Decentralised Distributed Fountain Coding: Asymptotic Analysis and Design

Dino Sejdinovic, Robert Piechocki, Angela Doufexi, and Mohamed Ismail

***Abstract*—A class of generic decentralised distributed fountain coding schemes is introduced and the tools of analysis of the performance of such schemes are presented. It is demonstrated that the developed approach can be used to formulate a robust code design methodology in a number of instances. We show that two non-standard applications of fountain codes, fountain codes for distributed source coding and fountain codes for unequal error protection lie within this decentralised distributed fountain coding framework.**

***Index Terms*—Rateless codes, asymptotic analysis, decentralised code design**

I. Introduction

We assume that the reader is acquainted with standard fountain code design and analysis, cf. [1], [2], [3] for the overview of fountain codes.

The aim of a decentralised distributed fountain coding (DDFC) scheme is to reliably recover the data possibly distributed across a set of nodes in a network, called source nodes, at another set of nodes, called collector nodes, with minimal number of transmissions. In our setting, we assume that each collector node seeks to recover a data sequence $\mathbf{x} = (x_1, x_2, \ldots, x_k)$, consisting of k data packets $x_i \in \mathbb{F}_2^b$, $i = \overline{1,k}$, which are vectors of length b over $\mathbb{F}_2$, and that each source node in a network has access to a subset of data sequence $\mathbf{x}$. Furthermore, each source node is oblivious of which data packets are available at other source nodes and the sets of packets available at different source nodes are not necessarily disjoint. Each source node uses a fountain code, i.e., LT or Raptor code, to produce encoding packets over its subset of data packets and multicasts these encoding packets to collector nodes. The challenge in designing an efficient and robust DDFC scheme lies within the fact that source nodes do not cooperate and thus are not able to produce a resulting bitstream resembling that of a good fountain code. Rather, they produce *localised* encoding packets, linear projections restricted to their respective subsets of coordinates in data sequence $\mathbf{x}$. However, we will show that by using an appropriate generalised version of standard techniques for analysis of sparse graph codes and fountain codes, we can formalise a robust code design methodology for a number of important instances of the DDFC framework. It is an interesting insight that for some typical single source multicast fountain coding problems such as fountain codes for unequal error protection (UEP), it may still be useful to study these problems as instances of DDFC, where a single source node produces various classes of localised encoding packets in order to attain the UEP property.

Manuscript received July 29, 2009. The associate editor coordinating the review of this letter and approving it for publication was V. Stankovic.

D. Sejdinović, R. Piechocki, and A. Doufexi are with the Centre for Communications Research, Department of Electrical and Electronic Engineering, University of Bristol, Bristol, UK (e-mail: {d.sejdinovic, r.j.piechocki, a.doufexi}@bristol.ac.uk).

M. Ismail is with Toshiba Research Europe Ltd, Telecommunications Research Laboratory, Bristol, UK (e-mail: mohamed@toshiba-trel.com).

This work was supported by Toshiba TRL Ltd.

