $k = 1024$, $CR = 2/3$, $N1 = 7$, $G = 1$ and symbols of size 1024 bytes.

k	*CR*	*decoding speed (Mbps)*
32	0.9	509,0 Mbps
32	2/3	338,3 Mbps
170	0.9	240,7 Mbps
170	2/3	91,9 Mbps

TABLE II. Reed-Solomon decoding speed, 1024 byte symbols.

B. 3GPP-eMBMS Performance Results

c) Methodology: In the 3GPP-eMBMS context, the tests rely on the same target smartphone but follow a different methodology as explained below. The figures we report are those achieved by an independent neutral company, with the latest RS+LDPC and RaptorQ softwares available mid of January 2013 [20] (http://www.3gpp.org/ftp/tsg_sa/WG4_CODEC/TSGS4_72/Docs/S4-130061.zip). Additionally, Qualcomm provided their own results for Raptor codes (the reference) a few months earlier in [21], but did not made them available for cross-verifications. We report them as such. [6] Two types of use-cases are considered:

- **file download use-cases:** here all the FLUTE/ALC packets that are not erased during transmission (the exact loss pattern to apply, representative of LTE networks, is specified by the use-case) are first stored on the smartphone SD memory card (no decoding yet). Once all of them are available locally, the time required to read from SD card, decode and write back the result to the SD card is measured and the global throughput calculated. Note that the time considered here is the "user" plus "system" times actually used

[6] NB: we do not report here the results achieved with SuperCharged codes, the third candidate from Broadcomm, as the decoding speed and maximum memory consumption are significantly worse.

for processing operations, as reported by the ”time” Unix command, not the total elapsed time. All of this occurs block per block (potentially on a per sub-block manner in case of Raptor/RaptorQ) and it is required that decoding be successful for all the blocks since the decoded object integrity is finally checked.

- **streaming use-cases:** here the video content is split into segments of predefined duration (and size) that are transmitted as separate FLUTE/ALC objects. All the transmissions are subject to packet losses representative of LTE networks when used either by a pedestrian or a vehicle (the exact loss pattern is specified by the use-case). The FLUTE/ALC application of the smartphone is used to receive and process each packet and to launch AL-FEC decoding when appropriate. The ”user” plus ”system” times (not the total elapsed time) are measured and the decoding speed is calculated.

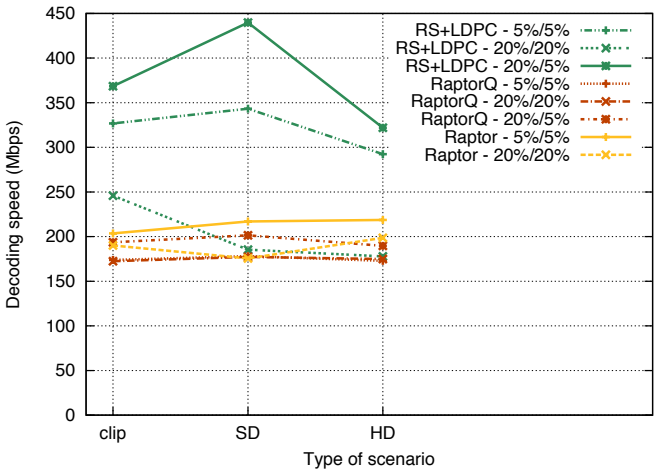
Most scenarios are in ”bad reception conditions”, with a loss ratio close to the theoretical limit (said differently the minimum number of repair packets is sent in order to recover from all the erased packets). This is the case of ”L/R = 20%/20%” situations (e.g. in table III), where the loss ratio is 20% and the number of repair packets is just a little more than 20% of k (the exact code rate may differ a little bit).

However some scenarios are in ”good reception conditions”, with a number of repair packets that is large compared to the number of erased packets. This is the case of ”L/R = 20%/5%” situations (e.g. in table III), where the number of repair packets is slightly more than 20% of k but the loss ratio is only 5%. This is a common situation in practice since the code rate used by FLUTE/ALC is set to a fixed value that should enable the vast majority of receivers to recover from erasures.

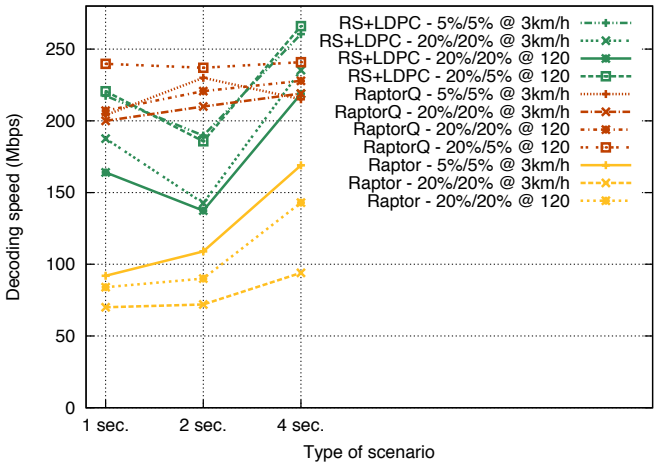
d) Results: The results of table III / figure 5(a) show that our RS+LDPC solution (here LDPC-Staircase codes) is faster than Raptor and RaptorQ in almost all download use-cases, often significantly faster. Only in one use-case (HD - 20%/20%) is our solution slightly slower than Raptor, but still slightly faster than RaptorQ. The maximum memory requirements during decoding operations show a small benefit for RS+LDPC for the clip and SD use-cases, and a slight advantage for Raptor/RaptorQ for the HD use-cases. In any case the difference is not significant to provide a clear advantage of one solution over the other.

