HAL
open science

Unequal Erasure Protection and Object Bundle Protection with the Generalized Object Encoding Approach

Aline Roumy, Vincent Roca, Bessem Sayadi, Rodrigue Imad

To cite this version:

Aline Roumy, Vincent Roca, Bessem Sayadi, Rodrigue Imad. Unequal Erasure Protection and Object Bundle Protection with the Generalized Object Encoding Approach. [Research Report] RR-7699, INRIA. 2011. inria-00612583

HAL Id: inria-00612583

https://inria.hal.science/inria-00612583

Submitted on 29 Jul 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Unequal Erasure Protection and Object Bundle Protection with the Generalized Object Encoding Approach

Aline Roumy — Vincent Roca — Bessem Sayadi — Rodrigue Imad

N° 7699

29 Juillet 2011

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7699--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

Unequal Erasure Protection and Object Bundle Protection with the Generalized Object Encoding Approach

Aline Roumy*, Vincent Roca†, Bessem Sayadi‡, Rodrigue Imad§

Theme :
Équipes-Projets Temics et Planète

Rapport de recherche n° 7699 — 29 Juillet 2011 — 25 pages

Abstract: When a data flow contains information of different priority levels, it is natural to try to offer an unequal protection where the high priority data benefits from a higher protection than the rest of data. In this work we focus on the "erasure channel", for instance the Internet where the UDP/IP datagram integrity is guaranteed by the physical layer FCS (or CRC) and the UDP checksum. In this context UEP refers to an Unequal Erasure Protection (rather than Error) and the FEC code being used is one of the various Application-Layer Forward Erasure Correction (AL-FEC) codes that have been designed and standardized in the past years, like Reed-Solomon, one of the LDPC variants, or Raptor(Q) codes. Offering an unequal protection in this context can be achieved by one of the following three general approaches: by using dedicated UEP-aware FEC codes, by using a dedicated UEP-aware packetization scheme, or by using an UEP-aware signaling scheme. In this work we ignore the first approach as we want to reuse existing AL-FEC codes. Instead we focus on and compare the last two approaches and more precisely the well known Priority Encoding Transmission (PET) scheme that belongs to the UEP-aware packetization category and a Generalized Object Encoding (GOE) scheme, we propose, that belongs to the UEP-aware signaling category. We compare them both from an analytical point of view (we use an N-truncated negative binomial distribution to that purpose) and from an experimental, simulation based, point of view. Since we want to derive practical recommendations, we consider erasure recovery metrics, but also processing load and peak memory requirements metrics that can be of high importance. We show that the GOE approach, by the flexibility it offers, its simplicity, its backward compatibility and its good recovery capabilities, is highly recommendable for practical systems requiring UEP.

* INRIA, France, {firstname.name}@inria.fr
† Alcatel Lucent - Bell Labs, France, {firstname.name}@inria.fr
‡ Alcatel Lucent - Bell Labs, France, {firstname.name}@inria.fr
§ Alcatel Lucent - Bell Labs, France, {firstname.name}@inria.fr

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

In a second part of the paper we consider the use of PET, more precisely an extension called Universal Object Delivery (UOD), and GOE in situations where one needs to send a bundle of small object (e.g. files). If both solutions can address this need, we show that once again the GOE scheme is highly recommendable for practical realizations.

Key-words: Application-Layer Forward Erasure correction code, Unequal Erasure Protection

Protection inégale aux effacements et protection de fagots d'objets par l'approche Generalized Object Encoding

Résumé : Nous proposons une nouvelle méthode qui permet à la fois de protéger des objets différents de manière inégale aux effacements mais qui peut aussi regrouper des objets en un fagot afin de fournir à chacun des objets du fagot, le même niveau de protection. Cette technique est apppelée Generalized Object Encoding (GOE). Nous la détaillons et l'analysons dans ce rapport.

Mots-clés : Application-Layer Forward Erasure correction code, Unequal Erasure Protection

1 Introduction

1.1 Providing Unequal Erasure Protection

When a data flow contains information of different priority levels, it is natural to try to offer an unequal protection where high priority data benefits from a higher protection than the rest of data. To achieve this goal, FEC codes are required in order to add redundant information to the data flow. In this work we focus on the "erasure channel", for instance the Internet where the UDP/IP datagram integrity is guaranteed by the physical layer FCS (or CRC) and the UDP checksum (used most of the times even if not mandatory).

In this context FEC refers to one of the various Application-Layer Forward Erasure Correction (rather than Error) (AL-FEC) codes that have been designed and standardized in the past years. Similarly UEP refers to an Unequal Erasure Protection (rather than Error). Over the time several UEP schemes have been designed in order to address this needs. They can be classified into three categories.

The first category offers UEP services thanks to dedicated UEP-aware FEC codes. That's the case of DA-UEP [3] where the generator matrix of the code is tailored in order to integrate the natural dependencies within the data flow. A typical example is a GOP of a video flow where receiving P frames is useless until the I frame they depend on are available. This relationship is hard coded within the DA-UEP code internal structure which guaranties good UEP performances.

The second category offers UEP services thanks to a dedicated UEP-aware packetization scheme. It relies on traditional, unmodified AL-FEC codes. The well known Priority Encoding Transmission (PET) scheme [2] offers UEP as a consequence of the way packets are created. A recent proposal, Universal Object Delivery (UOD) [7], shares with PET its packetization approach in order to provide UEP capabilities. Although elegant, this solution features many constraints that can make its use in practical systems difficult, as we will see.

The third category offers UEP services thanks to a dedicated UEP-aware signaling scheme. Here also AL-FEC codes are kept unmodified. The Generalized Object Encoding (GOE) scheme [12] that we propose belongs to this category. This proposal is based on a simple solution that consists in encoding with different code rates each priority class. In order to apply this simple idea, GOE provides an appropriate, UEP-aware signaling mechanism that creates a mapping between the original object, composed of several chunks of different priority, from the so-called "Generalized Objects" (GO) that correspond to each priority class and over which FEC encoding is performed.

In fact UOD/PET and GOE both perform "Time Sharing" [5, section 15.6] and are therefore asymptotically optimal, in the sense that they achieve the capacity of the Broadcast erasure channel [4]. However, their implementation differ and we show that despite its simplicity, GOE outperforms UOD/PET when practical aspects are considered.

1.2 Providing an efficient protection to bundles of small objects

The efficient protection of a bundle of small objects is not easy. The trivial solution where each object is FEC encoded separately and packets sent inde-

pendently is confronted to the coupon collector problem: if all object have been decoded but one, for which a single redundancy packet would be sufficient, receiving this packet will happen with a probability $1/N$, where N is the number of objects (we assume packets are sent in a random order and no feedback channel exists).