Digital Object Identifier 10.1109/LCOMM.2010.01.091541

II. Problem Overview

Let $k, b \in \mathbb{N}$, $\varepsilon > 0$, and $n = \lceil k(1+\varepsilon) \rceil$. Let $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ be a data sequence of k data packets $x_i \in \mathbb{F}_2^b$, $i \in N_k$ that needs to be communicated to the collector nodes.[1] Assume that a collector node obtains a sequence $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ of n encoding packets produced in a decentralised fashion at a number of source nodes, i.e., the code overhead available to the collector node is ε. We will describe the decentralised generation of the encoding packets by a weighted complete bipartite graph $\mathcal{G} = (\mathcal{A}, \mathcal{B}, \Theta)$, illustrated in Fig. 1a. In $\mathcal{G}$, nodes $\mathcal{A} = \{A_1, A_2, \ldots A_r\}$ represent a disjoint partition of N_k, such that $\forall i \in N_r$, $|A_i| = \pi_i k$, for some $\pi_i \in [0,1]$, and nodes $\mathcal{B} = \{B_1, B_2, \ldots B_s\}$ represent a disjoint partition of N_n, such that $\forall j \in N_s$, $|Y_j| = \gamma_j n$, for some $\gamma_j \in [0,1]$, and $\Theta = (\theta_i^j)$ is an $k \times n$ matrix, such that θ_i^j is the weight associated with the edge $A_i B_j$. The weights are normalized such that $\forall j \in N_s$, $\sum_{i \in N_r} \theta_i^j = 1$. Note that also $\sum_{i \in N_r} \pi_i = 1$, $\sum_{j \in N_s} \gamma_j = 1$, by construction. We can think of A_i, $i \in N_r$, and B_j, $j \in N_s$ as determining the divison of raw data packets and encoding data packets, respectively, into classes: sequence x_{A_i} is the i-th class of raw data packets and sequence y_{B_j} is the j-th class of encoding packets. Each of the s source nodes has direct access to a certain portion of vector $\mathbf{x}$, and the set of coordinates available to source node $j \in N_s$ is x_{C_j}, where $C_j = \cup_{\theta_i^j \neq 0} A_i$. Graph $\mathcal{G}$ is a key part of our system model, as it characterises: (1) availability of data at source nodes: node j has access to sequence x_{A_i} if $\theta_i^j \neq 0$ (2) rate of production of encoding symbols at each of the source nodes: sequence y_{B_j} was produced at node j, and (3) *bias* introduced towards certain portions of data in formation of encoding packets: during the generation of each encoding packet, packets from sequence x_{A_i} are used with probability θ_i^j at node j. Similar code construction involving bias for the case of a single source node is used in UEP fountain codes [5], which we refer to as weighted LT coding. In addition to graph $\mathcal{G}$, overall encoding process is described by a set of degree distributions $\{\Omega_j(x)\}_{j=1}^s$, where distribution $\Omega_j(x)$ is used for LT encoding at user node j. Thus, with $\mathcal{G}$ and $\{\Omega_j(x)\}_{j=1}^s$, we have fully described an instance of data collection via independent decentralised distributed LT encodings, i.e., a particular code ensemble which we denote by $\mathrm{DDLT}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$. We are interested in the decoding performance of this ensemble as $k \to \infty$.

Example 1 (Uniform LT encodings): Source node $j \in N_s$ chooses uniformly from all the available data packets: values of θ_i^j are proportional to the sizes of A_i, whenever $\theta_i^j \neq 0$,

[1] Throughout the paper, we adopt the following notation. For any natural number m, denote by N_m the set $\{1, 2, \ldots, m\}$. For data sequence $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ and $A \subseteq N_k$, we denote $x_A := (x_i : i \in A)$.

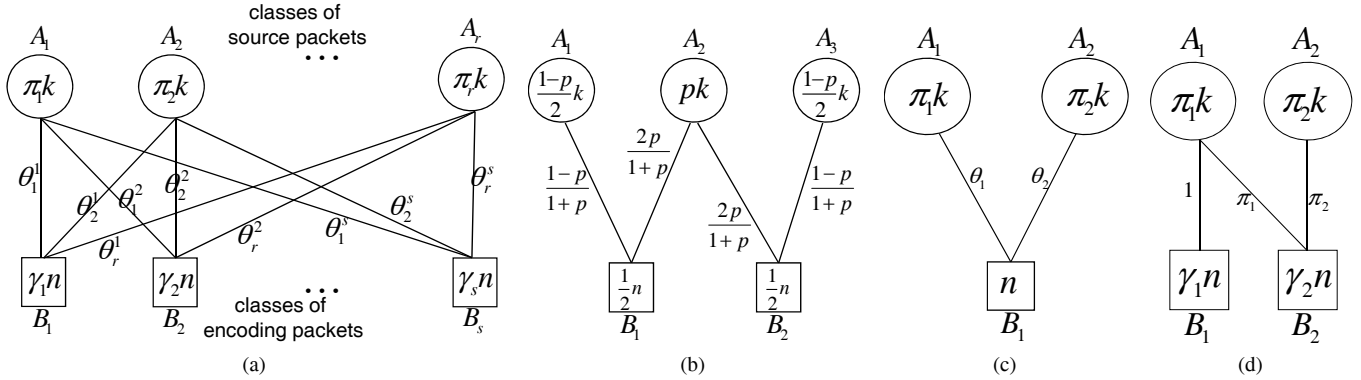

Fig. 1. (a) Generic DDFC scheme, (b) Symmetric DSC problem in Example 2 as an instance of DDFC, (c) Weighted UEP LT codes [5] for two classes of importance as an instance of DDFC, (d) EWF codes [6] for two classes of importance as an instance of DDFC.

i.e., $\theta_i^j = \frac{\pi_i}{\sum_{l\in C_j}\pi_l}$. This means that the user j is performing a simple LT encoding over vector x_{C_j}.

