Distributed LT Codes

Srinath Puducheri, Jörg Kliewer, and Thomas E. Fuja
Department of Electrical Engineering,
University of Notre Dame, Notre Dame, IN 46556, USA
Email: {spuduche, jkliewer, tfuja}@nd.edu

***Abstract*— This paper proposes a novel distributed encoding procedure to realize codes that resemble LT codes (rateless codes for erasure correction) in both structure and performance. For the case of two sources communicating with a single sink via a common relay, this technique separately encodes $k/2$ symbols of information onto slightly more than k code symbols at each source. These two codewords are then selectively XOR-ed at the relay, such that the result can be decoded by the sink to recover all k information symbols. It is shown that, for the case of four sources communicating to a single sink, the use of a similar distributed LT code leads to a 50% reduction in overhead at the sink, compared to the use of four individual LT codes.**

I. Introduction

LT codes were developed by Michael Luby [1] as rateless erasure correcting codes. These codes have the property that, for a fixed number of information symbols, any number of code symbols can be generated, and the data can be recovered with high probability from any subset of the code symbols that is only slightly larger than the data itself. The information and code symbols are bit-strings of length l, and all encoding and decoding processes involve only bitwise XOR operations. LT codes are very efficient as the data length grows, i.e., the fractional overhead of code symbols required (on average) to decode the data decreases as the block size of the data increases.

LT codes are a specific realization of the more general class of rateless erasure-correcting codes called Digital Fountain Codes [2], [3]. Such codes have potential applications in data transmission over communication networks (for both point-to-point and multicast scenarios), mass reliable data storage, etc. which can be modelled as erasure channels [4, pp. 589-594].

This paper considers the *decomposition* of an LT code, or, equivalently, the construction of an LT code from two or more codes by merely XOR-ing the symbols from those codes. This gives rise to the notion of *Distributed LT (DLT) codes* – "subcodes" from which LT code can be constructed. We propose a technique for designing such subcodes, specifically for the case of constructing an LT code from *two* subcodes – referred to as *DLT-2* codes.

This work is motivated in part by *network coding* [5], [6]. In designing linear network codes, it is often the case that the intended sinks in the network receive different linear combinations of the information symbols transmitted by the source [7], [8]. For the case when the network codes are over the binary field, and all network coding operations are just XORs (as in the examples given in [5]), if the source's symbols are chosen such that the final symbols received by the sinks resemble an LT code, then there is the added benefit of block erasure correction to combat lossy links in the network.

This paper, however, focuses on a different application of distributed LT codes – an application that exploits the fact that LT codes over longer blocks of data require less overhead. Consider a scenario in which two or more independent sources in a network are communicating with a sink through a common relay with limited data processing capability. All sources have the same amount of data to transmit to the sink. In such a situation, the proposed technique can be used to transmit the entire amount of data to the sink by effectively using one big LT code, rather than several small LT codes being used by the different sources, thereby taking advantage of the benefits of larger data blocklengths. This technique does not require communication between the sources and involves very little processing at the relay.

II. Description of LT Codes

This section reviews the results of [1].

A. Encoding and Decoding Procedures

Assume that the data consists of k information symbols. Then each LT code symbol is generated as follows:

- An integer d (called the *degree* of the code symbol) between 1 and k is chosen at random according to a probability distribution called the *degree distribution*. The degree distribution derived by Luby [1] for the construction of LT codes, called the *Robust Soliton Distribution*, will be described shortly.
- From the set of k information symbols, a subset of d symbols is chosen uniformly at random - this constitutes the set of *neighbors* of the code symbol. The neighbors are bitwise XOR-ed to produce the code symbol.

The decoder is assumed to know the degree of every code symbol and the set of neighbors associated with it. Given a block n ($> k$) of code symbols, the decoder recursively solves for the information symbols from the set of n linear equations connecting the information and code symbols, starting with degree one code symbols, as described in [1]. This decoding procedure is essentially equivalent to the graph-based decoding

This work was supported in part by the U.S. National Science Foundation under grant CCF 02-05310 and by the German Research Foundation (DFG) under grant KL 1080/3-1.

of low density parity check (LDPC) codes using the belief-propagation (BP) algorithm [9].

