Memory Consumption Analysis for the GOE and PET Unequal Erasure Protection Schemes

Aline Roumy*, Vincent Roca*, and Bessem Sayadi†
*INRIA, France, {firstname.name}@inria.fr
†Alcatel Lucent - Bell Labs, France, {firstname.name}@alcatel-lucent.com

***Abstract*—Unequal Erasure Protection (UEP) is an attractive approach to protect data flows that contain information of different priority levels. The various solutions that have been proposed can be classified into three families. The first one consists of specific, UEP-aware FEC codes, that map the information dependency within the code structure. It enables to design a specific solution, valid for a specific data flow, as in [3]. However, because this is a specific solution, its practical interest is also narrowed. In this work we focus on two additional solution families. One family implements UEP thanks to a dedicated packetization scheme, as it is the case with Priority Encoding Transmission (PET) [2], while the other family uses a dedicated signaling scheme, as is the case with the Generalized Object Encoding (GOE) [6]. These two solutions have the main benefit of being compatible with existing standardized Application Layer FEC (AL-FEC) schemes, which is a major practical benefit.**

Through a careful modeling of both proposals, we have demonstrated that the protection performance of both approaches are equivalent [7]. However additional key differences become apparent when considering such a practical metric as the peak memory consumption. Thanks to a modeling of the packet storage behavior at the receiver side, and by considering two major parameters, namely the channel loss probability and the permutation type, we show that the PET scheme is equivalent to the average uniformly permuted GOE scheme. We also show that, when no permutation is used, GOE allows to further reduce the peak memory with respect to PET.

I. Introduction

Unequal Erasure Protection (UEP) is an attractive approach to protect data flows that contain information of different priority levels. The various solutions that have been proposed can be classified into three families. The first one consists of specific, UEP-aware FEC codes, that map the information dependency within the code structure. It enables to design a specific solution, valid for a specific data flow, as in [3]. However, because this is a specific solution, its practical interest is also narrowed.

In this work, we focus on two additional solution families. One family implements UEP thanks to a dedicated packetization scheme, as it is the case with Priority Encoding Transmission (PET) [2], while the other family uses a dedicated signaling scheme, as is the case with the Generalized Object Encoding (GOE) [6]. These two solutions have the main benefit of being compatible with existing standardized Application Layer FEC (AL-FEC) schemes, which is a major practical benefit.

The PET scheme is an elegant solution. However this is also a complex scheme. It features many constraints that make its use in practical systems difficult as we have shown in [7]. On the opposite GOE is based on a simple solution that consists in encoding each priority class with an appropriate code rate. From a practical point of view, GOE defines an UEP-aware signaling mechanism that creates a mapping between the original object, composed of several priority classes, and the so-called "Generalized Objects" (GO) that correspond to each priority class. FEC encoding is performed independently for each GO. We have demonstrated in [7] that GOE provides the same UEP protection as PET, while being simple to understand, implement and use in practical systems.

The goal of this paper is to focus on one key practical aspect, namely the peak memory requirements during the decoding process (receiver), a topic that was missing in our previous work [7]. This metric can be of high importance for UEP deployments on lightweight terminals (e.g. smartphones), with limited processing and storage capabilities. Thanks to a careful analytical modeling of PET and GOE, we show in particular that GOE without permutation largely outperforms PET from this point of view.

In Section II we shall present the two implementations of the UEP scheme, namely PET and GOE. We shall compare the coding metrics of the two schemes (number of coded packets, coding rate, amount of uncoded and coded data, ...). This gives a basis for comparing the peak memory usage at the decoder for both schemes. Then, Section III analyzes the memory requirements of the PET scheme as a function of time. This allows to derive the peak memory usage as a function of the erasure probability. In Section III we use different criteria to analyze the memory requirements of the GOE scheme. In fact, we derive the peak memory usage as a function of the erasure probability and as a function of the interleaver.