III. Generalised And-Or Lemma

Assume that each user is producing encoding packets as described and once a collector node successfully receives a sufficient amount of these encoding packets, it attempts to recover $\mathbf{x}$. We can capture the asymptotic performance of the belief propagation decoder at the collector node as a function of the code overhead ε by the generalisation of the original And-Or tree analysis [4]. As in the standard And-Or tree analysis and other density evolution arguments, we derive the asymptotic performance of the decoder by looking at the structure, and specifically at the degree distributions, on the decoding factor graph. The proof of the lemma is omitted as it is a straightforward generalisation of the standard And-Or lemma. What distinguishes this generalised setting from the original one is that the nodes in the decoding factor graph are divided into classes, each of the classes of nodes possibly having a different degree distribution. Classes are introduced naturally - class i of the input nodes corresponds to the raw data packets x_{A_i} and class j of the output nodes corresponds to the encoding packets y_{B_j}.

Lemma 1 (Generalised And-Or lemma): For all $i \in N_r$, the packet error rate within sequence x_{A_i} of a belief propagation decoder for an ensemble $\mathrm{DDLT}(\mathcal{G}, \{\Omega_j(x)\}_{j=1}^s, k)$ at a collector node, as $k\to\infty$, is equal to $y_i = \lim_{l\to\infty} y_{i,l}$, where $y_{i,l}$ is given by:

$$\begin{aligned} y_{i,0} &= 1, \qquad (1)\\ y_{i,l+1} &= \exp\Big[-(1+\varepsilon)\sum_{j=1}^{s}\theta_i^j\frac{\gamma_j}{\pi_i}\Omega_j'(1-\sum_{m=1}^{r}\theta_m^j y_{m,l})\Big]. \end{aligned}$$

In some special cases of DDFC, derived asymptotic analysis allows us to formulate a robust linear programming optimisation routine to calculate asymptotically good degree distributions, similarly to, e.g., [3], and the remainder of this paper illustrates two simple examples of such use of the generalised And-Or lemma.

IV. Symmetric Distributed Source Coding (DSC): Two Nodes with Common Data

Example 2: Assume that two source nodes S_1 and S_2 contain more than a half of all the packets in the data sequence $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ (but the number of packets at S_1 and S_2 is the same). A certain portion $p < 1/2$ of the packets is common to S_1 and S_2 but they have no knowledge of which packets are available at the other source node. A collector node would ideally need just slightly more than k encoding packets to recover the entire information sequence.

This Example outlines a simple DDFC problem, and its graphical representation is given in Fig. 1b. Note that there are three different classes of packets: class A_1 of $\frac{1-p}{2}k$ packets available only at node S_1, class A_2 of pk packets available at both nodes, and class A_3 of $\frac{1-p}{2}k$ packets available only at node S_2. Let each source node produce encoding packets with an LT code with degree distribution $\Omega(x)$ over $(\frac{1-p}{2}+p)k$ packets available at that node and let us assume collector node obtains an equal number of encoding packets from each source node. The recursive equation describing the asymptotic packet error rate in class A_1 is thus given by:

$$y_{1,l+1} = \exp\Big[-\frac{1+\varepsilon}{1+p}\Omega'\Big(\frac{(1-p)(1-y_{1,l})+2p(1-y_{2,l})}{1+p}\Big)\Big]. \qquad (2)$$

It is easily checked that $y_{3,l} = y_{1,l}$ and that $y_{2,l} = y_{1,l}^2$, $\forall l \geq 0$. Thus, one can trace the asymptotic behaviour of all three packet error rates with a single parameter, which allows simple transformation of the above recursive equations into the linear program optimisation procedure.