In a second part of this work we show how UOD and GOE both support this need and solve the coupon collector problem. As we will show, the approaches are quite different and here also the GOE approach outperforms UOD when practical aspects are considered.

1.3 Assumptions and problem position

Let us discuss AL-FEC codes more in details, since they are the foundations of UEP and object bundle protection schemes. An AL-FEC code is a block code which transforms k message symbols into n encoded (or output) symbols. In theory a symbol corresponds to the smallest entity processed by the code. With a binary code, a symbol is a bit. With a Reed-Solomon code over GF(2^8), a symbol is a byte. However we work at the application level, over the Internet, which is regarded as a *packet* erasure channel: a UDP/IP datagram is either received without error or lost altogether. It is therefore more convenient to think in terms of the units of data that are managed by the AL-FEC codec (the code implementation), of size s bytes (e.g. $s = 1024$ bytes) and carried within the UDP datagram payload. With a binary code, this means that the same encoding (resp. decoding) operations are performed on all the $8 * s$ bits of the symbol, or said differently $8 * s$ bit-level encodings (resp. decodings) are performed in parallel, the erasure patterns being the same at each time. Similarly, with a Reed-Solomon code working on GF(2^8), the same operations are performed on all the s bytes, or said differently, s byte-level encodings (resp. decodings) are performed in parallel.

In the remaining of this paper we therefore assume that a symbol is the input and output units of data manipulated by the AL-FEC codec, from several bytes to several hundred bytes long, and that is carried in a packet (along with symbols coming from other objects with UOD/PET).

Several AL-FEC codes for the packet erasure channel have been standardized during the past few years and included for instance in IPTV or Mobile TV standards. Examples are Raptor codes [9] (that can be found in several places for instance in DVB-H/SH IPDC or 3GPP MBMS standards), more recently RaptorQ codes [10], Reed-Solomon codes for the erasure channel [13][6], or LDPC-Staircase codes [11] (that can be found in ISDB-Tmm standards). However we don't want to compare specific codes and codecs but the UEP techniques themselves. Therefore in the remaining of this paper we only consider an ideal, Maximum Distance Separation (MDS) code, capable of encoding a small or large number of symbols with same optimal efficiency. Each object, no matter its size (in number of source symbols), can be encoded in a single pass by this ideal FEC code.

Because of the target use cases, the UEP techniques considered in our work may be deployed in real equipments, often mobile terminals with limited resource (processing, memory, and battery). Therefore it is extremely important to consider the practical aspects rather than focusing only on the recovery capabilities. Such considerations as the processing load or the type of operations

performed within packets, the number of FEC encodings/decodings performed, or the maximum memory required during the decoding process have a high practical importance and may guide the choice of an UEP technique. In this work we will show that GOE is from this point of view highly recommendable, compared to UOD/PET.

2 Introduction to the PET and GOE UEP schemes

We first detail in this section the PET and GOE techniques, when used in order to provide UEP.

Let us consider an input object that needs to be unequally protected. This input object can therefore be split into several classes and each class be protected according to its priority constraint. Without loss of generality, we will use the term *original object* to denote the input object and *object* (or *generalized object* in case of GOE) to denote each class. The whole content of an object has therefore the same priority. The importance of this distinction will become more obvious when considering the problem of protecting a bundle of small objects with the UOD and GOE schemes in Section 5.

We do not detail GOE signaling aspects in this section and give some insights in Section 5.2 (the interested reader can refer to [12] for details). Instead we focus on efficiency considerations through modeling and simulation.

The input parameters of our UEP algorithm 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. High priority objects are assigned small ρ_i values and vice-versa;
- l, the target packet length in bytes. This size is usually determined by the maximum transmission unit size on the path between the source and the receiver(s), minus the protocol header sizes;

The output of the UEP algorithm is a set of n_{PET} (resp. 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.

2.1 PET Principles

Goal: PET provides UEP in such a way that each packet is interchangeable in order to provide a deterministic erasure recovery behavior. To that purpose, the n_{PET} encoding symbols of each object, after FEC encoding, are uniformly spread over the n_{PET} packets.

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:

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

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

  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 (we will see this is not the case in general). We have: $g = \sum_{i=1}^{d} \frac{\alpha_i}{\rho_i}$.

- Determine the number of source symbols in object i, β_i:

$$\beta_i = \lceil n_{PET}\rho_i \rceil \tag{2}$$

- 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 jth 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.

  Lemma 1 (Underprotection under PET)**.** *In a PET system, when the parameters are determined according to [2], data is less protected than desired.*

  Proof. The observed coding rate for object i is $\frac{\beta_i}{n_{PET}} = \frac{\lceil n_{PET}\rho_i \rceil}{n_{PET}} \geqslant \rho_i$. Therefore the protection constraint is not satisfied: data is under-protected. □

2.2 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:

- 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:

$$n_i = \left\lceil \frac{k_i}{\rho_i} \right\rceil \tag{3}$$

  Unlike PET, here each encoding symbol (source or repair) corresponds to a packet.

Lemma 2 (Overprotection under GOE)**.** *In a GOE system, data is more protected than desired.*

Proof. The observed coding rate for object i is $\frac{k_i}{n_i} = \frac{k_i}{\lceil \frac{k_i}{\rho_i} \rceil} \leqslant \rho_i$. Therefore data is more protected than desired (in practice this over protection is limited). □

- It follows that the total number of packets sent is:

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

- 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. An intermediate scheme consists in a Δ-permutation where only a subset of the packets are randomly permuted (see Section 3.3). In this work a uniform interleaving is assumed unless otherwise mentioned. Note that packet interleaving has no sense with PET since packets are all inter-changeable, which is another limit of PET.

Remark: Interestingly, even if PET under-protects the data whereas GOE overprotects it, there are situations where the number of packets is greater for PET than for GOE. This is the case with the parameters used in the original paper [2]: $n_{PET} = 558$ and $n_{GOE} = 548$ (see Table I). We conclude that the round-off errors in the computation of the number of packets make PET suboptimal with respect to GOE.

3 Erasure Recovery Metrics

We now analyze the two techniques under the angle of the decoding delay of each object, i.e. the number of packets required to decode a given object. We essentially focus on the decoding delay for each object and then consider the total number of successful decodings. We first show that PET and GOE (with a uniform interleaving) have equivalent decoding performances. Then we show that the decoding delay of GOE can easily be largely improved with an appropriate interleaving scheme, an optimization that PET does not enable.