II. Two Unequal Erasure Protection schemes

Let us first introduce some notations. We consider an input object that needs to be unequally protected. Without loss of generality, we use the term "original object" to denote this input object and "object" (or "Generalized Object" in case of GOE) to denote each data chunk of a given priority in the "original object". Each object is therefore FEC encoded as appropriate and independently from other objects with both PET and GOE. The input parameters are:

- m, the original object length, in bytes;
- d, the number of objects;
- α_i, $i = 1, ..., d$, the length in bytes of the object of index i;
- ρ_i, $i = 1, ..., d$, the desired coding rate for the object of index i.
- l, the target packet length in bytes. This size is usually determined by the maximum transmission unit size on the

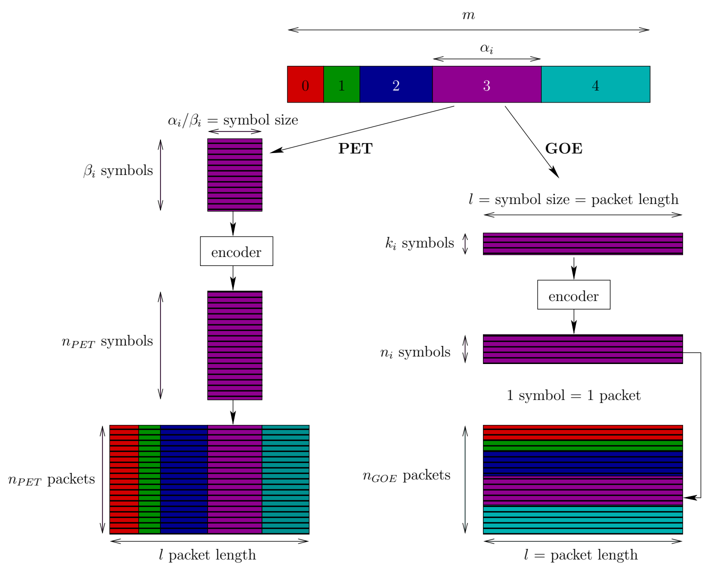

Fig. 1. PET vs GOE principles.

path between the source and the receiver(s), minus the protocol header sizes;

A. PET Principles

Goal: PET provides UEP in such a way that each packet is interchangeable in order to provide a deterministic erasure recovery behavior.

Detailed procedure: The number n_{PET} of packets and the number and size of symbols for each object need to be determined prior to FEC encoding. This is achieved as follows (Fig. 1):

- Determine the total number of packets: In [2], due to round-off errors, this number is given by:

$$n_{PET} = \left\lceil \frac{g}{l-d} \right\rceil = \left\lceil \sum_{i=1}^{d} \frac{\alpha_i}{\rho_i(l-d)} \right\rceil \quad (1)$$

  In this formula g stands for the girth, or more precisely the sum of all the encoded object lengths when the target coding rate of each object, ρ_i, is strictly respected: $g = \sum_{i=1}^{d} \frac{\alpha_i}{\rho_i}$.
- Determine the number of source symbols in object i, β_i:

$$\beta_i = \lceil n_{PET}\rho_i \rceil \quad (2)$$

- Determine the size of symbols in object i: $s_i = \left\lceil \frac{\alpha_i}{\beta_i} \right\rceil$. Note that the last symbol of an object, when shorter, is zero padded.
- FEC encode each of the d objects, thereby producing n_{PET} encoding symbols. For object i, the n_{PET} encoding symbols of index 1 to β_i are source symbols whereas the remaining $n_{PET} - \beta_i$ are repair symbols.
- Build packet j, $j = 1,..,n_{PET}$ by concatenating the j^{th} encoding symbol of each of the d objects. It can be verified that the sum of the d symbol sizes is less or equal to l, the target packet size.

B. GOE Principles

Goal: GOE provides UEP in a simple and flexible way while considering such practical aspects as minimizing the number of data copies, the number of FEC encodings, the maximum memory requirements or the number of packets to process.

Detailed procedure: GOE works as follows (Fig. 1):

- Segment each object i into k_i source symbols of length l bytes each (except the last one which may be shorter). We have: $k_i = \left\lceil \frac{\alpha_i}{l} \right\rceil$
- FEC encode each object into n_i encoding symbols according to the associated target coding rate:

$$n_i = \left\lceil \frac{k_i}{\rho_i} \right\rceil \quad (3)$$

  Unlike PET, here each encoding symbol (source or repair) corresponds to a packet.
- It follows that the total number of packets sent is:

$$n_{GOE} = \sum_{i=1}^{d} n_i = \sum_{i=1}^{d} \left\lceil \frac{\lceil \frac{\alpha_i}{l} \rceil}{\rho_i} \right\rceil \quad (4)$$

- Choose a packet interleaving scheme: Several schemes are possible that have major practical impacts. One of them consists in a uniform interleaving (random packet transmission order), in order to make the transmission robust in front of long erasure bursts. On the opposite packets can be sent in sequence (no interleaving), in decreasing object priority order, which offers limited robustness in front of erasure bursts but is most beneficial in terms of decoding delay and memory requirements.