B. Degree Distribution

The degree distribution used for constructing LT codes - the Robust Soliton Distribution - is constructed such that the decoder can recover the data from slightly more than k code symbols with probability at least $1-\delta$ [1].

Definition 1 (Robust Soliton Distribution [1])**.** *For constants $c > 0$ and $\delta \in [0,1]$, the Robust Soliton Distribution (RSD) $\mu(\cdot)$ is given by*

$$\mu(i) = \frac{\rho(i)+\tau(i)}{\beta}, \quad \textit{for} \quad 1 \le i \le k, \tag{1}$$

$$\text{where} \quad \beta = \sum_{i=1}^{k} \Big(\rho(i)+\tau(i)\Big). \tag{2}$$

Here, $\rho(i)$ (a probability distribution over $1 \le i \le k$) and $\tau(i)$ are given by

$$\rho(i) = \begin{cases} 1/k, & \textit{for} \quad i = 1, \\ 1/(i(i-1)), & \textit{for} \quad 2 \le i \le k, \end{cases} \tag{3}$$

$$\tau(i) = \begin{cases} S/ik, & \textit{for} \quad 1 \le i \le \frac{k}{S} - 1, \\ S\ln(S/\delta)/k, & \textit{for} \quad i = \frac{k}{S}, \\ 0, & \textit{otherwise.} \end{cases} \tag{4}$$

The parameter S represents the average number of degree one code symbols and is defined as

$$S = c \cdot \sqrt{k} \cdot \ln\Big(\frac{k}{\delta}\Big). \tag{5}$$

Luby [1] showed that, for a suitably chosen c (independent of k and δ), the decoder can recover the data from $n = k\beta = k + c \cdot \sqrt{k} \cdot \ln^2(k/\delta)$ LT code symbols, with probability at least $1-\delta$. However, as pointed out in [4, p. 592], in practice c can be treated as a free parameter. If c is chosen such that k/S is not an integer, then k/S should be replaced in (4) by $\lfloor k/S \rfloor$ or $\lceil k/S \rceil$. The value of $\tau(k/S)$ causes a spike in $\mu(i)$ at $i = k/S$ as pointed out in [1].

The contribution of the tail of the RSD is quite small. In fact, if we consider only the degrees between $k/S+1$ and k, the total contribution from these is

$$\sum_{i=\frac{k}{S}+1}^{k} \mu(i) = \frac{1}{\beta} \cdot \frac{S-1}{k}. \tag{6}$$

To put this in perspective, it is observed that $\beta > 1$ (because $\sum_{i=1}^{k} \rho(i) = 1$), and it follows from (5) that $(S-1)/k \to 0$ as $k \to \infty$. Also the number of degrees between $k/S+1$ and k is $k - k/S$, and $\frac{k-k/S}{k} = 1 - \frac{1}{S} \to 1$ as $k \to \infty$, which shows that this set of degrees constitutes a significant fraction of the range of allowed degrees.

The above observations will be useful in the ensuing sections.

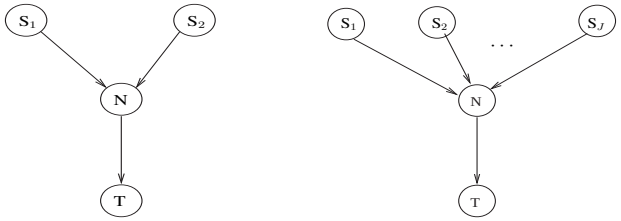

Fig. 1. (a) A two-source single-sink network, (b) A J-source single-sink network

III. Decomposition of an LT Code

This section derives a degree distribution such that, if two code symbols are chosen according to this distribution (in the same manner as an LT code), they can be combined to give a symbol with a degree that follows the RSD.

For motivation, consider the network in Fig. 1(a). Nodes S_1 and S_2 are associated with two data sets[1] - $\mathscr{D}_1$ and $\mathscr{D}_2$, respectively. Each node encodes its data set onto an LT-like codeword according to a degree distribution p. These code symbols are transmitted sequentially to node N. Node N then transmits to node T one code symbol for every pair of symbols it received from S_1 and S_2; how node N arrives at the symbol it transmits based on the pair received from S_1 and S_2 is described below. Our goal is to choose the degree distribution p such that the code symbols received by T follow the RSD in degree.

