The Design and Performance of Distributed LT Codes

Srinath Puducheri, Jörg Kliewer, *Senior Member, IEEE*, and Thomas E. Fuja, *Fellow, IEEE*

***Abstract*—This paper describes techniques to decompose LT codes (a class of rateless erasure-correcting codes) into distributed LT (DLT) codes. DLT codes can be used to independently encode data from multiple sources in a network in such a way that, when the DLT-encoded packets are combined at a common relay, the resulting bit stream (called a modified LT (MLT) code) has a degree distribution approximating that of an LT code, with simulations indicating comparable performance. In essence, DLT codes are designed so that the final stage of encoding for erasure correction can be carried out by a low-complexity relay that selectively XORs the bit streams generated at each source and transmits the result to the sink. This paper presents results for two-source and four-source networks. It is shown that, when the relay-to-sink link is the bottleneck, the DLT/MLT approach can yield substantial performance benefits compared with a competing strategy wherein each of the sources uses its own independent LT encoder and the resulting bit streams are time-multiplexed through the relay.**

***Index Terms*—Distributed coding, erasure correction, LT codes, relay networks.**

I. Introduction

THE erasure channel [1] is a good network-layer model for several packetized transmission scenarios. One example is data transmission over the internet, wherein packets may be dropped en route to the destination. Another example is given by a wireless fading channel; here, a packet may be declared erased if the underlying physical layer error protection fails to recover the packet.

A common technique to ameliorate erasures is to use forward error control codes for erasure protection—in particular, maximum distance separable (MDS) codes such as Reed–Solomon codes. When using MDS codes, k data symbols are represented by n code symbols $(n > k)$ in such a way that knowing *any* k code symbols makes it possible to recover the k information symbols—i.e., up to $n - k$ erasures in any n-symbol codeword can be tolerated. One drawback of this approach is that both the encoding and decoding complexity is quadratic in k, which might be infeasible for long data blocks.

Another way to deal with erasures is to adopt automatic repeat-request (ARQ) protocols wherein a packet is retransmitted if it is not positively acknowledged by the receiver. However, such protocols may lead to large overhead, for example, if the acknowledgment packets can be erased as well. Moreover, congestion can be a problem if ARQ protocols are used in a network in which data is multicast to many receivers; a large number of receivers can mean many retransmission requests, and even if only a few receivers request a retransmission, the retransmission must be sent out to all receivers—even to those which have already recovered the data.

To address these shortcomings, fountain codes [2], [3] were proposed. The key feature of fountain codes is their so-called *rateless* property: an arbitrary number of code symbols can be generated from a given set of k information symbols. The rateless property means that a transmitter can simply transmit code symbols until each receiver has enough unerased symbols to recover the associated information symbols—i.e., until each receiver has filled its cup from the digital fountain; moreover, the only acknowledgment required from the receiver is an indication that it has successfully recovered the information. Thus, fountain codes provide reliable dissemination of information without the complexity and congestion incurred by more conventional forward error correction and ARQ techniques.

Fountain codes are especially well suited to systems in which the channel statistics are unknown *a priori*; they are universal codes in the sense that the amount of redundancy used is not fixed by the code design. If a fountain decoder can (with high probability) recover a k-symbol information set from any $n(k)$ associated code symbols, and if $\lim_{k\to\infty} k/n(k) = 1$, then the fountain code's performance approaches the Shannon capacity of the memoryless erasure channel with symbol erasure probability p—without ever taking the value of p into account in the code design.

LT codes were developed by Luby [4] as the first practical realization of fountain codes. In LT codes, the information and code symbols are binary strings, and encoding and decoding require only bitwise XOR operations. LT codes are very efficient as the data length grows—i.e., the average fractional overhead required to decode the data decreases with increasing block size. Moreover, the encoding/decoding complexity is significantly smaller than for Reed–Solomon codes [4]. Following Luby's work, *Raptor codes* were developed by Shokrollahi [5] as rateless fountain codes with even smaller encoding and decoding complexities than LT codes. Raptor codes make use of *precoded* LT codes with *degree distributions* that are more general than those developed in [4].

This paper addresses the encoding of distributed data via "LT-like" codes [6]. Consider a scenario in which two independent

Manuscript received September 1, 2006; revised January 19, 2007. This work was supported in part by the University of Notre Dame's Center for Applied Mathematics and its Faculty Research Program. It was also supported in part by the U.S. National Science Foundation under Grants CCF 02-05310 and TF 05-15012, as well as the German Research Foundation (DFG) under Grant KL 1080/3-1. The material in this paper was presented in part at the 2006 IEEE International Symposium on Information Theory, Seattle, WA, July 2006, and the 44th Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, September 2006.

S. Puducheri and T. E. Fuja are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: spuduche@nd.edu; tfuja@nd.edu).

J. Kliewer was with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA. He is now with the Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM 88003 USA (e-mail: jkliewer@nmsu.edu).

Communicated by G. Kramer, Guest Editor for the Special Issue on Relaying and Cooperation.

Digital Object Identifier 10.1109/TIT.2007.904982

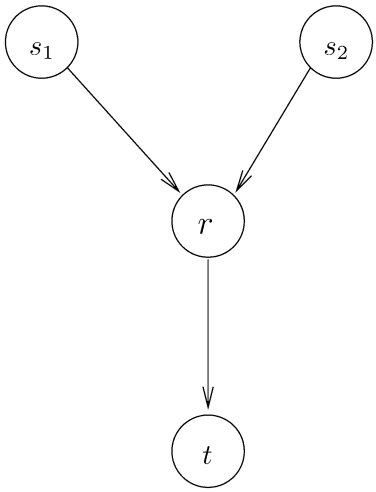

Fig. 1. A two-source single-sink relay network.

sources in a network transmit information to a sink through a common relay—i.e., the *multiple-access relay channel* (see Fig. 1). The relay is assumed to have limited data processing capability, and communication between the sources is impossible or not desired. We also assume that the relay has limited memory—specifically, it can store only one packet (symbol) per source at any given time. In such a situation, we may identify two distinct LT-based approaches to protect the data against erasures.

- Use two different LT codes to independently encode the two sources and time-multiplex the resulting two encoded sequences through the relay.
- Encode the data at the two sources in such a way that the relay—which combines its inputs in some low-complexity operation—transmits a sequence that "looks like" a single LT codeword.

This paper explores the second of these two approaches. In doing so, it seeks to exploit the efficiency inherent in long codewords; by having the relay transmit a single long codeword representing both sources—rather than two shorter codewords that each represent a single source—performance is enhanced. We refer to the codes used at each of the two sources as *DLT-2 codes*—"DLT" for "distributed LT"—while the codeword formed at the relay and delivered to the sink is called an *MLT-2 code*—"MLT" for "modified LT." This can be generalized to DLT-M and MLT-M codes for larger values of M.

The philosophy of this paper's approach is similar to that embodied by *network coding* [7]–[9]. Both techniques explore the advantages of going beyond mere *routing* at intermediate nodes in a network—of permitting intermediate nodes to carry out some form of encoding of the incident bit streams to produce the transmitted bit stream. While the goal of network coding is (typically) to minimize the use of network resources, the goal of the techniques in this paper is to provide protection against symbol erasures.

This paper is organized as follows. Section II provides a brief introduction to LT codes. Section III introduces a technique to decompose an LT code to obtain two DLT-2 codes and describes the formation of an MLT-2 code based on the DLT-2 codes; this is done for a two-source single-sink relay network, wherein all erasures occur on the relay-to-sink link. Section IV then extends this technique to the case of four sources and a single sink. The performance of these schemes obtained via simulation is presented in Section V. An improved technique to decompose LT codes into DLT-2 codes is presented in Section VI; this improved technique shows significantly better performance than the first scheme when there are erasures in the source-to-relay link. Finally, Section VII presents a summary and discussion of the results.

II. Some Background on LT Codes

LT codes were invented by Michael Luby as *random rateless* codes for the erasure channel [4], wherein each code symbol is generated as a linear combination of a *randomly selected* set of information symbols. This section briefly reviews the main results of [4].

A. Encoding and Decoding LT Codes

Assume that the goal is to encode k information symbols, where each symbol is a string of l bits. 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 used in [4] for the construction of LT codes is called the *robust soliton distribution* and is described in Section II-B.
- From the k information symbols, a set of d symbols is chosen uniformly at random—these constitute the *neighbors* of the code symbol. The neighbors are bitwise XORed to produce the code symbol.

The decoder is informed of the degree and the set of neighbors associated with every code symbol. Given a block of n $(> k)$ code symbols, the decoder recursively decodes the information symbols from the bipartite graph connecting the information and code symbols. The algorithm starts with degree one code symbols, removing their contribution from the graph to produce a smaller graph with another set of degree-one code symbols; the degree-one code symbols of this smaller graph are then removed, and the process continues, as described in [4]. This procedure is equivalent to graph-based erasure decoding of low-density parity-check (LDPC) codes [10] using the belief-propagation algorithm.

A *decoding failure* occurs if the decoder fails to recover all k data symbols. This happens if, at any stage prior to completion, there are no degree-one code symbols remaining.

B. Degree Distribution

The degree distribution used to form LT codes—the robust soliton distribution—is constructed such that the decoder can recover k data symbols from slightly more than k code symbols with high probability.

Definition 1: For constants $c > 0$ and $\delta \in [0, 1]$, the robust soliton distribution (RSD) $\mu(\cdot)$ is given by

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

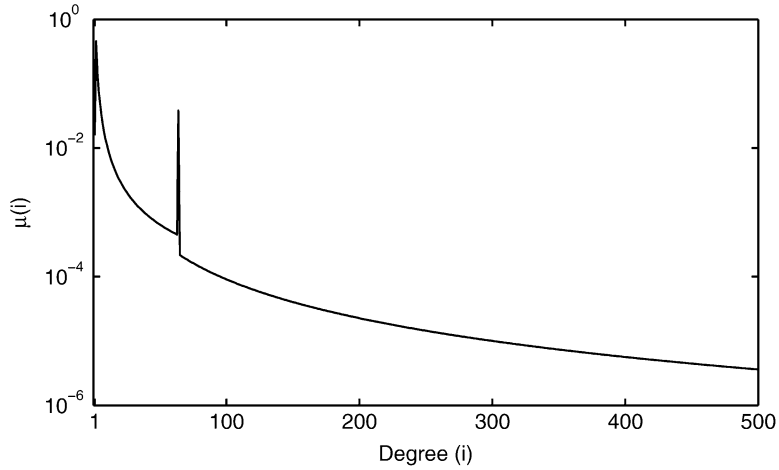

Fig. 2. The RSD $\mu(\cdot)$, with $k = 500, c = 0.05, \delta = 0.5$.

