

This is an author-deposited version published in: http://oatao.univ-toulouse.fr/
Eprints ID: 4867

To cite this document: TOURNOUX Pierre-Ugo, LOCHIN Emmanuel, LACAN Jérôme, BOUABDALLAH Amine, ROCA Vincent. On-the-fly erasure coding for real-time video applications. *IEEE Transactions on Multimedia*, vol. 13, n° 4, pp. 797-812. ISSN 1520-9210

Any correspondence concerning this service should be sent to the repository administrator: staff-oatao@inp-toulouse.fr

On-the-fly erasure coding for real-time video applications

Pierre Ugo Tournoux[1,2], Emmanuel Lochin[1,2], Jérôme Lacan[2], Amine Bouabdallah[1,2] and Vincent Roca[3]
[1] CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
[3] INRIA, Planète research team, Grenoble, France

***Index Terms*—Reliability, Delay recovery, Erasure code, Video-conferencing.**

***Abstract*—This paper introduces a robust point-to-point transmission scheme: Tetrys, that relies on a novel on-the-fly erasure coding concept which reduces the delay for recovering lost data at the receiver side. In current erasure coding schemes, the packets that are not rebuilt at the receiver side are either lost or delayed by at least one RTT before transmission to the application. The present contribution aims at demonstrating that Tetrys coding scheme can fill the gap between real-time applications requirements and full reliability. Indeed, we show that in several cases, Tetrys can recover lost packets below one RTT over lossy and best-effort networks. We also show that Tetrys allows to enable full reliability without delay compromise and as a result: significantly improves the performance of time constrained applications. For instance, our evaluations present that video-conferencing applications obtain a PSNR gain up to 7dB compared to classic block-based erasure codes.**

I. Introduction

Multimedia applications, even over best effort networks, are more and more pervasive today. This is the sign of an important need by end-users for such applications, no matter their location and the connection technology being used. If the networking conditions are sometimes appropriate, users might also experience long transmission delays and significant packet losses. When this happens, providing the level of data delivery timeliness and reliability required by multimedia applications seems to be really challenging [1]. In this context, this work aims at providing a transport-level reliability mechanism, called Tetrys, compliant with real-time applications requirements and able to recover lost packets in a given time threshold.

Currently there are two kinds of reliability mechanisms based respectively on retransmission and redundancy schemes. Automatic Repeat reQuest (ARQ) schemes recover all lost packets thanks to retransmissions. This implies that the recovery delay of a lost packet needs at least to wait one supplementary Round Trip Time (RTT). However, this can be problematic if this delay exceeds the threshold of the application (i.e. the threshold above the application considers a packet outdated).

A well-known solution to prevent this additional delay is to add redundancy packets to the data flow. This can be done with the use of Application Level Forward Error Correction (AL-FEC) codes[1]. The addition of $n-k$ repair packets to a block of k source packets allows to rebuild all of the k source packets if a maximum of $n-k$ packets are lost among the n packets sent. In practice, only Maximum-Distance Separable codes (MDS), such as Reed-Solomon codes [2], have this optimal property, whereas other families of codes (like LDPC [3] or Raptor codes [4]) need to receive a few number of extra symbols in addition to the k strict minimum. However, if more than $(n-k)$ losses occur within a block, decoding becomes impossible. In order to increase robustness (e.g. to tolerate longer bursts of losses), the sender can choose to increase the block size (i.e. the n parameter) with the price of an increase of the decoding delay in case of erasure. In order to improve robustness while keeping a fixed delay, the sender can also choose to add more redundancy while keeping the same block size with the price of a decrease of the goodput (which is not necessarily affordable by the application). These trade-off between *packet decoding delay; block length* and *throughput* are, for instance, addressed in [5]. Another approach is proposed in [6] where the authors use non-binary convolutional-based codes. They show that the decoding delay can be reduced with the use of a sliding window, instead of a block of source data packets, to generate the repair packets. However, both mechanisms do not integrate the receivers' feedbacks and thus, cannot provide any full reliability service.

Finally, an hybrid solution named Hybrid-ARQ which combines ARQ and AL-FEC schemes is often used. This is an interesting solution to improve these various trade-off [7]. However, when retransmission is needed, the application-to-application delay still depends on the RTT which might be not acceptable with real-time applications.

The present contribution totally departs from the above schemes. In fact, it inherits from the following two independent works on erasure coding which have converged to an on-the-fly coding mechanism where feedbacks from the receivers are considered during the encoding process:

1) In [8], Sundararajan *et al.* have proposed a coding scheme which includes feedback messages on the reverse path. The goal of this feedback path is to decrease the encoding complexity at the sender side without impacting on the

[1]AL-FEC codes are FEC codes for the erasure channel where symbols (i.e. packets) are either received without any error or lost (i.e. erased) during transmission.

communication transfer. This scheme allows to reduce the number of transmissions and as a result, the average decoding delay in the context of multiple receivers. In their evaluation, the authors neglect transmission delays and the resulting delays from the losses observed by different receivers. A noticeable contribution of their work is the concept of *seen packet* by which the receiver acknowledges the degrees of freedom of the linear system corresponding to the received packets. This scheme has the main benefit of optimizing buffers occupancy while reducing the encoding complexity;

2) Independently, Lacan and Lochin also proposed in [9] an on-the-fly coding system using feedbacks in the context of point-to-point communications with high transmission delays. Basically, the principle is to add repair packets generated as a linear combination of all the source data packets sent but not yet acknowledged. This scheme was proposed in order to enable full-reliability in Delay Tolerant Networks (DTN) and more specifically in Deep Space Networks (DSN) where an acknowledgment path might not exist and where the experienced delay might prevent the efficient use of standard ARQ schemes.

Unlike current reliability methods, these on-the-fly coding schemes allow to fill the gap between systems without retransmission and fully reliable systems by means of retransmissions. In our work, we propose to deeply investigate the recovery delay of the lost packets, which is one essential characteristic of these on-the-fly coding schemes, and we show that this delay is both tunable and independent of the RTT. The main contributions of this paper are the application of Tetrys [10] (augmented with the concept of *seen packets* [8]) to the context of real-time applications and the analysis of the performances achieved with a probabilistic approach.

We present the Tetrys mechanism in Section II and illustrate the simplicity of its configuration compared to FEC codes in Section IV. Then we demonstrate in Section V that Tetrys offers significant gains compared to standard erasure coding schemes, in particular in terms of delay versus reliability trade-off in the context of video-conferencing. An exhaustive analytical study of the mechanism is given in Section III. It is followed by a performance analysis in Section VI, that complements the experiments of Section V, and demonstrates that Tetrys is able to determine the minimal amount of redundancy required to fulfill the application requirements. We finally conclude this work in Section VII.

II. Proposal description

This section describes the Tetrys mechanism and the integration of the seen packet concept [8]. We choose to introduce the main Tetrys principle in Section II-A to allow the reader a quick understanding of the present coding scheme used while Section II-B further details Tetrys internal mechanisms.

A. Tetrys in a nutshell

Let us start with a quick overview of Tetrys. The Tetrys sender uses an elastic encoding window buffer (denoted BS) which includes all the source packets sent and not yet acknowledged. Let P_i be the source packet with sequence number i. Every k source packets, the sender sends a (single) repair packet $R_{(i..j)}$, which is built as a linear combination (with random coefficients) of all the packets currently in BS. The receiver is expected to periodically acknowledge the received or decoded packets. Each time the sender receives an acknowledgment, it removes the acknowledged packets from BS. A receiver can decode lost packets as soon as the rank of the linear system, which corresponds to the available repair packets, is higher or equal to the number of lost packets. In most cases, the decoding is successful as soon as the number of lost packets is lower or equal to the number of repair packets received.

P_i	The i^{th} source packet sent
$R_{(i..j)}$	A repair packet built as a linear combination of the source packets i to j: $R_{(i..j)} = \sum_{k=i}^{j} \alpha_k^{(i,j)} P_k$
k	The number of source packets between the transmission of two repair packets
n	The total number of source plus repair packets for each group of k source packets is denoted n (to keep the usual definition) and is always equal to $k+1$ for Tetrys
R	The redundancy ratio: $R = (n-k)/n = 1-(k/n)$ where k/n is the code rate. With Tetrys, we always have $R = 1/(k+1)$
Δ_R	The difference between the redundancy ratio and the packet loss rate: $\Delta_R = R - p$
p	The packet loss rate (PLR) experienced
b	The average burst size in case of a Gilbert Elliot channel. This value equals to 1 in the particular case of a Bernoulli channel. Therefore this parameter also defines the type of erasure channel used
L_i	The i^{th} lost packet
F_{sack}	The feedback (i.e. acknowledgment) transmission frequency, at the receiver
BS	The sender's (elastic) encoding window, composed of source packets not yet acknowledged
BR	The receiver's buffer where the packets received and decoded are kept until they are no longer needed to decode

TABLE I
NOTATIONS.

It results that: (1) Tetrys is tolerant to any burst of source, repair or acknowledgement losses, as long as the amount of redundancy exceeds the packet loss rate (PLR), and (2) the lost packets are recovered within a delay that does not depend on the RTT, which is a key property for real-time applications. These properties will be thoroughly studied in the remaining of this paper.

1) A simple data exchange: Fig. 1 illustrates a simple Tetrys exchange. Here $k = 2$ which means that a repair packet is sent each time two source packets have been sent. The right side of this figure shows the list of packets that are lost and not yet rebuilt, as well as the repair packets kept by the receiver in order to recover them. During this data exchange, packet P_2 is lost. However, the repair packet $R_{(1,2)}$ successfully arrives and allows to rebuild P_2. The receiver sends an acknowledgement for packets P_1 and P_2, in order to inform the sender that it can

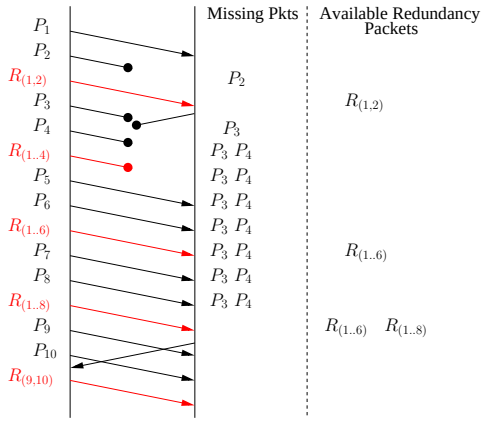


Fig. 1. A simple data exchange with Tetrys ($k = 2$).

compute the next repair packets from packet P_3. Unfortunately this acknowledgement is lost. However this loss does not compromise the following transmissions and the sender simply continues to compute repair packets from P_1. After this, we see that P_3, P_4 and $R_{(1..4)}$ packets are also lost. These packets are rebuilt thanks to $R_{(1..6)}$ and $R_{(1..8)}$ since the number of repair packets becomes higher or equal to the number of losses.

