Fountain Codes with Nonuniform Selection Distributions through Feedback

Morteza Hashemi, *Student Member, IEEE*, Yuval Cassuto, *Senior Member, IEEE*, and Ari Trachtenberg, *Senior Member, IEEE*

***Abstract*—One key requirement for fountain (rateless) coding schemes is to achieve a high *intermediate* symbol recovery rate. Recent coding schemes have incorporated the use of a feedback channel to improve intermediate performance of traditional rateless codes; however, these codes with feedback are designed based on *uniformly at random* selection of input symbols. In this paper, on the other hand, we develop feedback-based fountain codes with dynamically-adjusted nonuniform symbol selection distributions, and show that this characteristic can enhance the intermediate decoding rate. We provide an analysis of our codes, including bounds on computational complexity and failure probability for a maximum likelihood decoder; the latter are tighter than bounds known for classical rateless codes. Through numerical simulations, we also show that feedback information paired with a nonuniform selection distribution can highly improve the symbol recovery rate, and that the amount of feedback sent can be tuned to the specific transmission properties of a given feedback channel.**

***Index Terms*—Fountain codes, Feedback channel, LT codes, Nonuniform symbol selection**

I. Introduction

Reliable communication over erasure channels has emerged as a key technology for various networked applications, for example digital video broadcasting and over-the-air software updates. In applications where there exists a high-throughput feedback channel, automatic repeat request (ARQ) protocols guarantee reliability over erasure channels. However, when such feedback channels are not available, rateless codes, such as the capacity achieving Luby-Transform (LT) [3] and Raptor codes [4], can often provide reliable communication for sufficiently long block lengths. These codes have a well-known all-or-nothing decoding property (the so-called "waterfall" phenomenon), where a jump in the fraction of decoded input symbols occurs near the very end of the decoding process. For applications with real-time requirements, however, it is desirable to be able to recover symbols as decoding proceeds, i.e., to achieve a high intermediate symbol recovery rate.

In fact, the intermediate performance of classical fountain codes can be improved by incorporating the use of a feedback channel. For instance, a decoder in Real-Time (RT) oblivious [5] and Shifted-LT (SLT) [6] codes sends the number of recovered symbols back to the transmitter, and this feedback is used to modify the degree distribution at the encoder. Previous feedback-based rateless codes are mostly based on adjusting the degree of encoding symbols, e.g., by shifting the degree distribution in the SLT codes. However, after a degree d is picked for an encoding symbol, d input symbols are chosen *uniformly at random* and xored to form the symbol. Moreover, the receiver does not have full freedom in controlling the number of feedbacks transmitted.

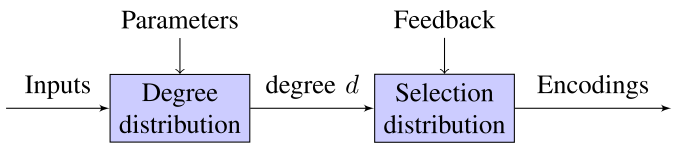


Fig. 1. Two-step rateless encoder with a degree distribution and nonuniform symbol selection distribution.

In this paper, we develop a class of rateless coding schemes that optimize for high intermediate symbol recovery rate. At its core, our encoder uses a nonuniform selection distribution that is dynamically adjusted based on feedback information. Fig. 1 depicts a schematic of our two-step encoder, where we illustrate that the inputs are chosen according to a feedback-based selection distribution, rather than uniformly at random. Feedback messages contain information on the distance between a received encoding symbol and the set of already decoded symbols at the receiver. In the *general form* of our codes, the encoder estimates the probability that each input symbol has been decoded (at the receiver), and these estimates are then used to dynamically tune the selection of input symbols within subsequent transmissions.

In fountain codes with a uniform selection distribution, all input symbols are *stochastically equivalent*, while with our nonuniform selection distribution input symbols are weighted based on the decoding progress, where we gracefully bias the distribution by changing selection probabilities. As a result, our biasing of the selection distribution is "soft", and similarly to the uniform family of fountain codes it admits a simple randomized encoding. If there is no feedback, the biasing scheme results in pure fountain codes with a uniform selection distribution, and thus our method is similarly robust to cases where there are additional receivers or imperfect feedback. On the other hand, if a feedback channel is available, the encoder can naturally track the decoding progress and generate encoding symbols that result in a faster decoding rate compared with a uniform selection of input symbols. Therefore, even

M. Hashemi and A. Trachtenberg are with the Department of Electrical and Computer Engineering at Boston University. Y. Cassuto is with the Department of Electrical Engineering, Technion, Israel Institute of Technology.

This paper was presented in part at the 51st and 52nd Annual Allerton Conference on Communication, Control, and Computing [1, 2]. This work was supported, in part, by the National Science Foundation under grant CNS-1012910. The work of Yuval Cassuto was supported in part by the Israel Science Foundation and by the German-Israeli Foundation.

though the performance may depend on erasure probability, the nonuniform selection method results in an improvement over the corresponding fountain code with uniform distribution. In this context, the *general form* of our codes is suitable for the scenarios with relatively large feedback budgets, although we give the receiver freedom as to when feedback transmissions occur (within the budget).

On the other hand, the *primitive form* of our code is designed based on a parsimonious use of the feedback channel. In this case, the encoder learns which symbols have been decoded, and those symbols will be assigned with a selection probability of zero for subsequent encodings. This coding scheme is suitable for applications with limited feedback capacity such as satellite networks [7], as we require the decoder to opportunistically send just one bit of feedback when certain conditions are met. Note that the coding schemes proposed in this work are presented as enhancements of LT codes for the case some feedback communication is available. The motive to base our codes on the LT degree distributions is to accommodate cases where feedback is extremely limited or completely unavailable, in which case we fall back to the standard LT performance. That said, the same methodology can apply to different rateless codes in the literature, and to others to be proposed in the future. In summary, the contributions of our work are as follows:

- We propose the use of distance-type feedback messages with rateless codes. Distance feedbacks provide implicit information such that the encoder learns about the state of individual symbols through feedback messages.
- We establish classes of rateless codes with distance feedback in which the decoder has full control over the amount of feedback sent. In this case, performance of the underlying codes is controlled based on the available feedback budget.
- We demonstrate that the distance-type feedback can be used to introduce nonuniform selection distributions into classical rateless codes that are mostly based on a uniformly at random selection of symbols at the encoder. The nonuniform selection distributions are tuned on-the-fly, which results in a significant improvement of the intermediate performance.

A. Organization

The rest of this paper is organized as follows. In Section II we define the problem setup and review various related coding schemes. Section III describes the most general form of our coding scheme. Section IV presents the primitive form of our codes adapted for constrained feedback applications, and a discussion on feedback overhead of the proposed coding schemes. Coding analysis for short block lengths is presented in Section V, followed by the decoder analysis in Section VI. Simulation results are presented in Section VII. We conclude with overall thoughts in Section VIII.

II. Background

This section describes the problem setup and some previous work on rateless coding.

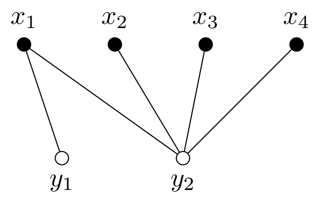


Fig. 2. Bipartite graph representation of fountain codes. In this example, x_1 is the only neighbor of y_1, and y_2 has x_1, x_2, x_3, and x_4 as its neighbor.

A. Preliminaries

In the rateless coding setup, it is assumed that an encoder (broadcaster) has k input symbols to transmit to all receivers over an erasure channel, and that there may exist a feedback channel through which receivers can send some information back to the encoder. Luby Transform (LT) rateless codes [3], as the first practical realization of fountain codes, support full recovery of k input symbols using an expected number of $k + O\left(\sqrt{k}\ln^2(k/\delta)\right)$ error-free transmissions with a given recovery failure probability δ. To generate an output symbol, the encoder first picks a coding degree d according to the Robust Soliton distribution [3]. Next, d input symbols are chosen uniformly at random without replacement, and their sum over an appropriate finite field forms the output symbol. Indices of the d selected input symbols, referred to as *neighbors* of the output symbol, are made available (i.e., as meta-information) to the decoder. For instance, Fig. 2 graphically illustrates a sample construction of encoding symbols wherein $y_1 = x_1$ and $y_2 = x_1 + x_2 + x_3 + x_4$, and thus y_1 has x_1 as its neighbor and y_2 has x_1, x_2, x_3, and x_4 as its neighbors. In total, the coding operations incur the computational cost of $O\left(k \ln(k/\delta)\right)$.

The LT decoder (so-called peeling decoder) uses a simple message passing algorithm, with a complexity typically less than traditional Gaussian elimination methods. In one variant, the decoder finds all encoding symbols with degree 1, whose neighbor can be immediately recovered. These recovered input symbols are then excluded from all output symbols that have them as neighbors, reducing the number of unknowns in those encoding symbols by one. This process continues until there exists no encoding symbol with degree 1. Decoding succeeds if all input symbols are recovered; alternatively, decoding fails if, at some point, there is no output symbol with degree 1.

B. Related work

Both fixed rate low-density parity-check (LDPC) codes [8] and Turbo codes [9] are capable of correcting bit errors, as well as erasures. Byers *et al.* in [10] have presented fixed rate Tornado codes as a class of simplified capacity-achieving LDPC codes. Within the context of rateless coding, random linear codes (see, for example, [11]) are well known due to their low communication overhead, but the encoding and decoding computations make them practical only for small message sizes. On the other hand, Luby Transform (LT) [3] codes and their extensions such as Raptor codes [4] are examples of rateless codes that are asymptotically optimal and also have computationally efficient encoding and decoding algorithms;

unfortunately, they usually have poor performance for small block sizes [12], and low intermediate decoding rate for general block sizes. Various optimization methods (e.g., [13]) have been proposed for these cases. In this context, we aim to enhance the performance of rateless codes using the feedback channel and based on three main components compared with previous works: (i) *feedback variations:* proposing distance-type feedback in order to learn the state of individual symbols at the decoder, (ii) *controllable feedback usage:* adding an additional coding flexibility (i.e., feedback interval s) to have full control over the amount of feedback sent, and (iii) *nonuniform coding:* enhancing the intermediate decoding rate through a dynamically-adjusted nonuniform selection distribution.

Feedback variations: there have been proposed rateless protocols that utilize side information fed back from the decoder to the encoder. Based on the type of feedback used, they can be divided in the following categories:

- the receiver sends the *number* of decoded symbols to the transmitter;
- the receiver suggests to the transmitter *what kind of degrees* it should use for future encodings; or
- the receiver notifies the transmitter of *which* input symbols have been recovered.

In the Real-Time (RT) oblivious codes [5], the encoder starts with degree one symbols, and it increments the degree of encoding symbols based on feedback messages. In this case, feedbacks contain information on the number of recovered symbols. Shifted LT (SLT) codes proposed in [6] use the same type of feedback information as the RT codes, but instead of explicitly increasing the encoded symbols degree, the encoder shifts the Robust Soliton degree distribution. There also exist rateless-type codes with real-time properties that allow intermediate knowledge of some input symbols as the decoding progresses. The authors in [14] propose Growth codes for the data collection within lossy sensor networks. Similar to the RT and SLT codes, feedback messages in Growth codes contain information on the number of decoded symbols, and degree of output symbols increases as the coding progresses.

As another type of feedback, the receiver in [15] has the ability to control the decoding progress by requesting particular degrees. In this method, the average number of output symbols required for decoding k input symbols is shown to be upper bounded by $1.236k$. Yet another type of feedback in [12] contains the identity of recovered symbols, which are used by the encoder to redesign the degree distribution for subsequent transmissions. Recently, the authors in [16] have proposed a heuristic to use a hybrid feedback-based rateless codes, called LT-AF, in which the receiver alternates between two types of feedback messages: the first type of feedback contains the number of decoded symbols as in the SLT and RT codes, while the receiver requests a specific input symbol through the second type of feedback, the same as in [17]. The type of feedback used in this paper is based on distance information by which the encoder learns about the state of individual symbols at the decoder. On the one hand, the distance feedback allows better adjustment of the encoding compared to schemes only feeding back the number of decoded symbols. On the other hand, the cost of feeding back distance information is shown to be smaller than the cost of feeding back the exact identities of decoded symbols.