C. Preliminary comparison of the PET and GOE schemes

The output of both UEP schemes is a set of n_{PET} or n_{GOE} packets ready to be sent to the receivers (usually $n_{PET} \neq n_{GOE}$). These packets contain one (GOE) or several (PET) encoding symbols (i.e. source or repair symbols) generated after a FEC encoding of the associated object.

However, even if the number of packets are not strictly equal they are equivalent as stated in the following lemma.

Lemma 1. *For the PET and GOE schemes, we have the following equivalences and ordering:*

$$\begin{aligned} n_{PET} &\approx n_{GOE} && (5)\\ \rho_i \approx \frac{\beta_i}{n_{PET}} &\approx \frac{k_i}{n_i} && (6)\\ \beta_i s_i &\approx k_i l && (7)\\ n_{PET} s_i &\approx n_i l && (8)\\ \beta_1 \leqslant \beta_2 &\leqslant ... \leqslant \beta_d && (9) \end{aligned}$$

Proof: (5) to (8) follow from the definition of the parameters (see Section II-A and II-B) by taking out all rounding operators, and since $d \ll l$. (9) follows from the definition (2) and from the ordering of the object in decreasing priority order (or equivalently increasing coding rate). ■

In order to illustrate the analytical results, we consider the same parameters as those used in the original PET paper [2] (Table I).

TABLE I
PARAMETERS FOR THE UEP PROBLEM, FROM [2]. ALL SIZES IN BYTES.

Input parameters	
m, original object length	100KB
d, number of objects (priorities)	5
α_i, length of object i	[10KB, 10KB, 20KB, 30KB, 30KB]
ρ_i, target coding rate for object i	[0.5, 0.6, 0.65, 0.8, 0.95]
l, target packet length	250 B
Output parameters with PET	
β_i, number of source symbols	[279, 335, 363, 447, 531]
s_i, symbol size of object i	[36 B, 30 B, 56 B, 68 B, 57 B]
actual packet size	247 B
number of encoded symbols	558
actual coding rate	[0.5, 0.6, 0.651, 0.801, 0.952]
n_{PET}, total number of packets	558
Output parameters with GOE	
k_i, number of source symbols	[40, 40, 80, 120, 120]
symbol size (=l)	250 B
n_i, number of encoded symbols	[80, 67, 124, 150, 127]
actual packet size	250 B
actual coding rate	[0.5, 0.597, 0.645, 0.8, 0.945]
n_{GOE}, total number of packets	548

III. PEAK MEMORY ANALYSIS FOR PET

By construction, packets are all inter-changeable with PET. Therefore the channel loss fraction, p_e, is the only "degree of freedom" to consider in the analysis (packet interleaving has no sense, unlike GOE). Two cases of interest are studied:

A. PET with no erasure

The number of symbols to be stored increases with the number of packets received. But once enough symbols have been received (i.e. β_i for object i), the decoding is successful and the received symbols can be deleted. Therefore, the memory usage across the packet reception for object i is:

$$t.s_i.\mathbf{1}_{1 \leqslant t \leqslant \beta_i} \tag{10}$$

where t is the current number of received packets, which varies from 1 to n_{PET} and where $\mathbf{1}_{1 \leqslant t \leqslant \beta_i} = 1$ if $1 \leqslant t \leqslant \beta_i$ and 0 otherwise (echelon function). Thus, the peak memory which is defined as the maximum over the sum of the memory usage for each object can be expressed as:

$$\begin{aligned} \text{peak mem}_{PET}^{no\ erasure} &= \max_{t \in [1, n_{PET}]} \sum_{i=1}^{d} t.s_i.\mathbf{1}_{1 \leqslant t \leqslant \beta_i} \\ &= \max_{1 \leqslant j \leqslant d} \left(\beta_j \sum_{i=j}^{d} s_i \right) \end{aligned} \tag{11}$$