B. A broader view of Tetrys

We now detail the key concepts of Tetrys, namely the encoding and decoding process, the notion of seen packet, and the use of acknowledgments.

1) Encoding process: A repair packet is sent every k source packets. This packet is computed as a linear combination of all the source packets currently in BS, as follows:

$$R_{(i..j)} = \sum_{l=i}^{j} \alpha_l^{(i,j)}.P_l$$

where all packets between P_i and P_j belong to BS, with $\alpha_l^{(i,j)}$ are coefficients randomly chosen in a finite field $\mathbb{F}_{2^m}$ with $m \in \mathbb{N}^*$, and where the multiplication of a coefficient by a packet is defined in [11]. From a practical point of view, instead of transmitting all the coefficients along with the associated repair packet (which introduces a potentially large transmission overhead), we use a Pseudo-Random Number Generator (or PRNG, e.g. [12]) and only transmit the seed which has been used.

The k value is directly related to the code rate which is equal to $\frac{k}{k+1}$. This is of course a key parameter that should ideally be adjusted dynamically depending on the network conditions. For the sake of simplicity, the code rate is chosen fixed. In section III-A, we analytically detail the code rate and evaluate with simulations its impact on the overall performance. We finally provide some guidelines to correctly set this value in Section VI.

2) Decoding process: Decoding (i.e. recovering lost source packets) consists in solving the system of linear equations currently available at the receiver side. The available source packets (received or decoded) are stored by the receiver as long as they might be used by the source to build the next repair packets $R_{(i..j)}$ while the repair packets are also stored as long as they can be used to recover lost packets. More precisely, when a new repair packet $R_{(i..j)}$ arrives, all the available source packets that are part of P_i .. P_j are subtracted from $R_{(i..j)}$. The result is $R_{(L_1..L_l)}$, where $(L_1..L_l) \in (P_i..P_j)$ is the subset of packets of the linear combination that have been lost.

Let us assume that the l source packets $(L_1..L_l)$ have been lost and that l repair packets have been received and stored in BR. Let R^i be the i^{th} packet of the set of l repair packets (for the sake of readability, this notation does not mention the set of source packets used by the linear combination). We obtain:

$$(R^1, .., R^l)^T = G \cdot (L_1, .., L_l)^T$$

with:

$$G = \begin{pmatrix} \alpha_{L_1}^{R^1} & .. & \alpha_{L_l}^{R^1} \\ . & .. & . \\ . & .. & . \\ \alpha_{L_1}^{R^l} & .. & \alpha_{L_l}^{R^l} \end{pmatrix} \quad (1)$$

and where $\alpha_{L_j}^{R^i}$ is the coefficient used to encode the j^{th} lost packet in R^i. If G can be inverted, the lost packets $(L_1..L_l)$ are recovered with:

$$(L_1, .., L_l)^T = G^{-1} \cdot (R^1, .., R^l)^T$$

Once the decoding is successful, all of these l repair packets can now be removed from BR. If the matrix G is singular, the repair packet whose coefficients are linearly dependent is discarded, and the receiver has to wait one more repair packet to do another attempt.

A solution to improve the probability of having an invertible matrix could consist in using super-regular matrices [13]. However the dynamic nature of Tetrys makes this solution complex to set up. Furthermore, it can be observed that with random coefficients, G has an extremely high probability of being invertible if the finite field is chosen sufficiently large [14].

3) Seen packet: A lost packet is considered as "seen" by a receiver when it receives a fresh repair packet built from a linear combination that includes this lost packet (i.e. the lost packet was part of BS at the time the repair packet has been created). Even if a seen packet cannot be decoded immediately, the received repair packet contains enough information to recover this packet later. This explains why a "seen" packet acknowledges a source data packet as if it has been effectively received. Of course, when several lost packets are covered by one repair packet, only the oldest lost packet is considered as seen.

4) Acknowledgment packet: A receiver periodically sends acknowledgment packets. Each acknowledgment contains the list (in the form of a SACK vector [15]) of the packets seen or effectively received or decoded. Upon receiving this acknowledgment, the sender removes the acknowledged packets from the encoding window (BS). Therefore these packets are no longer included in the linear combinations used to encode the next repair packets [8]. This reduces the encoding/decoding

complexity. We choose to set the acknowledgment transmission frequency F_{SACK}, as a function of the current RTT: $F_{SACK} = s \times RTT$ where typical values for s are ranging from 0.25 to 2 [16]. While the choice of F_{SACK} does not impact on the reliability of the mechanism, there is a trade-off to find between the increase of F_{SACK} which reduces the encoding/decoding complexity (evaluated in Section III-D) and the transmission overhead and acknowledgement processing cost.

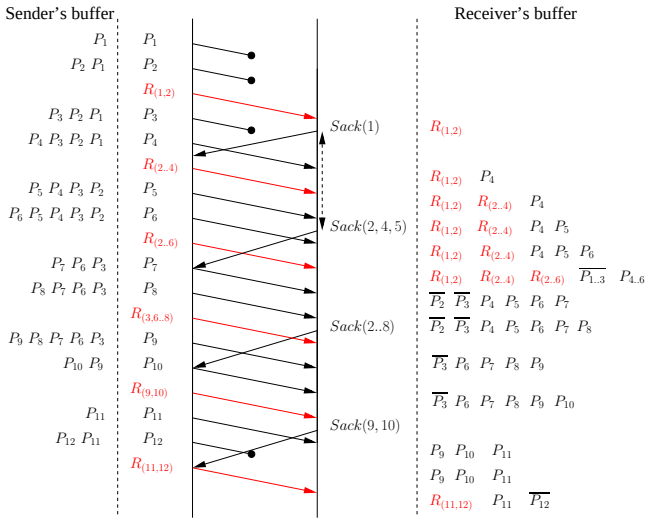


Fig. 2. A more elaborate data exchange, with selective acknowledgements and seen packets (k=2). Rebuilt packets are overlined.

5) A complete example: Let us consider the example of Fig. 2, where we assume the receiver sends back acknowledgments to a fixed frequency F_{sack}. The sender first transmits packets P_1, P_2 and $R_{(1,2)}$. Since the repair packet $R_{(1,2)}$ is the only one to be received, the receiver considers that P_1 and P_2 have been either lost or delayed. Then, the receiver acknowledges packet P_1 since $R_{(1,2)}$ contains a linear combination of P_1 which is considered as "seen". More generally, each time a repair packet is received, the receiver can acknowledge one of the source packets that are included in the linear combination. Then, the sender transmits P_3 and P_4. Just after, the sender receives an acknowledgement for packet P_1. So the sender creates a new repair packet starting from P_2: $R_{(2..4)}$. The receiver gets P_4 and $R_{(2..4)}$, meaning that the sender has received the previous SACK packet. Then, the receiver sends a new SACK packet which acknowledges P_2, P_4, P_5. The receiver cannot rebuild packets P_1 to P_3 since he did not receive enough repair packets. As a result, the receiver stores $R_{(1,2)}$ and $R_{(2..4)}$ for a future use. Since no loss occurs after that point, upon receiving a third repair packet, the receiver can now rebuild the missing packets. The received source packets included in the linear combination are subtracted, which results in $R_{(1,2)}, R'_{(2..4)}, R'_{(2..6)}$ such as:

$$(R_{(1,2)}, R'_{(2..4)}, R'_{(2..6)})^T = G \cdot (P_1, P_2, P_3)^T$$

with:

$$G = \begin{pmatrix} \alpha_{P_1}^{R_{(1,2)}} & \alpha_{P_2}^{R_{(1,2)}} & 0 \\ 0 & \alpha_{P_2}^{R_{(2..4)}} & \alpha_{P_3}^{R_{(2..4)}} \\ 0 & \alpha_{P_2}^{R_{(2..6)}} & \alpha_{P_3}^{R_{(2..6)}} \end{pmatrix} \quad (2)$$

where $\alpha_{P_z}^{R_{(i..j)}}$ is the coefficient used to encode P_z within the repair packet $R_{(i..j)}$.

With the assumption that G is invertible, G^{-1} is obtained thanks to a Gauss-Jordan elimination and packets P_1 to P_3 are given by:

$$(P_1, P_2, P_3)^T = G^{-1} \cdot (R'_{(1,2)}, R'_{(2..4)}, R'_{(2..6)})^T$$

These packets can be then considered as decoded. However, before removing them from BR, the receiver must still wait the reception of $R_{(3,6..8)}$ to be sure that the sender will not use these packets anymore to build new repair packets.

This example highlights the importance of several metrics: the decoding delay, the buffer size at the sender and at the receiver, and the number of operations needed to encode and decode. All these metrics will be studied and analyzed thoroughly in the Section III-A.

III. Evaluation of the Tetrys Parameters

This section includes both analytical and experimental evaluations of Tetrys. To that purpose, we have implemented a Tetrys prototype in C language. It borrows the finite field operations from Luigi Rizzo's Reed-Solomon codec [11]. For decoding, a Gauss-Jordan matrix inversion has been developed. This algorithm is modified in order to determine, in the case of a singular matrix, the repair packet which is a linear combination of the other received packets. This useless repair packet is then discarded and the decoder waits for additional repair packets. During experiments, the coefficients for the linear combination are randomly chosen on the finite field $\mathbb{F}_{2^8}$, except in Section III-H where other finite fields are used.

A. Tetrys general analytical model

We propose in this part a model allowing to assess the key properties of the Tetrys mechanism. We assume the packet losses follow a Bernoulli law of parameter p. Under this assumption, we introduce a Markov chain: $\{Y_n, n > 0\}$, which represents the difference between the number of lost packets and the number of received repair packets observed after the reception of each repair packet. As in section Sec. II, we assume to decode when $Y_j = 0$. This assumption is valid if the finite field is chosen sufficiently large (see [14] for theoretical arguments and Section III-H for simulation results).

As a first step, we focus on the probability distribution of $\{Y_n, n > 0\}$. Then, we use this distribution to estimate the decoding delay, the average buffer size and the computation complexity of the algorithm.

The evaluation of $\{Y_n, n > 0\}$ is done after each Tetrys block. We define a block as a set of $k+1$ consecutive packets that begins at the first source packet sent after a repair packet and ends at the next repair packet. We point out that our definition of block does not correspond to the usual definition in coding theory which is a set of symbols encoded together.

