Signal Processing: *Image Communication* 24 (2009) 229–242

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

Flexible forward error correction codes with application to partial media data recovery

Jari Korhonen [a,*,1], Pascal Frossard [b]

[a] *Norwegian University of Science and Technology (NTNU), Centre for Quantifiable Quality of Service in Communication Systems (Q2S), O.S. Bragstads plass 2E, NO-7491 Trondheim, Norway*[2]

[b] *Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland*

ARTICLE INFO

Article history:
Received 21 April 2008
Received in revised form
18 November 2008
Accepted 30 December 2008

Keywords:
Forward error correction
Multimedia communications
Video streaming

ABSTRACT

Conventionally, linear block codes designed for packet erasure correction are targeted to recover all the lost source packets per block, when the fraction of lost data is smaller than the redundancy overhead. However, these codes fail to recover any lost packets, if the number of erasures just exceeds the limit for full recovery capability, while it can still be beneficial to recover part of the symbols. In addition, common linear block codes are not well suited for unequal error protection, since different block codes with different rates must be allocated for each priority class separately. These two problems motivate the design of more flexible *forward error correction* (FEC) codes for media streaming applications. We first review the performance of short and long linear block codes. Long block codes generally offer better error correction capabilities, but at the price of higher complexity and larger coding delay. Short block codes can be more appropriate in media streaming applications that require smooth performance degradation when the channel loss rate increases. We study a new class of linear block codes using sparse generator matrices that permit to optimize the performance of short block codes for partial recovery of the lost packets. In addition, the proposed codes are extended to the design of unequal erasure protection solutions. Simulations of practical video streaming scenarios demonstrate that the flexible sparse codes offer a promising solution with interesting error correction capabilities and small variance in the residual loss rate. They typically represent an effective trade-off between short block codes with limited flexibility, and long block codes with delay penalties.

1. Introduction

Recovery from packet losses is an important problem in packet-switched networking. Traditionally, it has been solved by retransmitting the lost packets (*automatic repeat request* (ARQ)) or transmitting redundant data that can be used to regenerate lost data blocks (*forward error correction* (FEC)). In multimedia communications, such as video telephony or streaming, real-time requirements and limited link capacity often discourage the use of retransmissions. In this case, an error-free transmission cannot be guaranteed, and the application has to mask to effect the potential losses by regenerating the blocks of data that have been lost using different interpolation mechanisms. This approach is often referred as *error concealment*, since it aims to conceal the impact of errors instead of actually recovering fully the errors [1].

* Corresponding author. Tel.: +47 73592723; fax: +47 73592790.
E-mail addresses: jari.korhonen@q2s.ntnu.no (J. Korhonen), pascal.frossard@epfl.ch (P. Frossard).

[1] Part of the work for this paper has been done at Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland. Part of this work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

[2] "Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence" appointed by the Research Council of Norway, funded by the Research Council, NTNU and UNINETT.

0923-5965/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2008.12.005

In order to facilitate error concealment, FEC can be used to reduce the effective rate of lost data packets. A typical FEC coder is a block coder that takes a block of k source symbols as input and generates n FEC symbols as output ($n>k$). The code rate R of the code is defined as $R=k/n$. A data packet can be typically considered as a source symbol, or advanced techniques can use complex partitioning and packetization of data. According to the conventional wisdom, a good FEC code should be designed so that the redundancy overhead required to recover all the missing source symbols is minimized. Therefore, in an ideal case, FEC decoder can regenerate the original k symbols when any k of them out of n has been received. In other words, a random loss of a fraction p of the transmitted symbols can be recovered if $p\leqslant 1-R$. This can be achieved with *minimum distance separable* (MDS) codes, such as Reed–Solomon (RS) codes [2]. Unfortunately, if there are more than $n-k$ symbols erased, none of the lost source symbols can be recovered. Multimedia applications can however generally tolerate a small number of packet losses. It would be better to recover at least some of the lost data packets, even if it is not possible to recover all of them.

FEC schemes can be used to blindly recover lost data packets independently on the content they carry. Alternatively, the FEC protection can be computed in order to prioritize classes of packets with *joint source-channel coding* (JSCC) [3], where the protection can be adjusted to the channel conditions and to the content of the packets. Multimedia data for example typically contain components with different perceptual significance and benefit greatly from differential protection or *unequal erasure/error protection* (UEP). When conventional MDS codes are used for UEP, separate block codes are needed for data symbols belonging to different priority classes. This may cause serious problems with timing and scheduling, especially with strict delay constraints. In this case, long blocks are required to achieve distinctive protection levels. There is often a mismatch between the priority levels defined by the source encoder, and the flexibility of the UEP solutions based on conventional MDS codes.

In order to overcome the limitations of the conventional channel codes, we present FEC codes that are designed for partial recovery of lost source symbols and we study their performance in media streaming applications. In particular, we focus on Reed–Solomon type of codes with modified generator matrices that offer more flexibility in the FEC design. With this approach, some of the lost source symbols may be recovered even in the cases where MDS codes fail to recover any erasure. In addition, the proposed codes can easily be adjusted for efficient UEP with relatively short source blocks. These features are essential to overcome the abovementioned weaknesses of the conventional MDS codes, especially in delay-sensitive applications like media streaming scenarios, where very long block codes cannot be afforded.

The rest of this paper is organized as follows. In Section 2, the background and relevant related work is discussed. In Section 3, different FEC designs are proposed and studied analytically. In Section 4, the practical benefits of the proposed codes are demonstrated by analyzing their performance in realistic video streaming scenarios. Finally, the conclusions are given in Section 5.

2. Background and related work

There are several different alternatives to implement generic FEC codes that are usually driven by the characteristics of the target application. In this paper, we focus on systematic linear block codes for erasure channels, where each FEC symbol is a linear combination of source symbols. The simplest possible linear block code is generated by applying the binary exclusive OR (XOR) operation across the binary source symbols. If one of the source symbols is lost, it can be recovered by applying XOR operation to the FEC symbol and the remaining source symbols. Therefore, this kind of code is basically an MDS code, where $n=k+1$. Despite the limited range of code rates, XOR-based FEC is popular in practical applications, since it is easy to implement and the computational complexity is very low. RFC 2733 [4], for example, lists several ways of using XOR-based FEC in practical applications.

MDS codes, such as RS codes, with more flexible range of code rates can be implemented by using Galois field arithmetic to compute the linear combinations of source symbols. Traditionally, the high computational complexity is considered as a major weakness of MDS codes, making them impractical for long coding blocks. This is why suboptimal long block codes have gained serious attention during the past years. *Low density parity check* (LDPC) codes are linear block codes where each FEC symbol is generated by summing up different subsets of source symbols. The number of source symbols in a subset is the *degree* of the FEC symbol [5,6]. LDPC codes use binary XOR operations for parity checking, especially for erasure correction coding, but LDPC codes using higher order Galois fields have also been studied and reported to improve the bit error correction performance in a noisy channel [7,8]. Several more advanced variants of LDPC codes have also been proposed in the recent years. These codes have carefully designed degree distributions and generation algorithms. For example, Raptor codes generate the FEC symbols in several phases. The error recovery capability of Raptor codes approaches MDS codes asymptotically, when the source block length increases [9].

The *all-or-nothing* error correction capability that is characteristic for MDS codes and most of the other types of erasure correction codes is, however, not always desirable. To overcome this problem, a different approach has been taken recently with growth codes [10,11], which use degree distributions that intend to maximize the number of recovered symbols when $p\geqslant 1-R$. Therefore, the growth codes are capable of recovering some data in cases where traditional MDS or LDPC codes fail. It has been shown that this is a useful feature for applications such as sensor networks in emergency scenarios [10] and unequal error protection in video streaming [11]. Unfortunately, rather long source blocks are required for growth codes to achieve the desired performance (e.g., a

source block of 200 packets is used in [11]). This is one of the reasons that explain why growth codes are not ideal for time-critical multimedia communications. Furthermore, similar average performance can actually be achieved by conventional MDS codes with short blocks.

The *Partial Reed–Solomon* (PRS) codes proposed in [12] target similar purposes as growth codes for the partial recovery of source symbols even under high loss rates. With PRS codes, only part of the source data is protected by RS codes and the remaining part is left unprotected. The average recovery rate can thus be improved when the loss rate occasionally exceeds $1-R$. However, PRS codes do not treat the source data equally, and the unprotected source data suffers from higher loss probability than the protected data. Such a binary classification of source data between protected and unprotected information is certainly a limitation in practical streaming situations, where a finer adaptation might be required. It is possible to extend the PRS codes for more flexible protection of source data. For example, PRS can be applied hierarchically in several steps in order to define different protection levels. However, this approach is conceptually equivalent to using separate block codes for source data, with different priority levels. Mismatch between the fixed FEC code rate and the corresponding rate for generated data elements for different priority classes causes serious timing problems easily. This could be fixed by using adaptive FEC code rates for each priority class, but this would lead easily to complicated system design and undesirably long FEC blocks. For example, several UEP mechanisms rely on linear block codes and adapt the code rate to the data sections with different significance. For example, *Priority encoding transmission* (PET) divides payload of each packet in sections of different priorities and applies conventional MDS FEC coding hierarchically, so that stronger protection (lower code rate) is applied to the higher priority sections [13]. However, neither growth codes nor PRS can solve the problem of partial data recovery satisfactorily with solutions that are generic enough for the most relevant streaming application scenarios.