The results are more balanced with streaming tests (table IV / figure 5(b)), in particular because several of these scenarios rely on Reed-Solomon. For instance if RS+LDPC and RaptorQ performance are approximately equivalent in case of the 1 sec. and 4 sec. use-cases, on the opposite RaptorQ is faster than our solution for the 2 sec. use-cases. However in all cases RS+LDPC is always significantly faster than Raptor codes. The results in terms of decoding latency are similarly balanced. Our solution sometimes features the lowest latency, otherwise RaptorQ is better. In our opinion there is no clear advantage of one solution over the other.

e) Additional Criteria: Several additional criteria exist. We have shown that the RS+LDPC-Staircase combination is a very good choice, with many advantages:



(a) Download use-cases



(b) Streaming use-cases

Fig. 5. Decoding speed for 3GPP-eMBMS download/streaming use-cases.

- the true openness of the solution: open, free codecs are provided in http://openfec.org;
- the simplicity of the codes;
- last but not least, there are also situations (i.e. when $G = 1$) where our approach is, from our point of view, IPR free (e.g. prior arts exist on structured LDPC codes) which can be an advantage in particular for non commercial, open-source projects.

LDPC-Staircase codes are predefined rate codes, i.e. the code rate is fixed upon code creation: they are not rate-less codes unlike Raptor/RaptorQ codes. However many common use-cases do not rely on such a feature, which is confirmed by the fact that all the 3GPP-eMBMS call for technology did not involve tests requiring a code rate lower than $CR = 2/3$[1].

VII. Conclusions and Discussions

In this paper we introduce the Reed-Solomon + LDPC-Staircase AL-FEC codes that are well suited to file download and streaming applications. They both have been standardized

	Raptor			RaptorQ			RS+LDPC		
	clip	*SD*	*HD*	*clip*	*SD*	*HD*	*clip*	*SD*	*HD*
Decoding speed (Mbps)									
bad channel: L/R=5%/5%	203.6	216.9	218.6	173.8	178.1	172.9	326.8	343.3	292.4
bad channel: L/R=20%/20%	190.1	175.5	198.6	172.4	177.3	174.9	245.9	185.4	177.8
good channel: L/R=20%/5%	Not avail.	Not avail.	Not avail.	193.6	201.4	189.6	368.5	439.7	322.0
Maximum memory requirements (MB)									
all channels	6.8 to 6.9	21.2 to 21.3	21.9 to 22.2	6.7 to 7.0	21.0 to 21.9	22.4 to 23.4	4.8 to 6.2	13.8 to 18.6	25.3 to 30.3

TABLE III. DECODING SPEED AND MAXIMUM MEMORY REQUIREMENTS FOR 3GPP-EMBMS DOWNLOAD USE-CASES (SOURCES: [20], [21]).

	Raptor			RaptorQ			RS+LDPC		
	1 sec.	*2 sec.*	*4 sec.*	*1 sec.*	*2 sec.*	*4 sec.*	*1 sec.*	*2 sec.*	*4 sec.*
Decoding speed (Mbps)									
bad channel: L/R=5%/5%, 3km/h	92.6	109.5	169.1	204.2	230.2	215.5	217.4	189.7	260.4
bad channel: L/R=20%/20%, 3km/h	70.6	72.3	94.5	200.4	210.9	219.0	187.6	142.6	235.3
bad channel: L/R=20%/20%, 120km/h	84.3	90.4	143.4	207.1	220.7	227.9	164.0	137.4	185.7
good channel: L/R=20%/5%, 120km/h	Not Avail.	Not Avail.	Not Avail.	239.7	237.0	240.9	220.6	185.7	266.0
Decoding latency (sec.)									
all channels	Not avail.	Not avail.	Not avail.	1.8 to 4.3	5.0 to 7.9	11.7 to 17.4	2.0 to 4.0	7.4 to 10.4	10.6 to 14.2

TABLE IV. DECODING SPEED AND LATENCY FOR 3GPP-EMBMS STREAMING USE-CASES (SOURCES: [20], [21]).

by IETF, LDPC-Staircase codes are part of the ISDB-Tmm Japanese video push standard, and our proposal based on these codes has been ranked first (1 more vote than the RaptorQ proposal, 31 vs. 30) after more than one year of work in the context of the 3GPP-eMBMS competition. It was also recognized that the RS+LDPC proposal offers several benefits over Raptor codes for the 3GPP-eMBMS use-cases.