Here, $\beta = \sum_{i=1}^{k}(\rho(i) + \tau(i))$ is a normalizing constant, and $\rho(i)$ and $\tau(i)$ are given by

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

$$\tau(i) = \begin{cases} S/ik, & \text{for } 1 \le i \le \frac{k}{S} - 1 \\ S\ln(S/\delta)/k, & \text{for } i = \frac{k}{S} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

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

$$S = c \cdot \sqrt{k} \cdot \ln\left(\frac{k}{\delta}\right). \tag{4}$$

Luby showed that, for a suitably chosen[1] value of c (independent of k and δ), the decoder can recover the data from n LT code symbols with failure probability at most δ [4], where n is given by

$$n = k\beta = k + c \cdot \sqrt{k} \cdot \ln^2(k/\delta). \tag{5}$$

Thus, the fractional overhead is given by $\frac{c \cdot \ln^2(k/\delta)}{\sqrt{k}}$, which goes to zero with increasing k.

It was observed in [4] that the analysis that bounds the failure probability by δ is quite pessimistic, and the actual failure probability is typically much smaller than δ. As a result, LT codes can be designed with large values of δ and still exhibit good performance.

C. An Observation Regarding the RSD

It is easy to see that the RSD's support is mostly restricted to a vanishingly small set of degrees, as indicated by the following lemma.

Lemma 1: Asymptotically, the RSD is localized to the set of degrees from 1 to $N_k = k/S$. More precisely

$$\lim_{k\to\infty} \sum_{i=1}^{N_k} \mu(i) = 1 \quad \text{and} \quad \lim_{k\to\infty} [N_k/k] = 0. \tag{6}$$

[1]It was observed in [11, p. 592] that, in practice, c can be chosen as a free parameter.

Proof: Observe that

$$\sum_{i=N_k+1}^{k} \mu(i) = \sum_{i=\frac{k}{S}+1}^{k} \mu(i) \tag{7}$$

$$= \frac{1}{\beta} \cdot \frac{S-1}{k}. \tag{8}$$

Note from (1) that $\beta > 1$, as $\sum_{i=1}^{k} \rho(i) = 1$. Also, from (4), it follows that

$$\lim_{k\to\infty} \frac{S-1}{k} = 0 \tag{9}$$

Consequently, from (8), it follows that

$$\lim_{k\to\infty} \sum_{i=N_k+1}^{k} \mu(i) = 0 \tag{10}$$

which demonstrates the first part of (6). Now, $\frac{N_k}{k} = \frac{1}{S}$, and from (4) it follows that $\lim_{k\to\infty} \frac{1}{S} = 0$. This demonstrates the second part of (6). □

Another fact important to our construction is that the value of $\tau(k/S)$ causes a spike in the RSD at $i = k/S$. This is illustrated in Fig. 2.

III. Decomposing an LT Code Into Two DLT-2 Codes

Consider again the network in Fig. 1. Two sources s_1 and s_2 communicate with the same sink t via the relay r. Each source s_i has a block $\mathfrak{D}_i$ of $k/2$ information symbols to be conveyed to the sink. Let X_1 be a code symbol generated at s_1 by XORing d_1 information symbols from $\mathfrak{D}_1$, and X_2 is likewise generated at s_2 from $\mathfrak{D}_2$ and is of degree d_2. Both d_1 and d_2 have the same degree distribution. Our goal is to make the degree of $X_1 \oplus X_2$ be a random variable following the RSD. The motivation is that the relay can then simply XOR the symbols it receives from the two sources, and the resulting linear combination will constitute an "LT-like" sequence of code symbols generated by the k data symbols formed by concatenating $\mathfrak{D}_1$ and $\mathfrak{D}_2$. If the degrees of X_1 and X_2 are both generated independently according to the distribution $p(\cdot)$, then the degree of $X_1 \oplus X_2$ is $d_1 + d_2$ and

has the distribution $(p*p)(\cdot)$, where "$*$" indicates convolution. Thus, to determine $p(\cdot)$ requires the *deconvolution* of the RSD.

A. Deconvolution of the RSD

Direct deconvolution of the RSD $\mu(\cdot)$ does not necessarily yield a valid probability distribution. More generally, for any finite-length real sequence $y[n]$, there may not exist any other finite-length real sequence $x[n]$, such that $(x*x)[n] = y[n]$ for all n. Moreover, these problems are specific to the RSD.

1) For $k \geq 2$, the only way of ensuring that $(p*p)(1) = \mu(1) > 0$ is if we let $p(0) > 0$. However, this would imply that $X_1 \oplus X_2$ could have degree zero, which is clearly wasteful of resources.
2) Disregarding degree-one symbols in the RSD, if we try to reproduce $\mu(i)$ for $i > 1$ by recursively solving for $p(\cdot)$ from

$$\begin{aligned} p^2(1) &= \mu(2), \\ 2\cdot p(1)\cdot p(2) &= \mu(3), \\ 2\cdot p(3)\cdot p(1) + p^2(2) &= \mu(4), \\ 2\cdot p(4)\cdot p(1) + 2\cdot p(3)\cdot p(2) &= \mu(5), \\ &\vdots \end{aligned} \tag{11}$$

   then we obtain a negative value for $p(k/S)$ due to the "spiky" behavior of the RSD at k/S.
3) Recall that $\mathfrak{D}_1$ and $\mathfrak{D}_2$ each contain $k/2$ symbols. Hence, the support of $p(i)$ is $i = 1, \ldots, k/2$. Thus, only the first $k/2+1$ degrees of the RSD will be included in (11) above, and the "tail" of the RSD (for $i = k/2+2, \ldots, k$) cannot be reproduced.
4) Finally, if we restrict ourselves to deconvolving a *portion* of the RSD as dictated by the preceding constraint, then we are *not* guaranteed that the result $p(\cdot)$ will sum to one.[2]

We shall see that, due to Lemma 1, the last two problems above do not pose a significant difficulty.

To avoid direct deconvolution, we first split the RSD $\mu(\cdot)$ into two distributions—$\mu'(\cdot)$ and $\mu''(\cdot)$—such that $\mu''(\cdot)$ captures the problematic part of the RSD (i.e., the degree-one symbols and the spike at $i = k/S$) and $\mu'(\cdot)$ is a smooth distribution that is easier to deconvolve.

Assume that $\rho(\cdot)$ and $\tau(\cdot)$ are given by (2) and (3), respectively. Then $\mu'(\cdot)$ is defined as follows:

$$\mu'(i) = \begin{cases} 0, & \text{for } i = 1 \\ \frac{\rho(i)+\tau(i)}{\beta'}, & \text{for } 2 \leq i \leq \frac{k}{S} - 1 \\ \frac{\rho(i)}{\beta'}, & \text{for } \frac{k}{S} \leq i \leq k \end{cases} \tag{12}$$

with the normalization factor $\beta' = \sum_{i=2}^{k} \rho(i) + \sum_{i=2}^{k/S-1} \tau(i)$. Similarly, $\mu''(\cdot)$ is given by

$$\mu''(i) = \begin{cases} \frac{\rho(1)+\tau(1)}{\beta''}, & \text{for } i = 1 \\ \frac{\tau(k/S)}{\beta''}, & \text{for } i = \frac{k}{S} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

[2]On the other hand, if $\sum_{n=-\infty}^{+\infty} x[n] = 1$ and $(y*y)[n] = x[n]$ *for all* n, then we have $\sum_{n=-\infty}^{+\infty} y[n] = 1$.

with the normalization factor $\beta'' = \rho(1) + \tau(1) + \tau(\frac{k}{S})$. Thus, noting that $\beta' + \beta'' = \beta$, from (1) the RSD can be rewritten as

$$\begin{aligned} \mu(i) &= \frac{\beta'}{\beta}\cdot\mu'(i) + \frac{\beta''}{\beta}\cdot\mu''(i) \\ &= \frac{\beta'}{\beta}\cdot\mu'(i) + \left(1 - \frac{\beta'}{\beta}\right)\cdot\mu''(i), \quad \text{for } 1 \leq i \leq k. \end{aligned} \tag{14}$$

So the RSD $\mu(\cdot)$ is a mixture of the distributions $\mu'(\cdot)$ and $\mu''(\cdot)$ with mixing parameter β'/β.

The approach taken in this paper is to deconvolve the smooth distribution $\mu'(\cdot)$ and use the result in the construction of the DLT codes. To that end, define the function $f(\cdot)$ by this equation

$$(f*f)(i) = \mu'(i), \qquad \text{for } 2 \leq i \leq k/2+1. \tag{15}$$

Then the solution to (15) is given recursively by

$$f(i) = \begin{cases} \sqrt{\mu'(2)}, & \text{for } i = 1 \\ \frac{\mu'(i+1) - \sum_{j=2}^{i-1} f(j)f(i+1-j)}{2f(1)}, & \text{for } 2 \leq i \leq \frac{k}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

It is our conjecture that $f(i) \geq 0$ for all i; in all the codes we have designed based on $f(\cdot)$ we have never observed otherwise. An intuition for this behavior can be obtained by observing that the function $\mu'(i)$ for $i > 1$ exhibits an *inverse polynomial decay* (i.e., of the form $1/i^c$ for $c > 0$). Thus, the behavior of $\mu'(i)$ varies between that of an exponential $a^{-i} (a > 1)$, for small i, and a constant, for large i—both of which yield a positive sequence on deconvolution. Therefore, deconvolving $\mu'(i)$ for $i > 1$ will likely yield a positive sequence as well.

If we define $f^{\text{ext}}(i)$ as the solution of the deconvolution in (16) extended all the way up to k degrees, we have observed that, similarly, $f^{\text{ext}}(i) \geq 0$ for all i. If these conjectures are true, they imply the following.