Assume that each data set contains $k/2$ information symbols. Also, assume that node N only performs one of two operations - either it transmits the sum (bitwise XOR) of the symbols it receives from S_1 and S_2, or it transmits just one of them. Under this assumption, the degree of the resulting symbol is either the sum of the degrees, or the individual degree of one of the received symbols. This implies that the maximum allowed degree of the symbols received by T is k.

In determining $p(\cdot)$, we employ the "deconvolution" of the RSD $\mu(\cdot)$. If node N XORs the incoming symbols, the resulting symbol's degree is the sum of the degrees of the incoming symbols. Hence, if the degrees are independently chosen according to $p(\cdot)$, their sum will follow the distribution $p * p$, where '$*$' denotes the discrete-time convolution operation: $(p*p)(i) = \sum_{j=-\infty}^{+\infty} p(j) \cdot p(i-j)$.

However, a direct attempt at "deconvolving" $\mu(\cdot)$ does not yield satisfactory results. Because $\mu(0) = 0$, it is necessary that $p(0) = 0$ – i.e., we cannot permit a degree of zero, which implies that the smallest degree in $p*p$ is two. However, this does not match the RSD as $\mu(1) > 0$. Moreover, even if we neglect $\mu(1)$, and attempt to deconvolve the remaining part of $\mu(\cdot)$ by solving for $p(\cdot)$ from

$$\begin{aligned} p^2(1) &= \mu(2), \\ 2 \cdot p(1) \cdot p(2) &= \mu(3), \\ 2 \cdot p(3) \cdot p(1) + p^2(2) &= \mu(4), \end{aligned} \tag{7}$$

[1]By "data set", we mean a sequence of information symbols that each source wishes to convey to the sink.

$$2 \cdot p(4) \cdot p(1) + 2 \cdot p(3) \cdot p(2) \quad = \quad \mu(5),$$
$$\vdots$$

then, owing to the fact that $\mu(i)$ has a spike at $i = k/S$, we obtain negative values of $p(\cdot)$ for some arguments.

As an alternative to the above approach, we first split the RSD $\mu(\cdot)$ into two distributions - $\mu'(\cdot)$ and $\mu''(\cdot)$, such that deconvolving $\mu'(\cdot)$ yields a valid probability distribution function. Assume that $\rho(\cdot)$ and $\tau(\cdot)$ are given by (3) and (4), respectively. Then define

$$\mu'(i) = \begin{cases} 0, & \text{for} \quad i = 1, \\ \frac{\rho(i)+\tau(i)}{\beta'}, & \text{for} \quad 2 \leq i \leq \frac{k}{S} - 1, \\ \frac{\rho(i)}{\beta'}, & \text{for} \quad \frac{k}{S} \leq i \leq k, \end{cases} \tag{8}$$

with the normalization factor β' given by

$$\beta' = \sum_{i=2}^{k} \rho(i) + \sum_{i=2}^{k/S-1} \tau(i), \tag{9}$$

and

$$\mu''(i) = \begin{cases} \frac{\rho(1)+\tau(1)}{\beta''}, & \text{for} \quad i = 1, \\ \frac{\tau(k/S)}{\beta''}, & \text{for} \quad i = \frac{k}{S}, \\ 0, & \text{otherwise}, \end{cases} \tag{10}$$

with the normalization factor β'' given by

$$\beta'' = \rho(1) + \tau(1) + \tau\Big(\frac{k}{S}\Big). \tag{11}$$

Thus, noting that $\beta' + \beta'' = \beta$, from (1) the RSD can be rewritten as

$$\begin{aligned} \mu(i) \quad &= \quad \frac{\beta'}{\beta} \cdot \mu'(i) + \frac{\beta''}{\beta} \cdot \mu''(i) \\ &= \quad \frac{\beta'}{\beta} \cdot \mu'(i) + \left(1 - \frac{\beta'}{\beta}\right) \cdot \mu''(i), \\ &\quad \text{for} \quad 1 \leq i \leq k. \end{aligned} \tag{12}$$

Hence, the RSD $\mu(\cdot)$ can be viewed as a mixture of the distributions $\mu'(\cdot)$ and $\mu''(\cdot)$ with mixing parameter β'/β.