3.1 Infinite length analysis

Let us start with an infinite length analysis. PET decoding for object i is successful if β_i packets have been received. Thus the average decoding delay is the mean for the β_i^{th} success, where each success occurs with probability $1-p_e$, and where p_e is the packet erasure probability. Therefore the average decoding delay is the mean of the $\mathcal{NB}(\beta_i, 1-p_e)$ law (negative binomial distribution), and is given by (see Appendix 8.1):

$$\begin{aligned} \text{dec_delay}_{PET}^{\infty} &= \sum_{x \geqslant \beta_i} \binom{x-1}{\beta_i - 1} (1-p_e)^{\beta_i} p_e^{x-\beta_i} \\ &= \frac{\beta_i}{1-p_e} \end{aligned} \tag{5}$$

GOE decoding for object i is successful if k_i out of n_i packets have been received. Therefore we compute the mean for the k_i^{th} success, where each success occurs with probability $\frac{n_i}{n_{GOE}}(1-p_e)$, i.e. a packet of object i has been sent (due to uniform interleaving this occurs with probability $\frac{n_i}{n_{GOE}}$) and has not been erased, which occurs with probability $1-p_e$. The average decoding delay is given by:

$$\text{dec_delay}_{GOE}^{\infty} = \sum_{x \geqslant k_i} \binom{x-1}{k_i - 1} p_{GOE}^{k_i} (1-p_{GOE})^{x-k_i}$$

where $p_{GOE} = \frac{n_i}{n_{GOE}}(1-p_e)$. Therefore,

$$\text{dec_delay}_{GOE}^{\infty} = \frac{k_i}{n_i} n_{GOE} \frac{1}{1-p_e}. \tag{6}$$

It is therefore sufficient to compare the scaling factors of the decoding delay:

$$\beta_i = \lceil \rho_i \; n_{PET} \rceil \qquad \text{versus} \qquad \frac{k_i}{n_i} \; n_{GOE}$$

that are the decoding delays when there is no erasure.

Let us first compare the number of packets. Since both methods perform separate encodings of the same amount of data with the same protection constraints, the number of packets are equal up to some round-off errors (recall that the size of the packets are equal). Therefore, we claim that:

Claim 1. *The number of packets with PET and GOE are equivalent:* $n_{PET} \approx n_{GOE}$.

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	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

Table 1: Parameters for the UEP problem, from [2]. All sizes in bytes.

Second, by definition of the PET system, the ratio $k_i/n_i \approx \rho_i$ since the code of rate k_i/n_i must satisfy the priority constraint ρ_i. Combining these two results, we claim that:

Claim 2. *The average decoding delay of PET and GOE are asymptotically (as the number of packets goes to $+\infty$) equivalent.*

since $\beta_i = \lceil \rho_i \; n_{PET} \rceil$ and $\frac{k_i}{n_i} \; n_{GOE}$ are the scaling factor of the asymptotic average decoding delay (when the number of trials goes to $+\infty$).

3.2 Finite length analysis

We now consider the case where the number of trials is limited, which corresponds to the reality.

Let us first consider PET. For object i, decoding is successful if β_i packets out of the n_{PET} packets sent are received. Therefore the decoding delay is the mean of the $\mathcal{TNB}(\beta_i, 1-p_e, n_{PET})$ law (Truncated Negative Binomial Distribution, see Appendix 8.2). From Claim 4 this mean can be efficiently approximated by:

$$\text{dec_delay}_{PET} = \mathbb{E}[X] \lessapprox \min\left(n_{PET}, \frac{\beta_i}{1-p_e}\right) \tag{7}$$

where: $X \sim \mathcal{TNB}(\beta_i, 1-p_e, n_{PET})$. Therefore:

$$\text{dec_delay}_{PET} \lessapprox \min\left(n_{PET}, \text{dec_delay}^{\infty}_{PET}\right) \tag{8}$$

Figure 1 shows the observed decoding delay, obtained with our PET/GOE simulator, when using the parameters of Table I. We note that the decoding

delay increases with the erasure probability and is upper bounded by the number of packets sent, n_{PET}. We compare this simulated delay with the theoretical upper bound given in (7) and observe that the upper bound leads a very good approximation in the range $p_e \in [0, 1 - \beta_i/n_{PET}]$. If $p_e > 1 - \beta_i/n_{PET}$, then no decoding is successful and we use the convention that the average decoding delay $= 0$. With GOE, decoding of object i is successful if k_i packets have been

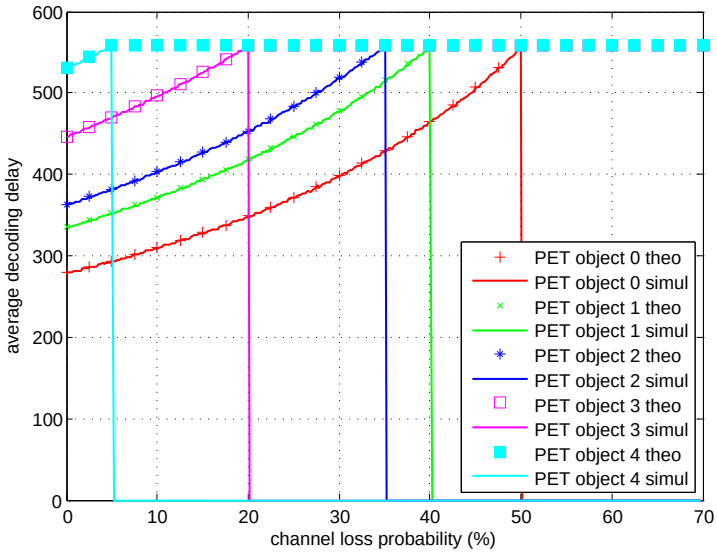

Figure 1: Decoding delay of the PET system.

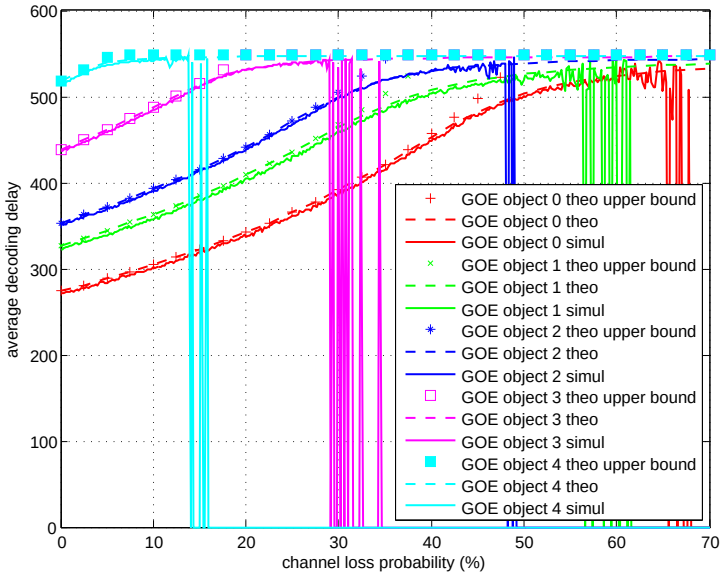

Figure 2: Decoding delay of the GOE system.

received. In this approach, n_i encoded packets have been sent and erased with probability p_e. We assume a uniform interleaving (i.e. random permutation) is

used in order to alleviate the erasure burst effects. Therefore, we compute the decoding delay averaged over all the possible permutations of the packets.

