Publication 3

Tuomas Tirronen and Jorma Virtamo. 2007. Performance analysis of divided random linear fountain. In: Proceedings of the 2007 IEEE Global Telecommunications Conference (GLOBECOM 2007). Washington D.C., USA. 26-30 November 2007, pages 520-526.

Performance Analysis of Divided Random Linear Fountain

Tuomas Tirronen
Networking Laboratory
Helsinki University of Technology
Finland
Email: tuomas.tirronen@tkk.fi

Jorma Virtamo
Networking Laboratory
Helsinki University of Technology
Finland
Email: jorma.virtamo@tkk.fi

***Abstract*—The random linear fountain (RLF) is an efficient form of fountain coding with an expected overhead of only 1.6 packets. Because of increasing computational complexity, however, it cannot be directly used for large message sizes. In this paper, we study the performance penalty of dividing the data into k parts which are coded using the RLF. To ease the performance problem, we propose the use of macropackets which are generated by using LT-coding over the different parts of divided RLF. We calculate the decoding probabilities for $k = 2$ and 3 and compare the results to the overhead obtained using the combination of divided RLF and data carousel. The results indicate that while the use of macropackets improves the performance over the divided RLF, still better performance is obtained by data carousel in the case of low channel loss rate.**

I. Introduction

Fountain coding is an interesting coding paradigm for efficient transmission of data over an erasure channel without the need to acknowledge all the packets separately [1]. The symbol generation in fountain codes is rateless, requiring a virtually infinite supply of possible packets. This rules out some traditional codes such as the Reed-Solomon codes [2], though these can be used to provide fountain code-like erasure coding [3]. The main feature of these schemes is that the only task of recipient is to collect packets and eventually, when enough packets are collected, to construct the original message, thus no feedback is required for successful operation. Some types of the different fountain codes, their applications and other considerations are presented in [4].

We study random linear fountain (RLF) coding [5] and extensions to it. In addition to computational complexity of the coding and decoding algorithms, the performance of a fountain code is determined by the number of overhead packets required in order to decode the message. It turns out that the random linear fountain requires only 1.6 overhead packets in average for decoding of files with practically speaking any number of file blocks. This is a clear advantage compared to, for example, LT codes [6], where the overhead is higher. Although the LT codes are asymptotically optimal, for shorter files with number of blocks in a few hundreds the overhead is significant. However, the decoding process of the random linear fountain is computationally more complex than with LT codes, as it corresponds to solving a dense linear system of equations.

We suggest a divide-and-conquer algorithm in which the file is divided into parts of equal sizes each to be transferred with the RLF. In ideal situation, this would mean average overhead of 1.6 times the number of the parts. However, the results indicate that division of the original problem is not without some performance degradation. Especially if the sender does not know which parts are already solved, the performance is notably worse with multiple parts compared to a single one. This degradation happens because the encoder sends redundant packets to the already solved parts.

To utilize the possibility to attain low overhead of random linear fountain, we further propose a method where we combine the random linear fountain with macropackets. The macropackets are generated by performing LT-coding over the different parts. This way we try lower the overhead resulting from the sent redundant packets as the macropackets can be used in the decoding process in situations where some of the problems have already been solved. The divided RLF with macropackets could be used with files where the number of blocks is in few hundreds, a regime where some of the more advanced fountain codes operate with higher overheads.

We also study the use of data carousel, where packets are sent from each of the subproblems cyclically in order. In contrast to the other methods, the encoded packets here are no longer i.i.d. and the performance is therefore channel dependent, i.e. it cannot be expressed solely in terms of the number of received packets without reference to which packets were lost. We show that for channels with low losses the performance of the data carousel is good but in the case of very heavy losses it gains nothing in comparison with the basic divided RLF.

This paper is organized as follows. First in Section II the idea of the random linear fountain is reviewed. After this, in Section III the division of the RLF into several subproblems is considered and the performance of the system analyzed. Then, in order to eliminate some of the extra overhead, we suggest the use of macropackets and analyze the achieved performance gain in Section IV. In Section V, we introduce and analyze still another mechanism, the data carousel, where the encoded packets are sent cyclically from different subproblems. Numerical comparisons are presented in Section VI and, finally, we conclude in Section VII.