Controllable feedback usage: in most of the previous works, the decoder does not have full control over the amount of feedback sent. For instance, the Shifted LT codes in [6] uses a non-uniform restriction heuristic to limit the total amount of feedback to $O(\sqrt{k})$ transmissions, the same as RT codes. The LT-AF codes require equal or smaller number of feedbacks compared to SLT codes, but still the decoder does not control the total amount of feedback sent. The decoder in [12] has more control over the number of feedback messages, but still a general method that enables the decoder to tune the feedback transmission rate is missing. Within this context, we enhance the decoder to control the total amount of feedback sent according to the feedback budget available.

Improved intermediate performance with nonuniform coding: in some applications, like video streaming, intermediate symbol recovery is important, as it is desirable to decode some symbols before an entire frame has been received. Intermediate performance of rateless codes was initially investigated in [18], wherein the author provides an upper-bound on the fraction of input symbols that can be recovered by rateless coding schemes. In that paper, asymptotic degree distributions are introduced to achieve an optimal intermediate decoding performance. In order to make the asymptotic distributions practical, Talari *et al.* in [19] use Pareto optimization to design an optimal degree distribution for high intermediate symbol recovery rates. In [20], the algorithm of Optimal Partial Decoder (OPD) is introduced that maximizes the intermediate performance of any rateless code.

In the context of nonuniform rateless coding, the authors in [21] propose the Unequal Error Protection (UEP) codes in which the nonuniform selection distribution is set a priori (i.e., offline) and based on importance of the input symbols. Similarly, [22] proposes a joint unequal loss protection and LT coding (ULP-LT) scheme for layered media delivery. Our work can be viewed as a bridge between these two categories in that the intermediate performance is indeed enhanced using a nonuniform selection distribution. Moreover, note that although there have been previous fountain codes with nonuniform distributions, our codes offer a systematic and flexible method to adjust the selection distribution with a clear tradeoff with feedback consumption, as the receiver has full control over the amount of feedback sent. In this work, we regard all symbols to be equally important and the nonuniform selection distribution is dynamically adjusted to result in a faster decoding rate. Our method can potentially be extended to accommodate UEP differentiation between input symbols.

III. Nonuniform Rateless Codes: General Form

Previous rateless codes with feedback are mostly designed based on modifying the output symbols degree distribution according to feedback information, e.g., by shifting the degree distribution, or by explicitly increasing the degree. In these schemes, when a coding degree d is picked, d input symbols are selected uniformly at random to construct an encoding symbol. Moreover, in most of previous works, feedback information does not provide a complete picture of the decoding state at the receiver side. Within this context, we present a nonuniform rateless coding scheme wherein various input symbols are selected based upon a nonuniform distribution. In particular, the selection distribution is tuned according to feedback messages, which contain the distance of received symbols to the set of already recovered symbols at the receiver. The definition of distance quantity is as follows:

Definition. *Given a set of recovered symbols $\mathcal{C}$ and an encoding symbol y that has a set of neighbors $\mathcal{A}$, the* ***distance*** *between y and $\mathcal{C}$ is defined as:*

$$dist(y, \mathcal{C}) = \sum_{x_i \in \mathcal{A}} \mathbb{1}_{x_i \notin \mathcal{C}},$$

where $\mathbb{1}_x$ is an indicator function that is equal to 1 if and only if x is true.

The distance quantity simply corresponds to the number of neighbors of y that are not already decoded. As an example, suppose that input symbols x_1, .., x_4 are encoded and transmitted in the following order: $y_1 = x_1 + x_2$, $y_2 = x_1 + x_4$, $y_3 = x_4$, and $y_4 = x_1 + x_2$[1]. The distance from either y_1 or y_2 to the set of recovered symbols is 2; thereafter, from y_3 the distance is 1, and finally, from y_4 the distance is 0 (as x_1 and x_2 will be decoded after receiving y_3). This example demonstrates that distance information may translate to explicit information about the state of decoder such that a distance 0 happens if and only if all neighbors of the received encoding symbol have already been decoded. Similarly, a distance 1 occurs in the case that there is only a single undecoded neighbor, which can then be recovered uniquely.

Ultimately, the goal of the encoder is to generate encoding symbols based on the state of the decoder in such a way that more "helpful" symbols have a higher selection probability. To this end, the encoder uses distance information to estimate the probability that each input symbol has been decoded (at the receiver), and these estimates are used to bias the selection of input symbols. In this paper, q_j denotes the probability that input symbol j has been decoded, and p_j is the probability that symbol j is included in the next encoding. Values of q_j and p_j are adapted on-the-fly and based on distance feedback information. In this approach, the receiver can further adjust the number of feedbacks using a parameter s so that one feedback transmission follows after every s received encoding symbols. The parameter s can be set to any arbitrary value, depending on the feedback channel available.

A. Processing distance information

In order to process distance information, the encoder constructs a bipartite graph wherein input symbols are placed on the top and encoding symbols at the bottom, as shown in Fig. 3. In this graph, *labels* are assigned to input and output symbols. Label of an input symbol corresponds to its probability of having been decoded, while label of an output

[1]For the sake of clarity, we assume that encoding and decoding are performed over the field $\mathbb{F}_2$.

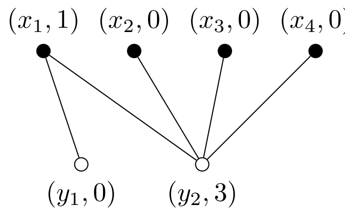


Fig. 3. Distance graph labeling: A label (x_i, q_i) implies that the input symbol x_i has been decoded with probability q_i up to the current state. Labels of output nodes y_i are defined to be the number of neighbors of y_i with a label of less than 1.

symbol y represents the number of neighbors of y with label less than 1. For instance, assume that after t feedbacks, n_t neighbors of y are labeled 1 (i.e, they have been decoded). Therefore, the label of y, denoted by l_t, is calculated as:

$$l_t = d - n_t; \tag{1}$$

where d is the degree of y. In this equation, the encoder excludes the recovered neighbors from the labeling process. Next, in order to calculate the label of an input symbol, we assume that the t-th feedback message contains the distance f_t corresponding to the encoding symbol $y = \sum_{j \in \mathcal{A}} x_j$. The label of a constituent symbol x_j is then defined as:

$$q_{j,t} = \max\left\{ q_{j,t-1}, \frac{\binom{l_t - 1}{f_t}}{\binom{l_t}{f_t}} \right\} = \max\left\{ q_{j,t-1}, \frac{l_t - f_t}{l_t} \right\}. \tag{2}$$

The arguments of the max term are justified as follows: label $q_{j,t}$ is the probability that input symbol j has been decoded conditioned that the encoding symbol y has distance f_t from the set of decoded symbols $\mathcal{C}$ (at the receiver side). Therefore, $q_{j,t}$ is updated based on the following rule:

$$\Pr\Big(x_j \text{ is decoded} \,\Big|\, dist(y, \mathcal{C}) = f_t\Big) = \frac{\Pr\Big(x_j \text{ is decoded and } dist(y, \mathcal{C}) = f_t\Big)}{\Pr\Big(dist(y, \mathcal{C}) = f_t\Big)} = \frac{\binom{l_t - 1}{f_t}}{\binom{l_t}{f_t}}. \tag{3}$$

This probability can be further simplified as $\frac{l_t - f_t}{l_t}$. Finally, after receiving a new feedback message, $q_{j,t}$ is updated to the maximum of its previous value (i.e., $q_{j,t-1}$) and the calculated probability at the current step based on (3). For instance, assume that the encoding symbol $y = x_1 + x_2 + x_3 + x_4$ has a distance of 2 with the current state of decoder, meaning that two neighbors of y have not been decoded yet (the encoder does not know which two symbols). If the encoder has already assigned label 1 to x_1 (i.e., x_1 has been decoded), then the encoder uniformly divides the distance of 2 between the remaining symbols (i.e., x_2, x_3, and x_4), suggesting that each of them has been decoded with probability $\frac{3-2}{3} = \frac{1}{3}$. It should be noted that the subscript t in $q_{j,t}$ represents the evolution of q_j as the coding proceeds, and for the sake of simplicity, we omit it in our discussion.

Our labeling process tracks the state of the decoder by answering this question: *what is the probability that an individual symbol x_j has been decoded up to this point?* As an example, Fig. 4 shows a realization of input symbols at the encoder,

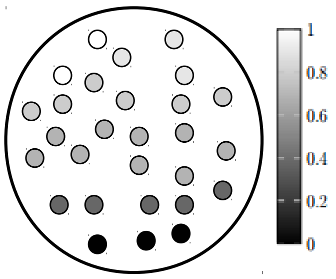


Fig. 4. The encoder estimates the probability of having been decoded (i.e., q_j's) for input symbols. Probability 1 (white color) implies that the symbol has been decoded, while probability 0 (black color) shows that the symbol has not been decoded yet.

where input symbols are assigned with a probability of having been decoded. In this case, $q_j = 1$ (white color) implies that symbol j has been recovered, while $q_j = 0$ (black color) means that symbol j has not been recovered yet. Therefore, input symbols are assigned with a weight between 0 and 1, which is used in the selection distributions defined in Section III-B.

To examine the accuracy of the estimated probability values against the actual decoder state, we use the Mean Absolute Error (MAE) quantity. Assume that b_j is an indicator function representing the state of symbol j at the decoder such that $b_j = 1$ if symbol j has been decoded and $b_j = 0$ otherwise. MAE is then calculated as:

$$\mathrm{MAE} = \frac{1}{k}\sum_{j=1}^{k} |q_j - b_j|, \tag{4}$$

in which q_j's are estimated using (2). Fig. 5 shows the simulation results of the MAE quantity averaged over all feedback messages transmitted as the interval of feedback transmission (i.e., parameter s) increases. The results illustrate that decreasing the interval of feedback transmission (i.e., higher feedback rate) decreases the estimation error. Next, we define nonuniform symbol selection distributions based on probability values q_j's.

B. Nonuniform symbol selection

We discussed that the encoder uses distance information to learn about the state of decoder. For the sake of concreteness, assume that the encoder estimates the input symbol x_j has been decoded with probability q_j, and it has probability p_j to be included in the next encoding symbol with degree d. We aim to design a symbol selection distribution that picks those d input symbols that can achieve a maximum decoding progress. In other words, we are interested in devising a relation between probability value q_j's (estimated by the encoder and based on feedback information) and selection probabilities p_j's. The main goal of a nonuniform selection distribution is to achieve a faster decoding rate compared with a uniform symbol selection rule. To this end, the nonuniform selection distribution should select $d-1$ symbols which have been recovered with a high probability, and a single symbol that with a high probability has not been recovered. This selection rule results in decoding

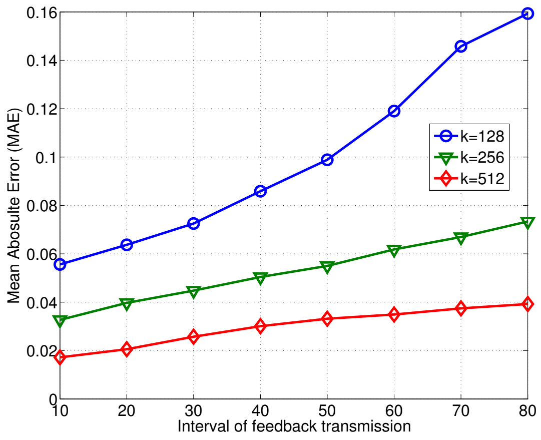


Fig. 5. Simulation results of the Mean Absolute Error (MAE) quantity for the probability of having been decoded as the interval of feedback transmission (parameter s) increases.

a single input symbol with a high probability. To put in a formal framework, we have the following definition.