First, we consider object i before the permutation. Let X denote the trial at which the decoding is successful, i.e. the trial at which the k_i^{th} (out of n_i) packet has been received. The average decoding delay (before permutation) is:

$$\mathbb{E}[X], X \sim \mathcal{INB}(k_i, 1-p_e, n_i) \tag{9}$$

Let us denote $S_{n_{GOE}}$ the symmetric group, i.e. the group of all the permutations on n_{GOE} elements. If $\sigma \in S_{n_{GOE}}$, we denote σ_i the restriction of σ for the object i. More precisely, σ_i maps the indices $(1,..,n_i)$ to a subset of $(1,..,n_{GOE})$ of size n_i:

$$\begin{array}{rccc} \sigma_i : & [1,n_i] & \longrightarrow & [1,n_{GOE}] \\ & x & \longmapsto & y \end{array}$$

Therefore, after permutation, the decoding delay averaged over all permutations is given by:

$$\begin{aligned} \text{dec_delay}_{GOE} &= \sum_{\sigma \in S_{n_{GOE}}} \sum_{x=k_i}^{n_i} \sigma_i(x) \mathbb{P}(X=x|X \leqslant n_i) \\ &= \sum_{x=k_i}^{n_i} \sum_{\sigma \in S_{n_{GOE}}} \sigma_i(x) \mathbb{P}(X=x|X \leqslant n_i) \\ &= \frac{n_{GOE}}{n_i} \mathbb{E}(X=x|X \leqslant n_i) \end{aligned} \tag{10}$$

where $\mathbb{E}(X=x|X \leqslant n_i) =$

$$\sum_{x=k_i}^{n_i} x \binom{x-1}{k_i-1} \frac{(1-p_e)^{k_i} p_e^{x-k_i}}{\sum_{j=k_i}^{n_i} \binom{j-1}{k_i-1} (1-p_e)^{k_i} p_e^{k_i-j}}$$

and where the third equality follows from the fact that $\sum_{\sigma \in S_{n_{GOE}}} \sigma_i(x) = \frac{n_i}{n_{GOE}} x$. From Claim 4, which gives a tight upper-bound of the $\mathcal{INB}$ mean, the decoding delay can be efficiently approximated by:

$$\begin{aligned} \text{dec_delay}_{GOE} &\lessapprox \min\left(n_{GOE}, \frac{n_{GOE}}{n_i} \frac{k_i}{1-p_e}\right) \\ &\lessapprox \min\left(n_{GOE}, \text{dec_delay}_{GOE}^{\infty}\right) \end{aligned} \tag{11}$$

Figure 2 shows the decoding delay observed, when the parameters are given by Table I. As for the PET system, the decoding delay increases with the erasure probability and is upper bounded by the number of packets sent n_{GOE}. First we observe that the upper bound curve (marker plus sign) given in (11) leads to a very good approximation when compared to the simulated decoding delay (solid line) for low p_e. As the erasure probability increases such that the average decoding delay gets closer to n_{GOE}, there is a small gap. However, the exact computation of the average decoding delay (dashed curve) (10) matches well the simulation. We can conclude, that even in the case of the GOE, the upper bound, which is easy to compute, gives a good approximation of the average decoding delay.

Since the PET and GOE upper bound laws are both of the form $\frac{constant}{1-p_e}$, as long as the maximum value is not reached, it is sufficient to compare the constant term. With the parameters of Table I, we can calculate these constant terms:
$\text{dec_delay}_{PET}(p_e = 0) = [279, 335, 363, 447, 531]$
$\text{dec_delay}_{GOE}(p_e = 0) = [274.0, 327.2, 353.5, 438.4, 517.8]$
We see that in this example GOE slightly outperforms PET for all the priorities. Since the asymptotic decoding delay provides a very tight upper bound of the finite decoding delay, we conclude from Claim 2 that:

Claim 3. *The average decoding delay of the PET and GOE (with a uniform interleaving) are equivalent.*

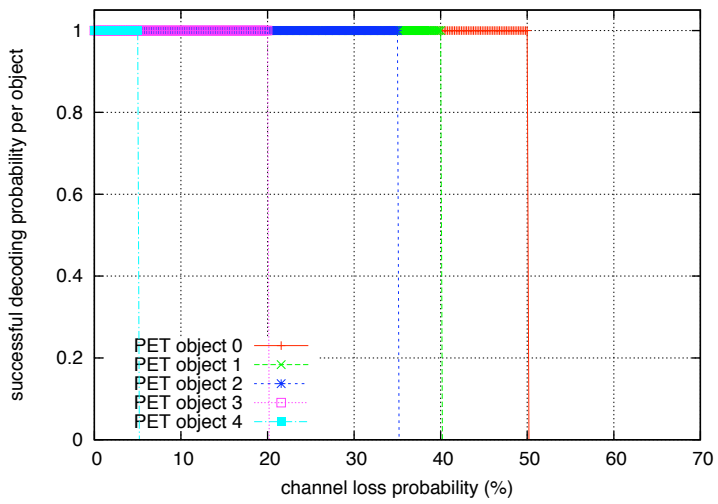

Figure 3: Number of successful decoding of the PET system.

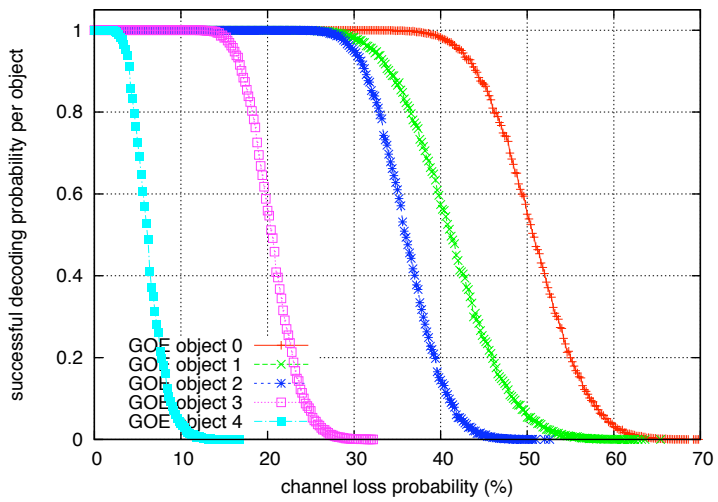

Figure 4: Number of successful decoding of the GOE system.