This paper studies a new class of short block codes based on RS coding that have been proposed recently in order to optimize the performance for UEP [14,15]. In traditional RS codes, each FEC symbol is a linear combination of all the source symbols in the coding block. However, to achieve UEP, it is possible to define the code such that some or all FEC symbols cover only a subset of source symbols. The idea has been proposed in [14], where FEC symbols are divided in several types. All FEC symbols cover the highest priority source symbols, but lower priority source symbols are covered only by a fraction of FEC symbols, respectively, to the relative priority. In practice, these codes can be implemented by using sparse RS generator matrices, where some of the coefficients are set to zero. Similar codes with more complicated sparse generator matrix structures have been studied in [15]. This type of codes are of particular interest in this paper.

3. Sparse FEC codes

3.1. Preliminaries

We focus now on a generic FEC code design based on RS codes, with a modified generator matrix structure, in order to overcome the *all-or-nothing* recovery characteristics of conventional codes and their lack of flexibility in UEP design. We focus on relatively short block codes, as we target media streaming applications with delay constraints. The use of MDS codes is thus reasonable in terms of computational complexity. We model the channel as a packet erasure channel with a uniform i.i.d loss process. The packets are the basic units for the block-based FEC design. We do not specifically consider the case of bursty loss processes, as short packet loss bursts can be dispersed with simple interleaving in the coding scheme. For example, it is possible to alleviate the potentially harmful effect of adjacent packet losses by concatenating several interleaved short blocks into longer blocks, as shown in Fig. 1. We do not specifically consider long bursts of lost packets either, as these ones cannot be recovered with the short coding blocks considered in this paper.

We analyze below the impact of the block length in the performance of FEC algorithms, and then we propose a sparse FEC coding scheme for partial data recovery. We eventually extend the sparse FEC scheme to unequal error protection.

3.2. Impact of block length

Even if the average packet loss rate remains rather stable when computed over a long sequence of packets, the number of lost packets per short sequence of packets can vary enormously. This is why the residual packet loss rate varies more also when the FEC block gets shorter. As the block length is an important parameter in the FEC scheme proposed in this paper, we illustrate now in more details the effect of the block length on decoding performances. Assuming that the probability of packet loss is p and each packet loss is an independent event (Bernoulli process), the probability of losing exactly l packets in a block of n packets follows a binomial

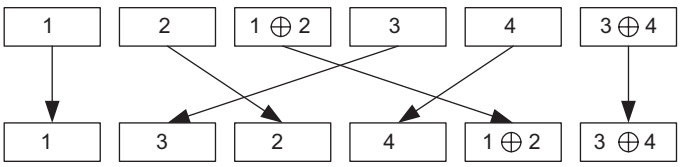

Fig. 1. Two (3, 2) codes interleaved to form a (6, 4) code.

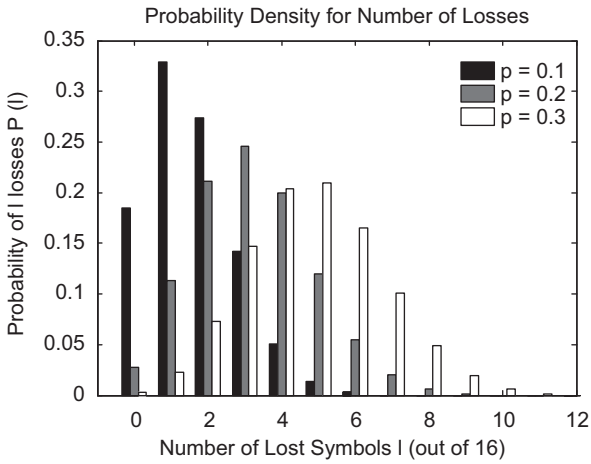

Fig. 2. Probability density of different number of lost symbols, when symbol loss probability $p = \{0.1,\ 0.2,\ 0.3\}$ and block length $n = 16$ symbols.

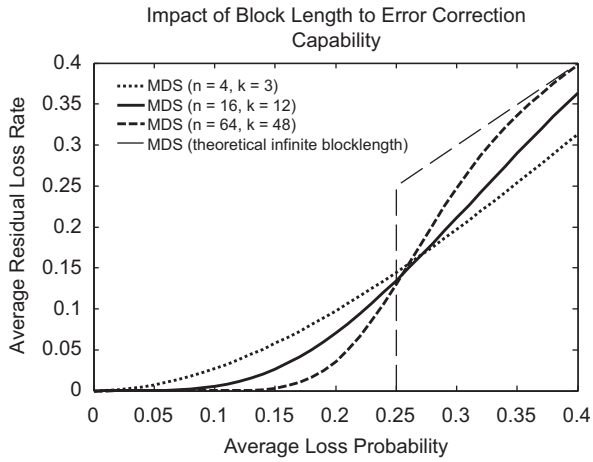

Fig. 3. Residual loss rate for MDS codes with code rate $\frac{3}{4}$, for different block lengths.

distribution, as expressed in Eq. (1) [16].

$$\Pr(N_{LOST} = l) = \binom{n}{l} p^l (1-p)^{n-l} \tag{1}$$

As an example, Fig. 2 shows the discrete probability density function for different number of packet losses within a block of 16 packets, derived from Eq. (1) with three different values of p. Assuming further that the block consists of k source packets and $n-k$ MDS FEC packets, the probability that l is smaller or equal to $n-k$, and that all lost packets can, therefore, be recovered, can be derived from Eq. (2). In this case, the average residual loss rate PLR_{RES} is given by Eq. (3). It is worth noting that even though the bursty error model is omitted in this work, it would be straightforward to extend our analysis to cover a two-state Markov packet erasure model with the approaches taken in [16,17].

$$\Pr(N_{LOST} > n-k) = \sum_{l=n-k+1}^{n} \binom{n}{l} p^l (1-p)^{n-l} \tag{2}$$

$$PLR_{RES}(p) = \frac{1}{n} \sum_{l=n-k+1}^{n} l \binom{n}{l} p^l (1-p)^{n-l} \tag{3}$$

Fig. 3 illustrates the impact of block length for the probability of full recovery of the packets. In this example, the code rate R is always $\frac{3}{4}$ and the average residual loss rate is derived from Eq. (3). The shortest possible MDS FEC code with this code rate is (4, 3) block code, where three source packets are XOR'ed to generate one redundant FEC packet. The same code rate is achieved with (16, 12) and (64, 48) RS codes. As shown in the illustration, longer blocks provide better error correction probabilities when the packet loss rate is low. In contrast, short MDS codes fail occasionally even if the packet loss rate is much lower than R, and this motivates the use of long source blocks for improved recovery performance. The code length is, however, limited by delay and complexity constraints in practical applications.

Smoother recovery performance can generally be achieved by using short blocks. As explained above, some applications benefit from partial recovery of data instead of losing or recovering everything. The choice of different block lengths becomes essentially a compromise between error correction capabilities at low and high packet loss rates. Note that it is also possible to achieve graceful packet loss performance for long blocks by using LDPC-based codes with appropriate degree distribution, as proposed in [12]. However, the benefit of LDPC codes compared against MDS codes lays particularly in their lower computational complexity, which is a major issue only when long blocks are concerned. This is why there is no real motivation to use LDPC-based codes in situations where the desired performance can be achieved with short MDS codes.