From (10) the peak memory for object i is reached when $t = \beta_i$. When considering all objects, the global peak can be searched over all local maxima that corresponds to a maximum of the per object peak memory. Moreover, the β_i are ordered (9). Therefore, when object j can be decoded, which occurs when β_j packets have been received, the objects that have not yet been decoded are: object $j+1$ to d, and their symbols must be kept. A local maximum, when object j can be decoded, is thus $\beta_j \sum_{i=j}^{d} s_i$. This explains the last equality in (11). Figure 2 illustrates this behavior. Each colored curve exhibits a regular slope until enough

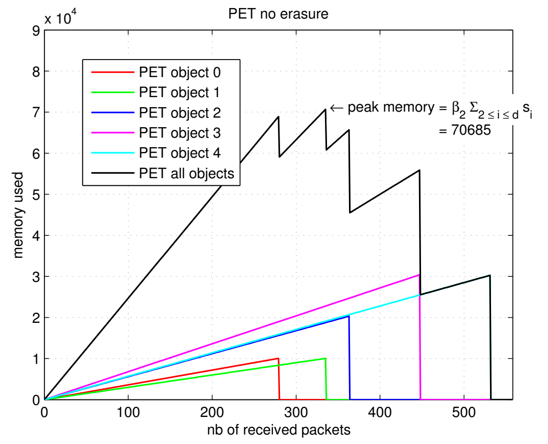

Fig. 2. PET memory usage as a function of the number of received packets, without erasure.

symbols are available for decoding to take place. The black curve is the total memory requirement over the time, and the peak is reached after receiving 335 packets.

B. PET with erasures

If erasures occur, only $(1-p_e)n_{PET}$ packets are received and some of the objects may not be decoded. The memory usage curve is therefore truncated. For instance figure 3 illustrates this behavior when $p_e = 0.45$. The memory usage for object i can be expressed as:

$$t.s_i.\mathbf{1}_{1 \leqslant t \leqslant \min(\beta_i, (1-p_e)n_{PET})}. \tag{12}$$

and the peak memory is:

$$\begin{aligned} \text{peak mem}_{PET} &= \max_{t \in [1, n_{PET}]} \sum_{i=1}^{d} t.s_i \mathbf{1}_{1 \leqslant t \leqslant \min(\beta_i, (1-p_e)n_{PET})} \\ &= \max \left(\max_{1 \leqslant j \leqslant \tau} \left(\beta_j \sum_{i=j}^{d} s_i \right), (1-p_e)n_{PET} \sum_{i=\tau+1}^{d} s_i \right) \end{aligned} \tag{13}$$

where τ s.t. $\beta_\tau \leqslant (1-p_e)n_{PET} < \beta_{\tau+1}$. In other words, we have received a number of packets such that all classes up to index τ have been decoded, but not classes of index $\tau+1$ and above.

Figure 4 shows the peak memory as a function of the channel loss fraction p_e. We observe a first step at low p_e, which corresponds to the maximum of the memory consumption shown in Figure 2 and expressed in the closed formula (11). If the erasure probability increases, this peak may not exist anymore (as shown in Figure 3) and a second step occurs at the second maximal value of all peaks (13). Depending on the relative values of the peaks and of their positions, there might be up to d steps. In our example, there are only 2. Finally, when p_e gets larger, no decoding will be performed successfully and no memory is freed. Therefore, the peak memory equals the total number of symbols received, which decreases linearly with p_e.

IV. PEAK MEMORY ANALYSIS FOR GOE

The GOE scheme is more flexible than PET. Here two "degrees of freedom" are considered: the channel loss fraction and the

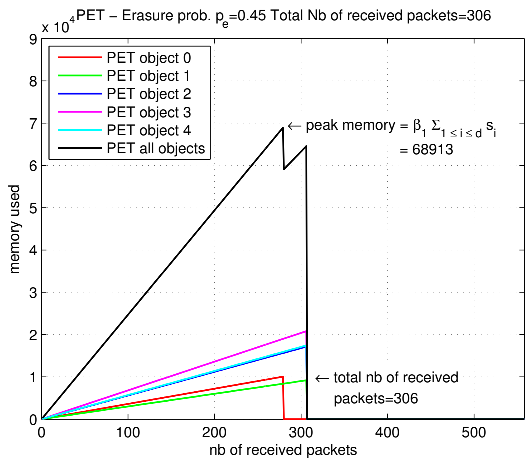

Fig. 3. PET memory usage as a function of the number of received packets, with $p_e = 0.45$.