In our context, a repair packet can be encoded from a set of source data packets belonging to several blocks.

The reception of each packet is represented by a random variable (r. v.) $X_{i,j}$, where $i > 0$ and $0 \leqslant j \leqslant k$. With this notation, i corresponds to the block and j to the position of the packet in the block.

On the Bernoulli channel, we have $P[X_{i,j} = 1] = p$ (the packet is lost), and $P[X_{i,j} = 0] = 1 - p$ (the packet is received). The variables $X_{i,j}$, where $0 \leqslant j \leqslant k-1$ thus corresponds to source packets and the variables $X_{i,k}$ corresponds to the repair packets. We then define the r.v. X_i, where $i > 0$, as follows:

$$X_i = \sum_{j=0}^{k} X_{i,j} - 1 \tag{3}$$

Indeed, this sum can be expressed as $X_i = \sum_{j=0}^{k-1} X_{i,j} + (X_{i,k} - 1)$. Then, the loss of one of the first k (source) packet increments the value of X_i while the reception of the repair packet decrements the value of X_i. Since X_i is obtained from a sum of Bernoulli variables, we have $P(X_i = u - 1) = \binom{k+1}{u} p^u (1-p)^{k+1-u}$ with $u = 0, \ldots, k+1$.

We then define the Markov chain $\{Y_n, n \geqslant 0\}$ as follows:

$$Y_n = \begin{cases} Y_{n-1} + X_n & \text{if } Y_{n-1} + X_n \geqslant 0 \\ 0 & \text{else} \end{cases} \tag{4}$$

Actually, the value of Y_n corresponds to the difference between the number of lost packets and the number of received repair packets since the previous decoding. Note that this value is considered at the end of each block, *i.* e. after the transmission of a repair packet.

Theorem 1: The success of the decoding and the decoding delay depend on the relationship between R and p as follows:

- if $R < p$, the recovery of a lost packet is not guaranteed;
- if $R = p$, all the lost packets are recovered, but the mean decoding delay is infinite;
- if $R > p$, all the lost packets are recovered, and the the mean decoding delay is finite;

Proof: From the definition of X_i, it can be shown that its expectation $E(X_i)$ is equal to $(k+1)p - 1 = \frac{p}{R} - 1$. If $R < p$, $E(X_i)$ is strictly positive and thus the chain is transient. Consequently, there is no guarantee to decode a lost packet.

For $R = p$, $E(X_i) = 0$ and the chain becomes null recurrent, i. e. any state can be reached, but in an infinite time. Since the state 0 corresponds to a decoding, it can be deduced that any lost packet is decoded but the mean decoding delay is infinite.

For $R > p$, $E(X_i) < 0$ and thus the state 0 is positive recurrent. This state is reached in a finite mean time and thus any lost packet is decoded in in finite decoding delay. □

Let us consider the case where $R > p$. Before studying the decoding delay in the next part, we can deduce additional informations on the decoding process from the Markov chain. Let us denote $a_{i,j} := P(Y_n = j | Y_{n-1} = i)$ the transition probabilities between the states i and j. Let us now define A the matrix $(a_{i,j})_{i,j \geqslant 0}$ and let us denote by $a_{i,j}^{(n)}$ the entries of A^n.

Proposition 1: If $R > p$, the chain $\{Y_n, n \geqslant 0\}$ admits a stationary distribution equal to :

$$P(Y_j = i) = \lim_{n \to \infty} a_{j,i}^{(n)} \tag{5}$$

for any $i, j \geqslant 0$.

Proof: Since the chain is irreducible and one state is positive recurrent, all the states are positive recurrent [17]. Thus the chain admits a stationary distribution whose values can be easily obtained with basic results in stochastic process theory [17]. □

B. Analytical model of the decoding delay

To study the decoding delay, we first need to obtain the distribution of the first hitting time. In our context, the first hitting time is denoted by H_i and is defined as follows:

$$H_i = \{\min h \text{ such that } Y_h = 0 | Y_0 = i\}$$

Intuitively, this hitting time corresponds to the time necessary to decode a packet knowing that, at the considered time, the difference between the number of lost packets and the number of received repair packets is i.

Lemma 1: The probability distribution of H_i can be obtained as follows :

$$P(H_i = h) = \frac{1}{h!} \frac{d^h (\sum_{t \geqslant 0} a_{i,0}^{(t)} z^t / \sum_{t \geqslant 0} a_{0,0}^{(t)} z^t)}{dz^h} |_{z=0} \tag{6}$$

Proof: Let us define

$$G_i(z) = \sum_{t \geqslant 0} a_{i,0}^{(t)} z^t \tag{7}$$

and

$$F_i(z) = \sum_{h \geqslant 0} P(H_i = h) z^h \tag{8}$$

the probability generating function (p. g. f.) of H_i. Following [18, chap. 2, lemma 25], we have :

$$F_i(z) = G_i(z) / G_0(z) \tag{9}$$

The probability distribution of H_i can be then obtained from the probability generating function by evaluating:

$$P(H_i = h) = \frac{1}{h!} \frac{d^h F_i(z)}{dz^h} |_{z=0}. \tag{10}$$

Combining Equations 7, 9 and 10 allows to obtain the expression of the probability distribution of H_i. □

Since this Markov chain concerns the decoding delay at the block level, we now need to refine the analysis at the packet level. Let us consider that a packet sent in position j ($j = 0, \ldots, k-1$) of a block i is lost. Let D_j be its decoding delay. This delay has necessarily the form $k - j + h(k+1)$ because the decoding can only be performed at the reception of a repair packet.

Proposition 2: The decoding delay of a packet sent in position j of a block has the following distribution :

$$P(D_j = k - j + h(k+1)) = \sum_{y \geqslant 0} \sum_{u=0}^{k} \binom{k}{u} p^u (1-p)^{k-u} P(H_{y+u} = h) P(Y_{i-1} = y) \tag{11}$$

Proof: Recall that Y_{i-1} and Y_i are the r. v. representing the states of the chain $\{Y_n, n \geqslant 0\}$ after the previous block and at the end of the current block.

Since the packet sent in position j is lost, we have:

$$P(Y_i = y + u | Y_{i-1} = y) = \binom{k}{u} p^u (1-p)^{k-u} \quad (12)$$

for $u = 0, \ldots, k$. We also have:

$$\begin{aligned} &P(D_j = k - j + h(k+1)) \\ &= \sum_{y \geqslant 0} \sum_{u=0}^{k} P(D_j = k - j + h(k+1), \\ &\qquad Y_{i-1} = y, Y_i = y + u) \\ &= \sum_{y \geqslant 0} \sum_{u=0}^{k} P(D_j = k - j + h(k+1) | Y_i = y + u) \\ &\qquad P(Y_i = y + u | Y_{i-1} = y) P(Y_{i-1} = y) \\ &= \sum_{y \geqslant 0} \sum_{u=0}^{k} P(H_{y+u} = h) \\ &\qquad P(Y_i = y + u | Y_{i-1} = y) P(Y_{i-1} = y) \end{aligned} \quad (13)$$

Combining this last expression with Equation 12 allows to obtain the expression given in the proposition. □

C. Analytical model of the matrix sizes

Like most of erasure codes, the decoding operation in Tetrys basically consists in inverting a matrix defined over a finite field. The size of this matrix, denoted by Z, corresponds to the number of repair packets involved in the decoding. Compared to classic block-based erasure codes (rateless or not), the main difference is that no theoretical bounds exist on the size of the matrix that must be inverted. This is due to the concept of elastic coding window. On the other hand, thanks to the elastic coding window, it can be observed that, with a good choice of parameters, the sizes of the inverted matrices by Tetrys is most of the time lower than the matrices used by classic erasure codes. For these reasons, the study of the sizes' distribution of the inverted matrices is important.

The first step in this study is the analysis of the recurrence time. This parameter, denoted by U, is the time between the first loss after a decoding and its recovery. This time is expressed in time units, where a unit time corresponds to the delay between the transmission of two consecutive packets.

With the notations introduced in the previous section, if we consider the block where the first packet is lost after a decoding, we define the r. v. F which corresponds to the position of the first lost packet in the block. When the first lost packet occurs in position j, its recovery delay, and thus the corresponding recurrence time U has the form $k-j+h(k+1)$, where h represents the number of complete blocks included in the recurrence time. Reciprocally, a recurrence time equal to $k - j + h(k+1)$ can only be observed with a first loss at position j.

Lemma 2: The recurrence time U has the following distribution :

$$\begin{aligned} &P(U = k - j + h(k+1)) = \\ &\quad \frac{1}{1-(1-p)^k} \sum_{u=0}^{k} \binom{k-j}{u} p^{u+1} (1-p)^{k-u} P(H_u = h) \end{aligned} \quad (14)$$

Proof: Basic combinatorial arguments show that

$$P(F = j) = p(1-p)^j / (1 - (1-p)^k), \quad (15)$$

for $j = 0, \ldots, k-1$.

Since the considered packet is the first lost after the previous decoding, the value of the next Y_i is necessarily in the range $[0, k]$. Thus, we have:

$$\begin{aligned} &P(U = k - j + h(k+1)) = \\ &\sum_{u=0}^{k} P(U = k - j + h(k+1), Y_i = u | F = j) P(F = j) \end{aligned} \quad (16)$$

It follows that:

$$\begin{aligned} &P(U = k - j + h(k+1)) \\ &= \sum_{u=0}^{k} P(D_j = k - j + h(k+1) | Y_i = u) \\ &\qquad P(Y_i = u | F = j) P(F = j) \\ &= \sum_{u=0}^{k} P(H_u = h) P(Y_i = u | F = j) P(F = j) \end{aligned} \quad (17)$$

It can easily be shown that $P(Y_i = u | F = j) = \binom{k-j}{u} p^u (1-p)^{k-j-u}$. By combining this result with Equations 15 and 17, we obtain the probability distribution of U given in the lemma. □

Proposition 3: The distribution probability of Z, representing the sizes of the decoded matrices, is equal to:

$$\begin{aligned} P(Z = i) = \frac{1}{1-(1-p)^k} \sum_{h \geqslant i} \sum_{j=0}^{k-1} \sum_{u=0}^{k} \binom{h}{i} \binom{k-j}{u} \\ p^{h-i+u+1} (1-p)^{i+k-u} P(H_u = h) \end{aligned} \quad (18)$$

Proof: To obtain the matrix size Z from U, we can first observe that in a recurrence time equals to $k - j + h(k+1)$, $h+1$ repair symbols are sent. This means that the matrix size is ranging from 1 to $h+1$. By considering that the last repair symbol is necessarily received, we have:

$$P(Z = i | U = k - j + h(k+1)) = \binom{h}{i} p^{h-i} (1-p)^i \quad (19)$$

On the other hand, we have:

$$\begin{aligned} P(Z = i) = \sum_{h \geqslant i} \sum_{j=0}^{k-1} P(Z = i | U = k - j + h(k+1)) \\ P(U = k - j + h(k+1)) \end{aligned} \quad (20)$$

By combining this expression with Equations 2 and 19, we obtain the given formula. □

D. Analytical model of the buffer size

Like for the matrix sizes, the elastic coding window of Tetrys implies that there is no theoretical bounds on the number of packets stored in the buffer at the sender and receiver sides. The aim of this part is to evaluate these parameters. In this section, we consider that a packet is sent by the sender each time unit.