The variance of the loss process after decoding is another important parameter in the design of channel coding solutions. In general, clustered losses result in smaller perceived quality than losses that are uniformly distributed with same average loss rate when the error concealment is efficient. For example, it has been shown that users are particularly sensitive to quality fluctuation in video streams [18]. Interleaving [19] is a manner to break bursts of losses, but long interleaving cycles lead to increased latency, which is not desirable in applications with strict timing constraints. For improved quality, it becomes, therefore, advantageous to design FEC codes that minimize the variance of the residual fraction of lost packets per block. The variance of the fraction of unrecoverable packets per block can be computed using Eq. (4). The equation has been derived straightforwardly from the definition of variance in a discrete case, $VAR(X) = \sum p_i (x_i - \mu)^2$, where p_i is the probability of the random variable X to take value x_i, and μ is the mean of X. In our equation, p_i is replaced by the probability of losing i symbols, $\Pr(N_{LOST} = i)$, μ by the mean residual loss rate PLR_{RES}, and x_i by the residual fraction of lost symbols in the particular case when i symbols are lost, that is l/n

when $l > n-k$, and 0 when $l \leqslant n-k$.

$$VAR_{RES\_n} = \sum_{i=0}^{n-k} \Pr(N_{LOST} = i) PLR_{RES}^2 + \sum_{i=n-k+1}^{n} \Pr(N_{LOST} = i) \times (i/n - PLR_{RES})^2 \quad (4)$$

The variance of the absolute number of residual lost packets is achieved by multiplying VAR_{RES} by n^2. Since the variance of a sum of random variables is the sum of their variances, the variance for a short blocks of length n considered as a long block of length $a \cdot n$ can be computed using Eq. (5).

$$VAR_{RES\_an} = \frac{an^2 VAR_{RES\_n}}{(an)^2} = \frac{1}{a} VAR_{RES\_n} \quad (5)$$

The variances of the residual packet loss rates within a block of 16 packets with one MDS (16, 12) code and four MDS (4, 3) codes together are shown in Fig. 4. Short block codes generally lead to a lower variance than long blocks, even at low packet loss rate. Together with the possibility to offer a lower residual loss rate and a smaller computational complexity, this provides an additional motivation for the use of short FEC blocks when perfect data recovery is not mandatory and the channel loss rate is variable.

3.3. FEC for short blocks

As discussed above, streaming applications may often benefit from an error correcting code whose performance represents a compromise between those of short and long MDS codes. Typically, the FEC algorithm should introduce low decoding delay, offer good recovery performance with small variance in the residual number of lost packets, and at the same time provide the possibility for partial recovery of the data, even if the channel loss rate is high. In the following, we focus on sparse RS codes optimized for partial loss recovery. The benefits of these codes can be achieved with relatively short source blocks, which is especially useful for time-critical streaming applications. We give here a detailed description of the sparse FEC codes first proposed in [14,15]. Performance analysis is also presented. In the following section, we extend our study further to UEP.

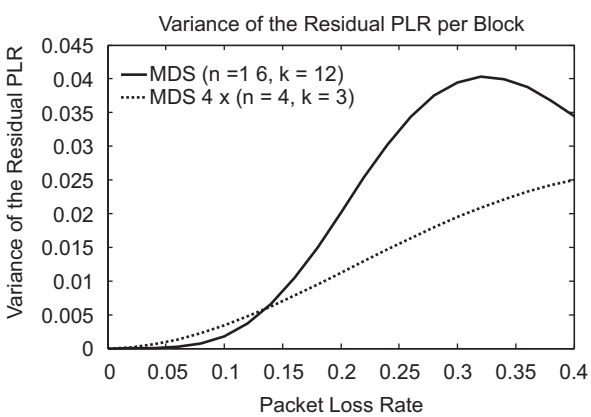

Fig. 4. Variance of the residual loss rate per block when two different FEC codes are used, with blocks of 16 packets.

An RS code can be defined as a system of linear equations, computed using Galois field arithmetic. Given source data vector $\boldsymbol{D} = \{d_j, j = 1 \ldots k\}$ and FEC code vector $\boldsymbol{C} = \{c_i,\ i = 1 \ldots n\text{–}k\}$, each FEC symbol can be computed from Eq. (6), where $\alpha_{i,j}$ is a coding coefficient in generator matrix $\boldsymbol{G}$. It is easy to see that when $\alpha_{i,j} = j^{i-1}$, rows in $\boldsymbol{G}$ are orthogonal and missing source symbols can be solved from the system of equations, assuming that there are no more than $n-k$ source and FEC symbols missing. Readers who are interested in more detailed description about RS codes may refer to relevant literature, such as [2].

$$c_i = \sum_{j=1}^{k} \alpha_{i,j} d_j \quad (6)$$

To achieve smoother performance in the region of $n-k$ packet losses per block, a sparse RS generator matrix structure has been proposed, where some of the coding coefficients in $\boldsymbol{G}$ are dropped and replaced by zeros. Obviously, the modified code is not any more capable of perfect recovery of data in all the cases where conventional RS is successful. However, partial recovery of data becomes possible with high probability even in the situation, where slightly more than n–k packets are missing. The improved recovery capability for loss rates close to $(n-k)/n$, however, reduces the loss recovery probability when the loss rate is small. Nevertheless, the average performance may still be improved in media streaming applications. We can note here that the design of the sparse FEC code has some similarities with the design of priority random linear codes used in network coding [20].

A few conditions have to be satisfied in the design of the sparse Reed–Solomon generator matrices. First of all, non-zero coefficients at each row shall partially overlap

a

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & \alpha_{1,6} & \alpha_{1,7} & \alpha_{1,8} & \alpha_{1,9} & \alpha_{1,10} & \alpha_{1,11} & \alpha_{1,12} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & \alpha_{2,6} & \alpha_{2,7} & \alpha_{2,8} & \alpha_{2,9} & \alpha_{2,10} & \alpha_{2,11} & \alpha_{2,12} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & \alpha_{3,6} & \alpha_{3,7} & \alpha_{3,8} & \alpha_{3,9} & \alpha_{3,10} & \alpha_{3,11} & \alpha_{3,12} \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \alpha_{4,4} & \alpha_{4,5} & \alpha_{4,6} & \alpha_{4,7} & \alpha_{4,8} & \alpha_{4,9} & \alpha_{4,10} & \alpha_{4,11} & \alpha_{4,12} \end{bmatrix}$$

b

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & \alpha_{1,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & \alpha_{2,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,7} & \alpha_{3,8} & \alpha_{3,9} & \alpha_{3,10} & \alpha_{3,11} & \alpha_{3,12} \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{4,7} & \alpha_{4,8} & \alpha_{4,9} & \alpha_{4,10} & \alpha_{4,11} & \alpha_{4,12} \end{bmatrix}$$

c

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & \alpha_{1,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & 0 & 0 & 0 & \alpha_{2,7} & \alpha_{2,8} & \alpha_{2,9} & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_{3,4} & \alpha_{3,5} & \alpha_{3,6} & 0 & 0 & 0 & \alpha_{3,10} & \alpha_{3,11} & \alpha_{3,12} \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{4,7} & \alpha_{4,8} & \alpha_{4,9} & \alpha_{4,10} & \alpha_{4,11} & \alpha_{4,12} \end{bmatrix}$$

Fig. 5. Examples of different generator matrices. (a) Original Reed–Solomon code generator matrix, (b) sparse Reed–Solomon generator matrix that actually generates two conventional RS codes and (c) a properly generated sparse generator matrix.

with non-zero coefficients at some other rows. Otherwise, the design would actually lead to a system of independent conventional RS codes with shorter source block lengths. Second, every row and column should have the same number of non-zero coefficients in order to achieve equal protection level for each source symbol. These constraints restrict efficiently the number of potential coefficient combinations and generator matrix dimensions. A few generator matrices are illustrated in Fig. 5. A conventional Reed–Solomon generator matrix for a (16, 12) RS code is shown in Fig. 5a, and a matrix that generates two (8, 6) RS codes in Fig. 5b. A sparse generator matrix that respects the abovementioned design constraints is finally shown in Fig. 5c.

Since the proposed code is intended for relatively short blocks and the number of reasonable generator matrix patterns is limited due to the abovementioned constraints, it is possible to analyze the performance that results from different sparse coefficient patterns for each of the code rates. For example, for a sparse (16, 12) code only two levels of sparseness are reasonable: 25% of zeros (containing 9 zeros per row and 3 zeros per column) and 50% of zeros (6 zeros per row and 2 zeros per column). In the first case, the 9 non-zero coefficients on each row must overlap with exactly 6 non-zero coefficients on every other row. This kind of structure is shown in Fig. 6a. With 50% of zeros, there are five reasonable alternatives: the 6 non-zeros are overlapping with exactly 2 non-zeros on every other row (2-2-2 overlap, Fig. 6b), 3 non-zeros on two other rows and none on one other row (3-3-0 overlap, Fig. 6c), or four non-zeros on one row, two on another and none on the last other row (4-2-0 overlap, Fig. 6d). Overlap pattern 5-1-0 would also be possible, although not shown in Fig. 6. Overlap of (6-0-0) would equal to two independent RS codes, as shown in Fig. 5b above. Finally, we can observe that the requirement for partially overlapping non-zeros on different rows can, however, not be achieved with 75% of zeros (3 zeros per row and 1 zero per column), unless some columns are left without non-zeros coefficients.