- We have

$$0 < (f*f)(i) < (f^{\text{ext}} * f^{\text{ext}})(i) = \mu'(i) \tag{17}$$

  for $i = k/2+2, \ldots, k$. This follows from the fact that, in the convolution of two nonnegative sequences with no all-zero terms, setting a portion of each sequence to zero can only reduce the convolution sum to a smaller positive value. This means that, from Lemma 1, the contribution of $\mu'(i)$ and $(f*f)(i)$ over $k/2+2 \leq i \leq k$ is quite small. Consequently, any differences over this range of degrees are of little significance.
- The following limit holds:

$$\lim_{k\to\infty} \sum_{i=1}^{k/2} f(i) = 1. \tag{18}$$

  From Lemma 1, (17), and the fact that $(f*f)(i) = \mu'(i)$ for $1 \leq i \leq k/2+1$ it follows that

$$\lim_{k\to\infty} \sum_{i=1}^{k} (f*f)(i) = 1. \tag{19}$$

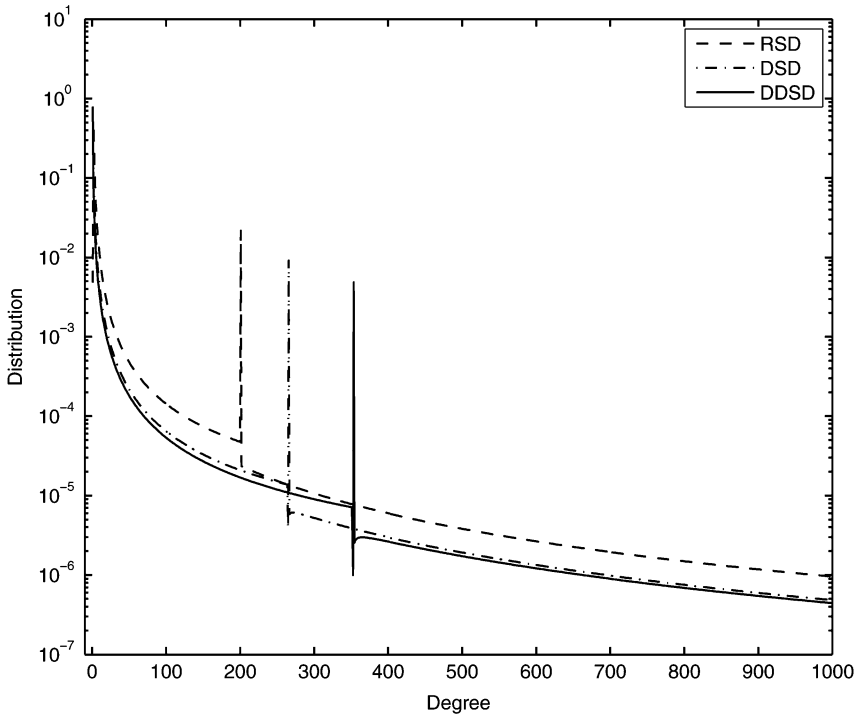

Fig. 3. Degree distributions with support over degrees 1 through 10000. (Only the first 1000 are shown.) (RSD: robust soliton distribution—DSD: deconvolved soliton distribution—DDSD: doubly deconvolved soliton distribution).

Inserting the relation

$$\sum_{i=1}^{k}(f*f)(i)=\left(\sum_{i=1}^{k/2}f(i)\right)^2 \tag{20}$$

into (19) finally leads to (20). In practice, $f(\cdot)$ is normalized so that it sums to one, but we still have $(f*f)(i)\approx\mu'(i)$ for $1\le i\le k/2+1$.

We now define a new distribution derived by mixing $f(\cdot)$ and $\mu''(\cdot)$.

Definition 2: The deconvolved soliton distribution (DSD) $p(\cdot)$ is given by

$$p(i)=\lambda\cdot f(i)+(1-\lambda)\cdot\mu''(i),\qquad \text{for } 1\le i\le k/2 \tag{21}$$

with the mixing parameter λ given by

$$\lambda=\sqrt{\frac{\beta'}{\beta}}. \tag{22}$$

A plot of the DSD and the RSD is shown in Fig. 3 for $k=10000$. (Only the first 1000 degrees are shown.) Also shown is a plot of the *doubly-DSD*, defined in Section IV-A.

B. Distributed Encoding: DLT-2 and MLT-2 Codes

This subsection addresses how the DSD can be used to encode information in the network of Fig. 1 such that the code symbols received by the sink t follow (approximately) the RSD in degree.

As a first step, the information at each of the two sources is encoded using the DSD as the degree distribution; it is assumed that a code symbol's degree d is generated by first randomly selecting either $f(\cdot)$ (with probability λ) or $\mu''(\cdot)$ (with probability $1-\lambda$) and then generating d with the selected distribution. Then, the symbol's neighbors are selected equiprobably from among the $\binom{k/2}{d}$ possibilities. This process produces a sequence of code symbols we refer to as a DLT-2 code.

The relay then *selectively* XORs the pair of DLT-2 code symbols it receives from the two sources as follows:

1) if both source symbols were chosen according to the $f(\cdot)$ component of the DSD, they are XORed and transmitted to the sink;
2) if exactly one source symbol was chosen according to $\mu''(\cdot)$, then it is transmitted to the sink and the other source symbol is discarded;
3) if both source symbols were chosen according to $\mu''(\cdot)$, one of them is randomly selected and transmitted to the sink while the other is discarded.

Consequently, with probability $\lambda^2=\beta'/\beta$, the symbol transmitted to the sink is the XOR of two source symbols and has a degree distributed (approximately) according to $\mu'(\cdot)$, while with probability $1-\lambda^2=(1-(\beta'/\beta))$, the symbol transmitted to the sink has a degree distributed according to $\mu''(\cdot)$. Thus, by (14), the degree of the symbol transmitted by the relay obeys (approximately) the RSD. We refer to the sequence of symbols transmitted by the relay as an *MLT-2 code*.

In the operation described above, the relay must know from which distribution—$f(\cdot)$ or $\mu''(\cdot)$—the degree of each encoded symbol it receives is drawn. This could be provided by appending a single bit to the l-bit string making up each code symbol. Alternatively, if the relay knows the actual *degree* of each symbol it receives, it is possible to construct a randomized decision protocol such that the relay transmits a symbol whose

degree obeys (approximately) the RSD. That randomized decision rule is as follows.

1) Let X_1 and X_2 denote the symbols received from the two sources, and let d_1 and d_2 denote their degrees.
2) The relay generates[3] two independent random variables U_1 and U_2, each uniformly distributed on $[0, 1]$.
3) The relay generates two binary random variables b_1 and b_2 as follows:

$$b_i = \begin{cases} 1, & \text{if } \left(d_i = 1 \text{and } U_i < 1 - \frac{\lambda \cdot f(1)}{p(1)}\right) \\ & \text{or } \left(d_i = k/S \text{ and } U_i < 1 - \frac{\lambda \cdot f(k/S)}{p(k/S)}\right) \\ 0, & \text{otherwise.} \end{cases}$$

4) The relay then transmits the binary random variable Y defined as follows:

$$Y = \begin{cases} X_1 \oplus X_2, & \text{if } b_1 = b_2 = 0 \\ X_1, & \text{if } b_1 = 1 \text{ and } b_2 = 0 \\ X_2, & \text{if } b_1 = 0 \text{ and } b_2 = 1 \\ \text{flip}(X_1, X_2), & \text{if } b_1 = b_2 = 1. \end{cases}$$

Here, $\text{flip}(X_1, X_2)$ is a random variable taking the value of either X_1 or X_2 with equal probability.

It can be shown that this randomized protocol generates a random variable Y with a degree distribution identical to that obtained when the source degree distribution—$f(\cdot)$ or $\mu''(\cdot)$—is encoded into each symbol and used at the relay as described previously.

IV. Extension to Four Sources: DLT-4 and MLT-4 Codes

Consider now the scenario in which four sources are communicating with a common sink through a relay, as shown in Fig. 4. If we combine the information from all four sources and encode them into a single codeword at the relay, then we can exploit the benefits of larger block length compared to alternate strategies such as time-multiplexing two MLT-2 or four LT codes to the sink. To accomplish this, we further decompose DLT-2 codes into a pair of DLT-4 codes, which are then used by each of the four sources. The four DLT-4 code symbols from the four sources are selectively XORed at the relay to yield an MLT-4 code.

We start by finding two codes which, when selectively XORed, will yield a DLT-2-like code—i.e., a code which (like the DLT-2 code) has the DSD as its degree distribution. As before, this requires the deconvolution of the target degree distribution, namely the DSD.

A. Deconvolution of the DSD

Each source s_i in Fig. 4 has a block $\mathfrak{D}_i$ of $k/4$ data symbols it wishes to communicate to the sink. Similar to the two-source case, the presence of degree-one symbols and the "spiky" behavior of the DSD make direct deconvolution problematic; therefore, we adopt a similar methodology and split the DSD

[3]We see from Steps 3) and 4) of the algorithm that if $d_1 \notin \{1, k/S\}$ and $d_2 \notin \{1, k/S\}$, then $b_1 = b_2 = 0$ and $Y = X_1 \oplus X_2$, so there is no need to generate U_1 and U_2.

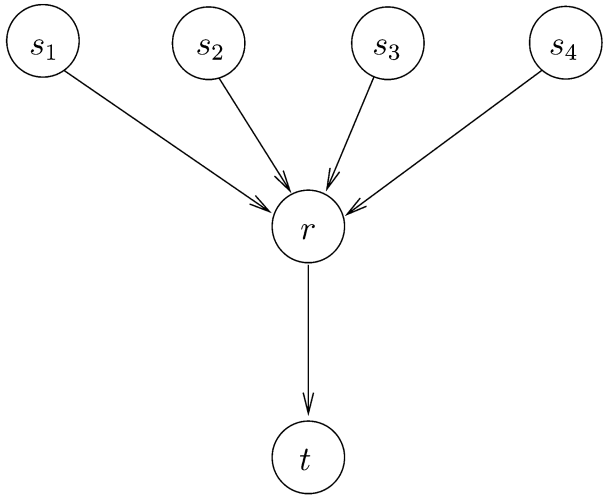

Fig. 4. A four-source single-sink relay network.

into two component distributions. The split is along the lines of (21), with some minor differences.