Definition. *For a given input symbol x and a set of $d-1$ input symbols $\mathcal{A}$, the Decoding Probability function $DP(x,\mathcal{A})$ is defined as the probability of immediate decoding the input symbol x after receiving $y = x + \sum_{i\in\mathcal{A}} x_i$.*

In order to decode an input symbol x_j within a transmission, the transmitted symbol with degree d should include the undecoded symbol x_j and $d-1$ already decoded symbols. Symbol x_j is not decoded with probability $1-q_j$, and $d-1$ symbols belonging to the set $\mathcal{A}$ have already been decoded with probability $\prod_{i\in\mathcal{A}} q_i$. Therefore, at each step to transmit a symbol of degree d, the encoder should choose d input symbols $\left(x_j^*, \mathcal{A}^*\right)$ satisfying:

$$\left(x_j^*, \mathcal{A}^*\right) = \underset{(x_j,\mathcal{A})}{\arg\max}\ DP(x_j,\mathcal{A}) = \underset{(x_j,\mathcal{A})}{\arg\max}\,(1-q_j)\prod_{\substack{i\in\mathcal{A}\\ i\neq j}} q_i. \tag{5}$$

The solution of this optimization problem is deterministic; in other words, the encoder always picks $d-1$ symbols with the largest value of q xored with a single symbol with the smallest value of q. However, it is desirable to preserve the same behavior with a probabilistic scheme such that if an input symbol j is included in the solution of the deterministic formulation, it would also have a high probability to be picked by the probabilistic method. This results in the following scheme to define the selection probability p_j:

$$p_j \propto \begin{cases} 1-q_j & \text{if } \ 0 \leqslant q_j < \frac{1}{2}; \\ q_j & \text{otherwise.} \end{cases}$$

In the second step of designing the selection distribution, we note that a single unrecovered (with high probability) symbol should be included within each encoding symbol. Therefore, based on the value of q_j's, input symbols are divided into two subsets: $\mathcal{U}$ containing undecoded symbols, and $\mathcal{D}$ containing decoded symbols. Input symbols with $0 \leqslant q < \frac{1}{2}$ are included in $\mathcal{U}$ and the rest are added to the set $\mathcal{D}$, and thus we construct an encoding symbol of degree d by selecting one symbol from $\mathcal{U}$ based on the selection distribution $P_{\mathcal{U}}$, and $d-1$

symbols from $\mathcal{D}$ according to the distribution $P_{\mathcal{D}}$. Selection distributions $P_{\mathcal{U}}$ and $P_{\mathcal{D}}$ are defined as follows:

$$P_{\mathcal{U}}(j) = \begin{cases} \frac{1-q_j}{\sum_{i=1}^{k-D} 1-q_i} & \text{if } j \in \mathcal{U}; \\ 0 & \text{otherwise.} \end{cases}$$

$$P_{\mathcal{D}}(j) = \begin{cases} \frac{q_j}{\sum_{i=1}^{D} q_i} & \text{if } j \in \mathcal{D}; \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

In the distributions, D is the size of subset $\mathcal{D}$, which results in $U = k - D$ as the size of undecoded symbols. Finally, the encoder transmits the xor of d selected symbols.

This scheme based on the distributions $P_{\mathcal{D}}$ and $P_{\mathcal{U}}$ is refereed to as the *All-Distance* codes since all distance feedbacks are needed to estimate probability value q_j's. Next, we relax this constraint in a way that, instead of sending all distance values, the decoder *quantizes* distance values and allocates only a single bit of feedback for each received encoding symbol.

C. Quantized distance codes

The All-Distance codes work based on estimating probability values q_j's from distance information. In the case of sending distance of n received symbols, the decoder on average needs to transmit $\bar{f} = \mathbb{E}_d[n \log(d)]$ bits of feedback, in which d is the degree of an encoding symbol. This expression can be simplified as follows:

$$\bar{f} = \mathbb{E}_d[n \log(d)] \leq n \log(\mathbb{E}[d]) \leq n \log\big(\log(k)\big), \tag{7}$$

in which, the first upper-bound is a result of Jensen's inequality, and the second inequality is based on the average degree of encoding symbols under the Robust Soliton distribution [3]. For low-overhead coding schemes, we have $n = (1+\epsilon)k$, and thus the *expected* number of bits of feedback needed by the all-distance codes is upper-bounded by $(1+\epsilon)k \log\big(\log(k)\big)$.

In order to further reduce amount of feedback, we consider a scheme with a single bit feedback per received encoding symbol. In particular, this scheme is based on the same idea of splitting input symbols into two subsets; however, instead of having an exact estimation of probability value q_j's, the decoder decides to send a feedback 0 or 1 based on the distance of a received symbol. More precisely, the decoder calculates the ratio of distance to degree for a received symbol, and if the ratio is larger than $\frac{1}{2}$, it implies that majority of neighbors within the received encoding symbol have not been recovered. In this case, the decoder allocates a single bit of 0 as the feedback message. On the other hand, if the calculated ratio is smaller than $\frac{1}{2}$, it shows that majority of neighbors have been decoded and feedback message would be 1. To limit the number of feedback transmissions, the receiver bundles the one-bit feedback messages together for every interval of s received encoding symbols, and sends the s-bit messages to the encoder.

At the encoder side and upon receiving a bit of 0 as feedback, corresponding neighbors are assigned with $q_j = 0$ and thus added to the subset $\mathcal{U}$. Conversely, if the received feedback contains a bit of 1, corresponding neighbors are assigned with $q_j = 1$ and grouped into $\mathcal{D}$. This *quantized* version of q_j is equivalent to evaluating $\lfloor q_j \rceil$ in (6) ($\lfloor x \rceil$ rounds x to its nearest integer). As a result, the $P_{\mathcal{U}}$ and $P_{\mathcal{D}}$ distributions would become uniformly distributed over the subsets $\mathcal{D}$ and $\mathcal{U}$ respectively. However, it should be noted that with a high probability only a single undecoded symbol is included within each transmission. Hence, splitting a single uniform distribution defined over all input symbols (as it has been used in previous rateless codes) into two disjoint uniform selection distributions can significantly improve the intermediate performance of rateless codes. In terms of the total amount of feedback, receiver sends exactly one bit feedback per received encoding symbol, where the total number of encoding symbols is $(1+\epsilon)k$ for a small value of ϵ.

IV. Nonuniform Rateless Codes: Primitive Form

In most of communication systems, a nominal utilization of the back channel is desirable as the bandwidth is mainly provisioned for forward transmissions. In the previous section, we presented a nonuniform coding scheme based on distance feedbacks, wherein all distance information are fed back to the encoder. In the scheme based on quantized distance information, decoder needs one bit feedback per received encoding symbol. In this section, we establish a coding scheme called Delete-and-Conquer with a more limited use of the feedback channel wherein the decoder is allowed to transmit one bit feedback for a small fraction of received encoding symbols, when certain conditions are met.

A. Delete-and-Conquer codes

Recalling the definition of the distance metric, a distance 0 happens if and only if all neighbors of the received encoding symbol have already been decoded. Similarly, a distance 1 occurs in the case that there is only a single undecoded neighbor, which can then be recovered uniquely. In other words, a distance of 0 or 1 provides information about the recovery of neighbors that are part of a received linear combination. The 0 and 1 distance feedbacks can be viewed as a generalization of the traditional acknowledgment to the coded cases as they notify the recovery of a group of input symbols involved in an encoding symbol. Our Delete-and-Conquer coding scheme is built upon a use of $0-1$ distance feedbacks.

A Delete-and-Conquer encoder performs similar to the LT encoder in that it first picks a coding degree d from the degree distribution. However, in the second step the encoder selects d symbols from a *subset* of input symbols. Specifically, upon receiving a feedback message, the encoder assigns a selection probability of zero to the neighbors of the acknowledged encoding symbol, while remaining symbols would have an equal selection probability. Intuitively, the encoder deletes recovered symbols and continues with a smaller block of symbols; in so doing, the encoder also rescales the primary degree distribution (e.g., the Robust Soliton distribution denoted by Ω_k) to the smaller set of input symbols with size $k - m$, in which m is the number of deleted symbols. Excluding recovered

Algorithm 1 Delete-and-Conquer Encoding $(x_1, x_2, .., x_k)$

```
1: $z \leftarrow 0$ and $m \leftarrow 0$
2: $\mathcal{A} \leftarrow \{x_1, x_2, ..., x_k\}$ and $\mathcal{B} \leftarrow \emptyset$
3: while $z < k$ do
4:     Pick a coding degree $d$ from the distribution $\Omega_{k-m}$
5:     Select $d$ symbols uniformly at random from set $\mathcal{A}$
6:     Send symbol $y$ as XOR of $d$ selected symbols
7:     if feedback($y$) = true then
8:         $\mathcal{C} \leftarrow$ Neighbors of $y$
9:         $\mathcal{B} \leftarrow B \cup \mathcal{C}$
10:        $m \leftarrow |\mathcal{B}|$
11:        $\mathcal{A} \leftarrow \mathcal{A} \setminus \mathcal{B}$
12:    end if
13:    if Terminate = true then
14:        $z = k$
15:    end if
16: end while
```

Algorithm 2 Delete-and-Conquer Decoding of k symbols

```
1: $\mathcal{S} \leftarrow \emptyset$          $\triangleright$ $\mathcal{S}$ is the set of recovered symbols
2: while $|\mathcal{S}| < k$ do
3:     $y \leftarrow$ Received encoded symbol
4:     if Distance($y, \mathcal{S}$) = 0 or 1 then
5:         Send a feedback and set feedback($y$) true
6:     end if
7:     call Peeling-Decoder
8:     Update $\mathcal{S}$
9:     if $|\mathcal{S}| = k$ then
10:        Terminate = true
11:    end if
12: end while
13:
14: function DISTANCE($y$, $\mathcal{S}$)
15:    distance $\leftarrow 0$
16:    for all neighbors $x_i$ of $y$ do
17:        if $x_i \notin \mathcal{S}$ then
18:            Increment distance
19:        end if
20:    end for
21:    return distance
22: end function
```

symbols from future transmissions reduces the computational complexity at the encoder and decoder. Algorithm 1 gives the pseudo-code of the Delete-and-Conquer encoding scheme. At the receiver side, the Delete-and-Conquer decoder performs based on peeling decoder with a slight modification that upon receiving a new encoding symbol, the decoder checks if the distance is equal to 0 or 1 in which case there is a one-bit feedback transmission. The pseudo-code of the Delete-and-Conquer decoding is provided in Algorithm 2.

In order to generalize the Delete-and-Conquer codes to the broadcast scenarios, we note that excluding a subset of recovered symbols from subsequent transmissions may increase the total number of transmissions (compared with when all recovered symbols are excluded), but it does not impede the decoding progress. In the worst case, no symbol is dropped from the encoding set, which reduces our codes to the original LT codes. Therefore, in a broadcast scenario, the encoder can simply take the intersection of collected feedbacks from different receivers, and proceed with excluding those symbols confirmed to be recovered by all receivers.