1) At the sender side: We denote by BS_t the number of packets stored in the buffer at time t. Basically, the buffer contains the packets that were not acknowledged. Let S_1 denotes the time between the reception of the last SACK and t. If we consider that a SACK is sent every $s.RTT$ time units and that it is lost with probability p, we have :

$$E(S_1) = s.RTT(1/2 + 1/(1-p)) \quad (21)$$

The factor $1/2$ corresponds to the average time to wait a received acknowledgment and the factor $1/(1-p)$ is the expectation of the geometrical law of parameter p representing the arrival of the last SACK.

This acknowledgment brings out the information on the reception of the packet sent by the sender one RTT ago. Thus,

the sender has to store the $RTT.k/(k+1)$ source packets sent during this period.

Finally, at the time $t - S_1 - RTT$, some source packets were not acknowledged because they were lost. Thanks to the use of the *ack-when-seen* mechanism (included in the SACK mechanism), each received repair packet acknowledges a lost source packet. Thus, the number of not acknowledged source packets is the difference between the number of lost source packets and the number of received repair packets, which is represented by the r. v. Y_n studied in Section III-B.

The average number of packets stored in the buffer is thus:

$$E(BS_t) = RTT(k/(k+1))(s/2 + s/(1-p)) + E(Y_n) \quad (22)$$

Since the RTT does not impact on the value of $E(Y_n)$, we can observe that, when we fix the other parameters (p, k and s), the number of packets in the sender buffer is a linear function of the RTT. This observation also holds for the parameter s representing the SACK frequency.

2) At the receiver side: The receiver has two buffers: the source buffer, which contains the received source packets necessary for future decoding and the repair buffer, which contains the received repair packets not yet decoded. The number of packets in the source buffer at the time t is denoted BRS_t and the number of packets in the repair buffer is denoted BRR_t.

We recall that, when a source packet is received by the receiver, it is acknowledged in the future SACKs. When the sender received the first of these SACKs, it deletes this source packet in its buffer and does not include it in the generation of the next repair packets. The receiver can delete this source packet as soon as it received a repair packet which does not include this source packet in its linear combination.

As shown in Fig. 3, it follows that the source packet is stored in the buffer during $S_2 + S_3 + RTT$, where $S_2 + RTT/2$ is the time needed by the sender to receive the first acknowledgment and $S_3 + RTT/2$ is the time needed by the sender to receive the next repair packet.

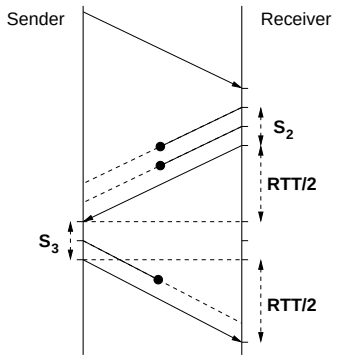


Fig. 3. Receiver buffer

Clearly, S_2 follows the same law than S_1. For S_3, the same method can be used to estimate the mean, excepted that a repair packet is sent each $k+1$ time units (instead of $s.RTT$ for the SACKs).

The average time spent by a source packet in the buffer is then:

$$E(S_2 + S_3 + RTT) = RTT + (k+1+s.RTT)(1/2 + 1/(1-p))$$

To obtain the number of packets stored in the buffer at a given time, we must consider that some of these packets are lost. Thus we have:

$$\begin{aligned} E(BRS_t) &= (k/(k+1))(1-p)E(RTT + S_2 + S_3) \\ &= (k/(k+1))(1-p)RTT + (k+1+s.RTT) \\ &\quad ((1-p)/2 + 1) \end{aligned}$$

To estimate the number of repair packets in the repair buffer, we can first estimate the probability of having no repair packet in the buffer. This probability is equal to $P(Y_n = 0)$ determined in Section III-B.

When there is at least one packet in the repair buffer, we can consider the probability distribution of the recurrence time U. Indeed, for $U = k - j + h(k+1)$, h repair packets are sent and we can estimate that, on average, $(1-p)h$ repair packets are received. It follows that the average number of packets in the buffer during this period is $(1-p)h/2$. We then have:

$$E(BRR_t) = \frac{\sum_{h>0}(1-p)h \sum_{j=0}^{k-1}(k-j+h(k+1))P(U=k-j+h(k+1))}{2.P(Y_n=0)}$$

Following this model, we can assess the minimum buffer size requested by Tetrys. In addition, source-based algorithms can also be envisaged to prevent buffer overflow.

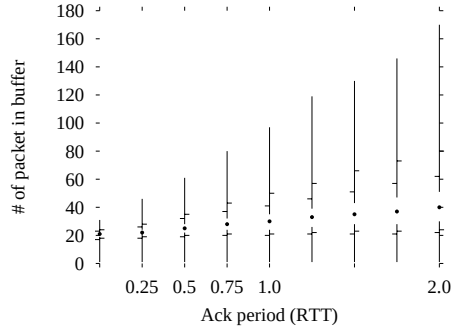
E. Experimental evaluation of the buffer size

In order to give an insight of the Tetrys requirements in a typical case, we evaluate the data source receiver buffer (BRS_t) evolution using our Tetrys prototype. We report only experiments over a Bernoulli channel[2] for the receiver's buffer as the receiver's buffer occupancy is always bigger than the sender. The RTT, repair ratio and sending rate are respectively set to $200ms$, $(3/4)$ and 100 packets per seconds. The two parameters that might affect the requested buffer sizes are the acknowledgment frequency (as presented Section II) and the PLR. We studied in Fig. 4(a) the impact of the acknowledgment frequency on the requested buffer size. Experiments are done with a fixed loss rate (10%). For the sake of completeness, we show the minimum, maximum and the (5, 10, 25, 50, 75, 90, 95) percentiles (the 50 percentile is the buffer size of the 50% highest buffer sizes) of the number of packets in buffer during the experiment. The samples used to compute these percentiles are selected at the reception of each data or repair packets.

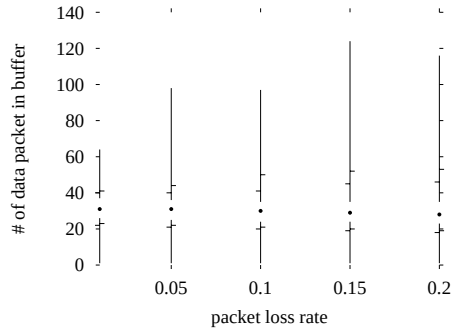
We can see that with one acknowledgment sent per packet, one per RTT and one for two RTT the 50th percentile are respectively around 20, 30 and 40 packets. The points in Fig. 4(a) also give the mean value which overlaps the 50th percentile. This confirms that as $E(BRS_t)$ suggests, the average number of packets kept in the buffer evolves linearly with the acknowledgment frequency.

The other parameter of interest is the PLR, since we have seen that when its value is closed to the repair ratio, the

[2]The results are in the same order of magnitude with bursty losses, using a Gilbert-Elliott channel.



(a) Number of data source packets in the receiver's buffer as a function of the acknowledgement frequency for a PLR=0.1



(b) Number of source data packets in the receiver's buffer as a function of the PLR for one acknowledgement sent per RTT

Fig. 4. Minimum, maximum and (5, 10, 25, 50, 75, 90, 95) percentiles of the number of packets requested to decode with a 3/4 repair ratio for Tetrys

recurrence time increases. Fig. 4(b) presents the result with an acknowledgment frequency of 1 and shows the number of packets in the buffer for a PLR varying from 1% to 20%. We can see that the (5, 10, 25, 75, 90, 95) percentiles remain close to their 50 percentile, implying a low number of packets in the buffer (most of the time around $30 \sim 40$ for one acknowledgement per RTT) and a reasonable peak size (a maximum of 160 packets) the rest of the time.

F. Tetrys encoding/decoding complexity analysis

This section introduces a complexity analysis of Tetrys operations, expressed in terms of the number of operations performed on packets. For example, the multiplication of a packet by a finite field coefficient or the XOR addition of two packets are considered as one operation.

1) Encoding Complexity: This complexity corresponds to the number of operations needed to generate one repair packet. Following the main principle of Tetrys, the number of source packets involved in the linear combination is the number of packets not acknowledged, i.e. the number of source packets in the buffer of the sender. The number of additions and multiplications performed to generate a repair packet at time t is exactly BS_t. An analytical expression of this parameter is given in Equation 22. Following discussions of Section III-D, for fixed packet loss rate and redundancy ratio, this complexity is linear according to the RTT and to the SACK frequency.

2) Decoding Complexity: The decoding process can be split into two separate processes. The first one is a continuous process which consists in subtracting all the available source packets (received or decoded) to the repair packets in which they are involved. The second one is the core decoding process which allows to recover a set of Z source packets from a set of Z repair packets. As explained before, the $Z \times Z$-matrix built from the finite field symbols used to generate the repair symbols is inverted and the obtained matrix is multiplied to the vector of repair symbols to recover the source symbols.

To evaluate the complexity of the first process, it is sufficient to estimate the number of available source packets in the source buffer of the receiver. This quantity, BRS_t, is studied in Section III-D. Figures 4(a) and 4(b) confirm these results with simulation results showing the evolution of the buffer size, and thus of this complexity, for typical parameters.

For the second process of the decoding operation, the decoder has to invert a matrix of size Z and then to multiply the $Z \times Z$-inverted matrix by the vector of Z repair packets. The matrix-vector multiplication only perform Z operations on each repair packets. The inversion of a general matrix has a cubic complexity, but it is done on finite field coefficients and not on packets. In practical, when the entries of the matrix are carefully chosen, it can be shown that this matrix inversion does not strongly impact on the decoding speed for moderate values of Z.