In practice, it is observed that the $f(i)$ component of the DSD must be slightly modified at $i = \frac{k}{S} - 1$, as otherwise the split-and-deconvolve approach used in Section III-A yields a negative value at $i = \frac{k}{S} - 2$ for the resulting distribution. Therefore, to "smooth" out $f(i)$, we replace the value of $f(\frac{k}{S}-1)$ with a linear interpolation between $f(\frac{k}{S} - 2)$ and $f(\frac{k}{S})$. Thus, we define a new distribution $f^{(\text{new})}(i)$ as follows:

$$f^{(\text{new})}(i) = \begin{cases} \frac{f(i)}{\gamma}, & \text{for } i \neq \frac{k}{S} - 1 \\ \frac{1}{\gamma} \cdot \frac{f\left(\frac{k}{S}\right) + f\left(\frac{k}{S} - 2\right)}{2}, & \text{for } i = \frac{k}{S} - 1 \end{cases} \tag{23}$$

where γ is chosen so that $f^{(\text{new})}(i)$ sums to one. Since $f(i)$ is quite small in the neighborhood of $i = k/S$, in practice, γ is approximately one. Similar to (21), we now define $p^{(\text{new})}(i)$ as a mixture of $f^{(\text{new})}(i)$ and $\mu''(i)$:

$$p^{(\text{new})}(i) = \lambda \cdot f^{(\text{new})}(i) + (1 - \lambda) \cdot \mu''(i), \quad \text{for } 1 \leq i \leq k/2 \tag{24}$$

with λ as defined in (22). In place of the DSD, $p^{(\text{new})}(i)$ now forms our new target distribution.

In the next step, $p^{(\text{new})}(i)$ is split into two distributions $p'(i)$ and $p''(i)$ as follows:

$$p'(i) = \begin{cases} 0, & \text{for } i = 1 \\ \frac{1}{\gamma'} \cdot \lambda \cdot f^{(\text{new})}\left(\frac{k}{S}\right), & \text{for } i = \frac{k}{S} \\ \frac{1}{\gamma'} \cdot p^{(\text{new})}(i), & \text{otherwise} \end{cases} \tag{25}$$

and

$$p''(i) = \begin{cases} \frac{1}{\gamma''} \cdot p^{(\text{new})}(1), & \text{for } i = 1 \\ \frac{1}{\gamma''} \cdot (1 - \lambda) \cdot \mu''\left(\frac{k}{S}\right), & \text{for } i = \frac{k}{S} \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

where γ' and γ'' are chosen such that $p'(i)$ and $p''(i)$ each sum to one. Thus, we have

$$\begin{aligned} p^{(\text{new})}(i) &= \gamma' \cdot p'(i) + \gamma'' \cdot p''(i) \\ &= \gamma' \cdot p'(i) + (1 - \gamma') \cdot p''(i), \quad \text{for } 1 \leq i \leq k/2. \end{aligned} \tag{27}$$

We finally proceed to deconvolve $p'(i)$ to obtain $g(i)$ according to

$$g(i) = \begin{cases} \sqrt{p'(2)}, & \text{for } i = 1 \\ \frac{p'(i+1) - \sum_{j=2}^{i-1} g(j)g(i+1-j)}{2\,g(1)}, & \text{for } 2 \le i \le \frac{k}{4} \\ 0, & \text{otherwise.} \end{cases} \quad (28)$$

The issues regarding the validity of $f(\cdot)$ as a probability mass function arise again here with respect to $g(\cdot)$. Once again, we observe that all realizations of $g(\cdot)$ we have generated have yielded $g(i) \ge 0$ for all i. Moreover, it can be shown that $\sum_{i=1}^{k/4} g(i) \approx 1$ for large k, and $g(\cdot)$ can be normalized to guarantee $\sum_{i=1}^{k/4} g(i) = 1$.

Definition 3: The doubly deconvolved soliton distribution (DDSD) $q(\cdot)$ is given by

$$q(i) = \eta \cdot g(i) + (1-\eta) \cdot p''(i), \qquad \text{for } 1 \le i \le k/4 \quad (29)$$

with the mixing parameter η given by

$$\eta = \sqrt{\gamma'}. \quad (30)$$

A plot of the DDSD is shown in Fig. 3 together with the RSD and the DSD for $k = 10000$. (Only the first 1000 degrees are shown.)

B. MLT-4 Codes: Encoding

The encoding of MLT-4 codes is similar to and based on the construction of an MLT-2 code for the two-source case, as outlined in Section III-B.

At each source in Fig. 4, the $k/4$ information symbols are encoded using the DDSD as indicated in (29). This is done by first randomly selecting either $g(\cdot)$ (with probability η) or $p''(\cdot)$ (with probability $1-\eta$) and then generating d with the selected distribution. Then, the symbol's neighbors are selected equiprobably from among the $\binom{k/4}{d}$ possibilities. This process produces a sequence of code symbols referred to as a DLT-4 code. (One bit is appended to each l-bit DLT-4 code symbol to indicate whether it was selected according to $g(\cdot)$ or $p''(\cdot)$.) At the relay:

- the four DLT-4 sequences are first combined to produce a pair of sequences that are (approximately) DLT-2 sequences; this two-to-one reduction makes use of the bit appended to each code symbol—i.e., two symbols are XORed if they were both drawn according to $g(\cdot)$, while if exactly one symbol was drawn according to $p''(\cdot)$ that symbol is inserted into the corresponding DLT-2 sequence, and if both symbols were drawn according to $p''(\cdot)$ then the relay randomly selects one to insert;
- the resulting pair of (approximately) DLT-2 sequences are then combined into a single sequence that is referred to as an MLT-4 code sequence. This is carried out using a randomized protocol similar to the one described in Subsection III-B.

The net result is an MLT-4 sequence with a degree distribution that (approximately) satisfies the RSD up to degree $k/4$.

V. Performance of MLT Codes

This section presents simulation results describing the performance of MLT-2 and MLT-4 codes.

A. Comparison With Parent LT Codes

MLT codes are designed to have (approximately) the same degree distribution as LT codes—viz., the RSD. However, there is clearly a critical difference between the two codes: The d source symbols that form the neighborhood of a given LT code symbol are picked uniformly over all k source symbols, whereas for MLT codes the constituent neighborhoods are formed independently at each source and then combined at the relay. (As an example, consider an MLT-2 code symbol: If its degree is anything other than 1 or k/S, then the symbol *cannot* have a neighborhood wholly produced by a single $k/2$-bit source; by construction, it must have neighbors coming from *both* sources. An LT code for the same k symbols would not be constrained as such.) The natural question arises: What effect does this have on code performance in practice?

The *parent LT code* of an MLT code is the LT code whose degree distribution was deconvolved to obtain the corresponding DSD or DDSD used to construct the MLT code. Thus, both the MLT code and its parent LT code possess (approximately) the same degree distribution and encode the same total number of information symbols.

Consider the fractional overhead Δ required to decode k data symbols when LT and MLT codes are used. The experimentally observed complementary cumulative distribution functions (CDFs) of Δ for these codes—i.e., $\Pr(\Delta > x)$ versus x—are plotted in Fig. 5 for $k = 1000$ information symbols. Both MLT-2 and MLT-4 codes exhibit performance inferior to that of the parent LT code, with the MLT-4 codes performing the worst. This may be explained by as follows: i) the fraction of degrees over which the MLT-4 code (by design) follows the RSD is much smaller $(1/4)$ than for the MLT-2 code $(1/2)$; ii) the deviation from picking information symbols uniformly to construct the code symbol is larger in the case of MLT-4 codes than MLT-2 codes.

B. Benefits of Using MLT Codes in a Relay Scenario

This section considers the benefits of using MLT-2 and MLT-4 codes to jointly encode information from multiple sources as opposed to using individual LT codes to encode each source separately. Specifically, consider the four-source network of Fig. 4; assume that there are $k/4$ information symbols to be encoded at each of the four sources. Then the three schemes under comparison are described as follows:

1) each source separately encodes its information using an LT code, and the four LT codes are multiplexed at the relay and transmitted;
2) the four sources encode using DLT-2 codes, and are grouped into two pairs—each pair contributes to an MLT-2 code at the relay, and the two MLT-2 codes are multiplexed and transmitted;
3) the four sources encode their information using DLT-4 codes which are combined into an MLT-4 code at the relay and transmitted to the sink.

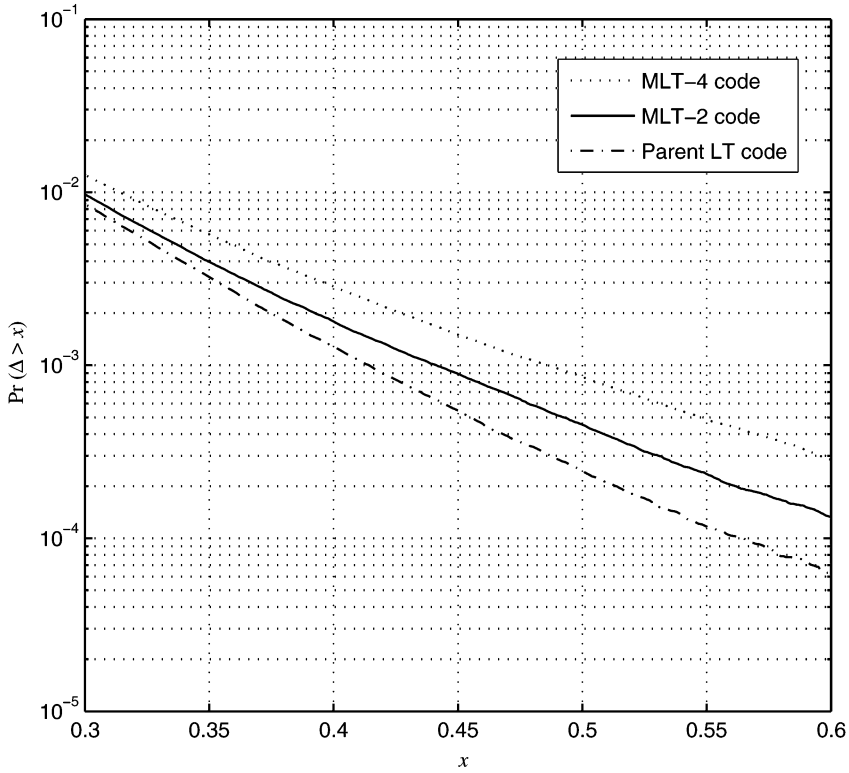

Fig. 5. Complementary CDFs of the fractional overhead (Δ) required to recover $k = 1000$ data symbols—MLT-2 and MLT-4 codes versus their parent LT code: $(c, \delta) = (0.05, 0.5)$.