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

II. Random Linear Fountain

We consider a file which consist of n blocks denoted $\mathbf{m} = (m_1, m_2, \ldots, m_n)$, where $\mathbf{m}$ is the whole file (message) and m_i ith block. Note that block sizes are arbitrary.

The encoding process generates random linear combinations of the blocks over modulo-2 addition. A packet c_i is generated by selecting each of the blocks to be included uniformly with probability $1/2$. The recipient collects these equations forming a linear system of equations and the goal is to obtain a system of full rank in order to reconstruct the original message.

From statistical point of view all of the generated packets are stochastically independent, thus no one packet is more important than another. This meant that the channel properties are irrelevant from the point of view of the process and its properties. The only assumption is that we use an erasure channel [7], i.e., the receiver can be assumed to receive packets intact as erroneous packets are removed by the channel.

A. Decoding Probability

Next we will calculate the decoding probability of RLF with a given number of received packets. If the file consists of n blocks, the number of possible different encoded packets is $2^n - 1$ excluding the all-zero packets, each representing an equation. When the recipient has m linearly independent vectors, the probability for the next one to be linearly dependent, i.e. the probability of failure, is $\mathcal{P}_m = (2^m - 1)/(2^n - 1)$,where the numerator gives the size of the subspace of all linear combinations of m linearly independent vectors, and the denominator is the size of the whole space, in both cases excluding the zero vector.

Let T_m be the random variable representing the number of extra trials needed for a new linearly independent equation when the recipient already has m independent equations. T_m obeys the geometric distribution,

$$\mathcal{P}\{T_m = i\} = \mathcal{P}_m^i(1 - \mathcal{P}_m), \quad i = 0, 1, 2, \ldots$$

The total number of extra packets, $T(n)$, needed for successful decoding of n source blocks is the sum of independent random variables $T(n) = \sum_{m=1}^{n-1} T_m$. Only a few last terms in the sum have an appreciable probability for being different form zero. Moreover, when n is large, say $n > 10$, we have $\mathcal{P}_{n-j} \approx 2^{-j}$ and the distributions of T_{n-1}, T_{n-2},... tend to given fixed forms. Correspondingly, their sum tends in distribution to the random variable T which is the sum of geometrically distributed random variables (starting from zero) with parameters 1/2, 1/4, 1/8,.... Its distribution function, $\tilde{F}(t) = \mathcal{P}\{T \leq t\}$, can easily be computed by convolution. The expected overhead is small, $\overline{T} = \sum_{i=1}^{\infty} 1/(2^i - 1) \approx 1.61$, regardless of the size of the problem. In the sequel we denote by

$$F_n(t) = \tilde{F}(t - n)$$

the distribution function of the number of packets needed for successful decoding of n blocks. Next we will study how the situation changes when the file is divided into parts and RLF is applied to each of these parts.

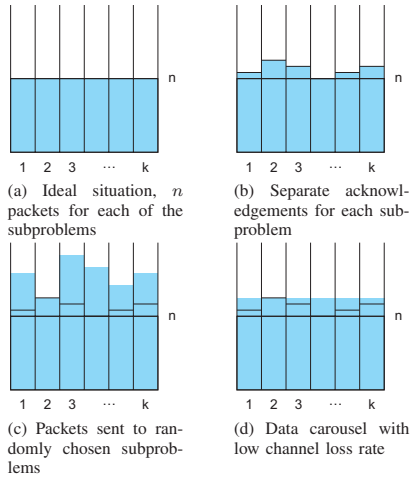
III. Dividing the Random Linear Fountain into Multiple Subproblems

The problem with the scheme presented in the previous Section is that the decoding is computationally very expensive for large n, as solving the system of linear equations has computational complexity of $O(n^3)$ with standard Gaussian elimination. The Strassen algorithm [8] could be used to obtain complexity $O(n^{\log_2 7})$, but this has a higher constant term and is a better algorithm only for n high enough. In any case, a lower bound for the complexity is at least $\Omega(n^2)$, which still grows too fast with n.

