Summary of Raptor Codes

Tracey Ho

October 29, 2003

1 Introduction

This summary gives an overview of Raptor Codes, the latest class of codes proposed for reliable multicast in the Digital Fountain model. This is a summary of a 2003 preprint by Amin Shokrollahi.

2 Background

2.1 Fountain Codes

A fountain code produces for given set of k input symbols $(x_1, \ldots, x_k)$ a potentially limitless stream of output symbols $z_1, z_2, \ldots$. The input and output symbols can be binary vectors of arbitrary length. Each output symbol is the sum of a randomly and independently chosen subset of the input symbols. Information describing the relations between input and output symbols is obtained at the receivers either in packet headers or by other application-dependent means of synchronization between sender and receivers.

A *reliable decoding algorithm* for a fountain code is one which can recover the original k input symbols from any set of m output symbols with error probability at most inversely polynomial in k. The ratio m/k is called the *overhead*. The *encoding cost* is the expected number of arithmetic operations sufficient for generating each output symbol. The *decoding cost* is the expected number of arithmetic operations sufficient to recover the k input symbols, divided by k. Desirable properties for a fountain code are an overhead close to 1, and constant encoding and decoding cost. A *universal fountain code* is one for which the decoder can recover with high probability the original symbols from any set of output symbols whose size is close to optimal.

2.2 Tornado Codes

Tornado Codes [1] are not strictly fountain codes since the number of output symbols is fixed for a particular code. They are however precursors to the LT Codes and Raptor Codes in that they are derived from sparse irregular random bipartite graphs, whose associated encoding and decoding algorithms compute one exclusive-or operation for each edge. The edge degree distributions of these graphs are carefully chosen so that with high probability, successful decoding is achieved by a simple belief propagation decoder that sets the value of a graph node only if the values of all its neighbors are known.

Tornado codes use a cascade of such graphs. For any given rate R and overhead $1+\varepsilon$, an $(n, k = Rn)$ tornado code with $O(\log(1/\varepsilon))$ encoding and decoding cost can be designed, such that a codeword can be recovered from $(1+\varepsilon)k$ output symbols with probability $1 - O(k^{-3/4})$.

2.3 LT Codes

LT Codes [2] were the first class of universal fountain codes to be invented. Each output symbol is generated by randomly choosing a degree d from some suitable degree distribution, choosing d distinct input symbols uniformly at random, and taking their sum.

If the output symbol degrees are chosen according to the Robust Soliton distribution proposed in [2], k input symbols can be recovered from any $k + O(\sqrt{k}\log^2(k/\delta))$ output symbols with probability $1-\delta$. The average encoding and decoding cost is $O(\log(k/\delta))$.

3 Limitation of LT Codes

The following proposition shows that the decoding graph of a reliable decoder for an LT code has at least $O(k\log(k))$ edges. It follows that if the number m of output symbols needed for decoding is close to k, then each output symbol is the sum of $O(\log(k))$ input symbols on average. The per symbol encoding cost is then $O(\log(k))$.

Proposition 1 *If an LT code with k input symbols as a reliable decoding algorithm, then there is a constant c such that the decoding graph has at least $ck\log(k)$ edges.*

Proof outline: Suppose m output symbols suffice to achieve error probability at most $1/k^u$ for some constant u. Let a be the average degree of an output node in the decoding graph, and $b = am/k$ the average degree of an input node. The probability of error is lower bounded by the probability that a particular input node v is not a neighbor of any output node, which can be expressed as $(1 - a/k)^n$. A series of mathematical manipulations leads to the inequality $b \geq c \log(k)$. ■

4 Raptor Codes

The key idea of Raptor Coding is to relax the condition that all input symbols need to be recovered. If an LT code needs to recover only a constant fraction of its input symbols, then its decoding graph need only have $O(k)$ edges, allowing for linear time encoding. We can still recover all input symbols by concatenating a traditional erasure correcting code with an LT code. n intermediate symbols are obtained by encoding k input symbols with an (n, k) erasure correcting block code capable of recovering all input symbols from a fixed fraction of intermediate symbols. The n intermediate symbols are then encoded with an LT code that can recover from its output symbols the required fraction of intermediate symbols. A Raptor Code is specified by parameters $(k, \mathcal{C}, \Omega(x))$, where $\mathcal{C}$ is the (n, k) erasure correcting block code, called the *pre-code*, and $\Omega(x))$ is the generator polynomial of the degree distribution of the LT code, i.e. $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$, where Ω_i is the probability that the degree of an output node is i.