Now, let us for the sake of simplicity assume $p = 1/3$, i.e., a third of all the packets are available at both source nodes. Simple transformations yield the following generic linear program[2]:

$$\begin{aligned} \mathrm{LP}: \qquad & \min \sum_{d}^{d_{\max}} \frac{\omega_d}{d} \qquad (3)\\ & \frac{3}{4}\omega(1-\frac{z_i}{2}-\frac{z_i^2}{2}) \;\geq\; -\ln(z_i),\ i\in N_m, \end{aligned}$$

where $1 = z_1 > z_2 > \cdots > z_m = \delta$ are m equidistant points on $[\delta, 1]$. The solution of this linear program is an edge perspective degree distribution with maximum degree $d_{\max}$ which reaches the packet error rate of δ within classes A_1 and A_3 (and δ^2 within class A_2) at the minimum overhead. The degree distribution we obtained using this linear program with $\delta = 0.01$ and $d_{max} = 100$ successfully takes advantage of the fact that two sources contain correlated information, and it is given by: $\Omega^*(x) = 0.0020x + 0.4305x^2 + 0.2205x^3 + 0.0793x^5 + 0.1097x^6 + 0.0508x^{12} + 0.0409x^{13} + 0.0343x^{30} +$

[2]In linear programs given in the paper, variables (to be optimised) are non-negative, i.e., $\omega_d \geq 0$, $d \in N_{d_{\max}}$; $\omega(x) = \Omega'(x)/\Omega'(1)$ is the edge perspective output degree distribution.

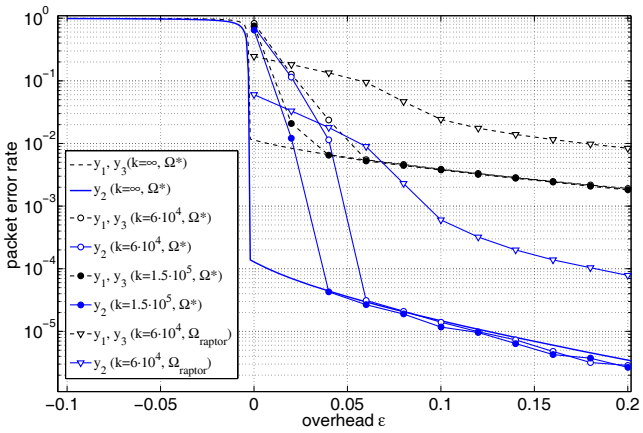

Fig. 2. Symmetric DSC with LT codes.

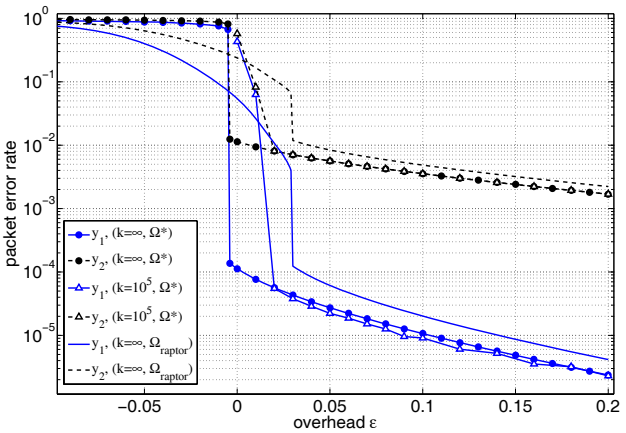

Fig. 3. Weighted LT codes with an optimised degree distribution.

$0.0106x^{32} + 0.0215x^{100}$. Simulation results for large blocklengths, $k = 6 \cdot 10^4$ and $k = 1.5 \cdot 10^5$ are consistent with our asymptotic analysis, as demonstrated in Fig. 2. For comparison, we included results for a typical Soliton-like degree distribution[3] $\Omega_{raptor}(x)$ proposed for Raptor codes [3], which is clearly penalised by higher error floors in this setting.

V. Fountain Codes for Unequal Error Protection

Whereas LT and Raptor codes typically provide equal error protection for all source data, there are cases where certain data parts are considered to be more important and require higher error protection, e.g., transmission of video compressed with a layered coder, and unequal error protection (UEP) fountain coding schemes have recently been proposed [5], [6]. Asymptotic analysis of these codes are special cases of Lemma 1. Although these codes can be fully centralised, it is useful to think of them as of instances of DDFC, as this allows rigorous asymptotic analysis and an insight into the design of code parameters.