We suggest a divide-and-conquer algorithm where the original problem (file) is divided into multiple subproblems (parts) and each of these parts if solved using the RLF. The subproblem is chosen uniformly at random and a packet is generated from the corresponding part of file. Note that the encoded packets are still i.i.d. retaining the rateless character of the coding scheme. The minimum expected overhead grows linearly with k as $k \cdot \overline{T}$. Ideally, one wants to keep the overhead as close as possible to this value, see Fig. 1(b). However, in the spirit of the fountain coding paradigm one wants to limit the use of the backward channel. Thus, we assume that different subproblems are not acknowledged separately. Scenarios where this would be preferable include for example massive parallel downloading and multicasting [4]. This strategy, however, has a performance penalty since the encoder keeps sending packets to subproblems which are already solved until all of them are solved, see Fig. 1(c).

A. Performance of Random Linear Fountain

The data consisiting of $N = k \cdot n$ source blocks are partitioned into k equally sized subproblems, each of which contains n source blocks. In the encoding algorithm, one of the



(a) Ideal situation, n packets for each of the subproblems

(b) Separate acknowledgements for each subproblem

(c) Packets sent to randomly chosen subproblems

(d) Data carousel with low channel loss rate

Figure 1. Illustration of the process during different packet encoding strategies for subproblems. The shaded areas indicate the total number of packets received and lines the number of needed packets.

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

Table I
MEAN NUMBER OF PACKETS REQUIRED FOR DECODING AND CORRESPONDING OVERHEAD PERCENTAGES

N	k	n	E[T]	overhead %
128	2	64	141	10
	4	32	160	25
	8	16	196	53
	16	8	268	109
200	2	100	215	7.5
300	2	150	317	5.7
	3	100	331	10
400	2	200	419	4.8
450	3	150	487	7.6
600	3	200	641	6.8

subproblems is first chosen uniformly at random in order to preserve the stochastic independence of the generated packets. Then RLF is applied to the selected part and the encoded packet is sent to the channel.

As discussed, dividing the whole problem into k subproblems that are not separately acknowledged leads to performance degradation, as one yet unsolved problem delays the complete solution of the whole problem while the encoder is spraying packets to all of them. Next we analyze the decoding decoding performance of this scheme.

Let ν be the total number of received packets. This further divides into categories of the number of received packets ν_κ for each of the k subproblems. We denote $\boldsymbol{\nu} = (\nu_1, \nu_2, \ldots, \nu_k)$ and $|\boldsymbol{\nu}| = \sum_\kappa \nu_\kappa$. The probability for a generated packet belonging to subproblem κ is $1/k$ for all κ. Thus $\boldsymbol{\nu}$ obeys the multinomial distribution:

$$\mathcal{Q}(\boldsymbol{\nu}) = \frac{\nu!}{\nu_1!\nu_2!\cdots\nu_k!}\left(\frac{1}{k}\right)^\nu = \nu!\prod_{\kappa=1}^{k}\frac{\left(\frac{1}{k}\right)^{\nu_\kappa}}{\nu_\kappa!}.$$

Given the composition $\boldsymbol{\nu}$, the probability of successful decoding of all the subproblems is $F_n(\nu_1)F_n(\nu_2)\cdots F_n(\nu_k)$. By deconditioning, the probability for successful decoding with at most ν received packets is

$$\mathcal{R}_k(\nu) = \nu!\sum_{|\boldsymbol{\nu}|=\nu}\prod_{\kappa=1}^{k}\frac{\left(\frac{1}{k}\right)^{\nu_\kappa}}{\nu_\kappa!}F_n(\nu_\kappa) = \nu!\sum_{|\boldsymbol{\nu}|=\nu}\prod_{\kappa=1}^{k}G(\nu_\kappa)$$

where $G(\nu) = \frac{(1/k)^\nu}{\nu!}F_n(\nu)$. Now, the sum is recognized to be of the type that can be calculated by convolution:

$$\mathcal{R}_k(\nu) = \nu!\bigotimes_{\kappa=1}^{k}\mathbf{G}[\nu],$$

where $\mathbf{G} = (G(0), G(1), \ldots, G(\nu))$.