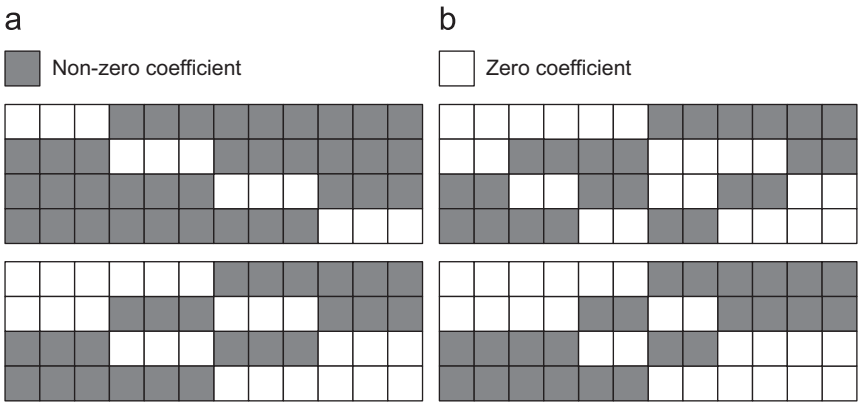

Fig. 6. Binary representations of different possible designs for sparse (16, 12) RS codes. White squares denote zero coefficients, dark squares normal RS generator matrix coefficients ($\alpha_{i,j} = j^{i-1}$). (a) 25% zeros, (b) 50% zeros, 2-2-2 overlap, (c) 50% zeros, 3-3-0 overlap (d) 50% zeros, 4-2-0 overlap.

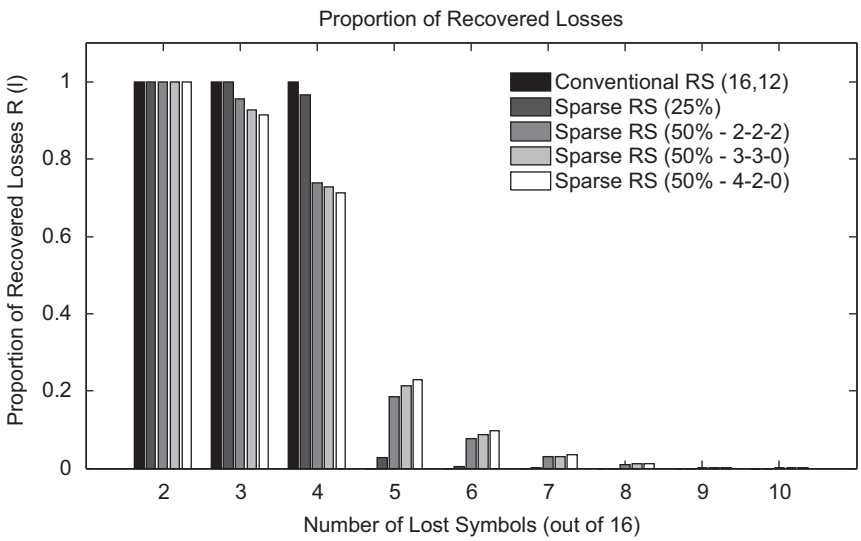

Fig. 7. Error recovery rate of sparse FEC codes versus the channel loss rate.

A solution to a linear system of equations can be found, if there are as many (or less) unknown variables as there are equations, where the variables are included. Let us define a function $C(i, l)$, that yields the proportional amount of combinations (out of all possible combinations) of l lost symbols (out of n), but for which the decoder can recover exactly i symbols. Then, the average fraction of recovered symbols $R(l)$ can be computed from Eq. (7)

$$R(l) = \frac{1}{l} \sum_{i=1}^{\min(l,n-k)} iC(i,l) \tag{7}$$

For conventional RS codes, the analysis is simple: $C(i, l) = 1$ when $l \leqslant n-k$ and $i = l$, and $C(i, l) = 0$ otherwise. With sparse RS codes, the function $C(i, l)$ depends on the sparseness level and the distribution of the non-zero coefficients. In most cases, it cannot be given in a short closed form, but rather requires iterative representations. In practice, $C(i, l)$ must be solved by exhausting all combinations. For example, four lost symbols ($l = 4$) can be selected among 16 symbols ($n = 16$) in 1820 different ways. When the sparse code with 50% zeros and 3-3-0 overlap is used, there are 1215 combinations that will result recovery of all four lost symbols. Therefore, $C(4, 4) = 1215/1820$. None of the symbols is recovered in 165 cases ($C(0, 4) = 165/1820$) and one symbol is recovered in 440 cases ($C(1, 4) = 440/1820$). In this example, none of the combinations result in recovery of two or three symbols ($C(2, 4) = 0$ and $C(3, 4) = 0$). Fig. 7 shows the fraction of recovered symbols as a function of the channel loss rate, for some of the design solutions given in Fig. 6.

The probability that exactly l packets are missing in a block of n packets has been given in Eq. (1) for a uniform and independent loss process. By combining Eq. (1) with Eq. (7), it is, therefore, possible to formulate the average fraction of recovered symbols as a function of the channel loss probability p, as given in Eq. (8). When the probability distribution of lost symbols per block is known (e.g., from Eq. (1)), the best sparse FEC code can finally be chosen among different candidate design solutions by finding the generator matrix that induces the minimal residual packet loss rate. It is worth noting that Eq. (8) is actually a generic form of Eq. (3). Correspondingly, the variance of the residual loss rate can be computed from Eq. (9), i.e., a generic form of Eq. (4).

$$PLR_{RES}(p) = p - \frac{1}{n} \sum_{l=1}^{n} l \Pr(N_{LOST} = l) R(l) \tag{8}$$

$$VAR_{RES}(p) = \sum_{l=0}^{n} \sum_{i=0}^{n-k} \Pr(N_{LOST} = l) C(i,l) (PLR_{RES}(p) - (l-i)/n)^2 \tag{9}$$

We now illustrate the performance of the sparse FEC code, using the same example as above. The sparse (16, 12) FEC codes have been implemented in Matlab, and a set of experiments have been performed in order to measure the residual packet loss rate when a lossy transport channel is simulated by dropping source and FEC packets randomly with uniform probabilities. We have compared the performance of a sparse RS code with 3-3-0 overlap pattern against a conventional (16, 12) RS code and the shortest possible (4, 3) MDS block code with equal FEC overhead, in terms of average residual packet loss rate per block and variance of the residual loss rate. To make the comparison fair, four short (4, 3) blocks are considered as one long (16, 12) block in our study. Fig. 8 shows the average residual packet loss for the three codes. At low loss rates (i.e., $p \leqslant 0.2$), the sparse RS performs only very slightly worse than the conventional RS, whereas at higher loss rates (i.e., $0.25 \leqslant p$) it outperforms the conventional RS codes. The short XOR-based code outperforms both other codes at very high loss rates (i.e., $0.3 \leqslant p$), but performs clearly worse at lower loss rates.

The corresponding variance of the residual loss rate per block is shown in Fig. 9. The sparse FEC codes are obviously advantageous when low variance is required by the application. In the range of loss probabilities where $0.1 \leqslant p \leqslant 0.4$, the proposed code outperforms the conventional (16, 12) RS code with a clear margin. Even lower variance can be achieved by using short (4, 3) blocks, but at the cost of the weaker error correcting capability at low loss rates, as we have seen in Fig. 8. Overall, the sparse FEC codes permit to improve the performance of common RS

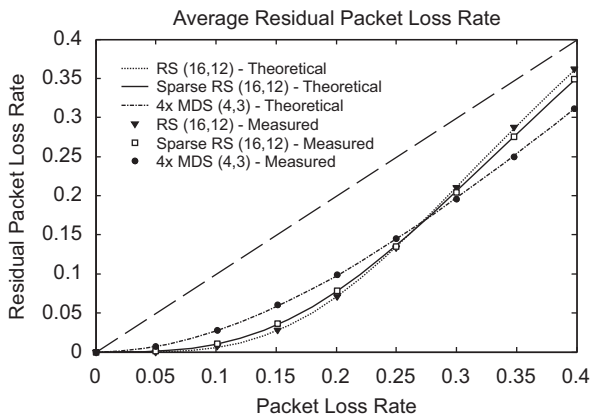

Fig. 8. Average residual packet loss rate for different FEC codes.

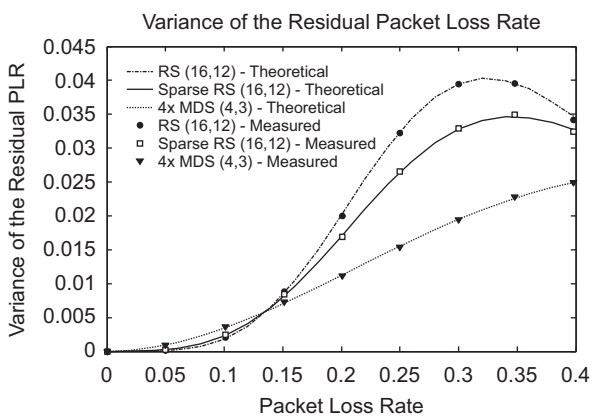

Fig. 9. Variance of the residual number of losses per FEC block.