The performance parameters overhead and decoding cost as defined in section 2.1 apply directly to Raptor Codes. However, the definition of the encoding cost of a Raptor Code differs slightly – it is the sum of the encoding cost of the pre-code divided by k, and the encoding cost of the LT code. Raptor Codes also require storage for intermediate symbols, so space consumption is another important performance parameter.

5 First Examples of Raptor Codes

5.1 LT Codes

An LT code with k input symbols and output distribution $\Omega(x)$ is itself a Raptor Code with parameters $(k, \mathbb{F}_2^k, \Omega(x))$, where $\mathbb{F}_2^k$ is the trivial code of dimension and block length k. The lack of a pre-code necessitates the use of a sophisticated output distribution $\Omega(x)$ with logarithmic encoding and decoding cost, as discussed in Section 3. The overhead and space however are close to 1.

5.2 Pre-Code-Only (PCO) Raptor Codes

At the other end of the spectrum are Pre-Code-Only (PCO) Raptor Codes. These codes have a sophisticated pre-code but a trivial output distribution $\Omega(x) = x$, which sets the value of every output symbol to that of a randomly and uniformly chosen input symbol. This approach in effect builds a fountain code from any block code.

The decoding algorithm collects a predetermined number m of output symbols, which determines the values of some number l of intermediate symbols. The decoding algorithm for the pre-code is then applied to these recovered intermediate values to obtain the values of the input symbols.

The performance of a PCO Raptor Code depends on the performance of its pre-code as follows:

Proposition 2 *Let $\mathcal{C}$ be an (n, k) linear block code such that $k\eta$ arithmetic operations suffice for encoding an arbitrary length-k input vector, and $k\gamma$ arithmetic operations suffice for decoding $\mathcal{C}$ with high probability over a binary erasure channel with erasure probability $1 - R(1 + \varepsilon)$, for some $\varepsilon > 0$, where $R = k/n$. Then the PCO Raptor Code with pre-code $\mathcal{C}$ has space consumption $1/R$, overhead $-\ln(1-R(1+\varepsilon))/R$, encoding cost η, and decoding cost γ with respect to the decoding algorithm for $\mathcal{C}$.*

Proof outline: The space consumption and encoding and decoding costs follow immediately from the properties of the pre-code. The overhead is obtained by noting that if the decoder collects $m = -k \ln(1 - R(1 + \varepsilon))/R$ output symbols, the probability that an intermediate symbol is not covered is $(1 - 1/n)^m$, which is upper bounded by $e^{-m/n} = 1 - R(1 + \varepsilon)$. ■

The overhead is at least $1 + \varepsilon$, since $-\ln(1 - R(1 + \varepsilon))/R \geq 1 + \varepsilon$ for $0 < R < 1/(1 + \varepsilon)$, and approaches this bound only if R approaches zero. Thus, improvement in overhead comes at the expense of the space consumption and the encoding and decoding time of the pre-code. PCO Raptor Codes can nevertheless be useful when the intermediate symbols can be calculated off-line in a preprocessing stage, and space for storage is not an issue.

6 Raptor Codes with Good Asymptotic Performance

Raptor Codes with good asymptotic performance (i.e. constant encoding and decoding costs, and space consumption and overhead arbitrarily close to 1) can be obtained by appropriately choosing the pre-code $\mathcal{C}$ and the output distribution $\Omega(x)$.