Table I shows some examples of the mean overhead in this scheme. The results clearly indicate that increasing the number of subproblems k rapidly deteriorates the performance. This is due to the fact that the actual numbers of packets received by the subproblems are not equal (even though their expectations are); the expectation of the minimum of a set of i.i.d. random variables is less than the mean, the more so the larger k is.

IV. MACROPACKETS

A typical scenario occurring in the decoding process is that most of the subproblems are solved but some of them are still missing a few packets for getting a full set of linearly independent equations. This problem of missing last pieces could be helped by the use of *macropackets*. A macropacket comprises of information from multiple different subproblems, i.e. RLF encoded packets from a few subproblems are combined by the XOR operation. If all but one of these subproblems have already been solved, the macropacket can be reduced to include the encoded packet of only the subproblem in question, thus is effect, making the number of received packets in different subproblems more even. As before, we assume that individual problems are not acknowledged when they are ready.

The macropackets are generated similarly as encoding symbols are generated with LT codes [6]. To this end, we define degree distribution $\rho(d)$ over the subproblems, the degree d here referring to the number of subproblems involved in the construction of the macropacket, $d = 1, \ldots, k$. Note that degree one corresponds to sending an RLF packet from a single subproblem. To be more precise, the following steps are performed:

1) Choose degree d from $\rho(d)$.
2) Choose uniformly at random d subproblems and generate a packet p_i, $i = 1, \ldots, d$ in each of the chosen subproblems by RLF.
3) Combine packets from the chosen d subproblems by XOR, i.e., $m = p_1 \oplus p_2 \oplus \ldots \oplus p_d$.
4) Send macropacket m.

The information of which subproblems are included in a macropacket has to be somehow made available to the decoder. This could be, for example, done by a header with bitmap expressing the composition or using pseudo-random number generators with same seed at both sender and receiver parts [6].

The decoding is a combination of LT decoding and RLF decoding. All received macropackets are stored. Once the number of degree-1 packets received in some of the subproblems is large enough, the number of linearly independent ones equalling n, that subproblem is solved (all blocks known) and blocks belonging to that subproblem can be subtracted (by XOR'ing) form the stored macropackets. In this process, a macropacket may eventually be reduced to a degree-1 packet, thus increasing the number of received RLF encoded packets in one of the yet unresolved subproblems.

A. Analysis of the Performance with Macropackets

We proceed by analyzing the decoding probabilities obtained by using macropackets. The type of a macropacket is defined by the subproblems that are involved in its construction. There are 2^k-1 different types of mackropackets (excluding an empty one). For instance, if there are two subproblems A and B, then the types of macropackets can be designated as A, B and AB. Let $\boldsymbol{\nu}$ now denote the 2^k-1-vector specifying the numbers of received mackropackets of each kind. In the

1930-529X/07/$25.00 © 2007 IEEE

encoding process, the probability that a macropacket of a given type i with degree d is generated is $p(i) = \rho(d)/\binom{k}{d}$ as there are $\binom{k}{d}$ different types of macropackets of degree d, each of them of equal probability once the degree d has been chosen.

After having received ν macropackets, the probability $\mathcal{Q}_k(\boldsymbol{\nu})$ (emphasizing the division into k subproblems) of a given vector $\boldsymbol{\nu}$, with $|\boldsymbol{\nu}| = \nu$, is given by the multinomial distribution with the above probabilities $p(i),\ i = 1, \ldots, 2^k - 1$. Then, the decoding probability can be written as

$$\mathcal{R}_k(\nu) = \sum_{|\boldsymbol{\nu}|=\nu} \mathcal{Q}_k(\boldsymbol{\nu})\mathcal{P}_k(\boldsymbol{\nu}), \tag{1}$$

where $\mathcal{P}_k(\boldsymbol{\nu})$ is the probability of decoding with the number of macropackets of different types as specified by $\boldsymbol{\nu}$. In the sequel, we will derive explicit expressions for $\mathcal{P}_k(\boldsymbol{\nu})$ for the cases $k = 2$ and $k = 3$.