We now deconvolve $\mu'(\cdot)$ to yield a valid probability distribution $f(\cdot)$. Specifically, we first solve for f such that $(f * f)(i) = \mu'(i), \quad 2 \leq i \leq k/2 + 1$. Using the approach in (7), with $p(\cdot)$ replaced by $f(\cdot)$ and $\mu(\cdot)$ by $\mu'(\cdot)$, we obtain the following set of recursive equations

$$f(i) = \begin{cases} \sqrt{\mu'(2)}, & \text{for} \quad i = 1, \\ \frac{\mu'(i+1) - \sum_{j=2}^{i-1} f(j) f(i+1-j)}{2f(1)}, & \text{for} \quad 2 \leq i \leq \frac{k}{2}, \\ 0, & \text{for} \quad \frac{k}{2} < i \leq k. \end{cases} \tag{13}$$

This deconvolution yields non-negative values of $f(\cdot)$ for all realizations of the RSD attempted so far – i.e., for all values of k, c, and δ. The same holds true if we solve for $(f*f)(i) = \mu'(i)$ for all i up to k, using (13).

Notice from (13) that, in order to compute $f(i)$, we only make use of $\mu'(1), \cdots, \mu'(i+1)$. Since $|\mathscr{D}_1| = |\mathscr{D}_2| = k/2$, we restrict ourselves to degrees up to $k/2$ in computing $f(\cdot)$. Hence, $\mu'(i)$ will *not* be accurately reproduced by $f * f$ for the range of degrees $k/2 + 2 \leq i \leq k$. However, this is not much of a problem because the tail of the RSD is very small, as observed in Section II. More precisely, given that we can solve for f such that $(f * f)(i) = \mu'(i), \quad 2 \leq i \leq k$ with the constraint $f(\cdot) > 0$, it follows that if we set the latter half of $f(\cdot)$ to zero (as we have done in the preceding equations), then $(f * f)(i) < \mu'(i)$ for $k/2 + 2 \leq i \leq k$. Since we have shown that the tail of $\mu(\cdot)$ is insignificant, the same is true for $\mu'(\cdot)$, and hence, also for $(f * f)(\cdot)$.

In formulating (13) we have only kept the requirement $f * f = \mu'$ in mind and have not explicitly ensured that $\sum_{i=1}^{k/2} f(i) = 1$, which is necessary for f to be a valid probability distribution. Hence, after performing the computations in (13), $f(\cdot)$ must be normalized. However, this does not change $f(\cdot)$ much, and we still have $(f * f)(i) \approx \mu'(i)$ for $1 \leq i \leq k/2 + 1$, after normalization.

Definition 2 (Deconvolved Soliton Distribution)**.** *The Deconvolved Soliton Distribution (DSD) $p(\cdot)$ is given by*

$$p(i) = \lambda \cdot f(i) + (1 - \lambda) \cdot \mu''(i), \quad \textit{for} \quad 1 \leq i \leq k/2, \tag{14}$$

with the parameter λ given by

$$\lambda = \sqrt{\frac{\beta'}{\beta}}. \tag{15}$$

From (14), we can view $p(\cdot)$ as the mixture of $f(\cdot)$ and $\mu''(\cdot)$ with mixing parameter λ. The motivation for the above choice of λ will be provided shortly.

We now propose an encoding procedure which ensures that the code symbols received by T satisfy the RSD.

1) At each instant, nodes S_1 and S_2 generate code symbols based on $\mathscr{D}_1$ and $\mathscr{D}_2$ using the DSD $p(\cdot)$ as the degree distribution. Denote these code symbols as X_1 and X_2, respectively.
2) X_1 and X_2 are transmitted to N. (Node N knows the degrees and neighbors of X_1 and X_2.)
3) Node N generates a symbol Y as a (randomized) function of X_1 and X_2 as follows:
   - If $\deg[X_1] \notin \{1, k/S\}$ and $\deg[X_2] \notin \{1, k/S\}$, then $Y = X_1 \oplus X_2$.
   - Otherwise, two independent random variables U_1 and U_2 are generated uniformly on the interval $(0, 1)$. Based on the values of U_1 and U_2, one of the following operations is performed.
     - If the event $\alpha_i = \big\{ \deg[X_i] = 1 \big\} \wedge \big\{ U_i < 1 - \frac{\lambda f(1)}{p(1)} \big\}$ occurs for either $i = 1$ *or* $i = 2$, then $Y = X_i$. It can be seen that $Pr(\alpha_i) = p(1) \cdot \big(1 - \frac{\lambda f(1)}{p(1)}\big) = (1 - \lambda) \cdot \mu''(1)$. This is the same as the probability that the degree one symbol was derived from the contribution of μ'' to p in (14).
     - If the event $\beta_i = \big\{ \deg[X_i] = k/S \big\} \wedge \big\{ U_i < 1 - \frac{\lambda f(k/S)}{p(k/S)} \big\}$ occurs for either $i = 1$ *or* $i = 2$, then $Y = X_i$. As above, it can be shown that