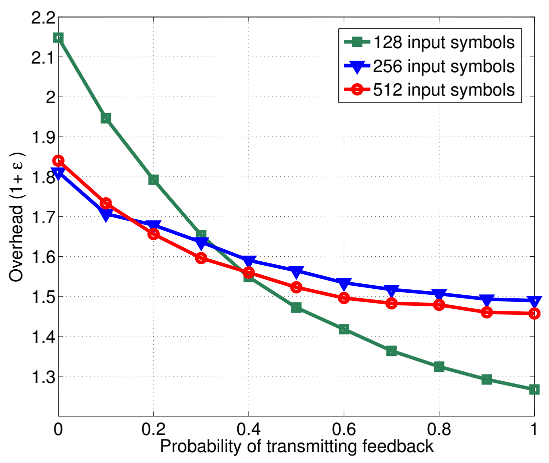
B. Probabilistic Delete-and-Conquer method

In the case of severely constrained feedback, the receiver adds the mechanism of probabilistic feedback control in which case feedbacks are only transmitted with a given probability. An optimal feedback probability is determined according to the capacity of back channel and the cost of feedback transmission. For instance, Fig. 6(a) shows simulation results of the coding overhead (i.e., number of forward transmissions normalized with respect to the number of input symbols) as the probability of sending 0 and 1 feedbacks increases. The results illustrate that when the probability of sending 0 and 1 distance feedbacks increases, amount of forward communications decreases. On the other hand, as Fig. 6(b) shows, the (normalized) number of transmitted feedback messages increases with the probability, as expected. Therefore, by adjusting the probability of feedback transmission, receiver would be able to trade off between the number of forward and feedback transmissions. In this paper, for the sake of exposition, we assume that the feedback channel is error-free; however, the probabilistic feedback mechanism resembles erasures on the back channel such that if the feedback transmission probability θ is set to 1, all feedback messages are received by the encoder, while probability $\theta = 0$ implies that none of feedback messages are received by the encoder. One can notice that this method simulates the same behavior of a feedback channel with an erasure rate of $1 - \theta$.

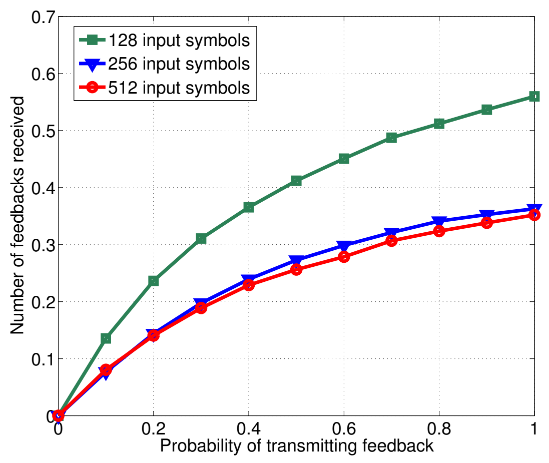
C. Feedback Analysis

The motivation behind the distance-type feedback is to learn about the state of individual symbols at the decoder side. For instance, Delete-and-Conquer encoder learns about the recovered symbols using a light-weight feedback and excludes the recovered symbols from subsequent transmissions. An alternative approach includes sending the identity of recovered symbols with up to $k \log(k)$ bits of feedback. The authors in [12] have proposed a method in which recovered symbols are excluded from future transmissions and the degree distribution is recomputed using least-squares computations. In this method, feedback messages contain the exact ID of recovered input symbols that incurs a feedback overhead larger than the Delete-and-Conquer scheme. Fig. 6(b) experimentally shows that the total amount of feedback transmitted by the Delete-and-Conquer decoder is strictly less than k bits. Hence, the differences between the codes in [12] and the Delete-and-Conquer method lie in the amount of feedback needed and the adaptation scheme after each deletion step. In Section VII, we numerically compare the performance of these two methods.

The Delete-and-Conquer receiver notifies the recovery of a group of symbols if the received symbol is within a distance of 0 or 1 from the set of recovered symbols. One-bit feedback messages can be regarded as an acknowledgement on a group



(a) Number of forward transmissions



(b) Number of feedbacks

Fig. 6. (a) Number of forward transmissions (normalized to the number of input symbols) needed by Delete-and-Conquer codes as the probability of sending feedback increases (b) Number of bits of feedback (normalized to the number of input symbols) as the probability of sending feedback increases.

of symbols. There are two use-cases where the feedback actually requires just one bit. The first one relates to the case where transmissions are time-delimited, in which case the timing of a one-bit feedback message encodes information about the encoding message to which it applies. In other words, feedback messages are received at the sender before the next transmission, similar to the assumption in [23]. The second use-case is suited for a situation where a lower-level protocol (such as TCP) that, itself, utilizes feedback messages, in which case our one bit can piggy-back on existing metadata. In [24], we demonstrated a prototype application of the one-bit feedback messages to develop Coded Datagram Protocol (CDP).

In order to relax the constraint of time-delimited transmissions, we consider a protocol based on the concept of transmission window. In this method, the sender keeps transmitting up to M encoded symbols. In this case, $\log(M)$ bits are needed to specify the identity of encoding symbols within each transmission window, and thus at most $M\log(M)$ bits of feedback are transmitted within each window. On the other hand, the method based on sending the identity of recovered symbols requires $\log(k)$ bits to specify the identity of a recovered input symbol within each window, and since at most M input symbols can be decoded using M encoding symbols, $M\log(k)$ bits of feedback would be sent within a window. In the case that $M \ll k$, our scheme provides savings (in terms of communication overhead) compared with the identity-based method. The parameter M is set according to typical end-to-end delay of the forward communication channels in order to require no additional metadata for distinguishing a block of M encoding symbols from other blocks. In order to compare the total amount of feedback overhead, Table I summarizes the established classes of rateless codes with their corresponding amount of feedback overhead (i.e., bits).

V. Short Block Length Analysis

In this section, we precisely analyze the performance gains of the primitive form of our codes for very short block length of $k = 2$ and $k = 3$ symbols and when the peeling decoder is used. Although such small block lengths are not practical, our analysis shows that even for such very short block lengths the Delete-and-Conquer method provides clear performance gains compared with the original LT codes without feedback. For larger block lengths, our exact calculation of overhead in terms of degree probabilities becomes unwieldy. For the analysis purposes, we assume that the forward channel is lossless.

A. Block length k=2

As the first case, we consider the block length of $k = 2$ symbols, in which two input symbols x_1 and x_2 are encoded. We assume that the probability of degree 1 transmission is equal to $2p$, and the probability of degree 2 transmission is $1 - 2p$ for $p \leq \frac{1}{2}$. Therefore, an encoded symbol is equal to x_1 or x_2 each with probability p, and x_1+x_2 with probability $1 - 2p$.

Lemma 1. *For the block length $k = 2$, if the probability of degree 1 transmission is $2p$, then the Delete-and-Conquer codes require an expected number of $\frac{4p^2+1}{2p}$ forward transmissions and $2p$ feedback transmissions for successful decoding.*

Proof. A Delete-and-Conquer decoder can successfully decode two symbols within $n = 2$ transmissions under the following possibilities for the received symbols:

$$\{x_1, x_2\}, \{x_2, x_1\}, \{x_1 + x_2, x_1\}, \{x_1 + x_2, x_2\}.$$

The probability of terminating after two transmissions is obtained as $4p-4p^2$. Similarly, the decoder would successfully recover x_1 and x_2 within $n \geq 3$ transmissions in the case of the following received symbols:

$$\{\overbrace{x_1 + x_2, ..., x_1 + x_2}^{n-1 \text{ symbols}}, x_1\},\ \{\overbrace{x_1 + x_2, ..., x_1 + x_2}^{n-1 \text{ symbols}}, x_2\}.$$

The probability of successful recovery in this case would be:

$$Q(n) = (1 - 2p)^{n-1}\,(p + p)\,, \quad n \geq 3;$$

and therefore, the expected number of forward transmissions for the Delete-and-Conquer scheme is equal to:

$$\bar{n}_{Del} = 2(4p - 4p^2) + \sum_{n=3}^{\infty} nQ(n) = \frac{4p^2+1}{2p}. \tag{8}$$

To calculate the expected number of feedbacks transmitted, we note that one feedback is transmitted only in the cases of received symbols $\{x_1, x_2\}$ and $\{x_2, x_1\}$ each happens with probability p, and thus the expected number of feedbacks transmitted would be $2p$. It should be noted that the last feedback message is excluded from the count, as it is also needed by other coding schemes to stop the encoder from further transmissions. □

Theorem 1. *For the block length $k = 2$, the Delete-and-Conquer codes provide a savings of $\frac{2p^2}{1-p}$ in forward transmissions compared with the LT codes.*

Proof. First, we calculate the expected number of transmissions required by the LT codes to recover all symbols. To this end, we obtain the probability of full recovery within $n \geq 2$ transmissions. For instance, in the case of $n = 2$, the decoder should receive one of the following combinations to successfully recover x_1 and x_2:

$$\{x_1, x_2\}, \{x_1, x_1 + x_2\}, \{x_2, x_1\}, \{x_2, x_1 + x_2\},$$
$$\{x_1 + x_2, x_1\}, \{x_1 + x_2, x_2\}.$$

Accordingly, the probability of decoding within two transmissions can be calculated as $4p - 6p^2$. For a general case of n transmissions, one can see that the probability of recovery within n transmissions is:

$$P(n) = 2p^{n-1}\left(p + (1-2p)\right) + (1-2p)^{n-1}\left(p+p\right);$$

and hence, the expected total number of transmissions is:

$$\bar{n}_{LT} = \sum_{n=2}^{\infty} nP(n) = \frac{4p^2 - p + 1}{2p(1-p)}. \tag{9}$$

Using (8) and (9), expected amount of savings $\bar{n}_{LT} - \bar{n}_{Del}$ is obtained.

□

Algorithm	Feedback type	Worst-case feedback	Average feedback
All-Distance	Distance value ($\leq k$) per received symbol	$(1+\epsilon)k\log(k)$	$(1+\epsilon)k\log\left(\log(k)\right)$
Quantized distance	Quantized distance value (0 or 1) per received symbol	$(1+\epsilon)k$	$(1+\epsilon)k$
Delete-and-Conquer	An ACK for a $0 \leq c \leq 1$ fraction of received symbols	$c(1+\epsilon)k$	$c(1+\epsilon)k$
Probabilistic Delete-and-Conquer ($0 \leq \theta \leq 1$)	An ACK for a θc fraction of received symbols	$\theta c(1+\epsilon)k$	$\theta c(1+\epsilon)k$

TABLE I
SUMMARY OF PROPOSED RATELESS CODES WITH NONUNIFORM SELECTION DISTRIBUTION

B. Block length k=3

For the block length $k = 3$, the authors in [13] have derived the expected number of encoding symbols required by the LT codes for full recovery. In this model, the set of received symbols at the decoder defines a state of an absorbing Markov chain, and the process (i.e., transmission of encoded symbols) ends when it reaches to the absorbing state that includes all input symbols decoded. We similarly adapt this approach to obtain the Markov chain for the Delete-and-Conquer scheme with 3 input symbols. The corresponding Markov chain shown in Fig. 7, includes states up to the permutations of input symbols, e.g., two states $\{x_1, x_2 + x_3\}$ and $\{x_2, x_1 + x_3\}$ are isomorphic and it is enough to consider a single unique state for each group of isomorphic states. In this figure, darker states are irreducible by the decoder in that no symbol can be further recovered, whereas other states can be immediately reduced by the decoder to the darker ones.. By constructing the state transition matrix $\mathbf{P}$ as

$$\mathbf{P} = \begin{pmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

we can compute the expected number of steps (transmissions) from the initial state to the absorbing state $\{x_1, x_2, x_3\}$. In the notation, matrix $\mathbf{Q}$ represents the transition probabilities between transient states, $\mathbf{R}$ denotes the probabilities between transient states and the absorbing state, and $\mathbf{I}$ is an identity matrix.