Let us now consider the number of successful decodings for object i with PET and GOE, when running 10,000 different tests for each p_e value. Figures 3 and 4 show this number as a function of the erasure ratio. With PET, as long as $p_e \leqslant 1 - \beta_i/n_{PET}$, the probability of successful decoding remains equal to 1 and changes to 0 as soon as $p_e > 1 - \beta_i/n_{PET}$. This deterministic behavior is a direct consequence of its packetization scheme. With GOE, as the erasure ratio tends to the critical value such that the decoding delay is close to the upper bound n_{GOE} the probability progressively tends to 0. However successful decodings still happen even after the erasure ratio corresponding to the object code rate.

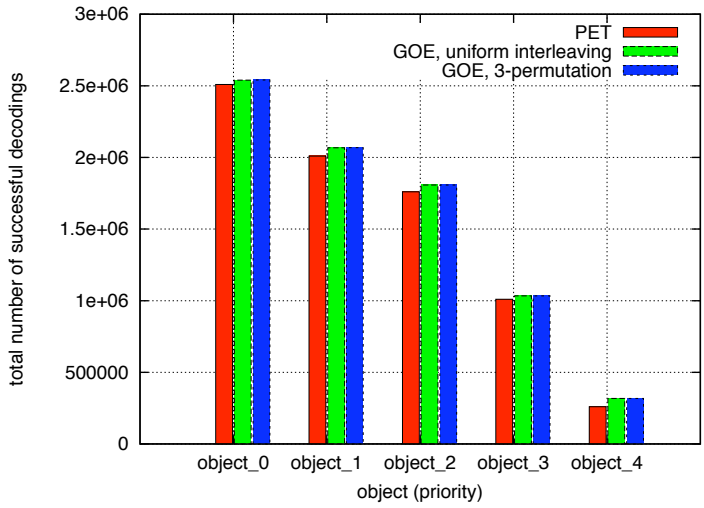

Figure 5: Total number of successful decoding for all p_e values.

If we sum up over all the possible channels the number of successful decodings (see Figure 5), we see that GOE performs slightly better than PET. This result confirms that **PET and GOE (with a uniform interleaving) provide equivalent UEP protection**.

3.3 Improving GOE with a Δ-permutation interleaving

Unlike PET, GOE's flexibility enables to change its packet interleaving strategy (Section 2.2). Therefore one can choose an interleaving that favors the average decoding delay. For instance, we can sort the packets in priority descending order and consider the permutations which only modify a subset of packets. The following is such a restricted permutation:

Definition 1 (Δ-permutation)**.** *Let $N_n = [1, n]$ be the set to be permuted. A Δ-permutation modifies the position of indices spaced by Δ. More precisely, each index whose remainder by the modulo Δ operation is 0 is permuted with another index chosen uniformly randomly from $N_n = [1, n]$.*

Note that if $\Delta = 1$, the set of Δ-permutation is the symmetric group S_n. It is intuitive that the average decoding delay under Δ-permutation is upper bounded by the decoding delay averaged over all possible permutations. Figure 6 shows the average decoding delay of the GOE system when the interleaving is

performed with a Δ-permutation. Even with a small value of Δ (here $\Delta = 3$ and recall that $\Delta = 1$ corresponds to an unconstrained permutation), the average decoding delay is significantly reduced with respect to the unconstrained permutation shown in Figure 2. From Figure 5, we see that the number of successful decodings with a Δ-permutation is the same as that of a uniform interleaving. However it must be noted that the bigger Δ, the smaller the resilience to erasure bursts. In future works, we plan to study the trade-off between the erasure burst resilience and decoding delay and optimally tune the parameter Δ.

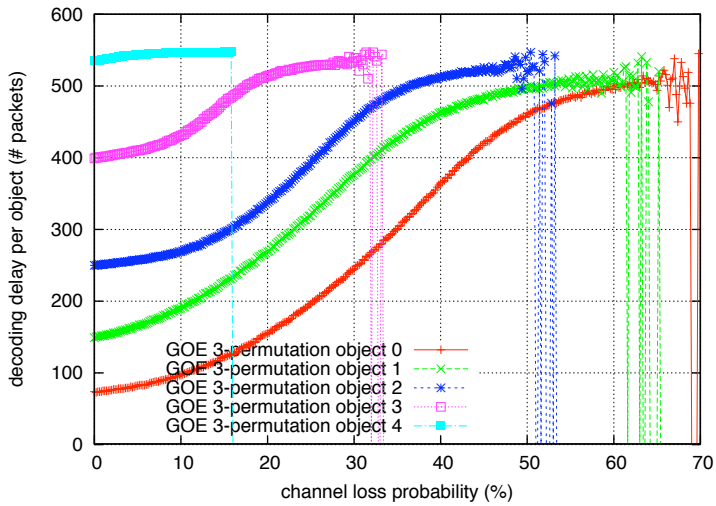

Figure 6: Decoding delay of the GOE system when the interleaving is performed with a Δ-permutation. $\Delta = 3$.

4 Processing metrics

Unlike PET, the GOE approach has been expressly designed to facilitate processing on both sides. This section analyses these considerations that are of high practical importance when it comes to deploying the solution for instance on lightweight terminals.

4.1 Processing load

In order to characterize the decoder processing load, we consider the number of packets "processed" (we explain what this term means below) by each method, PET and GOE. We assume that the decoding is performed continuously, each packet received being immediately analyzed and decoding launched if feasible, which is the usual way to proceed.