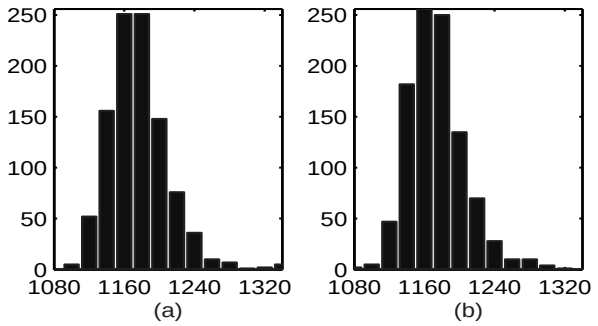

Fig. 2. Histograms of the number of code symbols needed to recover the data, over 1000 simulations – (a) LT code with $k = 1000$, (b) Proposed encoding scheme with $|\mathscr{D}_1| = |\mathscr{D}_2| = k/2 = 500$. The parameters of the RSD used for both cases are $c = 0.1$, $\delta = 0.5$.

$Pr(\beta_i)$ is the same as the probability that the symbol of degree k/S was actually derived from the contribution of μ'' to p in (14).

- If both $Y = X_1$ and $Y = X_2$ are dictated by the above conditions, then Y is picked to be either X_1 or X_2 with probability one-half.
- If none of the above conditions is satisfied, then $Y = X_1 \oplus X_2$.

4) The symbol Y is transmitted to T.

The code generated by node S_i using the DSD $p(\cdot)$ as the degree distribution is said to be a *DLT-2 code*.

The above scheme ensures that the degree of Y approximately obeys the RSD. Note that both X_1 and X_2 are generated according to the DSD $p(\cdot)$, and hence each follows $f(\cdot)$ with probability λ (see (14)). Whenever X_1 and X_2 have degrees that are neither 1 nor k/S, they are directly XOR-ed, and in such cases it is clear that they were derived from the contribution of $f(\cdot)$, as $\mu''(\cdot)$ is zero at all degrees other than 1 and k/S. On the other hand, when either of their degrees is 1 or k/S, the event that they are XOR-ed has exactly the same probability as the event that they were both derived from the $f(\cdot)$ component of the DSD. Thus, in essence, X_1 and X_2 are XOR-ed at node N whenever both of them are generated according to $f(\cdot)$, which occurs with probability λ^2. Whenever Y is given by $X_1 \oplus X_2$, it follows the degree distribution $f * f \approx \mu'(\cdot)$; otherwise it follows $\mu''(\cdot)$. In order for Y to follow the RSD, according to (12), Y must follow $\mu'(\cdot)$ with probability β'/β and $\mu''(\cdot)$ with probability $1 - \beta'/\beta$; this is ensured by the choice of λ in (15).

The code obtained by combining the two DLT-2 codes generated by S_1 and S_2 is not, strictly speaking, an LT code. This is because, in the construction of LT codes, the information symbols to be XOR-ed are picked uniformly at random from the entire set of information symbols. However, in the preceding development, our construction rules out the possibility that the final code symbols are derived from data only in $\mathscr{D}_1$ or only in $\mathscr{D}_2$, for all degrees except 1 and k/S. Nevertheless, in practice, it is observed that this construction performs as well as an LT code, as shown in Fig. 2.

Thus, we have shown how to construct two subcodes, called DLT-2 codes, such that a code similar to an LT code is obtained by little more than bitwise XOR-ing codewords from the two subcodes.