Theorem 2. *For the block length $k = 3$, given that p_j is the probability of transmitting an encoded symbol with degree j, the expected number of transmissions required by the Delete-and-Conquer scheme for successful decoding is:*

$$\bar{n}_{Del} = \frac{1}{p_1} + \frac{p_2}{3p_1 + 2p_2} + \frac{p_2^2}{p_1 + p_2} - \frac{8p_2^3}{(p_1 + 2p_2)(p_2 - 3)} + \frac{3p_1 - 4p_2 + 3p_1p_2 - 3p_2^3 + 3}{3 - p_2}. \tag{10}$$

Proof. In an absorbing Markov chain with a transition matrix $\mathbf{P}$ and the *fundamental matrix*

$$\mathbf{N} = \mathbf{I} + \mathbf{Q} + \mathbf{Q^2} + ... = (\mathbf{I} - \mathbf{Q})^{-1},$$

the expected number of steps (transmissions) from the initial state to the absorbing one is:

$$\bar{n} = \boldsymbol{\pi}_0 \mathbf{N} \mathbf{c}, \tag{11}$$

where $\boldsymbol{\pi}_0 = (1\ 0\ ...\ 0)$ is the initial probability corresponding to the state of no symbols been transmitted, and $\mathbf{c} = (1\ ..\ 1)^T$ [25]. From Fig. 7, we obtain matrix $\mathbf{P}$ as:

$$\mathbf{P} = \begin{pmatrix}
0 & p_1 & p_2 & p_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & p_1' & 0 & 0 & p_2' & 0 & 0 \\
0 & 0 & \frac{p_2}{3} & 0 & 0 & p3 & \frac{2p_1}{3} & \frac{p_1}{3} & \frac{2p_2}{3} & 0 \\
0 & 0 & 0 & p_3 & 0 & p_2 & 0 & p_1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & \frac{p_2+3p_3}{3} & 0 & \frac{p_1}{3} & \frac{2p_2}{3} & \frac{2p_1}{3} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{p_1'}{2} & 0 & 0 & 1 - \frac{p_1'}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 - p_1' & 0 & p_1' \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 - p_1 & p_1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

where we assume that after each symbol deletion at the encoder, the probabilities are normalized by dividing by the sum of the remaining degrees. For instance, after one exclusion, $p_1' \triangleq \frac{p_1}{p_1+p_2}$ and $p_2' \triangleq \frac{p_2}{p_1+p_2}$ would be the probability of degrees 1 and 2 transmissions respectively. This leads to the theorem statement. □

The expected number of transmissions for the LT codes has been derived in [13] as follows:

$$\bar{n}_{LT} = \frac{1}{p_1} + \frac{6p_1}{p_1 - 3} + \frac{18p_1}{(3 - p_2)(3 - 2p_1 - p_2)} + \frac{9p_1}{2(p_1 + p_2)(3p_1 + 2p_2)}. \tag{12}$$

If the encoder uses only degree 1 symbols (i.e., $p_1 = 1$), the expected number of required symbols for the LT codes is $\bar{n}_{LT} = 5.5$, illustrating the effect of the coupon collector's problem; on the other hand, Delete-and-Conquer scheme requires only $\bar{n}_{Del} = 3$ encoded symbols, which is the minimum possible number of forward transmissions. It should be noted that in this case, Delete-and-Conquer scheme turns into a no-coding ARQ method. An optimization in [13] results in a minimum number of 4.046 forward transmissions (with $p_1 = 0.524, p_2 = 0.366$, and $p_3 = 0.109$) for the LT codes, whereas Delete-and-Conquer coding with these same probabilities yields a total number of $\bar{n}_{Del} = 3.678$ forward transmissions. In general, we can numerically compare (10) to (12) to see that Delete-and-Conquer scheme can decrease the total number of forward transmissions up to 2.4-fold.

Theorem 3. *For $k = 3$ input symbols, the expected number of feedbacks transmitted by the Delete-and-Conquer scheme before conclusion (i.e. not including the termination signal) is:*

$$\bar{f}_{Del} = \frac{3p_1}{3p_1 + 2p_2} + \frac{6p_1}{3 - p_2} + \frac{p_1^2}{p_1 + p_2} - 2p_1. \tag{13}$$

Proof. In an absorbing Markov chain, the probability of ever visiting state j when starting at a transient state i is the entry h_{ij} of the matrix $\mathbf{H} = (\mathbf{N} - \mathbf{I})\mathbf{N_{dg}^{-1}}$, where $\mathbf{N}$ is the fundamental matrix and $\mathbf{N_{dg}}$ is the diagonal matrix with the same diagonal as $\mathbf{N}$, and $\mathbf{I}$ is an identity matrix [25]. In Fig. 7, a feedback is transmitted when transitions along the dotted-line occur, e.g. a transition from the state 1 to state 2. Accordingly, the probability of such transitions, and hence the expected number of feedbacks transmitted is given by:

$$\bar{f}_{Del} = h_{12} + h_{12}h_{25} + h_{13}h_{37} + h_{13}h_{38} + h_{14}h_{48};$$

from which the result follows. □

Based on the Theorem 2 and 3, we can calculate the optimal probability values p_1^* and p_2^* (and $p_3^* = 1 - p_1^* - p_2^*$) that minimize the total number of forward and feedback transmissions needed by the Delete-and-Conquer scheme. In other words:

$$(p_1^*, p_2^*) = \underset{(p_1, p_2)}{\arg\min} \left[\bar{n}_{Del} + \bar{f}_{Del}\right];$$

which results in $(p_1^*, p_2^*) = (0.644, 0.206)$ (and $p_3^* = 0.150$) with a minimum number of total transmissions 4.7247. In this case, we simply considered the sum of forward and feedback transmissions. In a more general sense, we can assume that

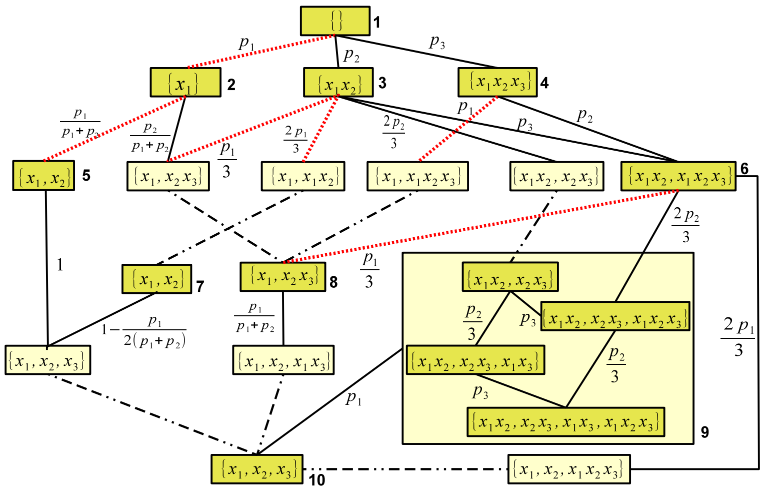


Fig. 7. State space of the Delete-and-Conquer scheme with 3 input symbols. The four states inside the box are considered as a single state. Notation x_ix_j represents the symbol $x_i + x_j$, and dotted red lines represent transitions with a feedback.

each transmission through the forward channel has a cost of C_1, while each feedback transmission has a cost of C_2. Therefore, the optimal probability values can be calculated as:

$$(p_1^*, p_2^*) = \underset{(p_1,p_2)}{\arg\min}\left[C_1\bar{n}_{Del} + C_2\bar{f}_{Del}\right].$$

Moreover, in comparison with the LT codes, one can notice that it is worthwhile to send feedback if:

$$C_1\bar{n}_{Del} + C_2\bar{f}_{Del} \le C_1\bar{n}_{LT} \Rightarrow \frac{\bar{f}_{Del}}{\bar{n}_{LT} - \bar{n}_{Del}} \le \frac{C_1}{C_2},$$

where $\bar{n}_{Del}$, $\bar{n}_{LT}$, and $\bar{f}_{Del}$ are calculated in (10), (12), and (13).

VI. Decoding Analysis

In this section, we first investigate the performance of the maximum-likelihood decoder when used with the Delete-and-Conquer method. Next, we provide upper-bounds on the number of ACK messages and the number of ACKed symbols when the peeling algorithm is employed by the Delete-and-Conquer decoder.

A. Maximum-Likelihood Decoder

We derive an upper-bound on the failure probability of the maximum likelihood (ML) decoder when used with the Delete-and-Conquer codes. It should be noted that ML decoding algorithms incur higher computational complexity compared with peeling decoders, and, thus, they are not typically used in practice. However, we provide our analysis results on the ML decoder performance to show that a use of distance-type feedback provides a tighter upper-bound on the decoder failure probability compared with the original LT codes. The performance of original LT codes when used with an ML decoder has been investigated in [21], and thus our results extend the previous works to the case of LT codes with distance-type feedback. To this end, we assume that there are k input symbols at the transmitter, and that n encoding symbols are received over a binary erasure channel (BEC). The ML decoding over a BEC is equivalent to recovering k information (input) symbols from n received encoding (output) symbols. Without loss of generality, we assume that each symbol is one bit; $\mathbf{x}$ is a row vector containing k input bits; and $\mathbf{y}$ is the vector of n output bits. Matrix $\mathbf{G} = [g_{i,j}]$ is an $n \times k$ adjacency matrix of the decoder graph, such that an entry $g_{i,j}$ is equal to 1 if the i^{th} output node has the j^{th} input node as a neighbor. The ML decoder is then equivalent to solving a system of linear equations (with unknowns $\mathbf{x}$ and received symbols $\mathbf{y}$) of the form:

$$\mathbf{G}\mathbf{x}^T = \mathbf{y}^T. \tag{14}$$

Encoding symbols with a distance of 0 or 1 trigger a feedback message that causes the corresponding symbols to be excluded from future transmissions. Excluding the recovered symbols from subsequent transmissions is equivalent to setting the subsequent elements of the corresponding columns in $\mathbf{G}$ to zero. For instance, Fig. 8 shows a realization of the matrix $\mathbf{G}$ in which the first feedback message acknowledges recovery of x_2. Thereafter the second column of $\mathbf{G}$ (i.e., the shaded part) is set to zero.

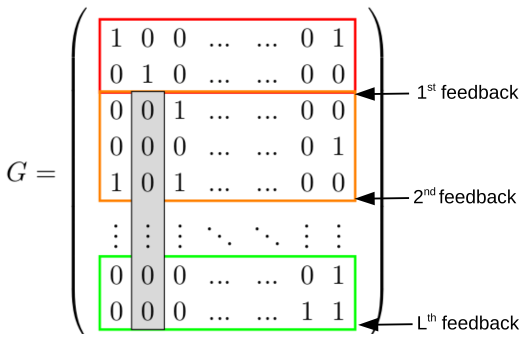


Fig. 8. Decoder matrix and the location of feedback transmission

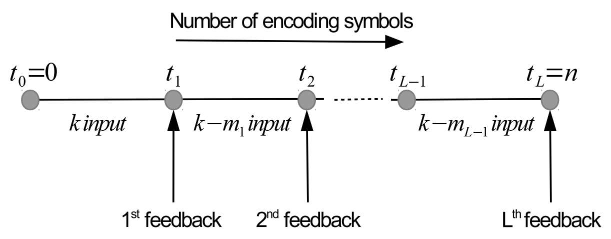


Fig. 9. Location of feedbacks and interval of coding

The ML decoder failure is equivalent to the event that the adjacency matrix $\mathbf{G}$ in (14) is not of full rank. Let p_e be the probability that an input bit j (for an arbitrary $j \in \{1, 2, ..k\}$) is not recoverable under the ML decoding rule. From [21]:

$$\begin{aligned} p_e &= \Pr\left\{\exists \mathbf{x} \in GF(2^k), x_j = 1 : \quad \mathbf{G}\mathbf{x}^T = \mathbf{0}^T\right\} \\ &\leq \sum_{\substack{\mathbf{x} \in GF(2^k) \\ x_j = 1}} \Pr\left\{\mathbf{G}\mathbf{x}^T = \mathbf{0}^T\right\}. \qquad (15) \end{aligned}$$

In order to calculate $\Pr\{\mathbf{G}\mathbf{x}^T = \mathbf{0}^T\}$, we separately consider the rows of $\mathbf{G}$ between consecutive feedback messages. We assume that L feedback messages are transmitted in total such that after receiving t_1 encoding symbols the first feedback message is transmitted, after receiving t_2 encoding symbols the second feedback is sent, and so forth. At the boundary points, we define $t_0 = 0$ and $t_L = n$. Therefore, there is no feedback within each interval of [0, t_1], (t_1, t_2], ..., (t_{L-1}, n], and there is one feedback at the end of each interval, as shown in Fig. 9. We assume that within the i^{th} interval ($i = 0, ..., L-1$) the coding window contains $k-m_i$ symbols, and thus the encoder uses a fixed degree distribution $\Omega_{k-m_i}(d)$ defined over the set of $k - m_i$ unacknowledged symbols.