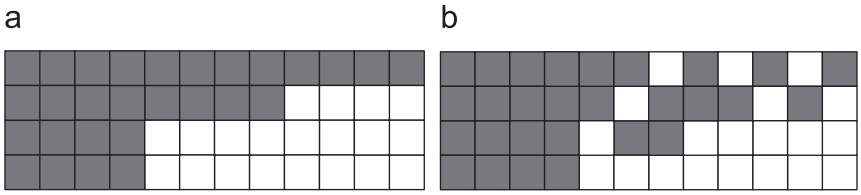

Fig. 10. Sparse code generator matrix designed for unequal error protection (notation is similar to Fig. 6). (a) Hierarchical design and (b) sparse hierarchical design.

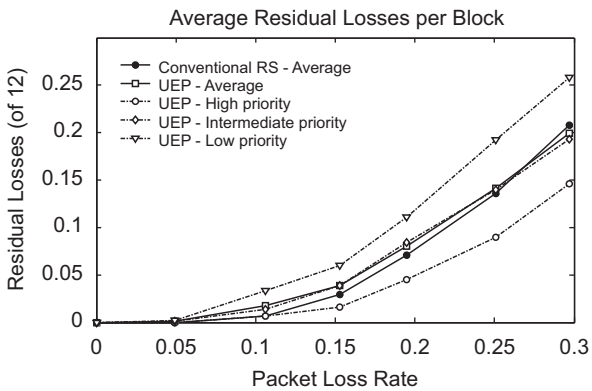

Fig. 11. Performance of a sparse RS code optimized for unequal error protection.

codes with a smaller variance in the residual loss probability when the channel loss probability increases.

3.4. Unequal error protection

The sparse FEC codes discussed in this paper can easily be extended to unequal error protection. Each column of the generator matrix is actually related to a source symbol. Therefore, the larger the number of non-zero coefficients in each column, the stronger protection for the corresponding source symbol. A flexible UEP scheme can, therefore, be implemented by relaxing the second condition in the design of sparse FEC codes.

Two examples of a generator matrix for UEP are shown in Fig. 10, where we assume that the source symbols in the columns 1–4 belong to the highest priority class, symbols in columns 5–8 to the intermediate priority class, and symbols in columns 9–12 have the lowest priority. Each row is related to a FEC symbol. Fig. 10a represents the same type of code as proposed in [14], where FEC symbols have been arranged hierarchically so that the first FEC symbol protects all the source symbols 1–12, the second FEC symbol protect high and intermediate priority symbols 1–8 and the third and fourth FEC symbols protects source symbols 1–4. However, this kind of strict hierarchy is not the only possibility, and more complex matrix structures can also be used, as proposed in [15] and shown in Fig. 10b. The number of zeros on each column is the same in both examples, but in case b, the zeros have been distributed more evenly among rows. This helps to make the relative priorities of FEC symbols more even.

Fig. 11 finally illustrates the performance of the proposed code for unequal error protection using a similar generator matrix, as shown in Fig. 10. It can be seen that the average error correcting performance computed over all priority classes is slightly worse than the performance achieved with conventional RS FEC code that do not implement any unequal error protection. However, the high priority data clearly present a lower residual loss rate. The residual loss rate of intermediate priority packets follows approximately the average residual loss rate, while low priority packets present higher residual loss rates, as expected. By using different distributions of the non-zero coefficients in the generator matrix, the relative residual packet loss rate curves can be adjusted in a rather flexible manner. Such flexibility cannot be achieved with short conventional RS codes: for example, the only possible combination of RS codes with three protection levels for similar configuration would be RS(7, 4) for high priority symbols, RS(5, 4) for intermediate priority symbols and no protection (4, 4) for the low priority symbols. Finally, it should be noted that the LDPC codes require long code blocks for efficient UEP implementations [12].

4. Streaming simulation results

4.1. Simulation setup

We analyze now the performance of the proposed sparse codes for equal or unequal error protection and typical media streaming scenarios. We have encoded video sequences in the H.264/AVC format using the H.264/AVC JM reference codec version 13.2 [21]. The encoder includes three basic types of frames: Intra frames (I-frames) that are independent on other frames, and inter frames that use either unidirectional (P-frames) or bidirectional (B-frames) temporal prediction from other frames [22]. A set of mutually dependent frames forms a *group of pictures* (GOP).

In H.264/AVC, the basic elements for transmission and eventually decoding are called *network abstraction layer units* (NALUs) and they correspond to one compressed frame, or part of it. Typically, the network packets then include one NALU [22,23]. This is also the basic assumption in our experiments. Relatively small NALUs (packets) have been used (maximum size of the NALU is 500) in order to minimize the probability of losing all the information from one frame. *Flexible macroblock ordering* (FMO) tool [23,24] has also been enabled in order to

facilitate error concealment. Note that FMO does not change the relative performance of the proposed error correction algorithms, even though the quality degrades more aggressively along the loss rate when FMO is disabled.

We run a set of simulations that reproduce the behavior of a lossy packet network, and the residual loss patterns after FEC decoding are applied to encoded H.264/AVC bitstreams. The "motion copy" error concealment feature implemented in the JM reference decoder [25] is used to conceal the impact of residual NALU losses. We measure the *peak signal-to-noise ratio* (PSNR) quality of the video after decoding, for the different FEC coding algorithms. Our results show that some benefits can be achieved with equal FEC, especially in terms of stability of the decoded video quality. When UEP FEC is applied, the benefits of the proposed sparse FEC codes become even more significant.

4.2. Equal error protection (EEP)

First, we consider a streaming scenario where *equal error protection* is a reasonable coding strategy. We consider video streams with relatively high quality and CIF resolution (376×288 pixels), where all NALUs represent approximately the same proportional priority, despite the temporal prediction in the compressed sequence. In this case, each I-frame is divided in a greater number of NALUs than the P-frames, and error concealment has a similar efficiency in both I- and P-frames when losses do not appear in bursts. EEP is assumed to be a reasonable error resilient streaming solution in this case.

We have used two different bitstreams ("Foreman" with IPPPPPPPPP GOP structure and "Soccer" with IPPPP GOP structure, both encoded with quality parameter $QP = 30$) and six different packet loss rates (0.1, 0.15, 0.20, 0.25, 0.30 and 0.35). These GOP structures were selected, since we wanted to experiment two different GOP lengths and GOPs in streaming applications cannot be very long. B-frames have not been used, as they are not considered suitable for the EEP scenario. Three different FEC codes have been experimented: conventional RS with long coding blocks (16, 12), conventional RS with short coding blocks (4, 3) and the proposed sparse RS(16, 12) code with (3-3-0) overlap pattern (similar as in Fig. 6c). The overall PSNR quality curves as a function of the packet loss rate are shown, respectively, in Figs. 12 and 13 for both sequences.

The performance of the proposed sparse FEC is similar to the RS(16, 12) codes when *PLR* is between 0.1 and 0.25 and slightly higher when PLR is above 0.25. The performance of the RS(4, 3) code is notably worse than both long codes at low PLR ($p < 0.25$), but better at high PLR ($p > 0.25$). It confirms the results shown in Fig. 8 that illustrates the evolution of the residual PLR for the same codes. Typically, the use of conventional RS(16, 12) is advantageous at low PLR, whereas the use of short RS(4, 3) codes is justified at high PLR. The proposed sparse FEC code represents an effective compromise between these two alternatives.

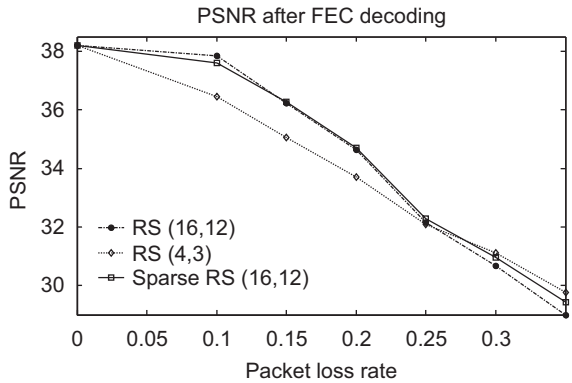

Fig. 12. Quality in terms of PSNR at different packet loss rates with conventional RS code and sparse RS code ("Soccer" sequence).

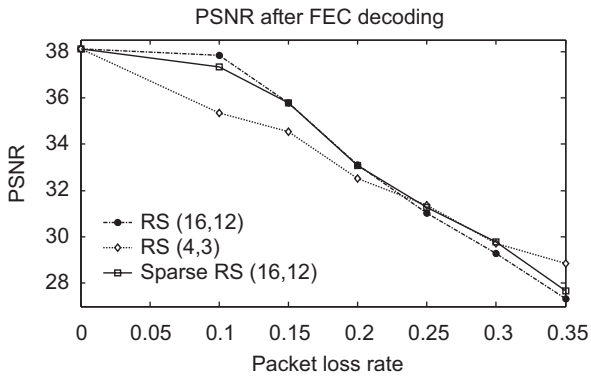

Fig. 13. Quality in terms of PSNR at different packet loss rates with conventional RS code and sparse RS code ("Foreman" sequence).