IV. An application of LT Code Decomposition

Consider the network in Fig. 1(b), in which there are $J \geq 2$ source nodes – $\{S_i : 1 \leq i \leq J\}$, each associated with its own data set $\mathscr{D}_i$. Assume the data sets each contain m information symbols, and let k denote the total number of data symbols at all sources, i.e., $k = Jm$.

The J sources want to communicate their data to the destination (sink) T via the relay N. The relay node N is assumed to have limited resources for processing and storage. Also, we do not allow for communication between the source nodes. Further, the link between N and T is lossy (in that certain symbols may be lost/corrupted), and hence we want erasure correction capability built into the sequence of symbols transmitted to T.

One way to provide this robustness is to let every source employ an LT code for its data. These LT codes are multiplexed in time at N and sent to T. This approach works well, except when the data blocks at the sources are relatively small (500-1000 information symbols); for such cases, the fractional overhead inherently required by the LT codes is fairly large (about 15-20%). It would be beneficial if we could form an LT code over the entire set of $k = Jm$ information symbols at N and send this to T. However, we have the constraint that N has limited processing and storage capabilities. So, we propose the use of a mechanism whereby code symbols are chosen according to an appropriate degree distribution at the sources such that, when code symbols from different sources are XOR-ed together at N, it yields symbols that obey the RSD over the range of degrees 1 through $k = Jm$. This requires a J-fold deconvolution of the RSD. Though an exact deconvolution may not be possible, the decomposition technique discussed in Section III clearly works for $J = 2$, with the processing at node N consisting of *selective* XORs. This motivates us to extend this technique to $J > 2$.

Specifically, consider the case $J = 4$. The DLT-2 code obtained in the last section can be decomposed further into two more subcodes called DLT-4 codes, such that when code symbols from four different DLT-4 codes are selectively XOR-ed, the resulting symbols follow the RSD in degree. This is done by performing a partial deconvolution of the DSD $p(\cdot)$ in a manner very similar to how the DSD was derived from the RSD. Specifically, we write the DSD $p(\cdot)$ as a mixture of two distributions $p'(\cdot)$ and $p''(\cdot)$ (akin to (12)), where $p''(\cdot)$ contributes to the degrees 1 and k/S. The distribution $p'(\cdot)$ is then deconvolved to get $g(\cdot)$ and the final degree distribution $q(\cdot)$ is obtained by taking the appropriate mixture of $g(\cdot)$ and $p''(\cdot)$. This degree distribution $q(\cdot)$ is used by the source nodes to generate code symbols from their data, and these code symbols constitute the DLT-4 codeword. The node N performs a selective XOR of the four DLT-4 code symbols it receives from the four sources, in a manner similar to the encoding

scheme discussed in the last section, and forwards the resulting code symbol to T. Node T needs to collect slightly greater than $k = Jm$ code symbols from N to recover all the data.

The deconvolution of $p'(\cdot)$ to yield $g(\cdot)$ is performed along the lines of (13), with $f(\cdot)$ replaced by $g(\cdot)$ and $\mu'(\cdot)$ replaced by $p'(\cdot)$. Also, while $f(i)$ is computed for i up to $k/2$ in (13), we need to compute $g(i)$ only for i up to $k/4$, as the data present at each source is only of size $m = k/4$ symbols. However, in practice, the above deconvolution of $p'(\cdot)$ still violates the non-negativity constraint on $g(\cdot)$. This is overcome by adjusting $f(\cdot)$ via linear interpolation, which alters the DSD $p(\cdot)$ slightly prior to deconvolution; however, this has little effect on the end result in terms of performance.

Simulation Results

We compare the performance of the following two schemes for $J = 4$, $m = 500$, $k = Jm = 2000$, in terms of the number of code symbols that must be received by the sink T to recover all the sources' data, assuming lossless transmission.

- *Scheme A:* Each source generates LT code symbols from its 500-symbol data set. The code symbols from different sources are time-multiplexed at N and transmitted to the sink T. The sink separately decodes each LT code to recover the data. Let N_A denote the total number of code symbols that are needed by T to recover all the data.
- *Scheme B:* Each source generates DLT-4 codewords from its 500-symbol data set. The DLT-4 codes are formed using a degree distribution that is derived from an underlying RSD. At every instant, the four code symbols generated by the sources are selectively XOR-ed at N and forwarded to the sink T. The sink thus receives a big LT code over all the sources' data, which is then decoded to recover all the data simultaneously. Let N_B denote the total number of code symbols that are needed by T for successful decoding.