Performing the 2^k-1-fold summation in (1) is directly feasible only for $k = 2$. Therefore, Monte Carlo summation is used for obtaining the results in Section VI,

$$\mathcal{R}_k(\boldsymbol{\nu}) \approx \frac{1}{S}\sum_{s=1}^{S} \mathcal{P}_k(\boldsymbol{\nu}_s), \tag{2}$$

where the samples $\boldsymbol{\nu}_s$ are drawn from the multinomial distribution $\mathcal{Q}_k$ and S is the sample size.

Sampling from multinomial can be done sequentially using the marginal distributions, i.e., binomial distribution, by first sampling the number of type-1 packets ν_1 from $\mathrm{Bin}(\nu, p(1))$, and for the rest always subtracting the already generated packets from ν and using conditional probabilities,

$$\nu_i \sim \mathrm{Bin}\Big(\nu - \textstyle\sum_{j=1}^{i-1}\nu_j,\ \frac{p(i)}{\sum_{j=i}^{k} p(j)}\Big), \quad i = 2, 3 \ldots$$

1) Case $k = 2$: The two subproblems are denoted by A and B. There are three types of macropackets, A, B and AB, with the corresponding numbers of received packets, ν_A, ν_B and ν_{AB}. Let $A[\nu]$ and $B[\nu]$ denote the events that problem A or B is resolved when there are ν macropackets of type A or B available, respectively, let them be original degree-1 packets or packets obtained by reduction from macropackets of higher degree. By the result in Section II-A we have $\mathrm{P}[A[\nu]] = F_n(\nu)$.

The decoding probability (1) can now be written as

$$\mathcal{R}_2(\nu) = \sum_{\nu_A+\nu_B+\nu_{AB}=\nu} \frac{\nu!}{\nu_A!\nu_B!\nu_{AB}!}\left(\frac{\rho_1}{2}\right)^{\nu_A+\nu_B} {\rho_2}^{\nu_{AB}}\mathcal{P}_2(\boldsymbol{\nu}),$$

where $\mathcal{P}_2(\boldsymbol{\nu})$ is the probability that the problem is solved with received packet composition $\boldsymbol{\nu} = (\nu_A, \nu_B, \nu_{AB})$.

To derive a formula for $\mathcal{P}_2(\boldsymbol{\nu})$ we first identify the different ways how the problem can be solved. We have four mutually exclusive events (i) $A[\nu_A] \cap B[\nu_B]$, (ii) $A[\nu_A] \cap \overline{B[\nu_B]}$, (iii) $\overline{A[\nu_A]} \cap B[\nu_B]$, (iv) $\overline{A[\nu_A]}\,\overline{B[\nu_B]}$. In the first case the whole problem is surely solved and in the last one surely not. In the second case, the problem is not directly solved with the original degree-1 packets but it is solved in the event $B[\nu_B + \nu_{AB}]$ that problem B is solved taking into account the reduced type AB macropackets; the same holds for the third case with A and B interchanged. The contribution of the first case to the total success probability is

$$\mathcal{P}\left(A[\nu_A] \cap B[\nu_B]\right) = F_n(\nu_A)F_n(\nu_B).$$

The contribution of the second case is

$$\mathcal{P}\left(A[\nu_A] \cap \overline{B[\nu_B]} \cap B[\nu_B + \nu_{AB}]\right) = F_n(\nu_A)(F_n(\nu_B + \nu_{AB}) - F_n(\nu_B)),$$

and similarly for the third case. Collecting these together we have

$$\mathcal{P}_2(\nu_A, \nu_B, \nu_{AB}) = F_n(\nu_A)F_n(\nu_B + \nu_{AB}) + F_n(\nu_B)F_n(\nu_A + \nu_{AB}) - F_n(\nu_A)F_n(\nu_B). \tag{3}$$