The distribution of the parameter Z was analytically studied in Section III-C for the Bernoulli channel. Simulation results obtained for typical parameters perfectly fit these theoretical estimations (see Figure 5). For a Gilbert-Elliott (GE) channel, additional simulations presented on Figure 7 show the behavior of the Z parameter on bursty channels.

To have roughly estimations of the practical decoding speed, Tetrys decoding can be compared to a block code decoding with dimension equal to Z. In Fig. 5 and 7, the highest average matrix size is equal to 14. As a result, we can compare the cubic complexity of the matrix inversion process to an erasure code of equivalent dimension defined over a non-binary finite field such as Reed-Solomon. If we now consider the subtraction process of source symbols from redundancy packets, Tetrys could be compared to common Reed-Solomon code of dimension 32 (assuming the source data buffer size from Fig. 4(b)). To roughly have an order of magnitude, the authors in [19] show that several implementations of Reed-Solomon code of dimension 32 can reach a decoding speed up to 600 Mbps with a standard personal computer. As a result, Tetrys is perfectly compliant with real-time video constraints both in terms of computation overhead and memory footprint (also practically observed with our real prototype).

G. Experimental analysis of the impacts of the channel type

1) Case of a Bernoulli channel: impact of the PLR: We first consider the impact of the PLR, p, using a simple Bernoulli channel model, on Tetrys performance. In Fig. 5, the Tetrys performance in terms of average matrix size, decoding delay, and recurrence time is illustrated as a function of the PLR, using a Bernoulli channel, when $R = 0.25$. The first y-axis

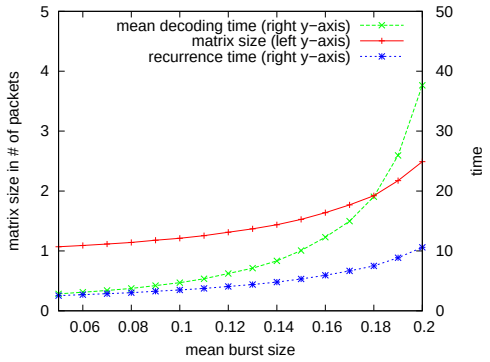


Fig. 5. Average matrix size, decoding delay and recurrence time as a function of the PLR, p, using a Bernoulli channel model.

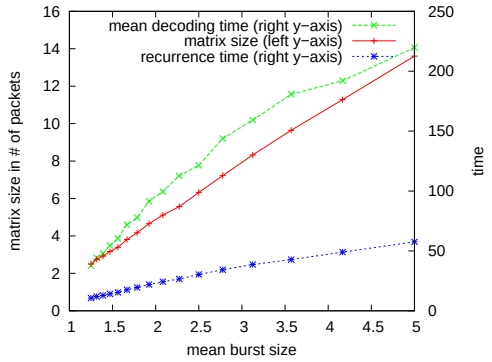


Fig. 7. Average matrix size, decoding delay and recurrence time as a function of the average loss burst size, using a GE channel model.

scale (left side) is expressed in number of packets and is used for the average matrix size. The second y-axis scale (right side) is expressed in time units and is used for the average decoding delay and average recurrence time (recall that a time unit corresponds to the delay between the transmission of two consecutive packets).

The first observation is that the three curves increase with the PLR. This is easily explained by the fact that when the error probability is small compared to R, then decoding happens quickly, and vice-versa. This is also in line with a previous result showing that he average recurrence time is equal to $1/(R-p)$ and thus, is infinite when $R = p$. The second observation is that the average decoding delay curve gets higher than the recurrence time curve. This can be explained by the fact that the decoding delay is related to packet while the recurrence time is related to decoding. In the case of a large "recurrence walk", a large number of packets have a large decoding delay, and thus this walk has a larger influence on the average decoding time than on the average recurrence time.

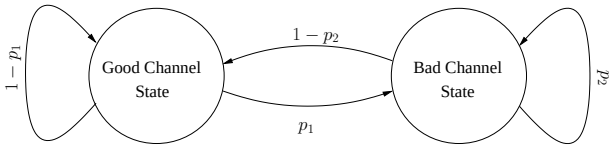


Fig. 6. The first-order two-state Markov chain representing the Gilbert-Elliott channel model

2) Case of a Gilbert Elliot channel: impact of the average loss burst size: We now consider the impact of loss bursts on Tetrys performance, using the well-known first-order, Gilbert-Elliott channel model (Fig. 6). With this model, which considers two input probabilities, p_1 and p_2, it is well known that the mean PLR is equal to $p = p_1/(1+p_1-p_2)$ and the average loss burst size to $1/(1-p_2)$. Thus: $p_2 = 1 + p_1 - p_1/p$.

Fig. 7 shows the Tetrys performance, using the same metrics as before, as a function of the average loss burst size, when $R = 0.25$. During the tests, p_1 and p_2 vary in such a way that the mean PLR is kept constant, equals to 0.2.

Compared to Fig. 5, the curve representing the average loss burst size (equal to $1/(1-p_2)$) is added. We can observe that a small value of p_1 implies a large value of p_2 and thus a large mean burst size. On the opposite, when $p_1 = p_2$, the Markov channel becomes a Bernoulli channel of parameter p_1 and thus, the mean burst size reaches its minimum.

The main information of the Fig. 7 is that the burst losses have a negative impact on Tetrys performance. We can observe that when p_1 varies from 0.1 to 0.2, the burst size varies from 2.5 to 1.25. In this range, the matrix size, mean decoding time and recurrence time are also divided by 2.

Even this rate of 2 is very specific to this simple example, more generally, we can observe that the only consequence of bursts is the increase of the decoding delay, recurrence time and of the matrix size at the decoder side. Indeed, the property to decode all packets if $R > p$ remains true.

Note that in the case of channels with variable parameters (with a fixed PLR), Tetrys adapts automatically to the variable conditions without any external intervention.

H. Experimental analysis of the impact of the finite field size

Section II says that decoding is not necessarily possible as soon as the number of received repair packets is equal to the number of lost source packet. This is explained by the fact that the corresponding matrix can be singular (i.e. non invertible). In this case, the receiver must wait additional repair packets, which increases both the decoding delay and the matrix size. In this section we analyze the impacts of the finite field size (over which the coefficients used to build the repair packets are randomly chosen) on these performance metrics.

More precisely we carried out experiments where the finite field size varies from 2 to 2^8, with $PLR = 0.15$ and $R = 25$, with either a Bernoulli or Gilbert Elliott channel. The results are plotted in Fig. 8.

The main result is that the two smallest finite fields ($\mathbb{F}_2$ and $\mathbb{F}_{2^2}$) lead to poor performances in terms of decoding delay and matrix size to invert. Even if the binary field ($\mathbb{F}_2$) is attractive because all operations are implemented with extremely fast XORs operations, this field must be avoided in our case. The best compromise seems to be the field $\mathbb{F}_{2^3}$ which obtains excellent decoding performance while supporting very fast operations. The decoding performance differences between $\mathbb{F}_{2^3}$ and larger finite fields are relatively negligible for both channels. This observation remains true for other loss patterns.

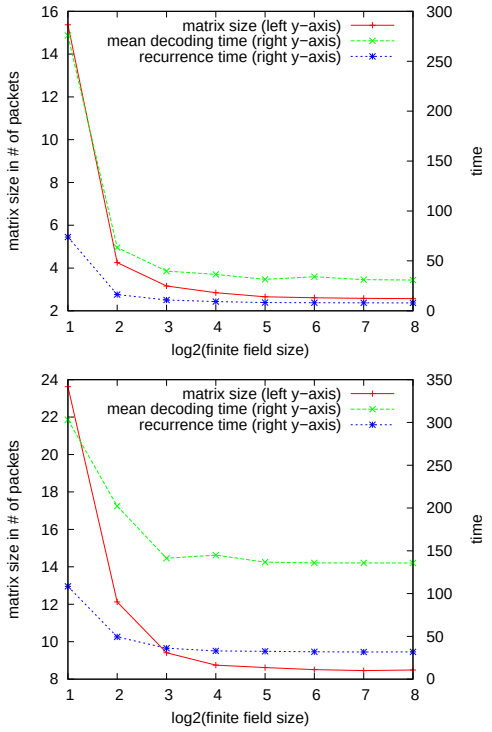


Fig. 8. Impact of the finite field size on the average matrix size, decoding delay and recurrence time, using a Bernoulli channel (top) or Gilbert Elliott channel (bottom). PLR=0.2, average loss burst size of 3 (GE channel case), and R=0.25.

We therefore suggest to always use $\mathbb{F}_{2^3}$. Additionally [20] explains that a multiplication in the field $\mathbb{F}_{2^m}$ (in our case $m = 3$) can be implemented on average with $m/2$ XOR operations per data unit (in our case $3/2$) which can be a useful way of mitigating the processing load of operations over $\mathbb{F}_{2^8}$.

IV. On the robustness of Tetrys versus FEC block codes in dynamic environments

This section compares Tetrys with another usual loss recovery scheme, namely FEC block codes, focusing on the decoding delay metric, a key performance metric with real-time multimedia applications. In particular, this section emphasizes the simplicity of Tetrys configuration (controlled by a single parameter) and the stability of the performance achieved as the network conditions change.

A. Comparison with FEC block codes

FEC block codes for the erasure channel are a usual way of mitigating packet losses. For instance the IETF FECFRAME working group[3] aims at defining a generic framework between the RTP and UDP protocols to plug various FEC block codes in a very flexible way, to protect one or several application flows, separately or together. The FEC Framework architecture being defined [21] is similar to the robust streaming solution that can be found for instance in the 3GPP MBMS or DVB IP Datacasting services [22]. Rather than focusing on a particular FEC block scheme (e.g. the Raptor codes used in the 3GPP or DVB streaming services [22] or one of the codes considered in [23]), we consider an MDS FEC code, i.e. a code optimal in terms of correction capabilities. Note that, even if Raptor codes are often used in streaming services, their rateless feature is totally useless in these environments ([21] (Section 8.1) forbids the code rate to be lower than 0.5). Similarly the large block feature of Raptor codes is totally useless in these environments, because of the application real time constraints.

In the remaining of this paper, the term "FEC scheme" will refer to the streaming solution, compliant with the FEC Framework architecture, using an MDS FEC block code. The exact nature of the code is irrelevant, we just know that practical codes will not perform better than the one we are considering in our tests.

This FEC scheme works as follows. Source packets are sent as soon as the application makes them available. Then, after the transmission of the k source packets, $n - k$ FEC repair packets are sent (instantaneously). Since we want to compare Tetrys with the best FEC scheme, we assume that the link bandwidth is sufficiently important to absorb the burst resulting from the introduction of these $n - k$ repair packets.