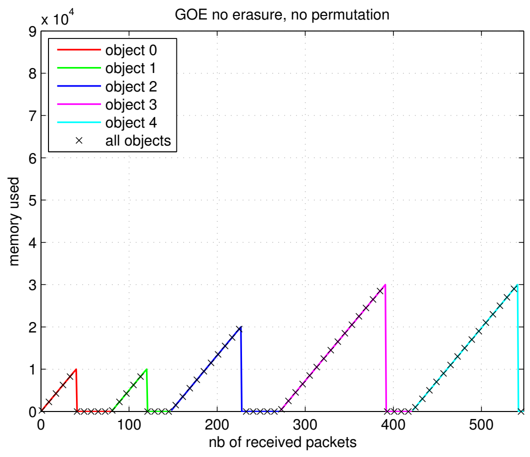

Fig. 5. GOE memory usage as a function of the number of received packets, with no erasure and no permutation.

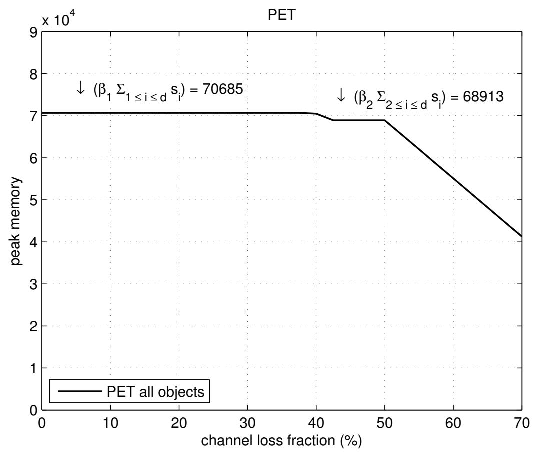

Fig. 4. PET peak memory usage as a function of the loss fraction p_e.

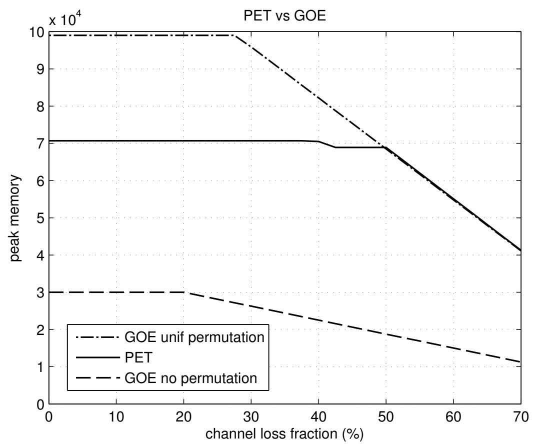

Fig. 6. PET and GOE peak memory usage as a function of the loss fraction p_e.

packet permutation (interleaving). Five cases of interest are studied:

A. GOE with no erasure and no permutation

At the receiver, packets are kept in memory as long as the object they belong to has not been decoded. Once decoded, all the packets of this object are immediately deleted. Since k_i packets are sufficient to decode object i, the memory consumption for this object after receiving t packets $(1 \leqslant t \leqslant n_i)$ is:

$$t.l.\mathbf{1}_{1\leqslant t\leqslant k_i} \tag{14}$$

The peak memory for object i occurs when $t = k_i$:

$$\max_{1\leqslant t\leqslant n_i} t.l.\mathbf{1}_{1\leqslant t\leqslant k_i} = k_i l. \tag{15}$$

Let us first assume that no erasure occurs and that no permutation has been used. Objects are received in their index increasing order and objects are processed in sequence. Therefore the memory usage is the time concatenation of the functions given in (14) and is shown in Figure 5. The peak memory is then given by the maximum of each per object peak memory:

$$\text{peak mem}_{GOE}^{no\ erasure\quad no\ perm} = l \max_{1\leqslant i\leqslant d} k_i \tag{16}$$

B. GOE with erasures and no permutation

If erasures occur, $n_i(1-p_e)$ packets for object i are received. If this is greater than k_i, then the memory is freed when k_i packets are received. Otherwise, when the reception of object $i+1$ starts, the receiver knows that no more packets for object i will be received and it can free the memory. The peak memory is then given by:

$$\text{peak mem}_{GOE}^{no\ perm} = l \max_{1\leqslant i\leqslant d} \min\big(k_i, n_i(1-p_e)\big) \tag{17}$$

where the maximum is computed over all the per object peak memories.