To calculate an upper bound on the decoder failure probability, we start with a single row of $\mathbf{G}$. Let $\mathbf{r}$ to be a row of degree d, and assume that the total number of m symbols are acknowledged before transmitting $\mathbf{r}$; in other words, m indices out of k indices in $\mathbf{r}$ are forced to be zero. We define a row vector $\mathbf{f}$ such that $f_l = 1$ if the l^{th} symbol has been acknowledged, and 0 otherwise (i.e., an indicator function on the index of acknowledged symbols). For a given input vector $\mathbf{x}$ with $||\mathbf{x}||_0 = w$ ($||.||_0$ is the 0-norm), we have the following lemma:

Lemma 2. *Given that the row vector* $\mathbf{r}$ *has degree* d *(i.e.,* $||\mathbf{r}||_0 = d$*), the probability of* $\mathbf{r}\mathbf{x}^T = 0$ *is:*

$$p\big(\mathbf{x}, ||\mathbf{r}||_0 = d\big) = \frac{\sum_{u=0,2,..,\min(2\lfloor \frac{d}{2} \rfloor, \bar{w})} \binom{\bar{w}}{u}\binom{k-m-\bar{w}}{d-u}}{\binom{k-m}{d}},$$

in which $\bar{w} = w - \langle \mathbf{x}, \mathbf{f} \rangle$ with $\langle \mathbf{x}, \mathbf{f} \rangle$ denoting the dot product of two vectors.

Proof. The event $\mathbf{r}\mathbf{x}^T = 0$ happens if and only if $\mathbf{r}$ has an even number of 1's in those indices of j in which x_j is equal to 1 as well. Assume that $J = \{j_1, j_2, .., j_w\}$ is the set of indices in which $\mathbf{x}$ is 1, and $A = \{a_1, a_2, ..., a_m\}$ is the set of acknowledged indices. Therefore, we need to choose an even number u of indices that belong to J but not to A. The number of these non-overlapping indices is given by $\bar{w} = w - \langle \mathbf{x}, \mathbf{f} \rangle$. Because the degree of $\mathbf{r}$ is d, we then need to choose $d - u$ symbols from the remaining $k - m - w + \langle \mathbf{x}, \mathbf{f} \rangle$ indices that belong neither to J nor to A. Finally, given that the vector $\mathbf{r}$ is generated randomly (i.e., d neighbors are selected uniformly at random from $k - m$ unacknowledged symbols), the result follows. □

Using Lemma 2 and the fact that $\mathbf{r}$ has degree d with probability $\Omega_{k-m}(d)$, we have:

$$p(\mathbf{x}) = \sum_{d=1}^{k-m} \Omega_{k-m}(d) p\big(\mathbf{x}, ||\mathbf{r}||_0 = d\big). \qquad (16)$$

Now, we can extend this result to more than one row of $\mathbf{G}$ in the following manner. Let us denote the $t_i - t_{i+1}$ rows of $\mathbf{G}$ by $\mathbf{G}_i$. Rows in $\mathbf{G}_i$ are generated independently according to the degree distribution $\Omega_{k-m_i}(d)$. Thus, we have:

$$\Pr\{\mathbf{G}_i\mathbf{x}^T = \mathbf{0}^T\} = (p_i(\mathbf{x}))^{t_{i+1}-t_i}, \qquad (17)$$

in which $p_i(\mathbf{x})$ is calculated as in (16), and based on the number of acknowledged symbols and the degree distribution within the i^{th} interval, i.e.,:

$$p_i(\mathbf{x}) = \sum_{d=1}^{k-m_i} \Omega_{k-m_i}(d) p_i\big(\mathbf{x}, ||\mathbf{r}||_0 = d\big).$$

Given that there are L transmit intervals (i.e., L feedback messages), we can calculate the probability of $\mathbf{G}\mathbf{x}^T = \mathbf{0}^T$ for a given vector $\mathbf{x}$ as follows:

$$\Pr\{\mathbf{G}\mathbf{x}^T = \mathbf{0}^T\} = \prod_{i=0}^{L-1} \Pr\{\mathbf{G}_i\mathbf{x}^T = \mathbf{0}^T\}. \qquad (18)$$

Assembling these steps together, the ML decoder failure probability of the Delete-and-Conquer scheme is given by the following theorem.

Theorem 4. *Given that* L *feedbacks are transmitted in total (i.e, one feedback after receiving the* t_i*-th (*$i = 1, ..., L$*) encoding symbol), the ML decoder failure probability of recovering an input symbol* j *(for an arbitrary* $j \in \{1, 2, ..k\}$*) is upper bounded by*

$$p_e \leq \min\left\{1, \sum_{w=1}^{k} \Big(\sum_{\substack{\mathbf{x} \\ ||\mathbf{x}||_0 = w \\ x_j = 1}} \prod_{i=0}^{L-1} \Pr\{\mathbf{G}_i\mathbf{x}^T = \mathbf{0}^T\}\Big)\right\}. \qquad (19)$$

Proof. From Lemma 2 and its following results, we obtain that for a given input vector $\mathbf{x}$ with Hamming weight w and L feedback messages, the probability of $\mathbf{G}\mathbf{x}^T = \mathbf{0}^T$ is calculated as in Eq. (18). Therefore, summing over all possible input vectors $\mathbf{x}$ with the j^{th} index equal to 1, yields the theorem statement. It should be noted that we assume the values of L and t_i's (i.e., the total number of feedbacks and their trigger points) are known. □

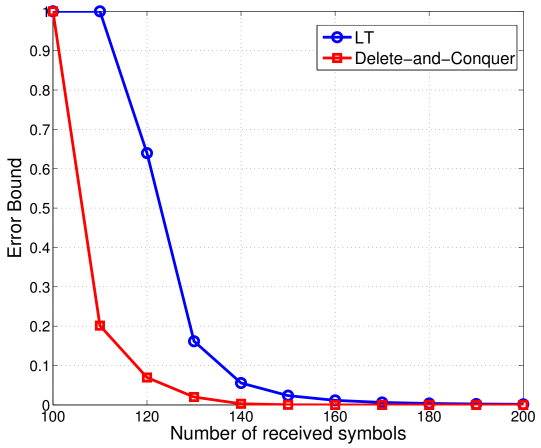


Fig. 10. Upper bound on the maximum likelihood decoder failure probability for the LT and the Delete-and-Conquer codes

From [21], the upper bound on the ML decoder failure probability when there is no feedback is calculated as:

$$p_e \leq \min \left\{ 1, \sum_{w=1}^{k} \binom{k-1}{w-1} \left(\sum_d \Omega_k(d) \frac{\sum_{s=0,2,..,2\lfloor \frac{d}{2} \rfloor} \binom{w}{s}\binom{k-w}{d-s}}{\binom{k}{d}} \right)^n \right\}. \tag{20}$$

Fig. 10 numerically compares the upper bound in (19) with that in (20) for $k = 100$ input symbols. The values of L (number of feedback messages), and m_i (number of excluded symbols) in Fig. 9 are obtained from simulation and plugged into (19), and the result is averaged over 1000 runs. For instance, with 120 collected encoding symbols at the receiver, we have $L = 47$, and the values of m_i increases from 1 to 80 deleted symbol. The results in Fig. 10 confirm that collecting more encoding symbols reduces the bound on ML failure probability, as expected. However, Delete-and-Conquer codes with 0 and 1 feedback messages achieve a tighter upper-bound on the decoder failure probability.

Asymptotic results: We conclude our analysis of the Delete-and-Conquer scheme by providing its asymptotic performance metrics. To this end, we keep the assumption that initially there exist k input symbols at the encoder, and at some point, m symbols are acknowledged. The Delete-and-Conquer distribution is thus given by $\Omega_{k-m}(i)$ (for $i = 1, ..., k-m$). Adapting the results of [3] yields that the average degree of an encoding symbol generated by the encoder is given by $\bar{D} = O(\ln(k-m))$. Furthermore, an encoder that deletes m symbols out of k symbols, needs to transmit *at most* $k-m+O\left(\sqrt{k-m}\ \ln^2(\frac{k-m}{\delta})\right)$ encodings so that the decoder be able to recover all input symbols with probability at least $1-\delta$. Furthermore, computational complexity of the coding process is given by $O\left((k-m)\ln\frac{k-m}{\delta}\right)$. One can notice that as the number of acknowledged symbols (i.e., parameter m) increases, performance metrics improve. In the case of no feedbacks (i.e., $m = 0$) Delete-and-conquer codes reduce to the original LT codes.

B. *Expected Number of ACKs and ACKed symbols*

In the previous section, we derived an upper-bound on the failure probability of the maximum-likelihood decoder when used with the Delete-and-Conquer coding scheme. However, ML decoders typically incur higher computational complexity compared with the peeling decoder, and thus, it is desirable to analytically investigate the performance of peeling decoder paired with the Delete-and-Conquer scheme. In this subsection, we derive results on the expected number of feedback messages and the expected number of excluded symbols by the Delete-and-Conquer encoder.

Recall that the Delete-and-Conquer method is based on sending ACKs on groups of symbols. However, not all recovered input symbols are acknowledged, as ACKs are only sent in response to an encoding symbol with a distance 0 or 1. For instance, Fig. 11 shows a sample simulation result on the number of ACKed symbols compared with the number of decoded symbols by the decoder. From the results, we observe that the number of ACKed symbols behaves similar to the number of decoded symbols, but it is smaller than the number of decoded symbols.

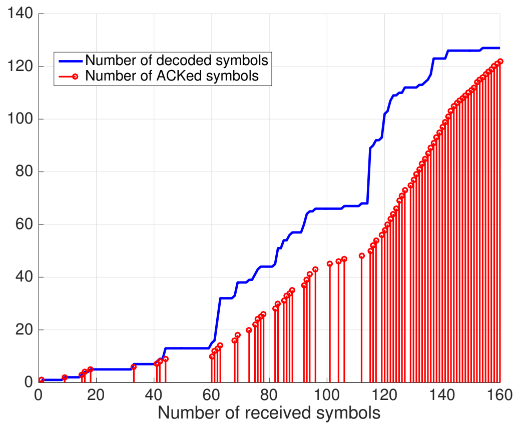


Fig. 11. Number of decoded symbols compared with the number of ACKed symbols using the Delete-and-Conquer method ($k = 128$).

In order to calculate the probability of sending an ACK message, we assume that A_i and B_i, respectively, denote the expected number of ACKs and the expected number of ACKed symbols after receiving the i^{th} encoding symbol. Let us define the following notations:

$$\gamma_i(d) := \Omega_{k-B_i}(d) \frac{\binom{k-B_i-U_i}{d}}{\binom{k-B_i}{d}};$$

$$\rho_i(d) := \Omega_{k-B_i}(d) \frac{\binom{k-B_i-U_i}{d-1} U_i}{\binom{k-B_i}{d}}, \tag{21}$$

in which U_i is the number of undecoded symbols after receiving the i^{th} encoding symbol. $\gamma_i(d)$ is the probability of receiving an encoding symbol with degree d and of distance 0, while $\rho_i(d)$ is the probability of receiving an encoding symbol with degree d and of distance 1. Hence, we obtain the following recursions for A_i and B_i:

$$
\begin{aligned}
A_{i+1} &= A_i + \sum_{d=1}^{k-B_i-U_i} \gamma_i(d) + \sum_{d=1}^{1+k-B_i-U_i} \rho_i(d) \\
&= A_i + \sum_{d=1}^{k-B_i-U_i} [\gamma_i(d) + \rho_i(d)] + \rho_i(1+k-B_i-U_i);
\end{aligned}
$$

$$
B_{i+1} = B_i + \sum_{d=1}^{k-B_i-U_i} d\gamma_i(d) + \sum_{d=1}^{1+k-B_i-U_i} d\rho_i(d) = B_i + \sum_{d=1}^{k-B_i-U_i} d[\gamma_i(d) + \rho_i(d)] + (1+k-B_i-U_i)\rho_i(1+k-B_i-U_i). \tag{22}
$$

These recursions are based on the fact that only an encoding symbol with degree d and of distance 0 (contributed by $\gamma_i(d)$) or distance 1 (contributed by $\rho_i(d)$) triggers a feedback message. In turn, corresponding d input symbols are added to the set of ACKed symbols. Note that the probability of receiving a symbol of distance 0 or 1 is the sum of all contributing encoding symbols with degree between 1 and $k - B_i - U_i$ (or $1 + k - B_i - U_i$ for distance 1), which yields the recursions in (22). In this method, each input symbol is ACKed at most once, as the Delete-and-Conquer encoder deletes ACKed input symbols from future transmissions. For the initial condition, we note that the first encoding symbol triggers an ACK only if it has degree 1. Therefore, we have: $A_1 = \Omega_k(1)$ and because this encoding is of degree one, the expected number of ACKed symbols right after receiving the first encoding symbol is $B_1 = A_1 = \Omega_k(1)$.