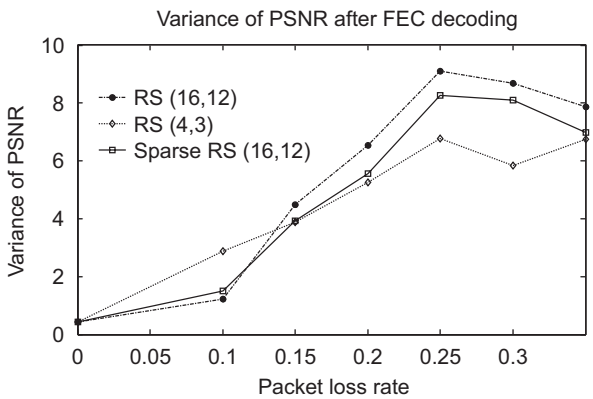

Fig. 14. Stability of quality in terms of variance of PSNR at different packet loss rates with conventional RS code and sparse RS code ("Soccer" sequence).

Since the variation of the video quality is an important parameter for the evaluation of the video quality perceived by the end user [18], the variance of the PSNR quality is also reported in Figs. 14 and 15 for the three coding schemes. The variance has been computed by using the per-frame PSNR values of the concerned video

sequence as a random variable. It can be seen that the sparse FEC codes outperform the two other schemes when both low and high loss rates are considered. Typically, the variance of the quality is significantly smaller than the one observed for the conventional RS(16, 12) code when PLR is larger than 0.25. Therefore, the user experience is clearly improved with the sparse FEC codes, as the quality is more stable.

In order to study the impact of coding block length in practice, we have conducted some experiments also with sparse and regular RS(12, 9) and (32, 24) codes, using the "Soccer" sequence. The binary representations of the sparse matrices are shown in Fig. 16. These matrices have been designed according to the principles explained in Section 3.2. For the (12, 9) code, the used number of zeros (9 out of 27) is the only option that fulfills the design constraints given in Section 3.2. For the sparse (32, 24) code there are more possibilities. In this case, we have chosen a rather sparse matrix structure with $\frac{5}{8}$ of zero coefficients to make the code clearly distinctive to conventional RS codes. Trial experiments show that the generator matrix structure depicted in Fig. 16 produces results that can be well generalized for sparse RS codes of that block length.

In Figs. 17 and 18, the example curves for PSNR and PSNR variance are plotted as a function of coding block length n with three different average packet loss rates, 0.15, 0.2 and 0.3. The results confirm the observation that the use of long code block improves the PSNR results when the loss rate is below 0.25, but worsens it at higher loss rates. In most of the cases, the use of sparse codes improves the average performance both in terms of PSNR and PSNR variance. The relative gain is at the largest when $n = 16$. The lower gain with sparse RS(12, 9) code is expected, since the low number of zero coefficients is not sufficient to make the code very distinctive from the conventional RS code. On the other hand, the slight performance improvement that can be achieved at low loss rates by using long blocks comes at the cost of higher variance, except when *PLR* is low. This is because the

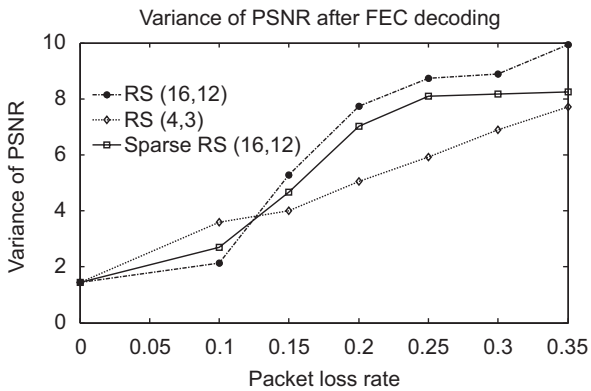

Fig. 15. Stability of quality in terms of variance of PSNR at different packet loss rates with conventional RS code and sparse RS code ("Foreman" sequence).

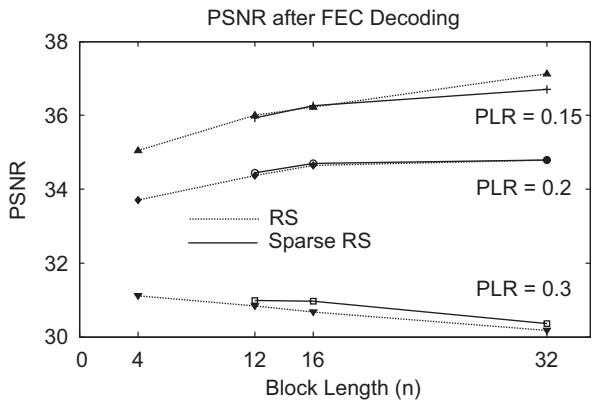

Fig. 17. PSNR curves shown for (4, 3), (12, 9), (16, 12) and (32, 16) codes.

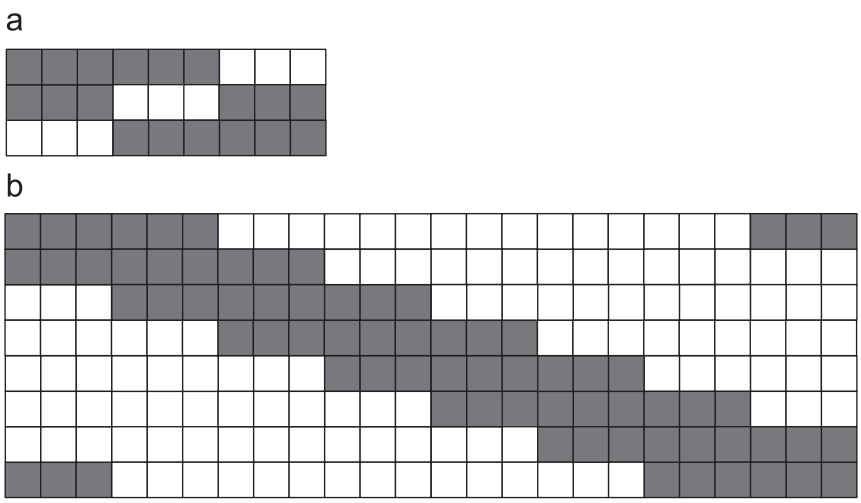

Fig. 16. The generator matrices used for sparse (12, 9) and (32, 24) codes (notation is similar to Figs. 6 and 10). (a) Sparse RS(12, 9) generator matrix and (b) sparse RS(32, 24) generator matrix.

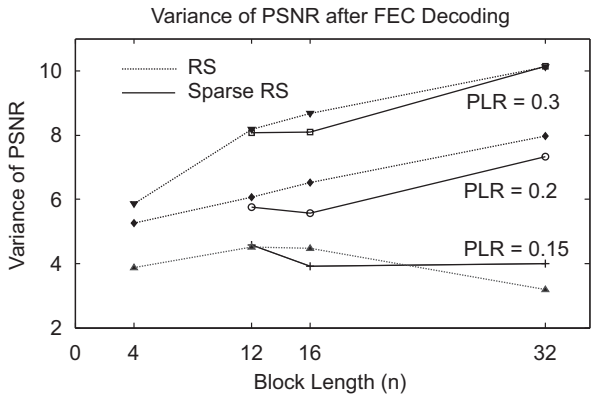

Fig. 18. PSNR variance curves shown for (4, 3), (12, 9), (16, 12) and (32, 16) codes.

clusterization of residual losses cannot be efficiently avoided by any other means than interleaving, when long coding blocks are used. The results show that the use of coding blocks longer than 16 can be justified in our application scenario only if it is guaranteed that the PLR remains low (0.15 or below).

4.3. Unequal error protection

In the following, we consider scenarios where the use of unequal error protection is beneficial in practice. We have used video sequences with relatively low bandwidth and QCIF resolution (176×144 pixels), coded with reasonable quality ($QP = 30$) and FMO enabled. Instead of using the data partitioning tool of JM reference codec, we have simply used the frame type as a basis for priority classification. The impact of error propagation is more important than in the scenario considered in the previous section, and UEP is clearly justified in this case. However, we can still provide a fair comparison with a baseline FEC scheme with equal error protection, since even the loss of a NALU in an I-frame can still be concealed quite efficiently due to FMO. The priority distinction is therefore not as strict as in scalable coding with hierarchical layers for example, where the loss of a base layer element renders all the related enhancement layer elements useless.