Theorem 1 (PET vs. GOE no permutation)**.** *The peak memory for PET is much larger than the peak memory of the GOE scheme with no permutation.*

Proof: From the equivalence of the amount of useful data (7), we have $\beta_j \sum_{i=j}^{d} s_i \approx lk_j + \beta_j \sum_{i=j+1}^{d} s_i \gg lk_j$. Moreover, from the equivalence of the amount of encoded data (8) $n_{PET} \sum_{i=\tau+1}^{d} s_i \approx \sum_{i=\tau+1}^{d} n_i l \gg n_i l$. Finally, comparing (13) and (17), we get that the peak memory for PET is much larger than the peak memory of GOE without any permutation. ∎

Figure 6 shows the peak memory difference between the GOE (no permutation) and the PET schemes.

C. GOE with a uniform permutation

Theorem 1 shows that GOE without permutation allows to save memory with respect to the PET scheme. However, GOE without permutation is not robust to burst of erasures. Therefore, we study the case of GOE under a uniform permutation in order to increase the robustness to bursts.

When a uniform permutation distribution is used, the peak memory consumption occurs when each object can almost be decoded (i.e. $k_j - 1$ packets received for object j) except one object that can be decoded. Therefore, the peak memory (without erasures) is:

$$\text{peak mem}_{GOE}^{no\ erasure\quad unif\ perm} = l\left(\sum_{j=1}^{d} k_j - d + 1\right) \tag{18}$$

If erasures occur, the same memory peak as (18) can be achieved unless not enough packets have been received. Therefore, the peak memory under erasure becomes:

$$\text{peak mem}_{GOE}^{unif\ perm} = l.\min\left(\sum_{j=1}^{d} k_j - d + 1, n_{GOE}(1-p_e)\right) \tag{19}$$

Theorem 2 (PET vs. GOE uniform permutation)**.** *The peak memory for GOE with a uniform permutation is larger than the peak memory of the PET scheme, unless the erasure probability is so large that no object can be decoded, in which case the peak memories are equal.*

Proof: Let us first compare the peak memory without erasure and let us compare each term to be maximized in (11) and (18). $\forall j, 1 \leqslant j \leqslant d$, we have $\beta_j \sum_{i=j}^{d} s_i = \min_{j \leqslant i \leqslant d} \beta_i \sum_{i=j}^{d} s_i \leqslant \sum_{i=j}^{d} \beta_i s_i \leqslant \sum_{i=1}^{d} \beta_i s_i \approx l.\sum_{i=1}^{d} k_i$, where the first equality follows from the ordering (9) and the last one from (7). Thus, by neglecting d in (19), we get that $\text{peak mem}_{PET}^{no\ erasure} \leqslant \text{peak mem}_{GOE}^{no\ erasure\quad unif\ perm}$. Moreover, since the number of packets are equivalent, see (5), we have $n_{PET}(1-p_e) \approx n_{GOE}(1-p_e)$. Therefore, comparing (13) and (19), we get that if the erasure probability is so large that no object can be decoded, then the peak memories are equal. ∎

This is illustrated in Figure 6.

D. GOE with no erasure but an average permutation

In order to further investigate the GOE scheme, we analyze the peak memory of an *average permutation*, which is typical of the permutation behavior. This average permutation results from the average decoding delay that has been studied in detail in [7]. For the sake of completeness, we recall that the mean decoding delay (under GOE) can be efficiently approximated by:

$$\begin{aligned} \text{dec\_delay}_{GOE} &\lessapprox \min\left(n_{GOE}, \frac{n_{GOE}}{n_i}\frac{k_i}{1-p_e}\right) \\ &\lessapprox \min\left(n_{GOE}, \text{dec\_delay}_{GOE}^{\infty}\right) \end{aligned} \tag{20}$$

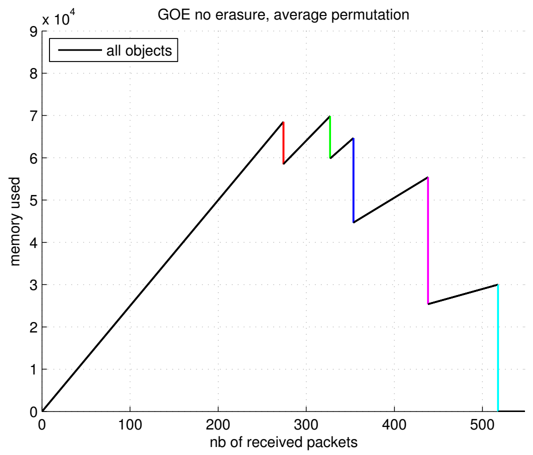

Fig. 7. GOE memory usage as a function of the number of received packets, with no erasure but an average permutation.