2) Case $k = 3$: The analysis of the decoding probability is similar to the case $k = 2$. Now we have 7 different types of macropackets: A, B, C, AB, BC, AC and ABC. We also have 8 mutually exclusive events according to which subproblems are directly solved with the degree-1 macropackets. Taking symmetry into account the cases where decoding is possible can be divided into three classes (i) either all of the problems are solved directly, (ii) two of the problems are solved directly and one with the help of the macropackets, and (iii) one problem solved directly and the two others with the help of macropackets. The contributions from the first one is

$$\mathcal{P}(A[\nu_A] \cap B[\nu_B] \cap C[\nu_C]) = F_n(\nu_A)F_n(\nu_B)F_n(\nu_C)$$

and from the second one (to show just one permutation of the indices)

$$\mathcal{P}(A[\nu_A] \cap B[\nu_B] \cap \overline{C[\nu_C]} \cap C[\nu_C + \nu_{AC} + \nu_{BC} + \nu_{ABC}]) = F_n(\nu_A)F_n(\nu_B)(F_n(\nu_C + \nu_{AC} + \nu_{BC} + \nu_{ABC}) - F_n(\nu_C)).$$

Class (iii) is a little bit trickier and needs further division into subcases. Because of lack of space, we omit the details and only give the final result where all the terms have been collected. Denoting $\boldsymbol{\nu} = (\nu_A, \nu_B, \nu_C, \nu_{AB}, \nu_{AC}, \nu_{BC}, \nu_{ABC})$, it can most compactly be written in terms of $\mathcal{P}_2$ of (3) as

$$\begin{aligned}\mathcal{P}_3(\boldsymbol{\nu}) = {} & F_n(\nu_A)F_n(\nu_B)F_n(\nu_C) + \\ & \mathcal{P}_2(\nu_A, \nu_B, \nu_{AB})F_n(\nu_C + \nu_{AC} + \nu_{BC} + \nu_{ABC}) + perms \\ & - F_n(\nu_A + \nu_{AC})F_n(\nu_B + \nu_{BC})F_n(\nu_C) - perms,\end{aligned}$$

where *perms* stands for two other structurally identical terms obtained by cyclic permutations of the indices.

V. Data Carousel

The major problem with the division of the problem into subproblems and selecting the encoded packet from one of the subproblems at random is that the number of packets generated into different subproblems are not equal but fluctuate around their expectations. An obvious question then is, why not try to make the numbers as equal as possible by selecting the subproblem cyclically in a data carousel fashion.

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

The main argument against this kind of scheme is that it means abandoning one of the basic principles of fountain coding, namely that the encoded packets should be independent and stochastically identical. With the data carousel, the packets are of course correlated. The coding scheme is no longer rateless and the performance cannot be expressed solely in terms of the number of received packets (no matter what has been lost) but depends also on the channel properties.

It is, anyhow, of interest to study how the data carousel performs under some channel model. Before doing this, we discuss some simple limits. If there are no channel losses, the data carousel works as ideally designed, delivering equal amounts of packets into all subproblems, see Fig. 1d. When ν is a multiple of k, the decoding probability is

$$\mathcal{R}_k(\nu) = F_n(\nu/k)^k. \tag{4}$$

as each of the k subproblems has to be solved with ν/k packets. At the other extreme, with heavy channel losses the packets get through only on rare occasions and the situation from the receiver's end looks similar as in the basic decoding scheme, i.e. the subproblem a received packet belongs to is random (unless the loss process has very long correlations).

A. Independent loss model

As shown above the performance of the data carousel can be inferred easily for the cases of no losses and very heavy losses. In the intermediated cases the performance depends on the exact loss characteristics of the channel Here we study the most obvious simplistic channel model, i.e., that of independent losses, where each packet is lost independent of others with a given probability p.