We make comparisons assuming that the code symbols are transmitted from the relay to the sink at the same transmission rate (i.e., symbols/s) for all three schemes. For the first scheme, these are just four multiplexed LT sequences, while for the second scheme they are two multiplexed MLT-2 sequences, and for the third scheme they represent a single MLT-4 sequence. However, it should be pointed out that the third scheme requires a *source-to-relay* transmission rate that is *four times* that required by the first scheme; with the third scheme, four long sequences are selectively XORed to create another equally long sequence that is transmitted to the sink, while in the first scheme four shorter sequences are concatenated to form the sequence transmitted to the sink. In a similar fashion, the second scheme requires a source-to-relay transmission rate that is twice that required in the first scheme. Thus, the proposed approaches are best suited to scenarios in which the relay-to-sink link is the bottleneck.

1) Overhead: Simulation results for the above schemes are shown in Fig. 6. In these simulations there are $k = 2000$ source symbols, and the RSD parameters are $c = 0.05$ and $\delta = 0.5$ for each code. Fig. 6 shows the observed CDF of the number of code symbols required to decode all $k = 2000$ source symbols; from these curves it can be shown that the average overhead for the LT scheme is about 342 code symbols, whereas it is about 263 for the MLT-2 scheme and about 205 for the MLT-4 scheme. Thus, there is a reduction in overhead of about 23% with the MLT-2 scheme and about 40% with the MLT-4 scheme, compared with the LT scheme.

2) Frame Erasure Rate for Lossless Source–Relay Links: Consider the case of Fig. 4 in which the source–relay channels are error-free, and the relay–sink channel is a memoryless erasure channel with symbol erasure probability $p_{\rm rs}$.[4] Once again, assume that each source generates $k/4$ data symbols and these are to be conveyed to the sink via three different approaches. These approaches all use a fixed rate of $1/2$—i.e., $n = 2k$ symbols are delivered to the sink, and those n symbols are either sufficient or insufficient to recover all the data.

- **Scheme I (LT codes):** The four sources each encode their $k/4$ data symbols as $n/4$ LT code symbols, which are multiplexed at the relay and transmitted to the sink. A frame erasure occurs at the sink if a received LT codeword cannot be decoded.
- **Scheme II (MLT-2 codes):** The sources transmit $n/2$ DLT-2 code symbols each, contributing to two MLT-2 codewords of length $n/2$ at the relay; these two MLT-2 codewords are multiplexed and transmitted to the sink. A frame erasure occurs if a received MLT-2 codeword cannot be decoded.
- **Scheme III (MLT-4 codes):** Each source encodes its data onto n DLT-4 code symbols resulting in a length-n MLT-4 codeword generated at the relay and transmitted to the sink. A frame erasure occurs if a received MLT-4 codeword cannot be decoded.

Fig. 7 shows the frame erasure rate (FER) versus $p_{\rm rs}$ for a particular source as seen at the sink. Both Schemes II and III have a significantly lower FER than Scheme I. Moreover, the MLT-4 code provides a significant improvement in performance

[4]Such a scenario could describe a network in which multiple sources are clustered around a relay, which gathers the sources' information and forwards it over a long distance.

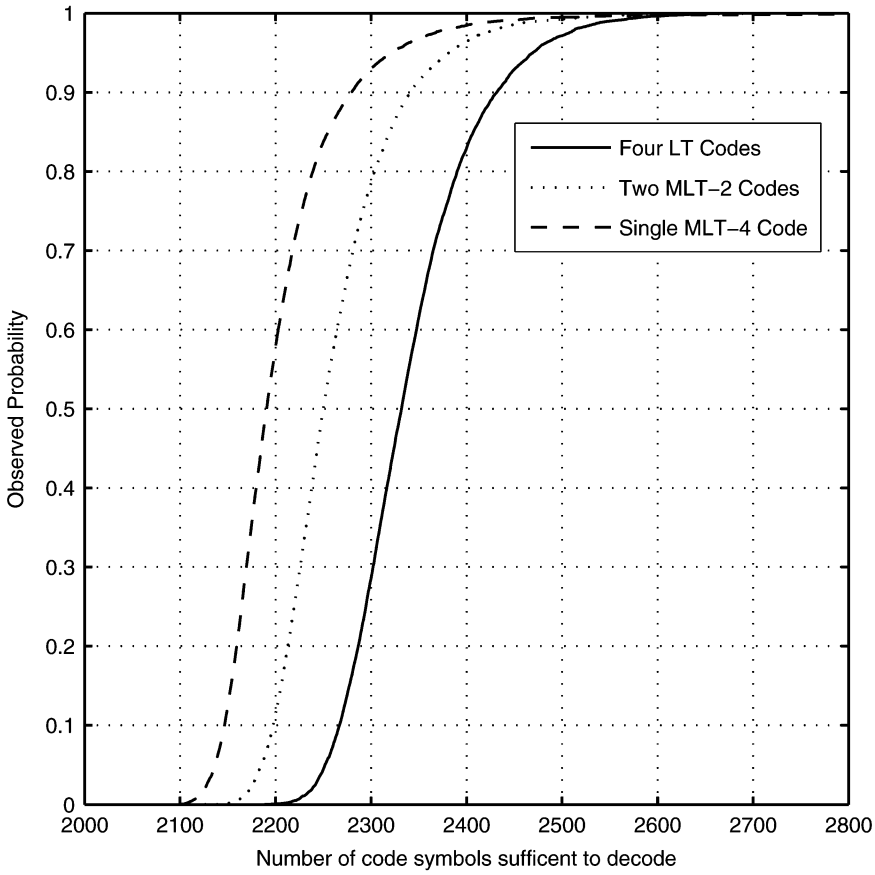

Fig. 6. Observed CDFs of the number of code symbols required to recover k symbols of data—using LT, MLT-2, and MLT-4 codes: $(k, c, \delta) = (2000, 0.05, 0.5)$.

over the MLT-2 codes as well. This is expected, given that the MLT-4 codewords are the longest, followed by the MLT-2 and LT codewords.

3) Frame Erasure Rate With Lossy Source–Relay Links: Up to now we have assumed perfect source–relay channels; we now consider source–relay channels in which each symbol is erased with probability $p_{\rm sr}$. The case when the source–relay channels are *asymmetric* is considered in Section VI-C.

For brevity, we focus on the MLT-2 scheme applied to the two-source network in Fig. 1; each source has $k/2$ information symbols to be conveyed. Assume the relay reacts to erasures as follows.

- If the relay receives a pair of unerased symbols from the two sources, the relay proceeds as described previously—transmitting either the XOR of the two symbols or just one of them.
- If exactly one of the two symbols received at the relay is erased, then the relay transmits the unerased symbol.
- If both symbols are erased, then the relay transmits nothing (saving energy)—or, equivalently, the sink observes an erasure.

In assessing an MLT-2 code, it is important to keep in mind that the DLT/MLT scheme requires the transmission of (approximately) twice as many symbols from each source to the relay as the comparable LT code. (The LT scheme multiplexes two shorter LT codewords from the sources through the relay, while the DLT/MLT scheme XORs two longer codewords.) Thus, the comparisons in Fig. 7 implicitly assume that the source-to-relay transmission is "free"—also reflected by the assumption that the source-to-relay link is lossless. However, if we now assume the source-to-relay link has erasures, it is appropriate to equalize the number of transmissions on the source–relay link for both schemes under consideration. We do this by assuming that, in the LT scheme, the source uses a rate-$1/2$ repetition code to transmit its LT code symbols to the relay; so each LT code symbol is transmitted twice, and the relay uses this redundancy to mitigate the effects of source-to-relay erasures before it multiplexes the received LT sequences on to the sink. (A repetition code is used to be consistent with the constraint that the relay be kept as simple as possible—and that, in particular, the relay has a memory constraint of one symbol (packet).)

The resulting FER curves as a function of the relay-to-sink erasure rates are shown in Fig. 8 for several different values of the source-to-relay erasure rate. For small values of $p_{\rm sr}$ (≈ 0.01), the LT scheme has essentially the same performance as the lossless $(p_{\rm sr} = 0)$ case. Moreover, the MLT-2 scheme matches or outperforms the LT scheme for small to moderate $p_{\rm sr}$ (0.01 to 0.3). For $p_{\rm sr} \geq 0.4$ (not shown in Fig. 8) the performance of the MLT-2 schemes falls below that of the LT scheme.

An interesting observation is that the MLT-2 scheme performs *better* for small values of $p_{\rm sr}$ (around 0.01) than for the lossless case. This could be due to the fact that, in this regime, *simultaneous* erasures on both source–relay links rarely occur. Thus, the code symbols transmitted to the sink form some appropriate random mixture of DLT-2 and MLT-2 code symbols that perform better than the MLT-2 code itself.

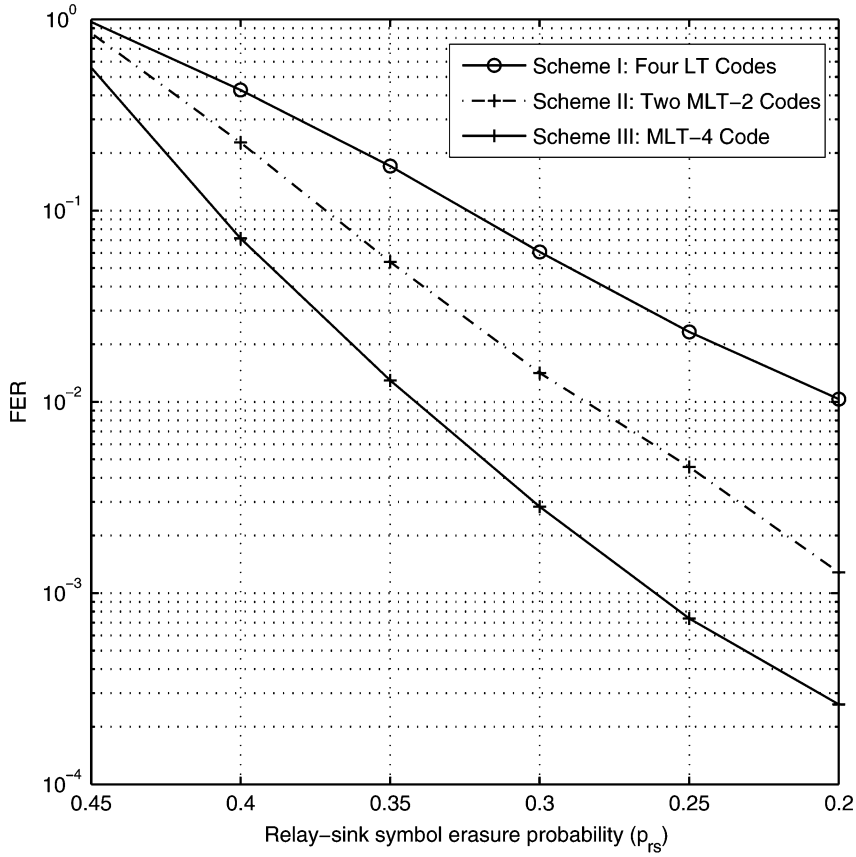

Fig. 7. FER versus relay–sink erasure probability $p_{\rm rs}$ for the LT, MLT-2, and MLT-4 coding schemes: $k = 1000$, $R = 0.5$, $n = 2000$ ($c = 0.05, \delta = 0.5$ are the parameters used for the RSD for all three codes).