Moreover, this upper bound (20) is exact for low value of p_e, i.e. $p_e \to 0$. Therefore, when there is no erasure, the average decoding delay is:

$$\text{dec\_delay}_{GOE}^{no\ erasure}(\text{object } i) = \frac{k_i}{n_i} n_{GOE}. \tag{21}$$

Definition 1 (The average permutation)**.** *The average permutation is a permutation such that, for each object, the decoding delay occurs at exactly the average decoding delay* (21)*.*

First, the objects are sent in a decreasing priority order (i.e. high priority first or low coding rate k_i/n_i first). As a consequence, they are also ordered according to their increasing average decoding delay, see (21). Before object 1 is decoded, the memory usage increases by l for each packet received. It reaches a peak when object 1 is decoded, which value is $\text{dec\_delay}_{GOE}(\text{object } 1)$ and decreases by $k_1 l$, the memory used for object 1 being freed. Due to the permutation, packets from object 1 continue to be received, but discarded. Then, the memory further increases by an amount of l if the received packet does not belong to object 1. Therefore the slope is now $l(1 - \frac{n_1}{n_{GOE}})$. By repeating this procedure, we obtain that the peak values are:

$$l(1 - \sum_{i=1}^{j-1} \frac{n_i}{n_{GOE}})\frac{k_j}{n_j} n_{GOE} = l(\sum_{i=j}^{d} n_i)\frac{k_j}{n_j}.$$

It follows that the peak memory under GOE is given by

$$\text{peak mem}_{GOE}^{no\ erasure\quad avg\ perm} = \max_{1 \leqslant j \leqslant d} l(\sum_{i=j}^{d} n_i)\frac{k_j}{n_j} \tag{22}$$

From (6), we have $\frac{k_j}{n_j} \approx \frac{\beta_j}{n_{PET}}$ and from (8), we have $l.n_i \approx n_{PET} s_i$. Therefore, comparing (11) and (22) we conclude that, up to round-off errors, the peak memory of PET and GOE with an average permutation and without erasures are equal.

E. GOE with erasures and an average permutation

If erasures occur, we observe the same behavior as for PET. More precisely, the peak memory is given by:

$$\text{peak mem}_{GOE}^{avg\ perm} =$$

$$\max\left(\max_{1\leqslant j\leqslant \tau} l(\sum_{i=j}^{d} n_i)\frac{k_j}{n_j}, (1-p_e)l\sum_{i=\tau+1}^{d} n_i\right) \quad (23)$$

where τ is such that:
$\text{dec\_delay}_{GOE}(\tau) \leqslant (1-p_e)n_{GOE} < \text{dec\_delay}_{GOE}(\tau+1)$.

Theorem 3 (PET vs. GOE average permutation)**.** *The peak memory with PET and GOE with an average permutation are equal.*

Proof: Let us compare (23) to (13). The equivalence of the first term (i.e. when there is no erasure) has been shown in the previous section. The second term of this formula is equivalent to that of (13). Indeed, the $l\sum_{i=\tau+1}^{d} n_i$ is the total number of bytes after FEC encoding of the classes that will not be decoded with GOE, and this value is equivalent to $n_{PET}\sum_{i=\tau+1}^{d} s_i$, see (8), which is the same for PET. Finally, the τ of both formulas are equivalent since both techniques have the same recovery capabilities [7]. This proves that the peak memory with PET and GOE with an average permutation are equal. ■

Moreover, we conjecture that:

Conjecture 1 (Average peak memory vs. peak memory of the average permutation)**.** *The average peak memory for GOE with a uniform permutation is smaller than the peak memory of the average permutation.*

Example 1. *Consider two objects of the same length and priority. Therefore the two objects have the same average decoding delay. The memory usage (under the average permutation) is maximum when the number of received packets equals the $\frac{k_1}{n_1}n_{GOE}$ delay. Thus the peak memory is $2k_1 l$.*

As for the peak memory average over all permutations, the two peaks (one for each object) will occur at the same time. And when they occur at two different times, the maximum is always smaller than $2k_1 l$. Therefore, on average, the peak memory $\leqslant 2k_1 l =$ the peak memory obtained with the average permutation.