The decoding probability can again be expressed in the form (1) and estimated by Monte Carlo summation as in (2). In the present case $\boldsymbol{\nu}$ is the k-vector $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_k)$ and the decoding probability $\mathcal{P}_k(\boldsymbol{\nu})$ with a given vector $\boldsymbol{\nu}$ is $\mathcal{P}_k(\boldsymbol{\nu}) = F_n(\nu_1) \cdots F_n(\nu_k)$. In order to be able to apply (2) we still need a method to generate samples of $\boldsymbol{\nu}$ with given total number ν of received packets, $|\boldsymbol{\nu}| = \nu$. This is easily done, as with the assumed model the sending-sequence-number difference U of two consecutive received packets is geometrically distributed,

$$\mathcal{P}(U = u) = (1-p)p^{u-1}, \quad u = 1, 2, \ldots.$$

Thus from a given realization of independent random variables $U_1, U_2, \ldots, U_\nu$, we get the sequence number, and ultimately the subproblem number, of the ith received packet as $(\sum_{j=1}^{i} U_j) \bmod k$. Counting the number of packets, out of the total ν, belonging to each subproblem κ we get the components ν_κ of the sample vector $\boldsymbol{\nu}$.

In Fig. 2 we see how the performance degrades with the growing loss probability, in accordance with the previous discussion, starting from that of a lossless channel and tending to the performance of a system where for each sent (and the received, as well) packet the subproblem is chosen at random.

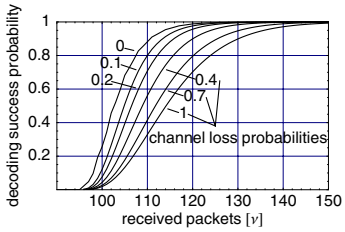


Figure 2. The decoding success probability in a system with data carousel and $k = 3$, $n = 32$ as a function of the number of received packets ν for the independent losses channel model with different loss probabilities.

VI. Numerical Results

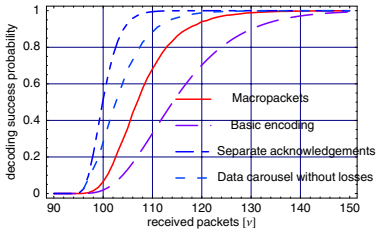
In Fig. 3 and Table II we compare the performance of the discussed variants of the coding scheme for the case of $k = 3$ subproblems each containing either $n = 32$ or $n = 100$ blocks (in Table II also $n = 64$ blocks). As a reference we use the scheme where each subproblem is acknowledged separately, with the mean overhead of $1.6 \cdot k$ packets, represented by the leftmost curve. We compare the basic scheme with subproblem division, the use of macropackets and the data carousel, here shown for the optimistic case of no packet losses.

The results for the case of macropackets depend on the degree distribution ρ. It was here numerically optimized to obtain the best results (studies on the optimization of the degree distribution of LT-codes are presented in [9], [10]). The optimized distributions are shown in Table II. For the cases of macropackets and data carousel the results were obtained using Monte Carlo summation with 10^5 sample points.

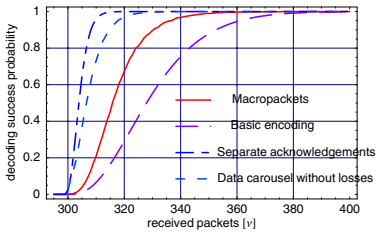
We note that the use of macropackets gives average performance roughly in the middle between the basic method and the optimal case (separate acks, cf. Fig. 1b). In a lossless channel the data carousel gives better performance than macropackets and is rather near to the ideal scenario. As the size of the problem grows the relative overheads become smaller. Comparing to Fig. 2 we see that the decoding probabilities with macropackets in scenario $k = 3, n = 32$ are nearly the same as with data carousel when channel loss probability $p = 0.2$. Thus, with lower channel loss probabilities and independent losses the performance of data carousel is better than with developed divided RLF method with macropackets.

The relative performance of the compared schemes can be studied by examining Fig. 1. Ideally, there is no overhead at all as all k subproblems are finished with just n packets. With separate acknowledgements for individual subproblems, plotted in Fig. 3, the required number of packets needed for decoding is random for each subproblem. However, the acknowledgement sent after completion stops the encoder from producing packets for a specific subproblem. Worst performance is obtained in scenario 1c where both the overhead and the number of required packets for decoding are random. Performance of data carousel falls in between the last two scenarios depending on the channel loss properties, such as the channel loss probability.