6.1 Edge Degree Distributions and And-Or Tree Evaluation

We first outline an analysis technique given in [3] that sheds some intuitive light on following proofs. The main tool is a simple analysis of the probability a tree consisting of alternating layers of AND and OR gates evaluates to 1. This technique yields a simpler and more intuitive way to obtain the criterion in [1] on the edge degree distributions required for decoding all missing message coordinates when a fixed fraction of them are erased independently at random.

Here it is useful to consider the BP decoding in terms of rounds: at each round all released output nodes (i.e. those connected to one unrecovered input node) recover their incident input nodes, and then all these recovered input nodes update their incident output nodes. All edges connected to recovered input nodes are then removed.

Consider an edge (v, w) chosen uniformly at random from the original graph. We can identify the generator polynomial of the output edge distribution, $\omega(x)$, with the probability that the output node w is released at some stage where (v, w) has not been deleted and a proportion x of input nodes have been recovered. Similarly, the generator polynomial of the input edge distribution, $\iota(x)$, is identified with the probability that the input node v is recovered at some stage where (v, w) has not been deleted and a proportion $1 - x$ of output nodes have been released.

Assuming that none of the input nodes are initially known, we obtain the following recursive formula for the probability p_i that a randomly chosen edge has carried a message from its incident output symbol by round i:

$$p_{i+1} = \omega(1 - \iota(1 - p_i)) \tag{1}$$

A similar recursive formula can be obtained for the probability q_i that a randomly chosen edge has not carried a message from its incident input symbol by round i:

$$q_{i+1} = \iota(1 - \omega(1 - q_i)) \tag{2}$$

This provides intuition for the condition

$$\iota(1 - \omega(1 - x)) < x, \qquad x \in [\delta, 1] \tag{3}$$

required for reliable recovery of a fraction δ of input nodes.

6.2 Proof of Asymptotic Performance

The output degree distribution $\Omega_D(x)$ used is a slight modification of the ideal soliton distribution of [2]:

$$\Omega_D(x) = \frac{1}{\mu+1}\left(\mu x + \frac{x^2}{1\cdot 2} + \frac{x^3}{2\cdot 3} + \ldots + \frac{x^D}{(D-1)\cdot D} + \frac{x^{D+1}}{D}\right)$$

where $D = \lceil 4(1+\varepsilon)/\varepsilon \rceil$ and $\mu = (\varepsilon/2) + (\varepsilon/2)^2$. An LT code with parameters $(n, \Omega_D(x))$ is used, along with an (n,k) pre-code $\mathcal{C}_n$ of rate $R = \frac{k}{n} = \frac{1+\varepsilon/2}{1+\varepsilon}$ that can be decoded on a BEC of erasure probability $\delta = \frac{\varepsilon}{4(1+\varepsilon)} = \frac{1-R}{2}$ with $O(n \log(1/\varepsilon))$ arithmetic operations. The paper mentions that a variety of codes can be used as the pre-code, such as tornado codes, right-regular codes, and certain types of repeat-accumulate codes.

Theorem 1 *The Raptor Code with parameters* $(k, \mathcal{C}_n, \Omega_D(x))$ *has space consumption* $1/R$*, overhead* $1+\varepsilon$*, and encoding and decoding costs of* $O(\log(1/\varepsilon))$.

Proof outline: First it is shown that any set of $(1+\varepsilon/2)/n+1$ output symbols of the LT code with parameters $(n, \Omega_D(x))$ are sufficient to reliably recover at least $(1-\delta)n$ intermediate symbols via BP decoding. This is done by showing that the input edge degree distribution $\iota(x)$ and the output edge degree distribution $\omega(x)$ induced by $\Omega_D(x)$ satisfy equation 3, which is sufficient to ensure reliable decoding of at least $(1-\delta)n$ intermediate symbols.

Recovery of at least $(1-\delta)n$ intermediate symbols allows the decoder for $\mathcal{C}_n$ to recover the k input symbols. The overall overhead is $n(1+\varepsilon/2)/k = 1+\varepsilon$.