TABLE I
THE RSD AND DSD FOR DEGREES $1 \leq d \leq 5$ (BOTH ARE DEFINED OVER DEGREES $1 \leq d \leq 500$; $c = 0.05$, $\delta = 0.5$)

Degree d	RSD	DSD
1	0.0156	0.6851
2	0.4552	0.1145
3	0.1541	0.0483
4	0.0782	0.0270
5	0.0476	0.0174

VI. IMPROVED DECOMPOSITION TECHNIQUE FOR LOSSY SOURCE–RELAY LINKS

This section presents an improved decomposition technique which, in contrast to the approach in Section III, specifically ensures enhanced performance for lossy source–relay channels.

Consider the situation in Fig. 1 when two DLT-2 code sequences are combined at the relay—ideally, combining them into an MLT-2 code. However, if an erasure has occurred on either of the two source–relay channels, the relay just forwards the unerased DLT-2 code symbol to the sink; as a result, the sink observes a random mixture of a DLT-2 and an MLT-2 code. While for small source–relay erasure probabilities this may be helpful (as observed in Fig. 8), for lossier channels the DLT-2 code symbols limit the overall performance. To see why this may be true, consider the significant differences in the *structure* of LT and DLT-2 codes shown in Table I, which compares the first few degrees of both the RSD and the DSD, defined over the range $1 \leq d \leq 500$.

From Table I, we observe that the DSD is dominated by the degree one term, unlike the RSD which is maximized for degree two. This is because the $d = 1$ term in the DSD must "cover" two kinds of symbols in the final MLT-2 sequence: 1) degree-one MLT-2 symbols derived from a single set of data; and 2) degree-two MLT-2 symbols derived from both sets of data, these MLT-2 symbols occurring with high probability because MLT-2 mimics the behavior of the RSD, which is maximized at $d = 2$. As a consequence, Table I indicates that symbols with a small degree greater than one (especially a degree of two) are significantly underrepresented in the DSD, vis-a-vis the RSD. This leads to a degraded performance of the resulting DSD-based code.[5]

To address this issue, we develop a new decomposition of the RSD that yields a new degree distribution to be used at the source—the *modified DSD* (MDSD). This new distribution reduces the dominance of degree-one symbols that characterized the DSD. For the sake of brevity we only consider DLT-2 codes, however, an extension to four sources and DLT-4 codes can be readily obtained as before.

A. Decomposition Approach

Our approach here will be similar to our approach in Section III-A—i.e., we will express the RSD as a mixture of two

[5]It is known for codes with an "LT-like" construction, a large fraction of degree-one symbols results in a "bad" code—cf. the "all-at-once" degree distribution in [4], for which $\Pr(\text{degree} = 1) = 1$.

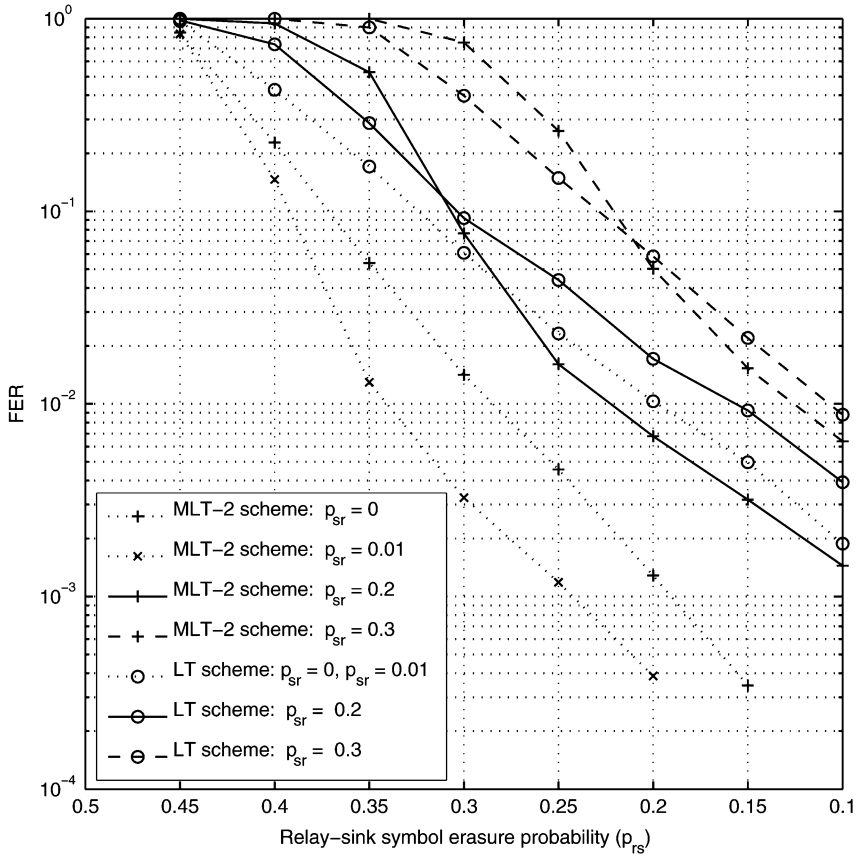

Fig. 8. FER versus relay–sink erasure probability p_{rs} for LT codes (with inner repetition code) and MLT-2 codes, under different source–relay erasure probabilities p_{sr}; $k = 500$, $R = 0.5$, $n = 1000$ ($c = 0.05$, $\delta = 0.5$ are the parameters used for the RSD for both codes).

component distributions and will deconvolve one of those component distributions, analogous to how, in Section III-A, we expressed the RSD as a mixture of $\mu'(\cdot)$ and $\mu''(\cdot)$ and deconvolved $\mu'(\cdot)$ to form $f(\cdot)$. We will then use that deconvolved distribution (as a component in another mixture) to encode the data at each source, just as before we used $f(\cdot)$ in a mixture with $\mu''(\cdot)$ to encode using the DLT-2 code.

We start our derivation by removing the $i = k/S$ "spike" from the RSD—i.e., define the distribution $\hat{\mu}(\cdot)$ as follows:

$$\hat{\mu}(i) = \begin{cases} \frac{\mu(i)}{\hat{\beta}}, & 1 \le i \le k,\ i \ne \frac{k}{S} \\ \frac{1}{\hat{\beta}} \cdot \frac{\mu(k/S-1)+\mu(k/S+1)}{2}, & i = \frac{k}{S} \end{cases} \tag{31}$$

with the normalization factor $\hat{\beta}$ given by

$$\hat{\beta} = \sum_{i \ne k/S} \mu(i) + \frac{\mu(k/S-1)+\mu(k/S+1)}{2}. \tag{32}$$

Then the RSD can be expressed as

$$\mu(i) = \hat{\beta} \cdot \hat{\mu}(i) + (1-\hat{\beta}) \cdot \delta_{\frac{k}{S}}(i), \qquad 1 \le i \le k \tag{33}$$

where $\delta_{\frac{k}{S}}(i) = 1$ for $i = k/S$ and zero otherwise.

While this does indeed express the RSD as a mixture, it is not the mixture we want, since it would suffer from some of the same shortcomings as the DSD decomposition obtained in Section III-A. Instead, we focus on expressing $\hat{\mu}(\cdot)$ itself as a mixture—a mixture motivated by the LT encoding process in which the neighbors of each code symbol are selected equiprobably from among all possibilities of appropriate degree.

So assume a degree d is chosen according to the distribution $\hat{\mu}(\cdot)$, and a degree-d LT code symbol is then formed based on k data symbols. Moreover, let us partition the k data symbols into two halves—$\{1, 2, \ldots, k/2\}$ and $\{k/2+1, k/2+2, \ldots, k\}$. We define two events—$D_1$ and D_2, respectively—to have occurred if the LT code symbol has neighbors in exactly one or exactly both (respectively) of the two halves

$$P(D_1 \,|\, d = i) = \begin{cases} \frac{2\binom{k/2}{i}}{\binom{k}{i}}, & 1 \le i \le \frac{k}{2} \\ 0, & \frac{k}{2}+1 \le i \le k \end{cases} \tag{34}$$

and

$$P(D_2 \,|\, d = i) = 1 - P(D_1 \,|\, d = i), \qquad 1 \le i \le k. \tag{35}$$

Applying Bayes' rule for $1 \le i \le k$ yields

$$P(d = i \,|\, D_1) = \frac{P(D_1 \,|\, d = i) \cdot \hat{\mu}(i)}{P(D_1)} \tag{36}$$

$$P(d = i \,|\, D_2) = \frac{P(D_2 \,|\, d = i) \cdot \hat{\mu}(i)}{P(D_2)} \tag{37}$$

where

$$P(D_1) = \sum_{i'=1}^{k} P(D_1 \,|\, d = i') \cdot \hat{\mu}(i') \tag{38}$$

$$P(D_2) = 1 - P(D_1). \tag{39}$$

(Note that $P(d = i \,|\, D_1) > 0$ if and only if $1 \le i \le k/2$ while $P(d = i \,|\, D_2) > 0$ if and only if $2 \le i \le k$.)

Define $\alpha = P(D_1)$. Then

$$\hat{\mu}(i) = \alpha \cdot P(d = i \,|\, D_1) + (1-\alpha) \cdot P(d = i \,|\, D_2). \quad (40)$$

This is the mixture of $\hat{\mu}(\cdot)$ we are seeking. By construction, the two components—$P(d=i \,|\, D_1)$ and $P(d=i \,|\, D_2)$—reflect degree distributions associated with code symbols derived from a single half and both halves of the data, respectively. They therefore naturally fit into our encoding framework in which the relay generates some symbols by combining symbols from the two sources and generates others by simply forwarding symbols from one source.