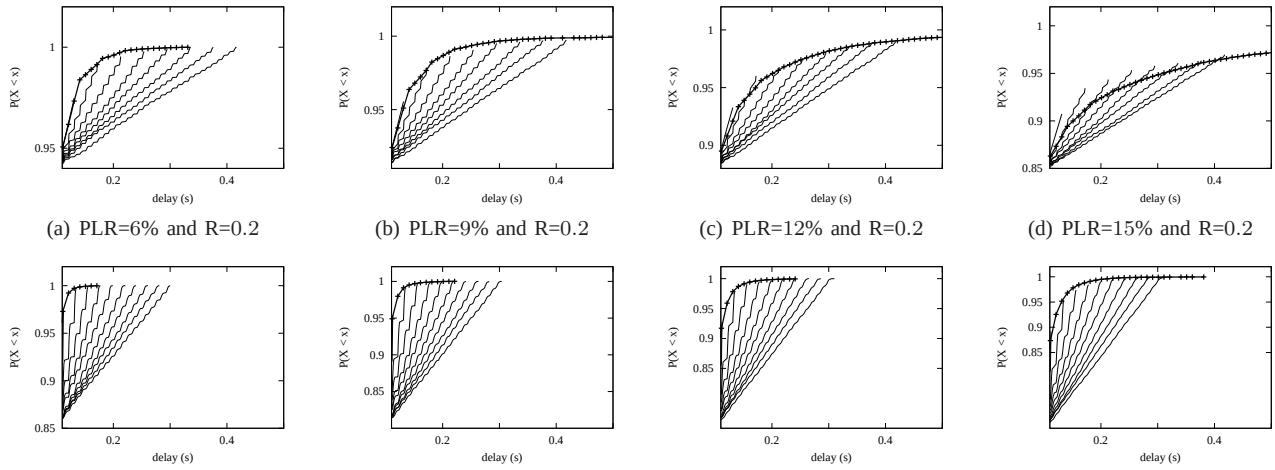
This approach faces two main limits: First of all, because of its per-block approach, the recovery of lost packets is only possible at the end, when at least k packets have been received for this block. This of course introduces a delay that depends on the chosen k parameter: the larger the k value, the better in terms of erasure recovery, but the higher the decoding delay, and the real-time feature of the application anyway incurs an upper limit to k. On the opposite Tetrys repair packets are uniformly spread among source packets. Therefore lost packets may be recovered without waiting for the end of a fixed length block and without any dependence on the RTT.

Additionally, in real conditions, the PLR is not constant over the time and two key parameters of the FEC scheme, namely the block size (k) and the code rate (k/n), should be adapted appropriately. Unfortunately, this adaptation requires feedback information which is, by definition, constrained by the RTT. Thus, the information is always returned at least one RTT later and might not reflect the current network state. As a result, the FEC parameters effectively used by the FEC scheme are not necessarily optimal. On the opposite, Tetrys is controlled by a single parameter and we will show in the following section that it is highly tolerant to varying network conditions.

B. Decoding delay performance evaluation

We carried out several tests to compare Tetrys to various FEC scheme configurations, i.e. different k and n values, in a Bernoulli channel. Considering many FEC scheme configurations is important since we do not have any reliable way to identify a priori the best FEC scheme configuration in a given channel. The results are depicted in Fig. 9. The redundancy ratio is set either to $R = 0.2$ (i.e. code rate=0.8) (upper row) or $R = 0.5$ (i.e. code rate=0.5) (lower row). Then, in each figure, there are as many FEC scheme curves as there are possible k values, while keeping the target R (which defines n). The PLR is then progressively increased to approach the R parameter.

[3]See http://www.ietf.org/dyn/wg/charter/fecframe-charter.html



(a) PLR=6% and R=0.2 (b) PLR=9% and R=0.2 (c) PLR=12% and R=0.2 (d) PLR=15% and R=0.2

(e) PLR=15% and R=0.5 (f) PLR=20% and R=0.5 (g) PLR=25% and R=0.5 (h) PLR=30% and R=0.5

Fig. 9. Cumulative Distribution Functions (CDF) of packets delivery delay for Tetrys (bold curve) and FEC (multiple staircase-like curves, corresponding to various block size configurations), for different packet loss rates and different R values (0.2 (upper row) vs. 0.5 (lower row)). The RTT is set to $200ms$ and the FEC scheme block size is set to k={4; 8; 12; 16; 20; 24; 28; 32} for the upper row (resp. k={2; 4; 6; 8; 10; 12; 14; 16; 18; 20} for the lower row).

For a given code rate we see that in all the studied cases, Tetrys provides full reliability as the CDF tends to one (but this is not the main goal). This is not the case for the different FEC schemes, essentially with short-dimension FEC codes. More importantly, the probability for Tetrys to decode below a given delay is higher than most FEC scheme configurations (i.e. the Tetrys curve is higher). When this is not the case, the FEC scheme features a lower correction capability (i.e. the curve stops earlier and never reaches 1, as in Fig. 9(d)). However, as the PLR approaches R (e.g. in Fig. 9(d)), the Tetrys recovery delay increases and the FEC schemes then overtake Tetrys.

In summary, Tetrys exhibits the same delay and resilience efficiency for most PLR, while being significantly more efficient than the best FEC scheme. The Tetrys redundancy ratio, R, only needs to be dynamically adapted when the PLR increases and be kept sufficiently high compared to the observed PLR. Since there is a single parameter, this one-dimensional problem is easily addressed. However we must point out that the main objective in this context is to reduce the recovery delay and not necessarily to optimize the bandwidth occupancy. An algorithm allowing both a dynamic adaptation of R and the minimization of the bandwidth occupancy will be introduced in Section VI.

V. Benefits of Tetrys with video-conferencing applications

A. Specificities of these applications and consequences

Video-conferencing applications have three main characteristics. First of all, the end-to-end delay must not exceed 100 ms (see [24] [25]) in order to preserve interactivity. They are also characterized by their Variable instantaneous Bit Rate (VBR). Indeed, Intracoded frames (I-frame), because they are coded from scratch, generate more data than predicted coded frames (P-frames), and even more than bipredicted frames (B-frames). Finally, losing an I-frame has, in general, a worse impact on the experienced video quality than losing a P or B-frame.

This has several impacts. First of all, FEC schemes are limited by their block size which must neither be too large (since it would impact the end-to-end decoding delay) nor too small (since it would reduce the robustness in front of loss bursts). Using both the optimal block size and redundancy ratio requires an intricate adaptation mechanism. On the opposite, Tetrys offers, as seen in Section IV, a better compromise between the decoding delay and the resilience than the best FEC scheme.

In the presence of VBR sources as video, this behavior is furthermore confirmed as FEC schemes lack adaptability compared to Tetrys. Indeed, recovering from a given number of losses means waiting for the reception of (at least) the same number of repair packets. With Tetrys, since two consecutive repair packets are spaced with k source packets, when the instantaneous packet rate increases during the transmission, the time needed to receive additional repair packets is reduced, and the probability to recover losses before the deadline increases. With video coded data, I-frames are the ones that will benefit the most from the adaptability of Tetrys. Although it could be considered only as a side effect of the Tetrys mechanism, this particularity has a major impact on the end user quality as the I-frames have the biggest weight in the video quality measure.

In this sense, Tetrys acts as an Unequal Erasure Protection (UEP) scheme such as DAUEP [26] or PET [27].

More generally, nothing would prevent the use of UEP schemes embedded in Tetrys just by allocating lower code rates to the set of important data or by nesting sources subsets. Hence, in this work we do not consider any of the FEC UEP schemes nor the Tetrys UEP schemes and let these aspects for a future work.

B. Experimental Setup

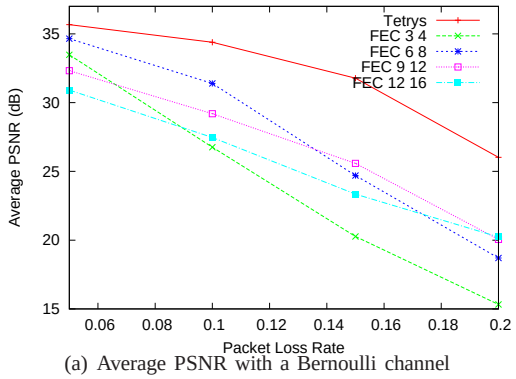
The goal of the tests is to compare Tetrys to various FEC schemes, using either a Bernoulli or GE channel model, during a video transmission. Various FEC schemes are used, of

parameters $(k, n) = (3, 4), (6, 8), (9, 12), (12, 16)$, all of them having the same code rate. We use the latest ITU-T's video codec recommendation, H.264, and the JM 15.1 H.264/AVC software [28]. We consider the Foreman sequence, in CIF size, with a frame skip of one picture, resulting in a frame rate of 15 fps. One I-frame is inserted every 14 P-frames and B-frames are not used at all because of the extra delay B-frames would generate. The average bitrate is about 384 kbps at the output of the video coder and the coded stream is packed into 500 bytes long packets. The maximum decoding tolerable delay is set to 100 ms, all the packets received after this due time being dropped. A total of 150 coded frames, corresponding to 10 seconds of video, is used. In order to obtain representative results, each sequence is repeated 20 times, leading to the transmission of a sequence composed of 3000 frames and 200 seconds long. This setup is derived from the common testing conditions mentioned in [24]. For evaluating the video we use the Evalvid framework described in [29], where the video quality is measured with the Peak Signal to Noise Ratio (PSNR) metric.