The values of A_i and B_i are recursively calculated with the assumption that the number of undecoded symbols U_i or, equivalently, the number of recovered symbols (i.e., $D_i = k - U_i$) by the decoder are given. The dependence of A_i and B_i on U_i complicates the analysis, because of the reverse dependence of U_i on B_i through the adapted degree distribution. Having an analysis that captures this cross dependence is certainly motivated, but we could not find way to extend the available analysis tools of the peeling decoder to this scenario. Instead, we use an approximation of the number of decoded symbols as $D_i = B_i - \alpha$, with a constant α. For instance, Fig. 11 demonstrates the difference of the number of decoded symbols and ACKed symbols as coding progresses. Using the approximation, we have $k - B_i - U_i = \alpha$ (for $1 \le i \le n$), and thus:

$$
\begin{cases}
A_{i+1} = A_i + \sum_{d=1}^{\alpha}[\gamma_i(d) + \rho_i(d)] + \rho_i(1+\alpha); \\
B_{i+1} = B_i + \sum_{d=1}^{\alpha} d[\gamma_i(d) + \rho_i(d)] + (1+\alpha)\rho_i(1+\alpha); \\
A_1 = \Omega_k(1); \\
B_1 = \Omega_k(1).
\end{cases} \tag{23}
$$

Within n encoding transmissions, the expected number of ACKs A_n and the expected number of ACKed symbols B_n are calculated using dynamic programming. In the following, we provide upper-bounds on A_i and B_i.

Lemma 3. *For a fixed i, let $\beta = k - B_i$, then $\gamma_i(d)$ is upper-bounded by $\Omega_\beta(d)\frac{\alpha}{\beta}$.*

Proof. We use the definition of $\gamma_i(d)$ such that:

$$
\gamma_i(d) = \Omega_\beta(d)\frac{\binom{\alpha}{d}}{\binom{\beta}{d}} = \Omega_\beta(d)\frac{\alpha!(\beta-d)!}{\beta!(\alpha-d)!} \le \Omega_\beta(d)\frac{\alpha}{\beta}, \tag{24}
$$

in which the inequality results from the fact that $\frac{\alpha!(\beta-d)!}{\beta!(\alpha-d)!}$ is upper-bounded by $\frac{\alpha}{\beta}$ for $d = 1$. □

Lemma 4. *For a fixed i and $\beta = k - B_i$, $\rho_i(d)$ is upper-bounded by $\Omega_\beta(d)d\frac{\beta-\alpha}{\beta}$.*

Proof. Similar to Lemma 3, we simplify the $\rho_i(d)$ expression as follows:

$$
\rho_i(d) = \Omega_\beta(d)\frac{d\alpha!(\beta-d)!(\beta-\alpha)}{\beta!(\alpha-d+1)!} \le \Omega_\beta(d)d\frac{\beta-\alpha}{\beta}. \tag{25}
$$

Note that the simplified expression of $\rho_i(d)$ is not a monotone function of d, and thus the last inequality is obtained by setting $d = 1$ in the increasing portion only (i.e., $\frac{\alpha!(\beta-d)!(\beta-\alpha)}{\beta!(\alpha-d+1)!}$). □

Theorem 5. *The number of ACKs transmitted by the Delete-and-Conquer method and within the $(i+1)^{th}$ transmissions is upper-bounded by:*

$$
A_{i+1} \le A_i + \frac{\alpha}{k-B_i} + (1 - \frac{\alpha}{k-B_i})\ln(k-B_i). \tag{26}
$$

Proof. From (23), Lemma 3, and Lemma 4, we have:

$$
\begin{aligned}
A_{i+1} &\le A_i + \sum_{d=1}^{\alpha}\Omega_\beta(d)\frac{\alpha}{\beta} + \sum_{d=1}^{\alpha+1} d\Omega_\beta(d)\frac{\beta-\alpha}{\beta} \\
&\le A_i + \sum_{d=1}^{\beta}\Omega_\beta(d)\frac{\alpha}{\beta} + \sum_{d=1}^{\beta} d\Omega_\beta(d)\frac{\beta-\alpha}{\beta} \\
&= A_i + \frac{\alpha}{k-B_i} + (1 - \frac{\alpha}{k-B_i})\ln(k-B_i).
\end{aligned}
$$

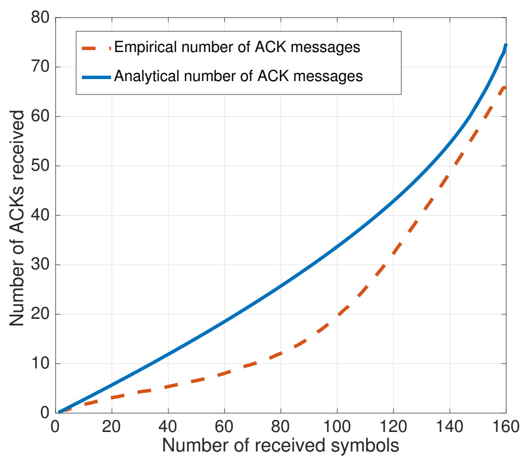
In which, we use the facts that $\alpha = k - B_i - U_i \le k - B_i = \beta$, and that $\sum_{d=1}^{\beta}\Omega_\beta(d) = 1$ as $\Omega_\beta(d)$ is the degree distribution defined over β symbols. Finally, we have $\sum_{d=1}^{\beta} d\Omega_\beta(d) = \ln(\beta)$ as it gives the average degree of the Robust Soliton distribution Ω_β defined over β symbols. The last expression is obtained from $\beta = k - B_i$. □

As a corollary, it is straightforward to show that the upper-bound on the number of ACK messages is a monotonically increasing function of B_i. In other words, as more symbols are excluded from the set of decoded symbols, the number of ACKs increases as well, as expected. Moreover, we can show that the following upper-bound on the number of ACKed symbols holds:

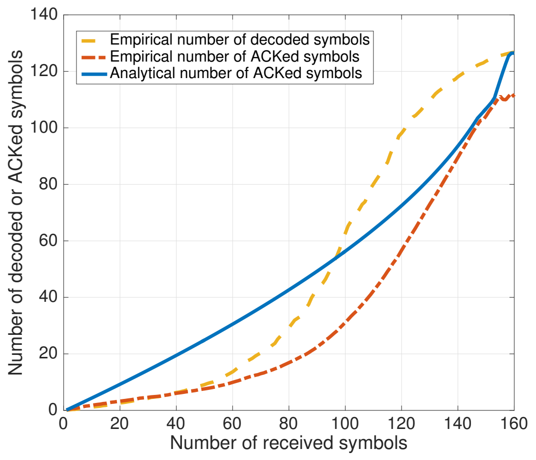
$$
B_{i+1} \le B_i + \frac{\alpha}{k-B_i}\ln(k-B_i) + (k-B_i)(1 - \frac{\alpha}{k-B_i}),
$$

where we use the second moment of the Robust Soliton distribution, i.e.,: $\sum_{d=1}^{\beta} d^2\Omega_\beta(d) = O(\beta)$.

Fig. 12 compares the analytical and empirical results on the number of ACK messages and the number of ACKed input symbols by the Delete-and-Conquer method. Difference between the number of decoded symbols and the number of ACKed symbols (i.e., value of α) is calculated empirically, and is used in A_i and B_i expressions. As the results show, the analytical upper-bounds closely approximate the behavior of the Delete-and-Conquer decoder in terms of the number of feedback messages (Fig. 12(a)) and the number of deleted symbols (Fig. 12(b)), noting that, however, the bounds are not tight.



(a) Number of ACK messages



(b) Number of ACKed symbols

Fig. 12. (a) Comparison of analytical and empirical results on the number of ACK messages transmitted by the Delete-and-Conquer codes (b) Comparison of analytical and empirical results on the number of ACKed symbols ($k = 128$, $\alpha = 15$).

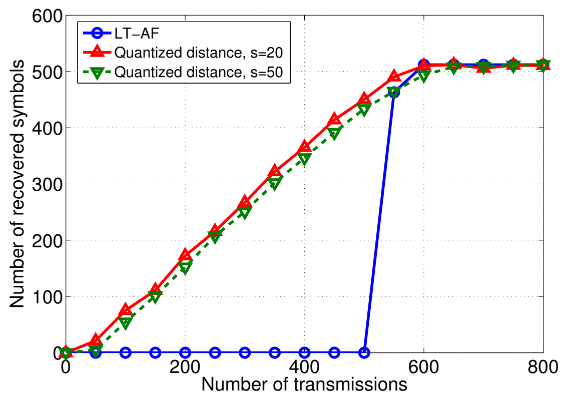


Fig. 13. Intermediate performance of codes with nonuniform symbol selection against the LT-AF codes ($k = 512$).

VII. Simulation Results

We evaluate the performance of rateless codes with nonuniform selection distributions against the Growth codes [14], Online codes proposed in [15], LT-AF codes [16], and LT feedback (LTF) codes [12].

A. General form

1) Intermediate performance: In many applications such as video streaming with real-time playback requirements, it is essential to partially recover some symbols before the recovery of entire frame. In this context, although LT codes are capacity-achieving, they lack real-time features; in other words, not many input symbols are decoded until the decoding process is almost complete. By incorporating a nonuniform selection distribution at the encoder, we aim to enhance the intermediate symbol recovery rate. Fig. 13 compares the performance of our codes with the LT-AF codes of Variable Node with Maximum Degree (LT-AF+VMD) [16], where the authors show that LT-AF codes can surpass previous rateless codes with feedback including SLT codes. One key point, however, is that the LT-AF decoder is not able to recover any symbol until at least k encoding symbols are received. As the results show, our scheme based on the Quantized distance method can achieve a high intermediate recovery rate. Moreover, the coding performance can be adjusted by tuning the parameter s (feedback transmission interval).

2) Coding overhead: Next we compare the total number of forward and feedback transmissions needed by our codes in comparison with the LT-AF+VMD codes. As the results in Table II show, our codes have a slightly better performance in terms of the number of forward transmissions, but the LT-AF codes require less feedback transmissions. It should be noted that amount of feedback in our codes can be adjusted using the parameter s, and that our codes are aimed to achieve a high intermediate symbol recovery rate, as the results in Fig. 13 show.

Algorithm	$k = 512$	
	Forward	Feedback
LT-AF + VMD	556.0	9.0
All-Distance	550.4	54.4
Quantized distance	555.2	55.0

TABLE II
Number of transmissions needed by the LT-AF codes and our codes with $s = 10$

Table III further illustrates the performance of our codes with the block length $k = 512$ symbols and as the feedback interval s increases. Similar to the previous results, the receiver is able to control the number of forward and feedback transmissions by changing the parameter s.

B. Primitive form

1) Intermediate performance: To investigate the progressive performance of the Delete-and-Conquer codes, we run simulations with the block length of $k = 512$. Results shown in Fig. 14, demonstrate that Growth codes proposed in [14] provides higher symbol recovery rate at the beginning, while Delete-and-Conquer achieves better performance when a small fraction of symbols are unrecovered (near the “knee”). On the other hand, Delete-and-Conquer scheme achieves better performance compared with the Online codes [15], noting that Delete-and-Conquer codes improve the intermediate performance with a lightweight utilization of the back channel (i.e., one bit feedback for each of a small fraction of received symbols).