Inserting (40) into (33) yields this formulation of the RSD

$$\mu(i) = (\alpha\hat{\beta} \cdot P(d = i \,|\, D_1) + (1-\hat{\beta}) \cdot \delta_{\frac{k}{S}}(i)) + (1-\alpha)\hat{\beta} \cdot P(d = i \,|\, D_2) \quad (41)$$

which can be rewritten as

$$\mu(i) = (1-(1-\alpha)\,\hat{\beta}) \cdot p_1(i) + (1-\alpha)\,\hat{\beta} \cdot p_2(i), \quad \text{for } 1 \le i \le k \quad (42)$$

where

$$p_1(i) = \frac{\alpha\hat{\beta}}{1-\hat{\beta}+\alpha\hat{\beta}} \cdot P(d = i \,|\, D_1) + \frac{(1-\hat{\beta})}{1-\hat{\beta}+\alpha\hat{\beta}} \cdot \delta_{\frac{k}{S}}(i)$$

and

$$p_2(i) = P(d = i \,|\, D_2). \quad (43)$$

Equation (42) indicates the decomposition of the RSD we will use to carry out our distributive coding. As noted above, $p_2(\cdot)$ (by construction) reflects the degree distribution of code symbols with neighbors in *both* $\mathfrak{D}_1$ and $\mathfrak{D}_2$ (from Section III), so it is well suited for deconvolution, which results in the following distribution:

$$P_{\text{dec}}(i) = \begin{cases} \sqrt{p_2(2)}, & i = 1 \\ \frac{p_2(i+1) - \sum_{j=2}^{i-1} P_{\text{dec}}(j) \cdot P_{\text{dec}}(i+1-j)}{2 \cdot P_{\text{dec}}(1)}, & 2 \le i \le \frac{k}{2} \\ 0, & \frac{k}{2} < i \le k. \end{cases} \quad (44)$$

Once again, extensive simulations indicate that the deconvolution in (44) yields nonnegative values for $P_{\text{dec}}(\cdot)$. Therefore, analogous to the considerations in Section III-A, $P_{\text{dec}}(\cdot)$ approximates a valid probability distribution, i.e., it sums to unity asymptotically. Furthermore, the distribution $p_1(\cdot)$ in (42) represents the contribution from a single source, which in contrast to the DSD-based approach, does not contain *only* degree one and the "spike" at $i = k/S$, but also all other degrees up to $k/2$. Encoding at the sources is now carried out via the degree distribution obtained by appropriately randomizing between $p_1(\cdot)$ and $P_{\text{dec}}(i)$. The processing rule at the relay is that if code symbols from both sources are drawn according to $P_{\text{dec}}(i)$, their XOR is transmitted to the sink, otherwise, only a symbol drawn according to $p_1(\cdot)$ is transmitted.

As a final step, we combine the distributions $p_1(\cdot)$ and $P_{\text{dec}}(\cdot)$ to obtain the MDSD.

Definition 4: The modified DSD $p'(\cdot)$ is given by

$$p'(i) = \lambda' \cdot P_{\text{dec}}(i) + (1-\lambda') \cdot p_1(i), \qquad \text{for } 1 \le i \le k/2 \quad (45)$$

TABLE II
COMPARISON OF THE BEHAVIOR OF THE DSD AND MDSD FOR DEGREES $1 \le d \le 5$ (BOTH ARE DEFINED OVER DEGREES $1 \le d \le 500$; $c = 0.05$, $\delta = 0.5$)

Degree d	DSD	MDSD
1	0.6851	0.4871
2	0.1145	0.2481
3	0.0483	0.0777
4	0.0270	0.0378
5	0.0174	0.0226

with the mixing parameter λ' given by

$$\lambda' = \sqrt{(1-\alpha)\,\hat{\beta}}. \quad (46)$$

Similarly to Table I, Table II shows the values of the DSD and the MDSD for a small set of degrees. As expected, the MDSD is less dominated by the $d = 1$ term than the DSD. Moreover, Fig. 9 shows the observed CDF of the number of code symbols required to decode, when the degree distribution used to encode a single set of data is chosen to be either the RSD, DSD, or the MDSD. There is a significant reduction in the overhead when the MDSD is used relative to the DSD; however, both DSD and MDSD perform worse than the RSD.

B. Distributed Encoding Using the MDSD

The encoding is similar to that described in Section III-B. For encoding the two sources now use the MDSD instead of the DSD for the degree distribution. The two components $f(\cdot)$ and $\mu''(\cdot)$ of the DSD are now replaced by the components $P_{\text{dec}}(\cdot)$ and $p_1(\cdot)$ of the MDSD, respectively. However, unlike the DSD-based scheme, the component distributions $P_{\text{dec}}(i)$ and $p_1(i)$ are *both* nonzero over the entire range of possible degrees $1 \le i \le k/2$. Consequently, when the randomized relay protocol is used, the relay must generate a pair of random numbers for *every* pair of received source symbols in order to decide whether to XOR the two symbols or not. This leads to higher complexity than the DSD-based scheme, wherein a random number is generated only when a source's symbol has either degree one or k/S.

C. Performance of MLT-2 Codes Based on the MDSD

The MDSD is an improvement over the DSD, and hence MLT-2 codes[6] using the MDSD are expected to perform better than those that use the DSD for lossy source–relay links. We consider two different types of block MLT-2 codes (Scheme II, Sections V-B2 and V-B3), with the DSD and the MDSD as the degree distributions. The corresponding FER curves (as a function of the relay–sink erasure probability p_{rs}) are plotted in Fig. 10 for different source–relay erasure probabilities p_{sr}. For small values of p_{sr} (≈ 0.01), the MDSD essentially matches the performance of the DSD, but for larger p_{sr} (about 0.2–0.4), the MDSD gives a significant performance improvement over the DSD.

Thus far, we have assumed the two source–relay channels have the same erasure probability. A comparison between

[6]We continue to refer to the code generated at the sources (using the MDSD as the degree distribution) as a DLT-2 code, and the final code at the relay as an MLT-2 code.

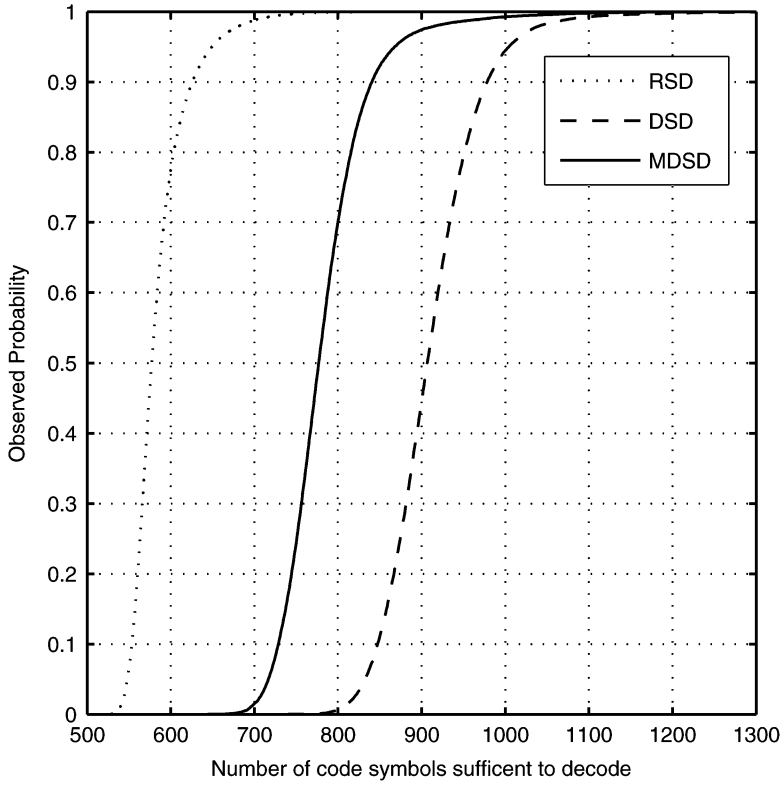

Fig. 9. Observed CDFs of the number of code symbols required to recover k symbols of data—with the RSD, DSD, and the MDSD as degree distributions: $(k, c, \delta) = (500, 0.05, 0.5)$.

MLT codes based on the DSD and the MDSD for *asymmetric* source–relay channels is shown in Fig. 11. The processing at the relay is kept symmetric—i.e., the same combining procedure used for the symmetric case is used here. The erasure probabilities between the sources s_1, s_2 and the relay r are taken to be $p_{s_1,r} = 0.2$ and $p_{s_2,r} = 0.3$. For comparison, the FER curves for the symmetric case with $p_{s_1,r} = p_{s_2,r} = 0.2$ and $p_{s_1,r} = p_{s_2,r} = 0.3$ are also included in the figure. The MDSD-based code is seen to perform better than the DSD-based code even with asymmetric source–relay channels.

Although not shown here, DSD- and MDSD-based MLT codes are seen to perform equally well when the source–relay channels are lossless.

VII. Summary and Discussion

We have introduced new distributed rateless codes (DLT codes) which enable multiple (two or four) sources to encode their information independently and then form a combined code sequence at a common relay with minimal complexity, via a "network-coding-like" operation involving the selective XOR of code symbols. With perfect source–relay channels, the resulting MLT code symbols resemble a long LT code covering all the sources. These codes offer significant performance gains over the use of individual LT codes at the sources (with mere routing at the relay) for protection against erasures on the relay–destination channel. The gains also extend to lossy source–relay channels for small erasure probabilities. The primary advantage of MLT codes is gained from encoding over longer information blocks.

The approaches presented in this paper serve to open up some interesting avenues for further research. Specifically, the following issues naturally arise from the work described here.