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.



(a) $n = 32$



(b) $n = 100$

Figure 3. Basic encoding compared to the use of macropackets, data carousel and separate acknowledgements in the case $k = 3$

Table II
MEAN OVERHEADS FOR CASE $k = 3$

		Mean overhead		
n	$\rho(d)$	Separate acks	Macropackets	No macro
32	$\{0.89, 0.10, 0.01\}$	4.82 (5.0%)	12.5 (13.1%)	20.3 (21.1%)
64	$\{0.911, 0.08, 0.01\}$	4.82 (2.5%)	15.6 (8.1%)	26.0 (13.5%)
100	$\{0.936, 0.05, 0.014\}$	4.82 (1.6%)	18.2 (6.1%)	31.0 (10.3%)

VII. CONCLUSIONS

We have studied the random linear fountain with a divide-and-conquer strategy to lower the computational complexity. Also, to retain the properties of fountain codes we chose not to allow separate acknowledgements for the resulting subproblems. This avoids the need for a backward channel save the possibility to send an acknowledgement when the whole file has been received completely.

First we presented a basic encoding scheme where the random linear fountain is divided into multiple subproblems and each of these problems is solved separately. The probability of decoding with a specified number of received packets was calculated. The results in Section III clearly show that the further we divide the original file, the worse performance we get in terms of the overhead. While this is natural, the results indicate that the performance is much worse than the optimal one of $1.6 \cdot k$ overhead packets, which would be acceptable in most scenarios. However, as the file size in terms of blocks grows, the relative overhead can supposedly be made arbitrary small.

To improve the performance we proposed the use of macropackets. Macropackets are generated by utilizing LT-coding over the packets generated from multiple subproblems. Explicit results for the decoding probabilities with macropackets were obtained for cases $k = 2$ and $k = 3$ (it remains a problem for further research to find similar formulae for larger k). For $k = 3$, Monte Carlo summation was used to calculate the numerical results. The results in Section VI show that the macropackets improve the performance by approximately $4-6$ percentage points for tested scenarios, where the performance gain seems to be smaller for higher values of n. Thus, the addition of macropackets provides the best performance gain for small file sizes, which are problematic for example when using LT codes.

Further, the developed strategies were compared with the data carousel. The results show that for a channel with low losses the performance of the data carousel is almost the same as with ideal strategy of separately acknowledging each of the subproblems. The data carousel, however, is not a genuine fountain coding strategy and the performance depends on the channel properties. It exhibits poor performance, equal to that of the basic divided RLF, for scenarios with a very high loss rate. Performance for intermediate losses was studied for independent loss model for the data channel.

To conclude, dividing the RLF into separate subproblems lessens the computational burden but this does not come without performance penalty, unless each of the subproblems is acknowledged separately. The performance degradation can be mitigated by the use of macropackets but it still remains a problem. The same performace as with macropackets can be obtained with the data carousel in a channel with independent losses and loss probability less than 20 %. This suggests that the data carousel, even though violating the pure fountain coding principle, might be an idea worth considering in many practical scenarios. An interesting question for further research is to go a step further and develop an optimal (non-stationary) sending strategy for a given channel model.

REFERENCES

[1] J. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communications, 20(8)*, October 2002.

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

[3] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, Apr. 1997.

[4] M. Mitzenmacher, "Digital fountains: A survey and look forward."

[5] D. J. MacKay, "Fountain codes," *IEE Proceedings Communications*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[6] M. Luby, "LT Codes," in *Proc. The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–282.

[7] D. J. C. Mackay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[8] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. The John Hopkins University Press, 1996.

[9] E. Hyytiä, T. Tirronen, and J. Virtamo, "Optimizing the degree distribution of LT codes with an importance sampling approach," in *Proc. The 6th International Workshop on Rare Event Simulation*, Oct. 2006, pp. 64–73.

[10] ——, "Optimal degree distribution for LT codes with small message length," in *Proc. IEEE INFOCOM'07*, May 2007, pp. 2576–2580.