Example 3 (UEP codes for two classes of importance): The data sequence $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ consists of two classes of importance: class A_1 consists of 10% of all data packets which require higher error protection, whereas remaining 90% in class A_2 are considered less important.
The encoder can utilise a weighted UEP LT code [5], which uses a nonuniform distribution across data packets, i.e., $\theta_1/\pi_1 > \theta_2/\pi_2$, which means that the data packet in class A_i is selected with probability $\frac{\theta_i}{\pi_i k}$, $i = 1, 2$. Alternatively, the encoder can utilise an EWF code [6], by dividing encoding packets into two classes - class B_1 of packets encoding only class A_1 of source packets (the first window of data packets), and class B_2 encoding the entire data sequence. Two kinds of UEP fountain codes with two classes of importance are illustrated in Fig. 1c and 1d as instances of DDFC. By applying Lemma 1, the recursive equation for asymptotic analysis of weighted UEP LT codes is given by:

$$y_{i,l+1} \quad = \quad \exp(-(1+\varepsilon)\frac{\theta_i}{\pi_i}\Omega'(1-\theta_1 y_{1,l} - \theta_2 y_{2,l})),$$

for $i = 1, 2$, which is consistent with the analysis in [5], and, in an analogous manner, the asymptotic analysis of EWF codes in [6] is another special case of Lemma 1. Furthermore, with appropriate transformations, we can optimise degree distribution for weighted LT codes using the following generic linear program:

$$\text{LP} \quad : \qquad \min \sum_{d}^{d_{\max}} \frac{\omega_d}{d} \tag{4}$$

$$\omega(1 - \theta_1 z_i^{\theta_1/\pi_1} - \theta_2 z_i^{\theta_2/\pi_2}) \quad \geq \quad -\ln(z_i),\ i \in N_m,$$

where $1 = z_1 > z_2 > \cdots > z_m = \delta$ are m equidistant points on $[\delta, 1]$. The solution of this linear program is an edge perspective degree distribution with maximum degree $d_{\max}$ which reaches the packet error rate of δ^{θ_1/π_1} within class A_1 and δ^{θ_2/π_2} within the class A_2 at the minimum overhead. A result of this optimisation for $d_{\max} = 100$, $\delta = 0.0075$ in the weighted LT instance with $\theta_1 = 0.184$, $\pi_1 = 0.1$, we obtain degree distribution $\Omega^*(x) = 0.0080x^1 + 0.4226x^2 + 0.2769x^3 + 0.1515x^6 + 0.0214x^7 + 0.0524x^{13} + 0.0123x^{14} + 0.0338x^{27} + 0.0212x^{74}$. We chose parameter $\theta_1 = 0.184$ in order to compare the asymptotic and large blocklength ($k = 10^5$) packet error rate of the calculated degree distribution with that of $\Omega_{raptor}(x)$, which was used in [5] with the bias parameter equivalent to $\theta_1 = 0.184$ (this bias parameter minimises packet error rate at the overhead $\varepsilon = 0.03$). The results are presented in Fig. 3 and clearly indicate how a designed degree distribution outperforms $\Omega_{raptor}(x)$ both in terms of the packet error rates and the code overhead. Nonetheless, we have noted a fragile behaviour of the designed degree distributions at smaller blocklengths and the robust finite length design with appropriately modified LP optimisation remains a problem for further investigations.

[3] $\Omega_{raptor}(x) = 0.0080x + 0.4936x^2 + 0.1662x^3 + 0.0726x^4 + 0.0826x^5 + 0.0561x^8 + 0.0372x^9 + 0.0556x^{19} + 0.0250x^{65} + 0.0031x^{66}$

References

[1] D. J. C. MacKay, "Fountain codes," *IEE Proc. Commun.*, vol. 152, no. 6, pp. 1062-1068, 2005.
[2] M. Luby, "LT codes," in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, Vancouver, Canada, Nov. 2002.
[3] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551-2567, June 2006.
[4] M. Luby, M. Mitzenmacher, and A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proc. SIAM Symp. on Discrete Algorithms (SODA)*, San Francisco, USA, Jan. 1998.
[5] N. Rahnavard, B. N. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1521-1532, 2007.
[6] D. Sejdinović, D. Vukobratović, A. Doufexi, V. Senk, and R. Piechocki, "Expanding window fountain codes for unequal error protection," *IEEE Trans. Commun.*, vol. 57, no. 9, pp. 2510-2516, 2009.