Practical Loss-Resilient Codes

Michael Luby[1], Michael Mitzenmacher†, Amin Shokrollahi‡, Daniel Spielman§, Volker Stemann¶
International Computer Science Institute
Berkeley, California
(luby@icsi.berkeley.edu)
† Digital Equipment Corporation Systems Research Center,
Palo Alto, California
(michaelm@pa.dec.com)
‡ International Computer Science Institute
Berkeley, California
(amin@icsi.berkeley.edu)
§ Department of Mathematics, M.I.T.,
Cambridge, Massachusetts
(spielman@math.mit.edu)
¶ International Computer Science Institute
Berkeley, California

Abstract — **We present randomized constructions of linear-time encodable and decodable codes that can transmit over lossy channels at rates extremely close to capacity. The encoding and decoding algorithms for these codes have fast and simple software implementations. Implementations of our algorithms are faster by orders of magnitude than the software implementations of previous algorithms. We expect these codes will be extremely useful for applications where lossy channels are common and fast decoding is a requirement, e.g., satellite transmission and multicast transmission over the Internet.**

I. Introduction

In many communication situations, data is lost in transit. A standard response to this problem is to request retransmission of data that is not received. When some of this retransmission is lost, another request is made, and so on. Such communication protocols can lead to delays due to the need for several rounds of communication between sender and receiver. An alternative solution is to use coding. The basic idea, well-known to the coding community, is to compute and transmit redundant information along with the original message. In transmission there may be losses, but if enough of the transmission is received then the entire original message can be decoded. A code with these properties is often called a *forward error-correcting* (FEC) code in the networking literature, and an *erasure* code in the coding literature. We call this type of code *loss-resilient*, to emphasize the type of environments where we believe they are most useful. Ideally, the portion of the transmission sufficient to recover the original message is the length of the original message itself. A code with this property is called *MDS*.

The problem is to design fast enough encoding and decoding algorithms to make this solution feasible. In this paper, we present loss-resilient codes that can be encoded and decoded in linear time while providing near optimal loss protection. Moreover, these linear time algorithms can be implemented to run very quickly in software.

Our results hold whether each symbol is a single bit or a *packet* of many bits. As usual, we assume that the receiver knows the position of each received symbol within the stream of all encoding symbols. We adopt as our model of losses the *erasure channel*, introduced by Elias [4], in which each encoding symbol is lost with a fixed constant probability μ in transit independent of all the other symbols. This assumption is not appropriate for the Internet, where losses can be highly correlated and bursty. However, losses on the Internet in general are not sensitive to the actual contents of each packet, and thus if we place the encoding into the packets in a random order then the independent loss assumption is valid.

Elias [4] showed that the capacity of the erasure channel is $1-\mu$ and that a random linear code can be used to transmit over the erasure channel at any rate $R < 1-\mu$. This means that a random linear code can be used to convert a message of length Rn into a transmission of length n from which the message can be recovered from most portions of length greater than Rn. Moreover, every linear code has quadratic time encoding algorithms and cubic time decoding algorithms. One cannot hope for better information recovery, but faster encoding and decoding times are desirable, especially for real-time applications.

Reed-Solomon codes easily implemented to transmit at the capacity of the erasure channel with order $n \log n$ encoding time and quadratic decoding time. In theory, it is possible to decode Reed-Solomon codes in time $O(n \log^2 n \log\log n)$ (see, [3, Chapter 11.7] and [7, p. 369]). However, for small values of n, quadratic time algorithms are faster than the fast algorithms for the Reed-Solomon based codes, and for larger values of n the $O(\log^2 n \log\log n)$ multiplicative overhead in the running time of the fast algorithms (with a moderate sized constant hidden by the big-Oh notation) is large, i.e., in the hundreds or larger.

We obtain very fast linear-time algorithms by transmitting just below channel capacity. We produce rate $R = 1-\mu(1+\epsilon)$ codes along with decoding algorithms that recover from the random loss of a μ fraction of the transmitted symbols in time proportional to $n \ln(1/\epsilon)$, with high probability. They can also be encoded in time proportional to $n \ln(1/\epsilon)$. The parameter ϵ is an input to the construction of the code, i.e., we can do this for any $\epsilon > 0$. The fastest previously known encoding and decoding algorithms [1] with such a performance guarantee have run times proportional to $n \ln(1/\epsilon)/\epsilon$. (See also [2] for related work.)

Our overall constructions are related to those introduced introduced in [2] for loss-resilient codes and to those introduced in [8] for error-correcting codes. Our encoding and decoding algorithms are almost symmetrical. Both are extremely simple, computing exactly one XOR operation for each edge in a randomly chosen bipartite graph. As in many similar applications, the graph is chosen to be sparse, which immediately implies that the encoding and decoding algorithms are fast. Unlike many similar applications, the graph is not regular; instead it is quite irregular with a carefully chosen degree sequence. The decoding algorithm can be described as a process on the graph. Our main tool is a model that characterizes almost exactly the performance of the decoding algorithm as a function of the degree sequence of the graph. The main argument given in the original paper [6] has been replaced by a much simpler and more intuitive analysis in [5]. The complete success of the decoding algorithm can then be demonstrated by combinatorial arguments.