In this paper we detail the code internals, some techniques we used to design high speed decoders, and provide extensive performance analyses. We show that these codes, in spite of their simplicity, achieve excellent results. The RS+LDPC-Staircase combination is therefore a very good choice, with many advantages in terms of openness, availability of free codecs, simplicity of the specifications, excellent erasure recovery performance that are either ideal or in a 1% margin of ideal codes, and very high decoding speeds, even on smartphones. There are also situations where our approach is, from our point of view, IPR free, which can be an advantage in some situations.

REFERENCES

[1] ETSI, "Evaluation of mbms fec enhancements (final report)," Apr. 2013, 3GPP TR 26.947 version 11.0.0 Release 11. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/26947.htm

[2] T. Paila, R. Walsh, M. Luby, V. Roca, and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport," Nov. 2012, IETF Request for Comments, RFC 6726 (Standards Track) (obsoletes RFC 3926). [Online]. Available: http://www.ietf.org/rfc/rfc6726.txt

[3] D. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, ISBN: 0521642981, 2003.

[4] V. Roca, C. Neumann, and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) schemes," Jun. 2008, IETF Request for Comments, RFC 5170 (Standards Track). [Online]. Available: http://www.ietf.org/rfc/rfc5170.txt

[5] V. V. Zyablov and M. S. Pinsker, "Decoding complexity of low-density codes for transmission in a channel with erasures," *Probl. Peredachi Inf.*, vol. 48, pp. 18–28, 1974.

[6] M. Cunche and V. Roca, "Optimizing the error recovery capabilities of LDPC-staircase codes featuring a Gaussian Elimination decoding scheme," in *10th IEEE International Workshop on Signal Processing for Space Communications (SPSC'08), Rhodes Island, Greece*, Oct. 2008.

[7] E. Paolini, M. Varrella, M. Chiani, B. Matuz, and G. Liva, "Low-complexity ldpc codes with near-optimum performance over the bec," in *Advanced Satellite Mobile Systems, 2008. ASMS 2008. 4th*, aug. 2008.

[8] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[9] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo, *Reed-Solomon Forward Error Correction (FEC) Schemes*, Apr. 2009, IETF Request for Comments, RFC 5510 (Standards Track). [Online]. Available: http://www.ietf.org/rfc/rfc5510.txt

[10] A. Shokrollahi and M. Luby, "Raptor codes," *Foundations and Trends in Communications and Information Theory*, vol. 6, no. 3-4, pp. 213–322, May 2011.

[11] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery," Oct. 2007, IETF Request for Comments, RFC 5053 (Standards Track). [Online]. Available: http://www.ietf.org/rfc/rfc5053.txt

[12] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery," IETF Request for Comments, RFC 6330 (Standards Track), Aug. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6330.txt

[13] V. Roca, M. Cunche, and J. Lacan, "Simple Low-Density Parity Check (LDPC) Staircase Forward Error Correction (FEC) Scheme for FECFRAME," Dec. 2012, IETF Request for Comments, RFC 6816 (Standards Track). [Online]. Available: http://www.ietf.org/rfc/rfc6816.txt

[14] V. Roca, M. Cunche, J. Lacan, A. Bouabdallah, and K. Matsuzono, *Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME*, Feb. 2013, IETF Request for Comments, RFC 6865 (Standards Track). [Online]. Available: http://www.ietf.org/rfc/rfc6865.txt

[15] A. Yamada, H. Matsuoka, T. Ohya, R. Kitahara, J. Hagiwara, and T. Morizumi, in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB'11)*, Jun. 2011.

[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.

[18] E. Paolini, B. Matuz, G. Liva, and M. Chiani, "Pivoting algorithms for maximum likelihood decoding of LDPC codes over erasure channels," *IEEE Global Telecommunications Conference (GLOBECOM'09)*, 2009.

[19] C. Thiennot, C. Seyrat, V. Roca, and J. Detchart, "Proposal for a candidate for emm-efec work item," May 2012, Expway, Document S4-120731, 3GPP TSG-SA4 meeting 69, Erlangen, Germany.

[20] "Performance verification results for MBMS EFEC candidates," Jan. 2013, Huawei Technologies France, Document S4-130061, 3GPP TSG-SA4 meeting 72, Valencia, Spain. [Online]. Available: http://www.3gpp.org/ftp/tsg_sa/WG4_CODEC/TSGS4_72/Docs/S4-130061.zip

[21] "Emm-efec, selection of the fec," Nov. 2012, document S4-121367, 3GPP TSG-SA4 meeting 71, Bratislava, Slovakia. [Online]. Available: http://www.3gpp.org/ftp/tsg_sa/WG4_CODEC/TSGS4_71/Docs/S4-121367.zip