Claim 1 (PET vs. GOE average permutation)**.** *The average peak memory of GOE with a uniform permutation is smaller than the peak memory of PET.*

Proof: This is a consequence of Theorem 3 and Claim 1. ■

Figure 8 shows the span of the peak memory for the GOE scheme with a uniform permutation. We observe that, as stated in Claim 1, the mean of the peak memory is smaller than the peak memory of the PET scheme. Note also that it is only slightly smaller.

V. Conclusions

This work has compared two techniques capable of providing an unequal erasure protection service, namely PET and GOE. We have recently demonstrated that the protection performance of both approaches are equivalent [7]. However key differences become apparent when considering such practical metrics as the peak memory consumption during decoding. We have shown that GOE without any permutation offers the smallest peak memory but with a limited burst erasure resiliency. In order to increase the burst erasure resiliency, we considered GOE with a uniform permutation and showed that the peak memory of PET is smaller than the one for GOE (uniform distribution), unless no object can be decoded in which case they are equal. However, we showed that PET corresponds to the average behavior of the GOE uniformly permuted scheme.

In future works, we will analyze alternative permutations (for instance a Δ-permutation [7]). Being somewhere between these two extrema, such a permutation is likely to achieve a good trade off between burst erasure resiliency and peak memory consumption (among other practical considerations).

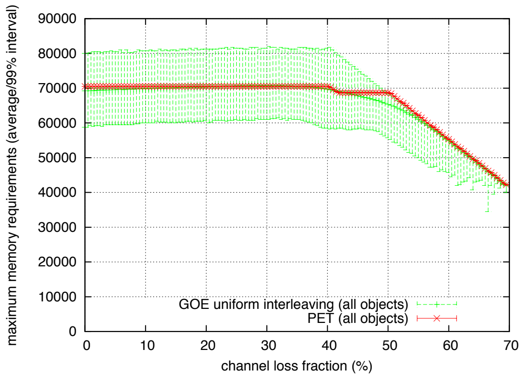

Fig. 8. Peak memory usage (average and 99% confidence intervals below and above the average) as a function of the channel loss fraction p_e.

VI. Acknowledgments

This work was supported by the ANR-09-VERS-019-02 grant (ARSSO project) and by the INRIA - Alcatel Lucent Bell Labs joint laboratory.
The authors would also like to thank the members of the ARSSO project for their comments that help improve the manuscript.

References

[1] N.L. Johnson, S. Kotz and A.W. Kemp, *Univariate Discrete Distributions*, (Third Ed.), New York: Wiley, 2005.
[2] A. Albanese, J. Blomer, J. Edmonds, M. Luby and M. Sudan, "Priority encoding transmission", *IEEE Trans. on Information Theory*, Vol. 42 Issue 6, November 1996.
[3] A. Bouabdallah and J. Lacan, "Dependency-aware unequal erasure protection codes", *Journal of Zhejiang University - Science A,* Vol. 7 Issue 1, ISSN 1673-565X, 2006.
[4] S. Boucheron and M. Salamatian, "About Priority Encoding Transmission", *IEEE Trans. on Information Theory,* vol. 46, no. 2, pp. 699-705, March 2000.
[5] M. Luby and T. Stockhammer, "Universal Object Delivery using RaptorQ", *IETF RMT Working Group*, Work in Progress: <draft-luby-uod-raptorq-00>", March 2011.
[6] V. Roca, A. Roumy and B. Sayadi, "The Generalized Object Encoding (GOE) Approach for the Forward Erasure Correction (FEC) Protection of Objects and its Application to Reed- Solomon Codes over GF(2x)", *IETF RMT Working Group*, Work in Progress: <draft-roca-rmt-goe-fec-00.txt>", July 2011.
[7] A. Roumy, V. Roca, B. Sayadi and R. Imad, "Unequal Erasure Protection and Object Bundle Protection with the Generalized Object Encoding Approach", INRIA Research Report, <http://hal.inria.fr/inria-00612583>, July 2011.