With PET, as long as at least one non decoded object remains, each incoming packet is processed and the new symbols for the non decoded objects are copied to their destination. If all objects are successfully decoded, $\max_{i\in[1,d]} \beta_i$ packets are processed. However as soon as the erasure probability is too high for at least one object not to be decoded (the receiver will not know this in advance in general), all the $n_{PET}(1-p_e)$ packets received are necessarily processed. Therefore the Number of Processed Packets (NPP) is:

$$\mathrm{NPP}_{PET} = \min\left(\max_{i\in[1,d]} \beta_i \ , n_{PET}(1-p_e)\right) \tag{12}$$

With GOE, a packet contains the contribution of a single object. Therefore if a packet of a decoded object is received, this packet is immediately discarded. It follows that the number of processed packets for object i is k_i, unless fewer packets have been received (i.e. if p_e is such that $n_i(1-p_e)\leqslant k_i$). Therefore the Number of Processed Packets is:

$$\mathrm{NPP}_{GOE} = \sum_{i=1}^{d} \min\left(k_i \ , n_i(1-p_e)\right) \tag{13}$$

Figure 7 shows the NPP values corresponding to formulas (12) and (13) and simulations (both results perfectly match). This figure shows the superiority of GOE with good channels and the equivalence of both solutions for higher values of the erasure probability. It is also worth noting that using GOE with a Δ-permutation does not modify the number of processed packets with a memoryless channel since it only affects the order but not the number of received packets for each object.

Lemma 3 (The number of packets processed is upper bounded)**.** *The number of packets processed by GOE is upper bounded by the number of packets processed for PET and the difference is very large when a successful decoding of all objects is possible.*

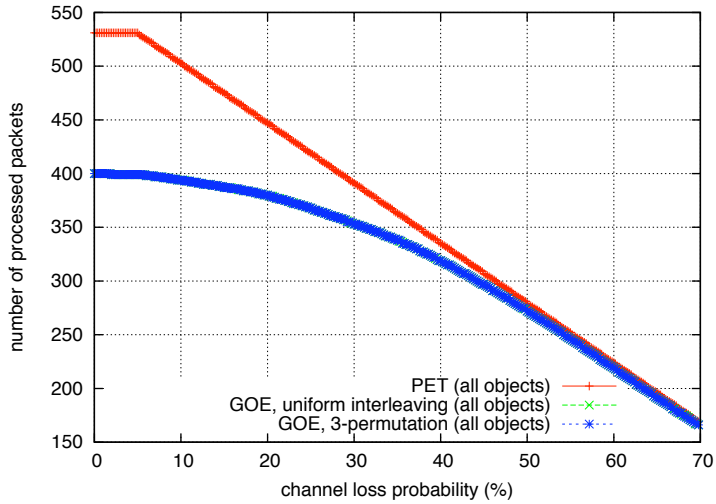

Figure 7: Number of processed packet.

Proof. We have:

$$\begin{aligned}\sum_{i=1}^{d} \min\left(k_i\ , n_i(1-p_e)\right) &\leqslant \min\left(\sum_{i=1}^{d} k_i\ , \sum_{i=1}^{d} n_i(1-p_e)\right)\\ &= \min\left(\sum_{i=1}^{d} k_i\ , n_{GOE}(1-p_e)\right)\\ &\approx \min\left(\sum_{i=1}^{d} k_i\ , n_{PET}(1-p_e)\right)\end{aligned}$$

The first inequality follows from the fact that $\min(a_1, b_1)+\min(a_2, b_2)\leqslant \min(a_1+a_2, b_1+b_2)$ since the LHS performs a minimization over a larger set than the RHS. More precisely, the LHS is $\min(a_1+a_2, b_1+b_2, a_1+b_2, b_1+a_2)$. The third equality results from Claim 1.

Since the second term in the minimization becomes predominant at high erasure probabilities, the number of packets processed for GOE and PET are then equivalent. On the opposite, if decoding is successful (i.e. small p_e), the minimum is obtained with the first operand and $\text{NPP}_{GOE} = \sum_{i=1}^{d} k_i$ which is much smaller than $\text{NPP}_{PET} = \max_{i\in[1,d]} \beta_i \approx n_{PET}$ □

Additionally, it should be noted that GOE processing of a packet is significantly simpler than PET processing. In the later case, for memory management reasons, each symbol must be copied to its final destination. Indeed, avoiding this copy means that each incoming packet is kept as such, which would lead to prohibitive memory consumption at a receiver, something which must be avoided, especially with a lightweight terminal. So this symbol copy is done to a remote location, for a few bytes (e.g. each symbol is a few tens of bytes long with the reference example of [2]), without any consideration for cache management. On the opposite GOE symbols are unmodified until they are used during FEC decoding.

4.2 Peak memory usage

Figure 8 shows the peak memory consumption as a function of the erasure probability. We see the peak memory usage of PET and GOE are equivalent whereas the Δ-permutation allows GOE to significantly decrease the peak. Thus, the pseudo-randomness of the Δ-permutation allows to better spread the decoding over the time, which frees regularly the memory. In future works, we will show the equivalence of the memory usage of PET and GOE for any parameter set.

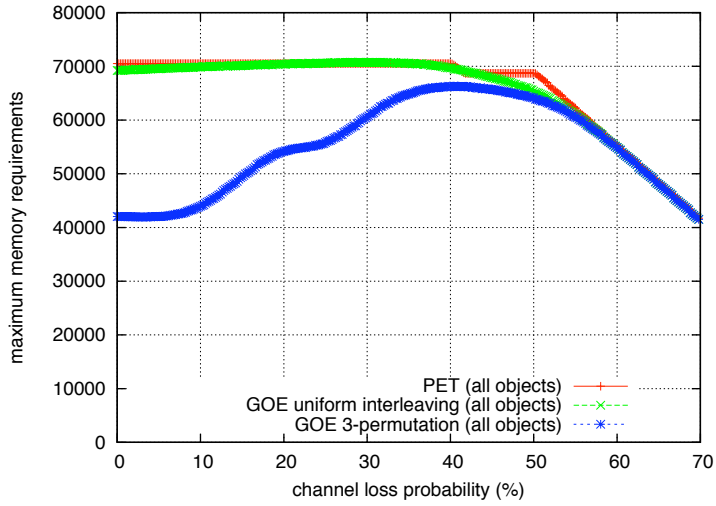

Figure 8: Peak memory usage as a function of the erasure probability p_e.

5 Object Bundle Protection

In this section we explain how to efficiently protect a bundle of (possibly small) objects, while solving the coupon collector problem (Section 1.2). We will show that the two approaches are quite different and that here also GOE outperforms UOD when practical aspects are considered.

5.1 UOD's handling of object bundles

PET being designed to provide UEP, [2] does not consider the object bundle problem. UOD extends PET and provides this extra functionality. If [7] does not detail how this is achieved, [8] gives more insights. In fact UOD's packetization scheme provides an elegant solution to the coupon collector problem since each packet received carries a symbol of each object of the bundle. However, although elegant, this approach has major practical consequences:

- the number of objects in a bundle is strictly limited to 255 for practical reasons;
- the larger this number, the smaller the symbol size, and therefore the higher the packet creation (at a sender) and processing (at a receiver) burden.