In the first set of simulations, we have used a GOP structure built on one I- and four P-frames. The I-frames are split into four NALUs, and each P-frame into two NALUs. The highest priority class contains the I-frames, the intermediate priority class represents the first two P-frames, and the lowest priority class corresponds to the last two P-frames. These configuration parameters produced a sequence with four NALUs per I-frame and two NALUs per P-frame, resulting in cycles of 12 NALUs, four in each priority class. Two sequences with different content have been used (“Carphone” and “Coastguard”). A sparse UEP FEC code with generator matrix similar to the one shown in Fig. 10 has been used, and the PSNR results for both sequences are compared against equal protection using the non-differentiating conventional RS(16, 12) code. The results have also been compared to the conventional UEP scheme using RS(7, 4) for the high priority class, RS(5, 4) for the intermediate priority class and no protection for the lowest priority class. The conventional UEP scheme uses the only possible combination of conventional MDS codes that allows three different priority levels with the given parameters (three sets of four source packets protected with four FEC packets altogether).

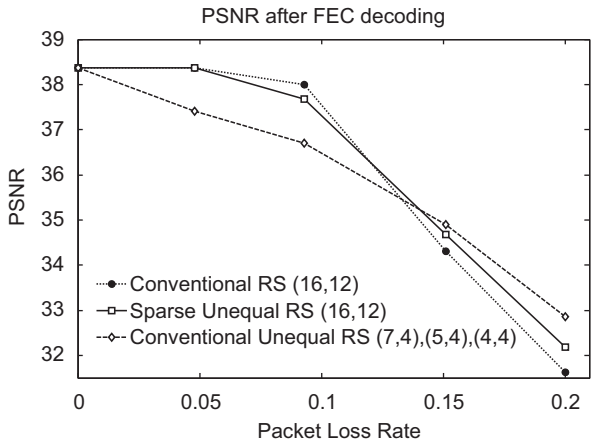

Fig. 19. Uneven sparse RS and conventional even/uneven RS codes compared in terms of video quality (“Carphone” sequence, GOP structure IPPPP).

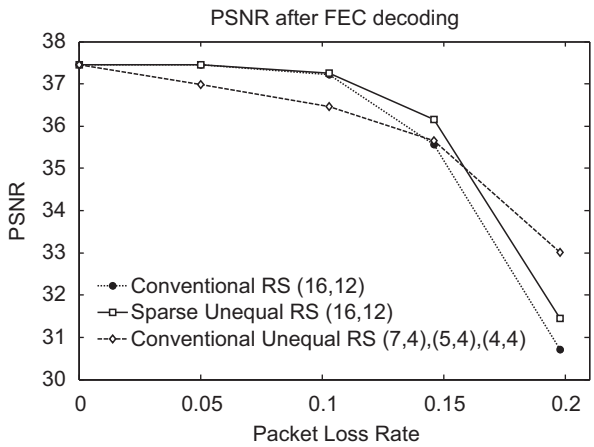

Fig. 20. Uneven sparse RS and conventional even/uneven RS codes compared in terms of video quality (“Coastguard” sequence, GOP structure IPPPP).

The resulting PSNR curves for both sequences are shown in Figs. 19 and 20, respectively. The non-differentiating RS performs slightly better at low loss rates, because both UEP schemes suffer from some low-priority packet losses, whereas equal erasure protection provides virtually perfect recovery. However, the benefit of the proposed UEP code becomes obvious at loss rates above 0.1. Even though the average residual packet loss rate is still slightly higher with UEP than with EEP RS codes, the resulting quality is better. In this case, a smaller number of

I-frames are affected by the packet loss, and the impact of error propagation is therefore reduced. The same tendency is even clearer when the variances of PSNR are compared, as shown in Figs. 21 and 22. The conventional UEP scheme using separate codes for each priority class becomes beneficial at high packet loss rates above 0.15, but performs significantly weaker than any of the two other codes at low packet loss rates. Therefore, UEP built on the sparse FEC codes can be considered as a good compromise between highly differentiating conventional UEP and EEP RS code.

In a second set of simulations, we have encoded the same two sequences with a GOP structure IBPB, where I- and P-frames are split into four NALUs and B-frames into two NALUs. This results in four high priority I-slice NALUs, four intermediate priority P-slice NALUs and four low priority B-slice NALUs. We have used the same FEC codes as before, and the PSNR quality curves for "Coastguard" sequence are plotted in Fig. 23, and the respective variance curves in Fig. 24. In this scenario, the advantage of using UEP instead of EEP FEC becomes even more obvious than in the previous scenario, and confirms that the proposed UEP schemes represent an interesting approach for error protection in video streaming applications. The results for "Carphone" sequence are roughly similar, and therefore not shown here for the sake of clarity.

Finally, Fig. 25 shows one example of visual results with "Coastguard" sequence (IPPPP structure, packet loss rate 0.15). On the left side, there is the first frame (I-frame) and on the right side the last frame (P-frame) of the GOP. In this particular case, equal RS code fails to recover one lost NALU from the first I-frame, leading to rather serious distortion in the upper part of the frame. Due to the temporal prediction, this distortion is propagated further and can be seen also in the last frame. With uneven codes, the I-frame is fully recovered. In this example, the UEP code based on conventional RS codes cannot recover one

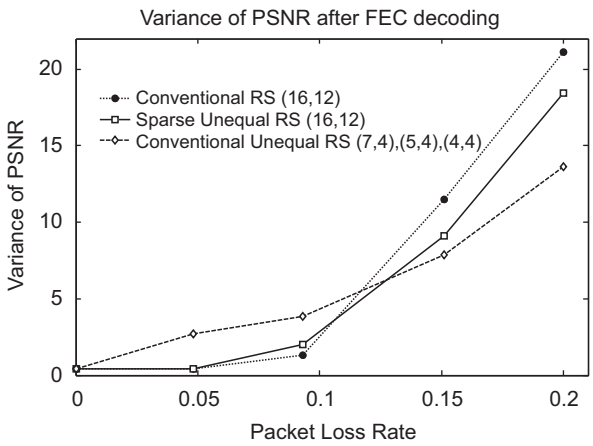

Fig. 21. Uneven sparse RS and conventional even/uneven RS codes compared in terms of variance of the video quality ("Carphone" sequence, GOP structure IPPPP).

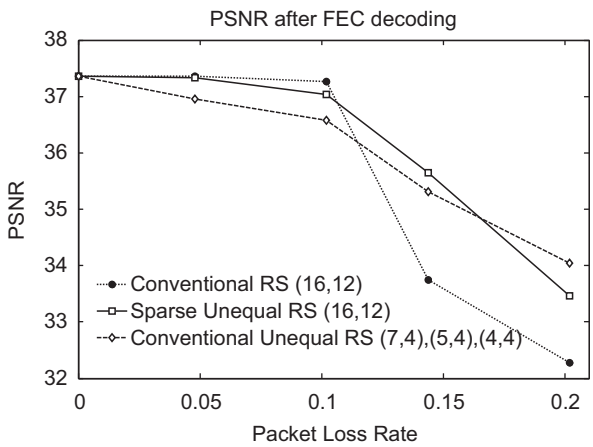

Fig. 23. Uneven sparse RS and conventional even/uneven RS codes compared in terms of video quality ("Coastguard", GOP structure IBPB).

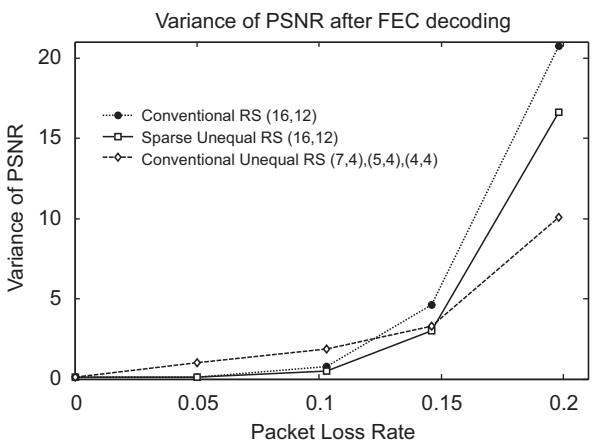

Fig. 22. Uneven sparse RS and conventional even/uneven RS codes compared in terms of variance of the video quality ("Coastguard" sequence, GOP structure IPPPP).

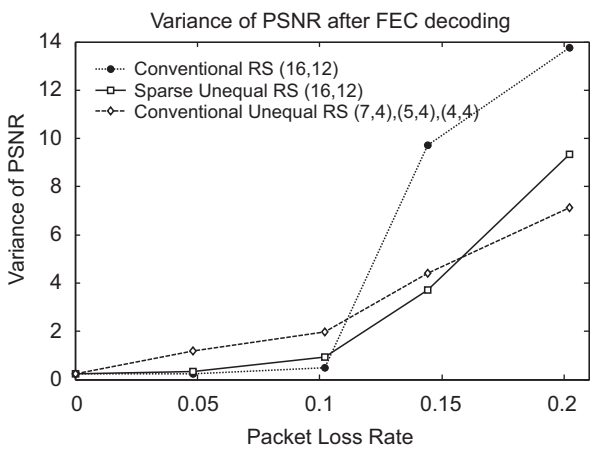

Fig. 24. Uneven sparse RS and conventional even/uneven RS codes compared in terms of variance of the video quality ("Coastguard" sequence, GOP structure IBPB).