The same values of c and δ are used for the RSDs in schemes A and B. (Note that the RSD in scheme A is only for degrees up to 500, whereas the RSD in B extends up to degrees of 2000.) Moreover, the value of δ chosen for constructing the RSDs in our simulations is fairly high (i.e., 0.5). This is due to the conservative nature of the analysis of failure probability in [1], as pointed out in [4].

The average values of N_A and N_B obtained via simulations are tabulated in Table I for different values of c and δ. Also, the observed cumulative distributions of N_A and N_B are plotted in Fig. 3.

TABLE I

AVERAGE NUMBER OF CODE SYMBOLS NEEDED TO RECOVER THE DATA ($k = 2000$ INFORMATION SYMBOLS)

c	δ	Average N_A	Average N_B
0.1	0.5	2423	2227
0.05	0.5	2345	2180

It is seen that scheme B compares favorably with scheme A, in terms of both the average number of symbols required to decode, as well as the probability of successful decoding for a fixed number of code symbols, for a wide range of code blocklengths. On average, scheme B requires only about *half* the overhead required by scheme A. Consequently, scheme B performs better even when the link connecting nodes N and T is lossy, i.e., constitutes an erasure channel.

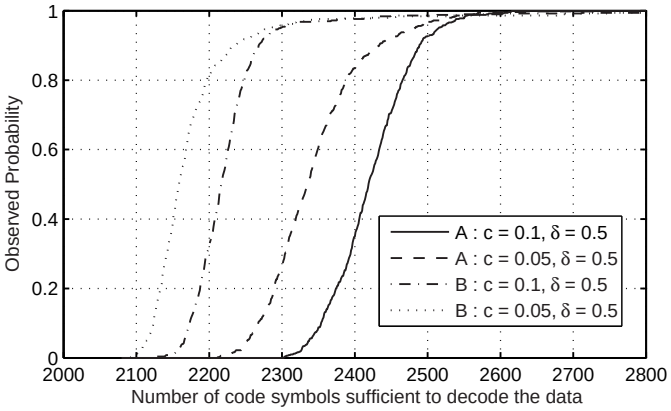

Fig. 3. Observed cumulative distributions of the number of code symbols required for schemes A and B

V. CONCLUSIONS

This paper presents a novel technique that allows distributed users to encode their data in such a way that, when the encoded packets are selectively XOR-ed, it results in an efficient rateless erasure correcting code, similar to an LT code, at a common destination. The primary advantage of the proposed approach is that it exploits the improved efficiency of such codes for large blocklengths; by forming a single codeword at the destination representing *all* the distributed data sets, there is a reduction in overhead compared with a system in which each data set is encoded with its own LT code having a smaller blocklength.

REFERENCES

[1] M. Luby, "LT Codes," *Proc. of the 43rd Annual IEEE Symp. on Foundations of Comp. Sc.*, pp 271-280, Vancouver, Canada, November 2002

[2] J. W. Byers, M. Luby, M. Mitzenmacher, "A Digital Fountain Approach to Asynchronous Reliable Multicast," *IEEE J. on Selected Areas in Communications*, vol. 20, no 8, pp. 1528-1540, August 2002.

[3] J. W. Byers, M. Luby, M. Mitzenmacher, A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," *Proc. of ACM SIGCOMM '98*, pp. 56-67, Vancouver, Canada, September 1998

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

[5] R. Ahlswede, N. Cai, S.-Y. R. Li, R. W. Yeung, "Network Information Flow," *IEEE Trans. on Information Theory*, vol. 46, no. 4, pp. 1204-1216, July 2000

[6] S.-Y. R. Li, R. W. Yeung, N. Cai, "Linear Network Coding," *IEEE Trans. on Information Theory*, vol. 49, no. 2, pp. 371-381, February 2003

[7] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain, L. Tolhuizen, "Polynomial Time Algorithms for Multicast Network Code Construction," *IEEE Trans. on Information Theory*, vol. 51, no. 6, pp. 1973-1982, June 2005

[8] R. Koetter, M. Médard, "An Algebraic Approach to Network Coding," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782-795, October 2003

[9] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. IT-8, pp. 21-28, January 1962