- Why choose the RSD as the "target" distribution for the code symbols delivered to the sink? The RSD was shown in [4] to be an appropriate distribution for d when each code symbol is the XOR of d source symbol "neighbors"—and the set of neighbors is selected equiprobably from among all $\binom{k}{d}$ possibilities. For the distributed coding problem, the set of d neighbors associated with each code symbol delivered to the sink is *not* selected equiprobably from among all $\binom{k}{d}$ possibilities, and therefore it is not obvious that the RSD is the best choice for the target distribution. It may well be that, if one were to start from first principles, one could derive an optimal target distribution for the distributed problem—one that takes into account the fact that each encoder at each source has access to only a subset of the data. Such an approach would presumably outperform the more *ad hoc* techniques developed in this paper. Still, our approach—"borrowing" the RSD from [4]—*does* benefit from longer block lengths and thus yields gain compared with a time-division multiple access (TDMA)-based solution using conventional LT codes. Furthermore, some of the techniques used in this paper, such as the "divide-and-conquer" method for deconvolution (to tackle degree one symbols) and the randomized decision protocol used at the relay (which helps in recursively constructing MLT-4 codes) would presumably also find application in a more general approach.
- This paper does not address the design of MLT codes taking into account the erasure probabilities for the source–relay links, especially when these are asymmetric. Indeed, when the channels are *highly* asymmetric, it is no longer clear if joint encoding of the two sources' data

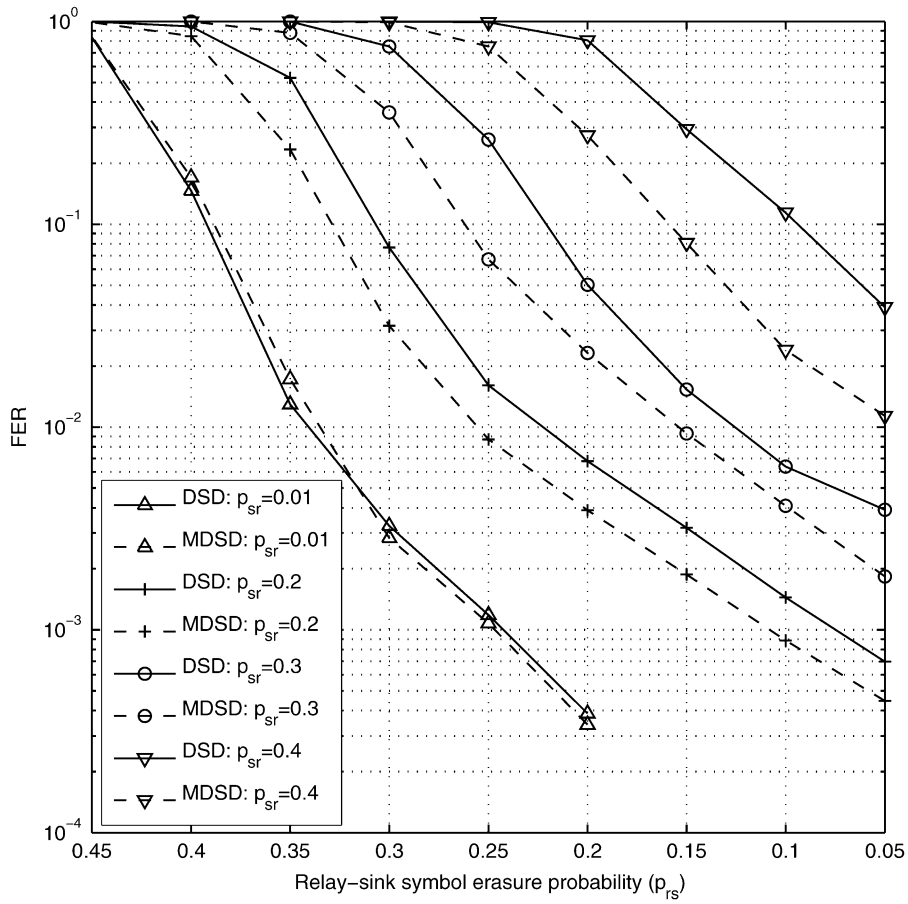

Fig. 10. FER versus relay–sink erasure probability $p_{\rm rs}$ for MLT-2 codes based on DSD and MDSD, under different source–relay erasure probabilities $p_{\rm sr}$; $k = 500$, $R = 0.5$, $n = 1000$ ($c = 0.05, \delta = 0.5$ are the parameters used for the RSD for all cases).

is beneficial. Nevertheless, although beyond the scope of this paper, the characterization of coding schemes for the asymmetric regime constitutes an interesting topic for further study.

- This paper considers MLT codes for two and four sources, and benefits are observed in going from two to four. A natural question: Can this design approach be extended to even larger powers of two? Although not presented here, an extension to eight or more sources is possible via similar techniques. However, this is accompanied by an increased performance gap between the parent LT code and the derived MLT code. This is presumably because, as the number M of distributed sources increases, there is an increasing "mismatch" between the resulting distributed coding problem and the original (nondistributed) problem formulated by Luby in [4]—and it was the problem in [4] that motivated the RSD that is used as the target distribution in this paper. Thus, effectively going to large values of M may require the derivation of a new target distribution based on first principles, referred to above. In addition, MLT-M codes require an M-fold increase in the transmission rate at each of the sources. Thus, MLT-M codes are primarily beneficial in practice when the relay–sink channel is the main bottleneck in the network relative to the source–relay traffic, and this limitation becomes even more constraining for large values of M. An extreme case, however, is when $M = k$—in other words, each source transmits a *single* symbol (packet); in this case, the relay can directly compute LT code symbols from the incoming symbols according to the RSD.
- A related question: What happens when the number of sources M is not a power of two? The same basic approach—e.g., inverting a threefold convolution or a fivefold convolution—may be applied, but the processing at the relay becomes more delicate. For instance, for the $M = 2$ case, when each source produces a symbol of degree at least one, then the relay cannot produce a symbol of degree less than two with simple XORing, and so some additional processing is required to deliver degree-one symbols to the sink; however, for the $M = 5$ case, the relay cannot produce a symbol of degree less than *five* with simple XORing. When M is a power of two, the required "surgery" at the relay may be performed recursively using $M = 2$ codes, but this is not possible when M is not a power of two. Moreover, the nonnegativity of the deconvolved distribution—already only conjectured for $M = 2$—becomes even more problematic for other values of M.
- In this paper, the processing permitted at the relay has purposely been kept very simple—i.e., only selective XOR operations and a memory (buffer) that can hold at most one packet per source. If there are no such restrictions on the relay, then clearly, the optimal strategy at the relay would

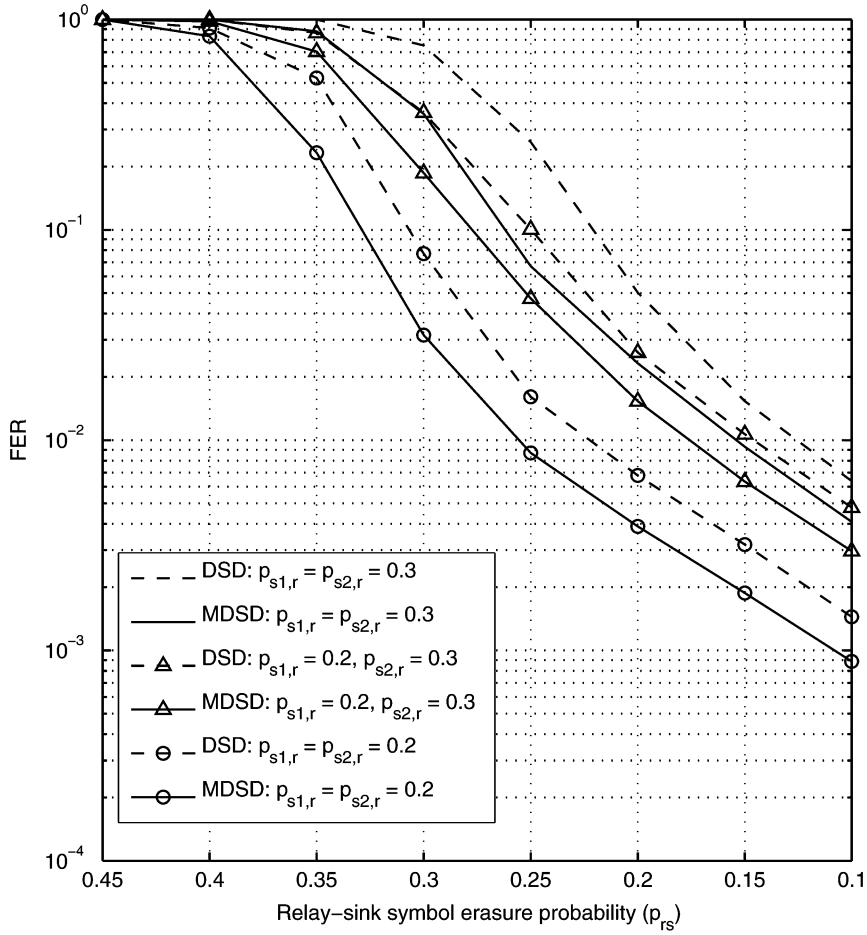

Fig. 11. FER versus relay–sink erasure probability $p_{\rm rs}$ plots for MLT codes (based on the DSD and the MDSD) with asymmetric source–relay channels; $k = 500$, $R = 0.5$, $n = 1000$ ($c = 0.05, \delta = 0.5$ are the parameters used for the RSD for both codes).

be to gather all the sources' information and encode them into a single LT code. An interesting "intermediate" case is when the relay has a buffer of *more* than one symbol per source. Now, the relay can combine DLT symbols from the past in constructing MLT symbols—specifically, in constructing a degree-i MLT symbol, the relay can ensure that its neighbors are distributed more "uniformly" from among the sources' data, thus reproducing an LT code more accurately. This would also call for a different decomposition of the RSD. However, a theoretical analysis of the case of other buffer sizes (with low-complexity combining at the relay) is beyond the scope of this paper.

- Finally, we note that MLT codes are not only useful in single-sink networks with multiple parallel unicast sessions (as in Figs. 1 or 4), but also in a multisource multisink network with parallel multicast sessions via a common relay. In this case, a subset of M sources could each encode their information with a DLT-M code, such that the relay can then generate MLT-M code symbols that are subsequently communicated to a subset of N sinks over (symbol) erasure channels. Furthermore, the *same* DLT-M code symbol from a particular source s can be used to construct multiple MLT-M code symbols (intended for different sets of sinks) for covering different sets of M sources that include s. These schemes entail minimal complexity at the relay and offer performance benefits over the use of individual LT codes for each multicast session.

REFERENCES

[1] P. Elias, "Coding for two noisy channels," in *Proc. 3rd London Symp. Information Theory*, London, U.K., 1956, pp. 61–76, Academic Press.
[2] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proc. of ACM SIGCOMM*, Vancouver, BC, Canada, Sep. 1998, pp. 56–67.
[3] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1528–1540, Aug. 2002.
[4] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science*, Vancouver, BC, Canada, Nov. 2002, pp. 271–280.
[5] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
[6] S. Puducheri, J. Kliewer, and T. E. Fuja, "Distributed LT codes," in *Proc. IEEE Int. Symp. Information Theory*, Seattle, WA, Jul. 2006, pp. 987–991.
[7] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.
[8] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.
[9] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.
[10] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. IT-8, no. 1, pp. 21–28, Jan. 1962.
[11] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2003.