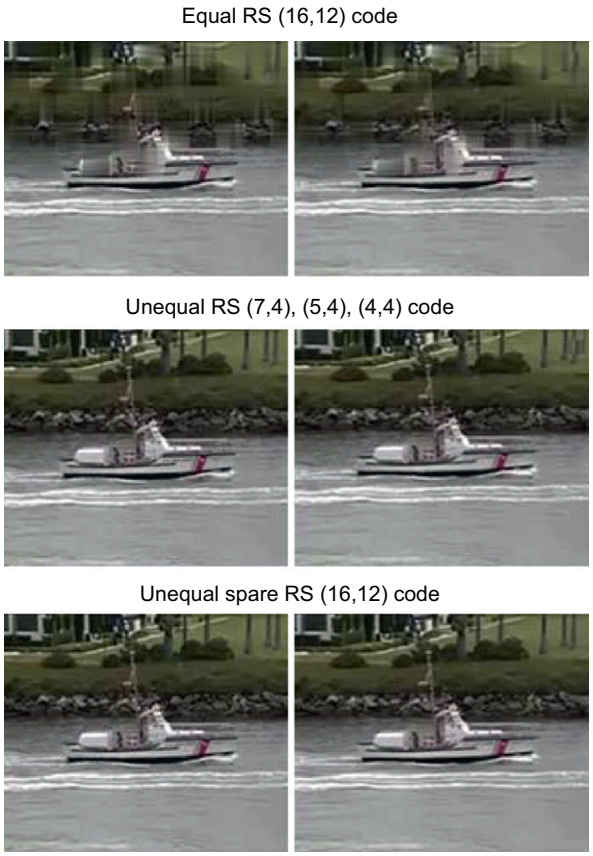

Fig. 25. An example of visual results ("Coastguard" sequence, GOP structure IPPPP). The first I-frame of the GOP is on the left, and the last P-frame of the same GOP is on the right.

of the lower priority NALUs, which can be seen as minor distortion in the upper right corner of the last P-frame. With the sparse UEP code, the P-frame seems intact. However, it should be noted that the packet loss occur randomly and this example represents the optimal situation for the sparse UEP code. Any conclusions cannot be derived by analyzing individual frames, since the performance of the schemes must be studied statistically at the sequence level.

Overall, the simulated scenarios demonstrate the capabilities and flexibility of the proposed sparse FEC codes, even though they do not cover all the possible implementations of real systems that use longer GOPs. They show that different sparse generator matrix patterns can lead to different proportional protection levels. This is not possible with conventional UEP and short coding. This is why the sparse RS codes could be easily adapted for a wide variety of different GOP structures and even media types (not limited to H.264 video), with different priorities and dependency relations between media units.

For the sake of clarity, we have restricted our analysis and experiments to a simple packetization scenario, where one NALU is allocated per packet. The most advanced UEP schemes proposed in the literature, such as PET, typically involve more complicated packetization schemes that allocate several source and MDS FEC data units in each transport packet [13,26]. It is assumed that the proposed codes will be beneficial also for this kind of scenarios, when the conventional RS codes are replaced by the sparse RS codes. It is part of the future work to study the applicability and potential benefits of the proposed codes in different kinds of advanced streaming and packetization frameworks.

5. Conclusions

In this paper, we have presented sparse FEC codes that are optimized for partial recovery of data in cases where conventional MDS FEC codes fail to recover the lost source packets. They permit to reduce the variations in the residual packet loss rate when the fraction of packet losses per block occasionally exceeds the FEC overhead. We have observed that common short source blocks tend to provide smoother average error correcting performance, at the cost of weaker average performance at low loss rates. In order to further facilitate partial error recovery in short FEC blocks, we have studied codes that use sparse RS generator matrices. These codes can easily be extended to provide unequal error protection where they permit flexible adjustment of different proportional priority levels even when short FEC blocks are used. The performance of the codes has been validated with the simulation of H.264 video streaming in practical scenarios. The simulation results show that the sparse FEC codes achieve smoother quality degradation when the packet loss rate increases, without sacrificing the quality significantly at very low packet loss rates. They represent an efficient trade-off between short block coded with limited flexibility, and long block codes with delay penalties.

References

[1] Y. Wang, S. Wenger, J. Weng, A.K. Katsaggelos, Error resilient video coding techniques, IEEE Signal Processing Magazine 17 (4) (2000) 61–82.
[2] S. Lin, D.J. Costello, Error Control Coding: Fundamentals and Applications, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
[3] F. Zhai, Y. Eisenberg, A.K. Katsaggelos, Joint source-channel coding for video communications, in: A. Bovik (Ed.), Handbook of Image and Video Processing, second ed., Elsevier Academics Press, San Diego, 2005.
[4] J. Rosenberg, H. Schultzrinne, An RTP Payload Format for Generic Forward Error Correction, IETF RFC 2733, 1999.
[5] V. Roca, C. Neumann, Design, Evaluation and Comparison of Four Large Block FEC Codecs: LDPC, LDGM, LDGM–Staircase and LDGM–Triangle, Plus a Reed–Solomon Small Block FEC Codec, INRIA Research Report RR-5225, 2004.
[6] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, V. Stemann, Practical loss-resilient codes, in: Proceedings of the STOC '97, 1997, El Paso, Texas, USA, pp. 150–159.
[7] M.C. Davey, M. MacKay, Low-density parity check codes over GF(q), IEEE Communications Letters 2 (6) (1998) 165–167.
[8] X.-Y. Hu, E. Eleftheriou, Regular and irregular progressive edge-growth tanner graphs, IEEE Transactions on Information Theory 51 (16) (2005) 386–398.
[9] A. Shokrollahi, Raptor codes, IEEE Transactions on Information Theory 52 (6) (2006) 2551–2567.
[10] A. Kamra, J. Feldman, V. Misra, D. Rubenstein, Growth codes: maximizing sensor network data persistence, in: Proceedings of the SIGCOMM '06, 2006, Pisa, Italy, pp. 255–266.
[11] A. Dimakis, J. Wang, K. Ramchandran, Unequal growth codes: intermediate performance and unequal error protection for video

streaming, in: Proceedings of the MMSP '07, 2007, Chania, Greece, pp. 107–110.
[12] S. Karande, H. Radha, Partial reed–solomon codes for erasure channels, in: Proceedings of the ITW '03, 2003, Paris, France, pp. 82–85.
[13] A. Albanese, J. Blömer, J. Edmonds, M. Luby, M. Sudan, Priority encoding transmission, IEEE Transactions on Information Theory 42 (6) (1996) 1737–1744.
[14] A. Bouabdallah, J. Lacan, Dependency-aware unequal erasure protection codes, Journal of Zheijiang University Science A 7 (1) (2006) 27–33.
[15] J. Korhonen, P. Frossard, Sparse FEC codes for flexible media protection, in: Proceedings of the ICME '08, 2008, Hannover, Germany, pp. 445–448.
[16] I. Cidon, A. Khamisy, M. Sidi, Analysis of packet loss processes in high-speed networks, IEEE Transactions on Information Theory 39 (1) (1993) 98–108.
[17] P. Frossard, O. Verscheure, Joint source/FEC rate selection for quality-optimal MPEG-2 video delivery, IEEE Transactions on Image Processing 10 (12) (2001) 1815–1825.
[18] M. Zink, O. Künzel, J. Schmitt, R. Steinmetz, Subjective impression of variations in layer-encoded videos, in: Proceedings of the IWQoS '03, 2003, Monterey, California, USA, pp. 137–154.
[19] M. Claypool, Y. Zhu, Using interleaving to ameliorate the effects of packet loss in a video stream, in: Proceedings of the MNSA '03, 2003, Providence, Rhode Island, USA, pp. 508–513.
[20] Y. Lin, B. Li, B. Liang, Differentiated data persistence with priority random linear codes, in: Proceedings of the ICDS '07, 2007, Toronto, Canada, pp. 47–54.
[21] H.264/AVC Reference software archive. Available online: 〈http://iphome.hhi.de/suehring/tml/download/old_jm/〉.
[22] S. Wenger, H.264/AVC over IP, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 645–656.
[23] ITU-T, Advanced Video Coding for Audiovisual Services, ITU-T Recommendation H.264, 2005.
[24] S. Wenger, M. Horowitz, FMO: Flexible Macroblock Ordering, JVT-C089, 2002.
[25] Y-K. Wang, M. Hannuksela, V. Varsa, A. Hourunranta, M. Gabbouj, Error concealment feature in the H.26L test model, in: Proceedings of the ICIP '02, 2002, Rochester, New York, USA, vol. 2, pp. 729–732.
[26] R. Hamzaoui, V. Stanković, Z. Xiong, Optimized error protection of scalable bit streams, IEEE Signal Processing Magazine 22 (6) (2005) 91–107.