The encoding and decoding cost of the LT code is proportional to the average degree of the distribution Ω_D, which is $\Omega'_D(1) = 1 + H(D)/(1+\mu) = \ln(1/\varepsilon) + \alpha + O(\varepsilon)$, where $H(D)$ is the harmonic sum up to D, $1 < \alpha < 1 + \gamma + \ln(9)$, and γ is Euler's constant. The encoding and decoding cost of $\mathcal{C}_n$ is also $O(\log(1/\varepsilon)$, which gives the result. ■

7 Design and Analysis of Finite Length Raptor Codes

The asymptotic analysis of the previous section is less useful in a practical setting. The error probability bounds given in the previous section are not tight for finite length codes, and the code design choices, while sufficient for proving asymptotic results, do not perform well in practice.

7.1 Design and Analysis of LT Code Component

The LT code component is obtained by linear programming on a heuristically obtained design problem. Consider running the decoding algorithm on $k(1+\varepsilon)$ output symbols. From equation 1 a recursion can be obtained on the probability u_i that an input symbol is unrecovered at round i:

$$u_{i+1} = 1 - e^{(1+\varepsilon)\Omega'(u_i)}$$

This shows that if an expected x-fraction of input symbols has been recovered by the start of a round, then the expected fraction of input nodes recovered during the round is $1-x-e^{(1+\varepsilon)\Omega'(x)}$. The degree distribution $\Omega(x)$ is chosen to minimize the objective function $\Omega'(1)$, subject to the constraint that the expected number of input nodes recovered during each round is larger by a constant factor c than the square root of the number of unrecovered input symbols, which is $\sqrt{(1-x)k}$.

The error probability of the LT decoder for a given degree distribution can be calculated exactly using a dynamic programming approach which is the subject of an upcoming paper [4].

7.2 Design and Analysis of Pre-code

The LDPC codes proposed as suitable pre-codes are constructed from a node degree distribution $\Lambda(x)$ as follows: for each message node a degree d is chosen independently at random from the distribution, and d random check nodes are chosen to be the neighbors of the message node. A complicated analytical expression for the error probability of the ensemble is given.

7.3 Combined Error Probability

Denoting by p_l^{LT} the probability that the LT decoder fails after recovering l intermediate symbols, and by $p_j^{\mathcal{C}}$ the probability that the pre-code $\mathcal{C}$ cannot decode j randomly chosen erasures, the probability that the k input symbols cannot be recovered from the $k(1+\varepsilon)$ output symbols is

$$\sum_{l=0}^{n} p_l^{LT} p_{n-l}^{\mathcal{C}}$$

8 Systematic Raptor Codes

An algorithm is given for constructing from a given Raptor Code of parameters $(k, \mathcal{C}, \Omega(x))$ a systematic version of the code. First, a set of k positions is computed, which will be the positions of the systematic output symbols. This is done by considering the generator matrix G of the pre-code and the generator matrix S of the first $k(1+\varepsilon)$ symbols of the LT code. Using Gaussian elimination, k rows of SG which form a full rank square submatrix R are identified. These rows correspond to the positions of the systematic output symbols.

Encoding of the first $k(1+\varepsilon)$ symbols is done by first left-multiplying the vector of input symbols with GR^{-1} to obtain the vector u, and then left-multiplying the result with S. Subsequent output symbols are obtained by application of the LT code to the vector u.

Decoding is done by applying the decoding algorithm of the original Raptor Code to obtain a vector of symbols y. The original input symbols are then given by Ry.

9 Conclusion

This summary has explained the motivation behind the idea of Raptor Codes, and has covered tools and approaches for the design and analysis of a range of Raptor Codes that includes simple Pre-code-only Raptor Codes, asymptotically good Raptor Codes, and practical finite length and systematic Raptor Codes.

References

[1] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, "Efficient Erasure Correcting Codes", IEEE Transactions on Information Theory, Vol. 47, Issue 2, pp. 569-584, February 2001.

[2] M. Luby, "LT Codes", 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002.

[3] M. Luby, M. Mitzenmacher, and M. Shokrollahi, "Analysis of Random Processes via And-Or Tree Evaluation", Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, 1998, pp 364-373.

[4] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT-codes", To appear, 2002.