- certain bundle configurations cannot be suitably handled. For instance, protecting with a target code rate 1/2 a bundle of 32 objects of size 32 KB plus a single object of size 10 bytes, with a packet size of 1 KB, leads to a severe over-protection of the small file (actual code rate is 0.00146!), an under protection of other files (actual code rate is 0.571), and a packet size significantly smaller (900 B) than its target. Too many constraints exist, in particular the 4-byte alignment granularity, for UOD's packetization scheme to correctly handle this situation;
- last but not least, with d objects, d independent FEC encodings (at a sender) and d independent FEC decodings (at a receiver) are required. This feature has major impacts on the processing load on both sides.

5.2 GOE's handling of object bundles

GOE proceeds in a totally different manner. Let us assume that objects to be protected together have been submitted in sequence to the content delivery protocol and have been assigned sequential Transport Object Identifiers (TOI) values. These d objects are first segmented into symbols of the same length l, and gathered in blocks[1]. We use a "No-Code" FEC encoding of these source objects[2]. Thanks to this fake encoding, each symbol is individually identified by a {TOI, block ID, symbol ID} tuple. A Generalized Object (GO) is now constructed as the concatenation of the $\sum_{i=1}^{d} k_i$ symbols of these objects. A new TOI value is then assigned to this GO. It is sufficient for the sender to inform the receiver(s) of the tuple of the first symbol and the total number of symbols. Receivers can then determine with precision the "object bundle to GO" mapping (we assume the receiver knows the meta-data associated to each of the original objects, sent separately, e.g. using FLUTE "FDT Instance" signaling mechanisms). The GO is then processed as a usual object by the sender.

Therefore GOE features:

- no restriction on the number of objects in a bundle;
- no restriction on the symbol size (e.g. this size can be freely determined according to the Path MTU);
- no impact of the precise bundle configuration. A one byte object is efficiently protected as part of a bigger GO;
- the GO is FEC encoded (at a sender) and FEC decoded (at a receiver) once only. A receiver has a single system of linear equations to solve, no matter the d value;
- by construction, all the files of the bundle are protected equivalently (same coding rate);

We therefore conclude that **GOE outperforms UOD when a bundle of objects need to be globally protected**.

[1] Content delivery protocols like FLUTE introduce a blocking structure in order to comply with possible code dimension or length limitations.

[2] This "No-Code" scheme is commonly used in FLUTE/ALC sessions to transport objects that are small enough, e.g. one packet long.

6 Conclusions

This work has compared two techniques capable of providing both an unequal erasure protection service and an object bundle protection service, namely the UOD/PET scheme and the GOE scheme. Through a careful modeling of both proposals, whose accuracy has been confirmed by simulations, we have demonstrated that the protection performance of both approaches are equivalent, not only asymptotically but also in finite length conditions.

In fact the key differences between these approaches become apparent when applying them in practical systems. Such metrics as the simplicity of the solution, the number of packets processed, the maximum memory requirements, the number of FEC encoding and decodings, as well as the system of linear equations complexity (number of variables) are in favor of the GOE approach.

Another important limitation of UOD/PET, even if this is not easy to quantify, is its lack of flexibility. For instance the limited size of a packet creates an upper bound to the number of objects that can be considered together (e.g. UOD limits this number to 255), the symbol size has a coarse granularity (e.g. UOD requires symbols to be multiple of 4 bytes when used with RaptorQ codes) which can create rounding problems with certain sets of objects (i.e. the actual packet size may be significantly shorter than the target, and/or the actual code rate significantly different than its target).

GOE has no such constraints. In particular GOE offers the possibility to adjust the packet interleaving to the use-case and channel erasure features. One can easily trade robustness in front of long erasure bursts for very short decoding delays of high prioirity objects and low memory requirements, which can be a key asset in case of small, lightweight terminals or timely delivery services. This feature may be sufficiently important to justify by itself the use of a GOE FEC Scheme.

7 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.

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] T. M. Cover and J. M. Thomas, *Elements of Information Theory*, (Second Ed.), New-York: Wiley, 2006.

[6] J. Lacan, V. Roca, J. Peltotalo and S. Peltotalo "Reed-Solomon Forward Error Correction (FEC) Schemes", *IETF Request for Comments*, RFC 5510 (Standards Track/Proposed Standard), April 2009.

[7] M. Luby and T. Stockhammer, "Universal Object Delivery using RaptorQ", *IETF RMT Working Group*, Work in Progress: <draft-luby-uod-raptorq-00>", March 2011.

[8] M. Luby and T. Stockhammer, "Universal Object Delivery using RaptorQ - slides for the RMT meeting", *IETF 80 meeting*, RMT Working Group, "http://www.ietf.org/proceedings/80/slides/rmt-3.pdf", March 2011.

[9] M. Luby, A. Shokrollahi, M. Watson and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery", *IETF Request for Comments*, RFC 5053 (Standards Track/Proposed Standard), October 2007.

[10] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery", *IETF RMT Working Group*, Work in Progress: <draft-ietf-rmt-bb-fec-raptorq-06>", May 2011.

[11] V. Roca, C. Neumann and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes", *IETF Request for Comments*, RFC 5170 (Standards Track/Proposed Standard)", June 2008.

[12] 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.

[13] L. Rizzo, "Effective erasure codes for reliable computer communication protocols", *ACM Computer Communication Review,* Vol. 27 Issue 2, April 1997.

8 Appendix

8.1 Negative binomial distribution: definition and properties

In a sequence of independent Bernoulli(p) (denoted $\mathcal{B}(p)$) trials, let the random variable X denote the trial at which the rth success occurs, where r is a fixed and positive integer. Then

$$\mathbb{P}(X = x|r,p) = \binom{x-1}{r-1} p^r (1-p)^{x-r}, \ x = r, r+1, ..., \tag{14}$$

and we say that X has a negative binomial(r,p) distribution (denoted $\mathcal{NB}(r,p)$).[3]

The mean and variance of X are:

$$\mathbb{E}[X] = \frac{r}{p} \qquad \mathrm{Var}(X) = \frac{r(1-p)}{p^2} \tag{15}$$

8.2 Truncated negative binomial distribution: definition and properties

The N-truncated negative binomial distribution $\mathcal{NB}(r,p)$, denoted $\mathcal{TNB}(r,p,N)$, is the distribution of X conditioned on $X \leqslant N$, where N is a non-negative integer and X has a negative binomial distribution. Therefore the N-truncated negative binomial distribution $\mathcal{TNB}(r,p,N)$ has pmf (probability mass function):

$$\mathbb{P}(X = x|X \leqslant N, r, p) = \binom{x-1}{r-1} \frac{p^r (1-p)^{x-r}}{\sum_{j=r}^{N} \binom{j-1}{r-1} p^r (1-p)^{r-j}} \tag{16}$$

for $x \in [r, N]$ and $\mathbb{P}(X = x|X \leqslant N, r, p) = 0$ for $1 \leqslant x \leqslant r-1$ or $x > N$.