Our analytical tools allow us to almost exactly characterize the performance of the decoding algorithm for any given degree sequence. Using these tools, we can show that regular graphs do not yield codes that are close to optimal, and hence irregular graphs are a necessary component of our design.

Not only do our tools allow us to analyze a given degree sequence, but they also help us to *design* good irregular degree sequences. We describe, given a parameter $\epsilon > 0$, a degree sequence for which the decoding is successful with high probability for a loss fraction δ that is within ϵ of $1 - R$. Although these graphs are irregular, with some nodes of degree $1/\epsilon$, the average degree of each nodes is only $\ln(1/\epsilon)$. This is the main result of our work, i.e., a code with encoding and decoding times proportional to $\ln(1/\epsilon)$ that can recover from a loss fraction that is within ϵ of optimal.

II. Experimental Runs

All of the experiments were benchmarked on a SUN 167 MHz Ultrasparc 1 with a 64M Bytes RAM. All runs used packets of length 1K Bytes each. The comparison FEC codes listed in the tables as *Vandermonde* and *Cauchy* are standard implementations of Reed-Solomon FEC codes. These codes are based on Vandermonde matrices and Cauchy matrices, respectively. They use straightforward quadratic time encoding and decoding algorithms. The *Tornado* codes are designed using some of the principles described in [6] and [5]. All implementations are written in C. The *Vandermonde* codes were implemented by Luigi Rizzo and the experiments for these codes were run by Christian Riechmann. The implementations and experiments for both the *Cauchy* codes and the *Tornado* codes were done by Michael Luby. None of the implementations are particularly optimized, and thus their running times could be improved by constant factors.

For the encoding experiments, a message of length SIZE was encoded by adding redundancy of length SIZE, and thus the total length of the encoding is 2*SIZE.

Encoding Benchmarks			
	FEC codes		
SIZE	Vandermonde	Cauchy	Tornado
250K Bytes	9.0 seconds	4.6 seconds	0.06 seconds
500K Bytes	39 seconds	19 seconds	0.12 seconds
1M Bytes	150 seconds	93 seconds	0.26 seconds
2M Bytes	623 seconds	442 seconds	0.53 seconds
4M Bytes	not available	1717 seconds	1.06 seconds
8M Bytes	not available	6994 seconds	2.13 seconds
16M Bytes	not available	30802 seconds	4.33 seconds

For the decoding experiments, for both the *Cauchy* and the *Vandermonde* codes, a portion SIZE/2 of the original message and SIZE/2 of the redundancy was used to recover the original message of length SIZE. For the *Tornado* codes, slightly more of each portion was used to recover the original message, i.e., on average 1.05*SIZE/2 of the original message and 1.05*SIZE/2 of the redundancy was used to recover the original message of length SIZE.

Decoding Benchmarks			
	FEC codes		
SIZE	Vandermonde	Cauchy	Tornado
250K Bytes	11.0 seconds	2.06 seconds	0.06 seconds
500K Bytes	32 seconds	8.4 seconds	0.09 seconds
1M Bytes	161 seconds	40.5 seconds	0.14 seconds
2M Bytes	1147 seconds	199 seconds	0.19 seconds
4M Bytes	not available	800 seconds	0.40 seconds
8M Bytes	not available	3166 seconds	0.87 seconds
16M Bytes	not available	13629 seconds	1.75 seconds

References

[1] N. Alon, J. Edmonds, M. Luby, "Linear Time Erasure Codes With Nearly Optimal Recovery", *Proc. of the* 36^{th} *Annual Symp. on Foundations of Computer Science*, 1995, pp. 512-519.

[2] N. Alon, M. Luby, "A Linear Time Erasure-Resilient Code With Nearly Optimal Recovery", *IEEE Transactions on Information Theory* (special issue devoted to coding theory), Vol. 42, No. 6, November 1996, pp. 1732–1736.

[3] R. E. Blahut, **Theory and Practice of Error Control Codes**, Addison Wesley, Reading, MA, 1983.

[4] P. Elias, "Coding for Two Noisy Channels" *Information Theory*, Third London Symposium, September 1955, Buttersworth's Scientific Publications, pp. 61-76.

[5] Michael Luby, Michael Mitzenmacher, Amin Shokrollahi, "Analysis of Random Processes via And-Or Tree Evaluation" *ICSI technical report TR-97-042*, November 1997, accepted to 9^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1998.

[6] Michael Luby, Michael Mitzenmacher, Amin Shokrollahi, Daniel Spielman, Volker Stemann, "Practical Loss-Resilient Codes", Proceedings of the 29^{th} *Annual ACM Symposium on Theory of Computing*, 1997, pp. 150 – 159.

[7] F. J. Macwilliams and N. J. A. Sloane, **The Theory of Error-Correcting Codes**, North Holland, Amsterdam, 1977.

[8] D. Spielman, "Linear-Time Encodable and Decodable Error-Correcting Codes" *IEEE Transactions on Information Theory* (special issue devoted to coding theory), Vol. 42, No. 6, November 1996, pp. 1723–1731.