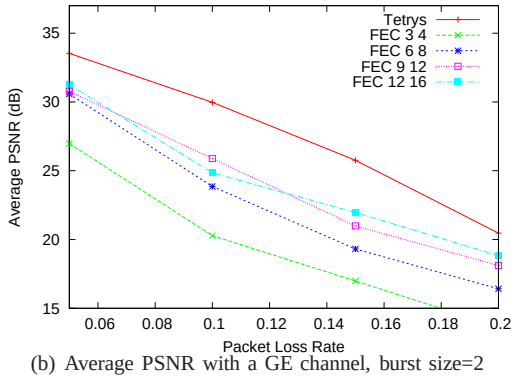
C. Video transmission performance evaluation

Let us consider the case of a Bernoulli channel first. Fig. 10(a) shows that Tetrys achieves an average PSNR gain of 7.19 dB over the best FEC scheme, namely FEC$(6, 8)$ at a PLR of 15%. The average PSNR drop for Tetrys does not exceed 4 dB when the PLR increases from 5% up to 16%, hence ensuring that the average PSNR still remains above 30 dB. When full reliability is impossible because of high time-constraints, Tetrys allows a graceful degradation of the video quality. If we consider instantaneous (rather than average) PSNR performances, a representative 10 second trace being shown in Fig. 11(bottom), Tetrys still outperforms FEC$(6, 8)$, the best FEC scheme for this scenario. Tetrys exhibits a significantly higher instantaneous PSNR, except between time 2.5 and 2.8, where the FEC scheme behaves momentarily better. By looking more carefully at the traces over this 10 seconds snapshot (not shown in the figure), we can see that Tetrys retrieved 9 I-frames out of 10, whereas FEC scheme retrieved only 5 I-frames. This behavior confirms what we said in Section V-A, namely that I-frames automatically benefit from a better protection compared to P frames with Tetrys. The reason is that Tetrys allows the use of more redundancy packets in the decoding process before the 100 ms than FEC which is constrained by its block size. As a matter of fact, if the FEC parameters were adapted with an oracle (instantaneoulsy and automatically), we should obtain similar performance than Tetrys (See Section IV for further details.). This UEP-like behavior is achieved transparently by Tetrys, without requiring any extra information exchange (data types, sizes, or importance) from the source coding application, whereas most of the existing UEP schemes do.

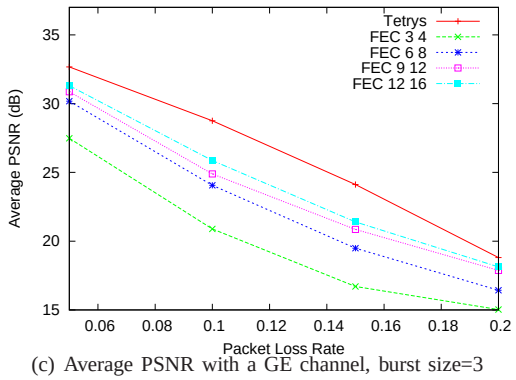
Let us now consider the case of the GE channel. The average PSNR performances, plotted in Fig. 10(b) and 10(c), show the same tendency even if the gains are less important: Tetrys still offers a 3.78 dB gain for burst length of 2 and 2.72 dB gain for burst length of 3 over the best FEC scheme.



(a) Average PSNR with a Bernoulli channel



(b) Average PSNR with a GE channel, burst size=2



(c) Average PSNR with a GE channel, burst size=3

Fig. 10. Average PSNR performance of Tetrys versus various FEC schemes during a video sequence transmission, for various channel types.

Therefore, the results achieved are unequivocal: Tetrys clearly outperforms all the tested FEC schemes in all the scenarios, in particular because of its transparent UEP-like behavior with video flows.

VI. Redundancy allocation in Tetrys under reliability and latency constraints

As for the video conference example, rather than full reliability, some multimedia applications require that a given proportion Pkt_{min} of packets arrive within a tolerable delay D_{max} (e.g. VoIP applications). After this delay, packets are considered as lost by the application although they might be delayed in the network and arrive later.

In order to verify whether the request given by an application defined by (Pkt_{min}, D_{max}) is feasible, we choose to

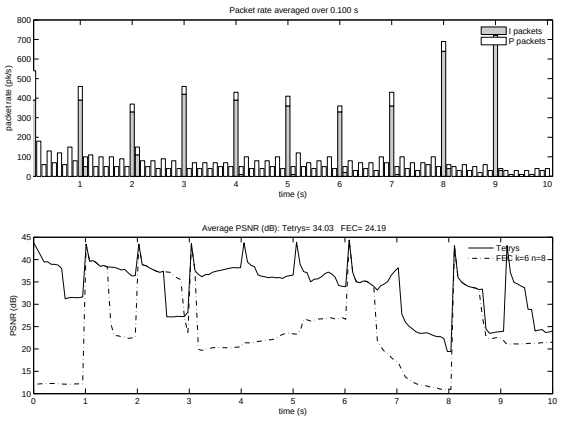


Fig. 11. Packet rate (top) and instantaneous PSNR of Tetrys versus FEC(6,8) (bottom) during a 10 second snapshot, with a Bernoulli channel and PLR=15%.

infer a Tetrys heuristic model θ following several experiments. We define this model as follows:

$$\theta(t)_{(d,p,b,T,R)} \tag{23}$$

This model gives the cumulative distribution function of the lost packets recovery delay where R is the redundancy ratio for an application that produces a packet every T seconds[4] according to the network characteristics (i.e. a delay d, a PLR p and a burstiness of losses b).

We then test the capability of Tetrys to satisfy the request (D_{max}, Pkt_{min}), given R, with a boolean function denoted $\Psi_{\theta(t)}(D_{max}, Pkt_{min})$. Ψ returns TRUE if the probability that a packet arrives before D_{max} is higher than Pkt_{min} and FALSE otherwise. As a result, by iterating R (starting from $R = p$), we find the set of solutions that satisfies the application requirements. Finally, among these possible solutions, the Tetrys sending application solves Equation (24) to find the smallest redundancy ratio needed denoted R_{min}:

$$R_{min} = min(R|\Psi_{\theta(t)}(D_{max}, Pkt_{min})) \tag{24}$$

The following sections detail the method used to build this model.

A. Model of the delay distribution

The behaviour of the Tetrys mechanism can be modeled by a Markov chain process with a random walk driven by the losses of source packets and the reception of redundancy packets. As in Section III-B, we could compute the recurrence and hitting times of the Markov chain and obtain an analytical model of θ. Unfortunately, the computational complexity of this model requires substantive computation time and prevents any implementation inside a real protocol. This motivates the use of our heuristic model θ previously introduced.

[4]We assume a Constant Bit Rate (CBR) where all the packets have the same size.

1) Experimental setup: We have performed several experiments with a redundancy ratio R ranging from 0.1 to 0.5, a PLR p ranging from 1% to 50% which follows either a Bernoulli model or a GE model with an average burst size of 2 or 3. For each experiment, 10^5 source data packets are generated.

2) Distributions fitting: We seek to estimate the delay in number of packets sent (and supposed to be received) between a lost packet and the redundancy packet that rebuild it. Following the distribution of packets recovery delay obtained by the experiments, we find out that the Weibull law fits our distribution[5].

A Weibull distribution is defined by two parameters: the scale and the shape. Such distribution captures both exponential distribution if the shape parameter κ is around 1 and the heavy tailed distribution if $\kappa < 1$ and is defined as follows:

$$P[X < x] = 1 - e^{-(x/\lambda)^{\kappa}} \tag{25}$$

B. Estimating the distribution parameters

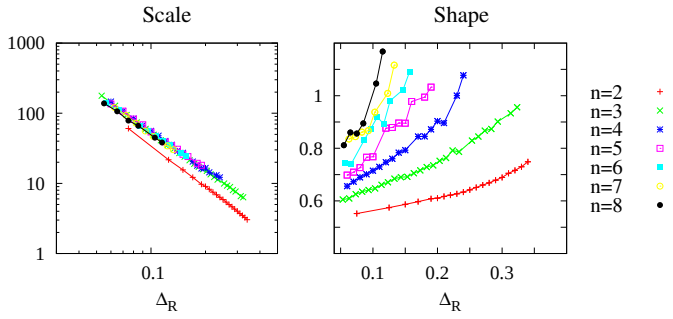


Fig. 12. Evolution of the scale (λ) and (κ) shape as a function of Δ_R

For a given loss distribution (e.g. Bernoulli or Gilbert-Elliott) the delay distribution is impacted by n ($n = k+1$) and p (as $\Delta_R = \frac{1}{n} - p$)). For each value of the block size n and each loss distribution the shape parameters evolves "linearly" as a function of Δ_R as seen in Fig. 12. The linear function coefficients obtained through a least square are stored in table II.

In the same way, the scale parameter is only impacted by n and the losses distribution. The scale can be approximated by:

$$\lambda(\Delta_R) = \frac{a_{c,n}}{{\Delta_R}^{b_{c,n}}}$$

with $a_{c,n}$ and $b_{c,n}$ the appropriate values in the table II where c is the channel (that takes the following values $1, 2, 3$ respectively for Bernoulli and Gibert-Elliott of burst size 2 or 3) and n the block size. It results that θ can be approximated by:

$$1 - e^{-(\frac{x}{\lambda(n,p,c)})^{\kappa(n,p,c)}} \tag{26}$$

with:

- $\lambda(n,p,c) = \frac{a_{c,n}}{(\frac{1}{n}-p)^{b_{c,n}}}$,
- $\kappa(n,p,c) = a_{c,n} * (\frac{1}{n} - p) + b_{c,n}$.

[5]We used R [30] statistical software environment

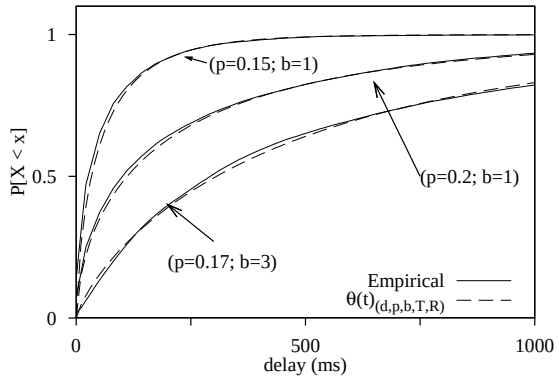


Fig. 13. Comparison between the empirical distribution obtain by experiments and $\theta(t)_{[d,p,b,T,R]}$; $T = 10ms$, $n = 3$.

Fig. 13 presents the good fitting obtained by the empirical distribution of the delay obtained by experimentation and the expected distribution obtained with θ. The results are shown for a PLR of 15% and 20% with $b = 1$ (i.e. a Bernoulli erasure channel) and a PLR with $b = 3$ (a Gilbert-Elliott losses with an average burst size of 3).

C. Accuracy of the approach

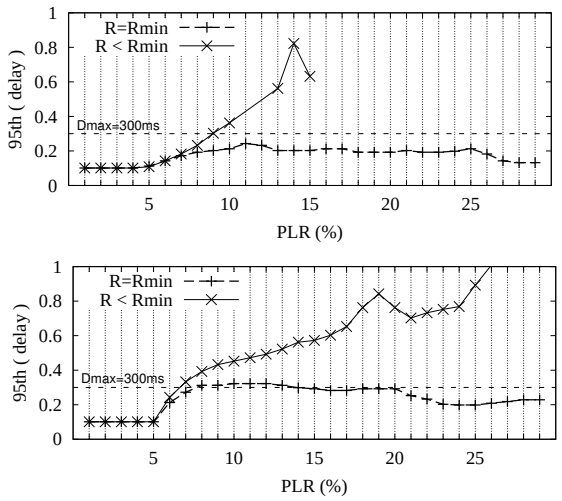


Fig. 14. Comparison between the optimal (i.e. $= R_{min}$) and suboptimal (i.e. $\leqslant R_{min}$) redundancy ratio for the Bernoulli (top) and GE with average burst size 3 (bottom) models. The metric is the 95^{th} percentile of the delay.