Note that there is no closed-form formula for neither the pmf (16) nor its moments, since there is no closed-form formula for the (untruncated) cumulative distribution function (cdf) $\mathbb{P}(X \leqslant x|r,p)$ [1, page 221]. However, the mean can be computed by summation:

$$\mathbb{E}[X|X \leqslant N, r, p] = \sum_{x=r}^{N} x \binom{x-1}{r-1} \frac{p^r (1-p)^{x-r}}{\sum_{j=r}^{N} \binom{j-1}{r-1} p^r (1-p)^{r-j}}. \tag{17}$$

8.3 Means of the untruncated and truncated negative binomial distribution

Lemma 4 (Upper bounds for the truncated mean)**.** *Let X be a discrete r.v. defined over $\mathbb{N}$ with pmf $\mathbb{P}(X = x)$. Let the N-truncated distribution of X be the distribution of X conditioned on $X \leqslant N$. The means of the untruncated and truncated distributions satisfy:*

$$\mathbb{E}[X = x|X \leqslant N] \leqslant \min(\mathbb{E}[X = x], N) \tag{18}$$

[3] Note that an alternative form of the negative binomial distribution consists in defining the random variable Y, which counts the number of failures before the rth success. This is equivalent to our definition, with $Y = X - r$.

Proof. First, note that $\mathbb{E}[X = x|X \leqslant N] \leqslant N$.
Let us now assume that $\mathbb{P}(X > N) > 0$, then

$$\begin{aligned}
\mathbb{E}[X = x] &= \mathbb{P}(X \leqslant N)\mathbb{E}[X = x|X \leqslant N] \\
&\quad + \mathbb{P}(X > N)\mathbb{E}[X = x|X > N] \\
&> \mathbb{P}(X \leqslant N)\mathbb{E}[X = x|X \leqslant N] \\
&\quad + \mathbb{P}(X > N)\mathbb{E}[X = x|X \leqslant N] \\
&> \mathbb{E}[X = x|X \leqslant N]
\end{aligned}$$

where the second inequality follows from the fact that $\mathbb{E}[X = x|X > N] > \mathbb{E}[X = x|X \leqslant N]$, since $\mathbb{E}[X = x|X \leqslant N] \leqslant N$ and $\mathbb{E}[X = x|X > N] > N$ if $\mathbb{P}(X > N) > 0$.
If $\mathbb{P}(X > N) = 0$, trivially $\mathbb{E}[X = x|X \leqslant N] = \mathbb{E}[X = x]$. Both results lead to:

$$\mathbb{E}[X = x|X \leqslant N] \leqslant \mathbb{E}[X = x].$$

Finally, combining the two upper bounds, we get (18). □

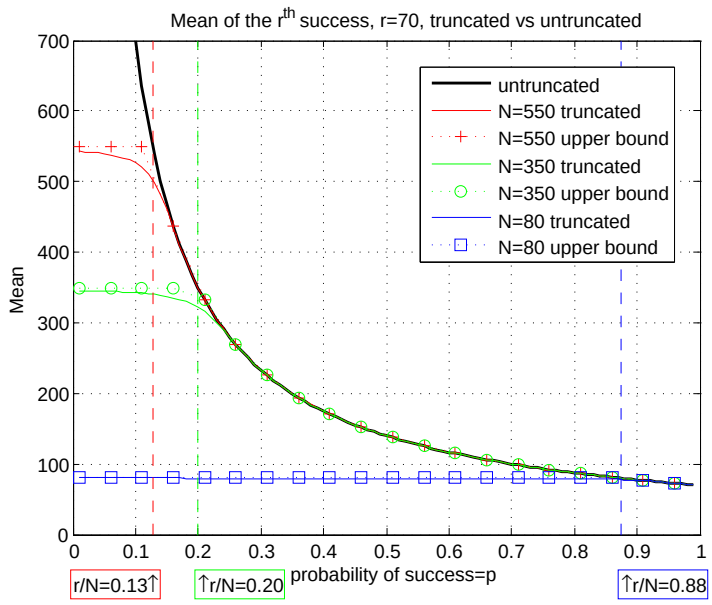

Figure 9: Mean of the truncated negative binomial distribution. Comparison with the mean of the untruncated law and with the upper bound (18). (18) is shown to be a tight bound and can serve as a good approximation for the mean.

Figure 9 shows the mean of the truncated negative binomial law $\mathcal{TNB}(r, p, N)$. First, we observe that the mean of the truncated only differs from the mean of the untruncated in a range of small probability of success p. More precisely, they differ when p is smaller than the value of p s.t. the untruncated mean equals N: $\mathbb{E}[X = x] = r/p \approx N$, therefore if $p \lessapprox r/N$. We further notice that the upper bound (18) for the mean of the $\mathcal{TNB}(r, p, N)$ is very tight especially when the total number of trials N is small. We can therefore claim that:

Claim 4 (Approximation for the truncated mean of the $\mathcal{TNB}(r,p,N)$ law.). *The mean of the truncated distributions is tightly approximated by:*

$$\mathbb{E}[X=x|X\leqslant N,r,p] \lessapprox$$

$$\min(\mathbb{E}[X=x],N) = \begin{cases} N & \textit{if } 0\leqslant p\leqslant \frac{r}{N}, \\ \frac{r}{p} & \textit{elsewhere} \end{cases} \tag{19}$$

Contents

1 Introduction **4**
1.1 Providing Unequal Erasure Protection 4
1.2 Providing an efficient protection to bundles of small objects . . . 4
1.3 Assumptions and problem position 5

2 Introduction to the PET and GOE UEP schemes **6**
2.1 PET Principles 6
2.2 GOE Principles 7

3 Erasure Recovery Metrics **9**
3.1 Infinite length analysis 9
3.2 Finite length analysis 10
3.3 Improving GOE with a Δ-permutation interleaving 14

4 Processing metrics **16**
4.1 Processing load 16
4.2 Peak memory usage 18

5 Object Bundle Protection **18**
5.1 UOD's handling of object bundles 18
5.2 GOE's handling of object bundles 19

6 Conclusions **20**

7 Acknowledgments **20**

8 Appendix **22**
8.1 Negative binomial distribution: definition and properties 22
8.2 Truncated negative binomial distribution: definition and properties 22
8.3 Means of the untruncated and truncated negative binomial distribution 22

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399