This mechanism has been implemented and evaluated with the ns-2 network simulator. Fig. 14 shows the results of the accuracy of R_{min} (see (24)) in a practical use case. The application emits at 100 pkt/s and requests a minimum of $(Pkt_{min}, D_{max}) = (0.95, 300ms)$ and the one-way delay is fixed to $100ms$.

The figure gives the 95^{th} percentile of the delay. According to the application requirements, it should remain below $300ms$. Considering a Bernoulli erasure channel, using $R_{min} = \frac{1}{n}$ allows to keep the 95^{th} percentile of the delay below D_{max} thus satisfying the application requirements.

Shape parameter κ							
n	2	3	4	5	6	7	8
$a_{1,n}$	0.72	1.25	2.0	2.65	3.44	3.866	5.6
$b_{1,n}$	0.473	0.51	0.512	0.525	0.53	0.55	0.46
$a_{2,n}$	0.48	1.31	1.92	2.15	3.69	5.15	4
$b_{2,n}$	0.57	0.6	0.61	0.62	0.56	0.48	0.67
$a_{3,n}$	0.62	1.8	2.8	4	4.54	5.5	5.4
$b_{3,n}$	0.65	0.61	0.57	0.53	0.6	0.62	0.72

Scale parameter λ							
n	2	3	4	5	6	7	8
$a_{1,n}$	0.83	0.35					
$b_{1,n}$	1.815	2					
$a_{2,n}$	4.2	7.15	9.9	10.48	5.6	2.7	6.3
$b_{2,n}$	1.14	1.35	1.3	1.3	1.65	1.94	1.57
$a_{3,n}$	11.8	11.4	18.2	9.3	7.1	19.1	36
$b_{3,n}$	1.04	1.44	1.3	1.6	1.7	1.28	1.05

TABLE II
TABLE OF LINEAR FUNCTION COEFFICIENTS TO GENERATE THE SHAPE AND SCALE PARAMETER.

When using $R = \frac{1}{n+1}$, the 95^{th} delay is higher than D_{max} and does not satify the application requirements. Considering a Gilbert-Elliott (GE) erasure channel with average burst of 3, the comparison between $R_{min} = \frac{1}{n}$ and $R = \frac{1}{n+1}$ remains the same. However, we observe when the loss ratio is between 8% and 12% that the 95^{th} percentile of the delay is slightly higher than D_{max}. The explanation comes from the moving average method used to compute the packet loss rate that sometimes under-estimate this value in the context of GE channel [31]. To conclude, R_{min} is effectively the smallest redundancy ratio compliant with the application requirements.

VII. CONCLUSION

In this paper we propose a novel reliability mechanism, Tetrys, based on on-the-fly erasure coding techniques. We demonstrate, through a detailed modeling of Tetrys performance as well as real measurements, that Tetrys can achieve a full reliability service even in case of an unreliable acknowledgment path (thanks to the non sensitivity of Tetrys to the loss of acknowledgments), or as the extreme case no acknowledgment at all, while ensuring faster data delivery to the application than pure FEC based techniques. In particular, we demonstrate that Tetrys offers key benefits when used in the context of video-conferencing (and more generally real-time applications) over best effort networks. In this case, the main challenge tackled by Tetrys is to combat loss and delay in order to bring a substantial gain in terms of end user perceived quality. We show that Tetrys allows a faster recovery of missing information compared to block codes, and at the same time avoids non-useful retransmitted packets. Although the contributions of this paper deal with real-time data flows, Tetrys can also be used with non real-time applications, or at a different protocol layers. We expect to investigate these considerations, as well as the interactions between Tetrys and a congestion control mechanism, in a future work.

N	1	2	3	4	5	6	7
a_{ber}	0.72	1.25	2.0	2.65	3.44	3.866	5.6
b_{be}	0.473	0.51	0.512	0.525	0.53	0.55	0.46
a_{b2}	0.48	1.31	1.92	2.15	3.69	5.15	4
b_{b2}	0.57	0.6	0.61	0.62	0.56	0.48	0.67
a_{b3}	0.62	1.8	2.8	4	4.54	5.5	5.4
b_{b3}	0.65	0.61	0.57	0.53	0.6	0.62	0.72

TABLE III
TABLE OF LINEAR FUNCTION COEFFICIENTS TO GENERATE THE SHAPE PARAMETER κ

N	1	2	3	4	5	6	7
a_{ber}	0.83	0.35					
b_{ber}	1.815	2					
a_{b2}	4.2	7.15	9.9	10.48	5.6	2.7	6.3
b_{b2}	1.14	1.35	1.3	1.3	1.65	1.94	1.57
a_{b3}	11.8	11.4	18.2	9.3	7.1	19.1	36
b_{b3}	1.04	1.44	1.3	1.6	1.7	1.28	1.05

TABLE IV
TABLE OF LINEAR FUNCTION COEFFICIENTS TO GENERATE THE SCALE PARAMETER λ

ACKNOWLEDGEMENTS

This work was supported by the French ANR grants 2006 TCOM 019 (CAPRI-FEC project) and ANR-09-VERS-019-02 (ARSSO project).

REFERENCES

[1] B. Ganguly, V. Subramanian, S. Kalyanaraman, and K. Ramakrishnan, "Performance of disruption-tolerant network mechanisms applied to airborne networks," in Military Communications Conference, 2007. MILCOM 2007. IEEE, Oct. 2007, pp. 1–7.

[2] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes," RFC 5510 (Proposed Standard), Apr. 2009.

[3] V. Roca, C. Neumann, and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes," RFC 5170 (Proposed Standard), June 2008.

[4] A. Shokrollahi, "Raptor codes," IEEE/ACM Transactions on Networking, vol. 14, no. SI, pp. 2551–2567, 2006.

[5] J. Korhonen and P. Frossard, "Flexible forward error correction codes with application to partial media data recovery," Signal Processing: Image Communication, vol. 24, pp. 229–242, 2009.

[6] E. Martinian and C.-E. W. Sundberg, "Burst erasure correction codes with low decoding delay," IEEE Transactions on Information Theory, Oct. 2004.

[7] A. Sahai, "Why do block length and delay behave differently if feedback is present?" IEEE Transactions on Information Theory, vol. 54, no. 5, pp. 1860–1886, May 2008.

[8] J. K. Sundararajan, D. Shah, and M. Médard, "ARQ for network coding," IEEE International Symposium on Information Theory (ISIT), pp. 1651–1655, July 2008.

[9] J. Lacan and E. Lochin, "Rethinking reliability for long delay networks," in International Workshop on Satellite and Space Communications (IWSSC'08), Toulouse, France, Oct. 2008.

[10] P.-U. Tournoux, A. Bouabdallah, J. Lacan, and E. Lochin, "On-the-fly coding for real-time applications," in ACM Multimedia 2009 Systems Track, Beijing, China, 2009.

[11] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," ACM Computer Communication Review, Apr. 1997.

[12] D. F. Carta, "Two fast implementations of the "minimal standard" random number generator," Communications of the ACM, vol. 33, no. 1, pp. 87–88, 1990.

[13] R. Hutchinson, R. Smarandache, and J. Trumpf, "On superregular matrices and MDP convolutional codes," Linear Algebra and its Applications, vol. 428, no. 11-12, pp. 2585 – 2596, 2008.

[14] J. Kahn and J. Komlós, "Singularity probabilities for random matrices over finite fields," Combinatorics, Probability and Computing, vol. 10, pp. 137 – 157, Oct. 2001.

[15] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," IETF, Request For Comments 2018, Oct. 1996.

[16] S. Landström and L.-A. Larzon, "Reducing the TCP acknowledgment frequency," SIGCOMM Computer Communication Review, vol. 37, no. 3, pp. 5–16, 2007.

[17] D. R. Cox and H. D. Miller, The Theory of Stochastic Processes. London, UK: Chapman & Hall, 1965.

[18] D. J. Aldous and J. A. Fill, "Reversible Markov Chains and Random Walks on Graphs," book in preparation: http://www.stat.berkeley.edu/~aldous/book.html.

[19] A. Soro and J. Lacan, "Fnt-based reed-solomon erasure codes," in Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE, jan. 2010, pp. 1 –5.

[20] J. Bloemer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-based erasure-resilient coding scheme," 1995, technical report TR-95-048, International Computer Science Institute, Berkeley, California.

[21] M. Watson, Forward Error Correction (FEC) Framework, July 2010, IETF FECFRAME Working Group, work in progress: <draft-ietf-fecframe-framework-09>.

[22] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, "Reliable multimedia download delivery in cellular broadcast networks," IEEE Transactions on Broadcasting, vol. 53, no. 1, Mar. 2007.

[23] K. Matsuzono, J. Detchart, M. Cunche, V. Roca, and H. Asaeda, "Performance analysis of a high-performance real-time application with several AL-FEC schemes," in 35th IEEE Conference on Local Computer Network (LCN'10), Oct. 2010.

[24] S. Wenger, "H.264/AVC over IP," IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 645–656, 2003.

[25] F. A. Tobagi and I. Dalgic, "Performance evaluation of 10base-T and 100base-T ethernets carrying multimedia traffic," IEEE Journal on Selected Areas in Communications, vol. 14, no. 7, pp. 1436–1454, 1996.

[26] A. Bouabdallah and J. Lacan, "Dependency-aware erasures protection codes," Journal of Zhejiang University (JZUS) - Science A, vol. 7 (Suppl. 1), pp. 27–33, 2006.

[27] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," IEEE Transactions on Information Theory, vol. 42, 1996.

[28] H.264/AVC JM Reference Software, http://iphome.hhi.de/suehring/tml/.

[29] J. Klaue, B. Rathke, and A. Wolisz, "Evalvid - A framework for video transmission and quality evaluation," in 13th International Conference of Computer Performance Evaluations, Modelling Techniques and Tools, vol. 2794, Urbana, IL, USA, Sept. 2003, pp. 255–272.

[30] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2009.

[31] F. Agharebparast and V. Leung, "A new traffic rate estimation and monitoring algorithm for the qos-enabled internet," in IEEE Global Telecommunications Conference, vol. 7, Dec. 2003.