As mentioned earlier, the authors in [12] have proposed a class of LT codes with feedback, which is referred to as LT feedback (LTF) codes. Similar to the Delete-and-Conquer scheme, LTF codes are also designed based on two principles: (i) excluding recovered symbols from future encoding, and (ii) degree distribution adaptation

Feedback interval	All-Distance		Quantized distance	
	Forward	Feedback	Forward	Feedback
$s = 5$	536.6	106.9	544.8	108.4
$s = 10$	550.4	54.4	555.2	55.0
$s = 50$	657.6	12.8	684.8	13.6
$s = 100$	758.6	7.0	759.4	7.2
$s = 500$	1155.7	2.0	1172.6	2.0

TABLE III
NUMBER OF TRANSMISSIONS NEEDED BY OUR CODES AS THE INTERVAL OF FEEDBACK TRANSMISSION INCREASES ($k = 512$)

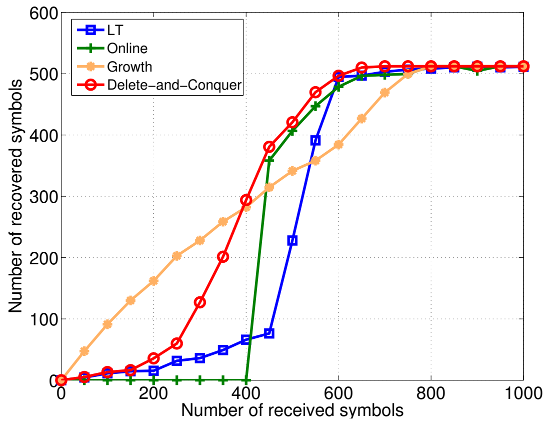


Fig. 14. Intermediate performance of the Delete-and-Conquer codes compared with other rateless codes ($k = 512$)

within feedback intervals. The feedback messages of the LTF codes include the identity of recovered symbols, and the receiver specifies the number of feedback and the point at which a feedback is initiated (obtained from an optimization step). The LTF encoder, in turn, optimizes the degree distribution within each feedback interval. Note that the Delete-and-Conquer degree distribution is merely based on the Robust Soliton distribution, and thus it does not introduce extra computation costs to the encoder.

In Fig. 15, we compare the intermediate performance of the LTF codes and the Delete-and-Conquer codes for $k = 128$ input symbols. From the results, one can notice that the Delete-and-Conquer method achieves a faster decoding rate at the beginning. However, due to the optimization steps used, the LTF codes achieve a smaller coding overhead compared with the Delete-and-Conquer codes. In particular, to successfully decode 128 input symbols, the LTF codes need 147.8 encoding symbols (averaged over 1000 runs of simulation), while the Delete-and-Conquer scheme requires 160.7 transmissions.

We also compare number of bits of feedback sent by the LTF method and the Delete-and-Conquer method. Note that in [12], the authors have not explicitly specified the format of LTF feedback messages, and thus we assume that the receiver allocates a vector of size k (number of input symbols) for the feedback message. Therefore, the receiver sets the value of those indices that correspond to the identity of recovered symbols, which results in a total of k bits of feedback. Alternatively, the receiver can explicitly feed the ID's of decoded symbols back, which requires $k \log(k)$ bits in total. In Fig. 16, we use the former method. As the results show, initially the Delete-and-Conquer method requires more feedback, but as the decoding progresses and more symbols are decoded, the number of bits of feedback by the LTF codes becomes dominant. In this simulation, we assume that transmissions are time-delimited, and thus each Delete-and-Conquer feedback is one bit. Finally, one can note

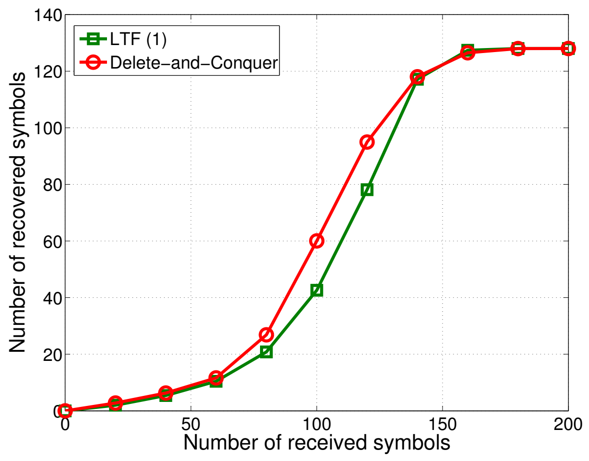


Fig. 15. Intermediate performance comparison of the LT feedback codes [12] with the Delete-and-Conquer codes for $k = 128$. LTF (1) denotes the LT feedback codes with a single feedback opportunity.

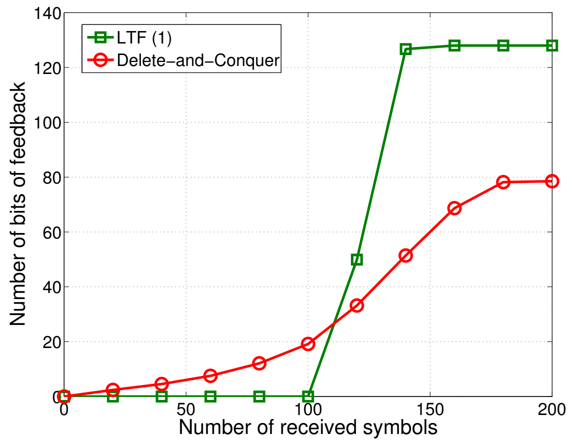


Fig. 16. Total number of bits of feedback sent by the LTF codes and the Delete-and-Conquer codes for $k = 128$.

that the second jump of the LTF feedback is due to the averaging effect over independent runs of simulation wherein for some runs there is one feedback message, and for some there is none (based on feedback condition).

2) Computational complexity: Computational costs at the encoder and decoder are mainly related to the average degree of input symbols. Fig. 17 shows the average degree of input symbols for different codes compared to the Delete-and-Conquer codes. As the results show, Delete-and-Conquer codes have a smaller average degree on input symbols, and hence they incur less computational complexity. Smaller average degree is due to incrementally dropping input symbols from the coding window.

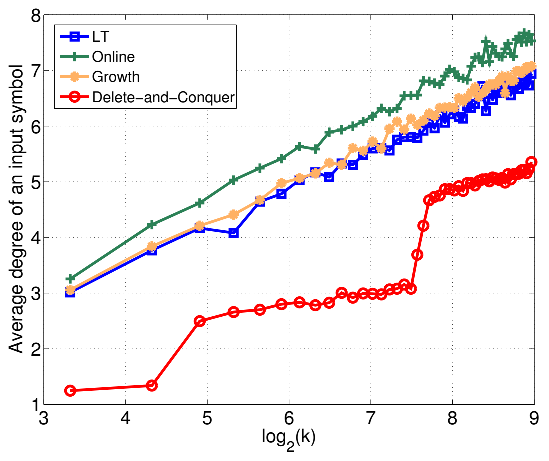


Fig. 17. Average degree of input symbols for various coding schemes

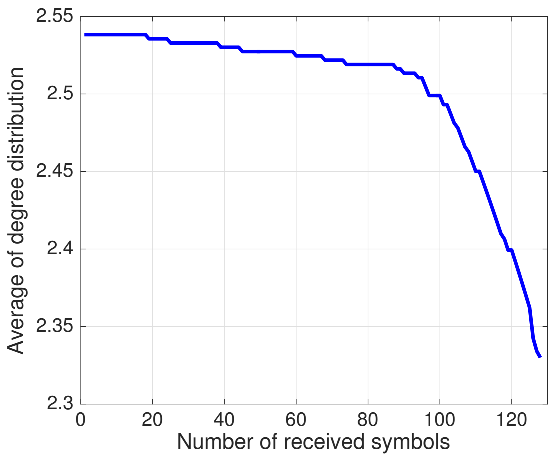


Fig. 18. Average of the degree distribution as coding progresses ($k = 128$).

3) Degree distribution evolution: Fig. 18 shows the average of the degree distribution as coding progresses. From the results, we notice that as more encoding symbols are received more input symbols are dropped and average degree of encoding symbols decreases. This, in turn, reduces the computational complexity of coding at both encoder and decoder.

VIII. Conclusion

In this paper, we have developed feedback-based rateless codes with nonuniform selection distributions. Our encoder estimates the decoder state using feedback information, and dynamically adjusts the selection distribution so that more helpful symbols (in terms of decoding progress) are assigned with a higher probability to be included in future encodings. As a result, we improve the intermediate performance of the underlying rateless codes and make them more suitable for applications with real-time decoding requirements. Our codes further support two important features: the decoder has full control of the *rate* and *timing* of feedback transmission. Our simulation results, backed by analysis, confirm that distance-type feedback paired with a nonuniform selection distribution achieves a high intermediate recovery rate. On the whole, rateless codes with nonuniform selection distributions help the encoder to optimize for the performance requirements dictated by the application. In the future works, we plan to further investigate the performance of nonuniform rateless codes in terms of decoding delay and *freshness* of recovered data at the decoder side.

Morteza Hashemi (S'10) is currently a postdoctoral researcher at the Ohio State University. He received his PhD in Electrical and Computer Engineering from Boston University (2015), and his B.S. in Electrical Engineering from Sharif University of Technology, Tehran, Iran (2011). His research interests include error correcting code, networks performance evaluation, and wireless communications.

Yuval Cassuto (S'02 - M'08 - SM'14) is a faculty member at the Department of Electrical Engineering, Technion – Israel Institute of Technology. His research interests lie at the intersection of the theoretical information sciences and the engineering of practical computing and storage systems.

During 2010-2011 he has been a Scientist at EPFL, the Swiss Federal Institute of Technology in Lausanne. From 2008 to 2010 he was a Research Staff Member at Hitachi Global Storage Technologies, San Jose Research Center. From 2000 to 2002, he was with Qualcomm, Israel R&D Center, where he worked on modeling, design and analysis in wireless communications.

He received the B.Sc degree in Electrical Engineering, summa cum laude, from the Technion, Israel Institute of Technology, in 2001, and the MS and Ph.D degrees in Electrical Engineering from the California Institute of Technology, in 2004 and 2008, respectively.

Dr. Cassuto has won the 2010 Best Student Paper Award in data storage from the IEEE Communications Society, as well as the 2001 Texas Instruments DSP and Analog Challenge $100,000 prize.

Ari Trachtenberg (S'96-M'00-SM'06) is a Professor of Electrical and Computer Engineering at Boston University. He received his PhD and MS in Computer Science from the University of Illinois at Urbana/Champaign, and his SB from MIT in Math/CS. His research interests include cyber security (smartphones, offensive and defensive), networking (security, sensors, localization); algorithms (data synchronization, file edits, file sharing), and error-correcting codes (rateless coding, feedback).

References

[1] M. Hashemi, A. Trachtenberg, and Y. Cassuto, “Delete-and-conquer: Rateless coding with constrained feedback,” in *51st Annual Allerton Conference*, 2013.

[2] M. Hashemi and A. Trachtenberg, “Near real-time rateless coding with a constrained feedback budget,” in *52nd Annual Allerton Conference*, 2014.

[3] M. Luby, “LT codes,” in *Annual Symposium on Foundations of Computer Science*, 2002, pp. 271–280.

[4] A. Shokrollahi, “Raptor codes,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[5] A. Beimel, S. Dolev, and N. Singer, “RT oblivious erasure correcting,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1321–1332, 2007.

[6] A. Hagedorn, S. Agarwal, D. Starobinski, and A. Trachtenberg, “Rateless coding with feedback,” in *IEEE INFOCOM*, 2009, pp. 1791–1799.

[7] J. W. Byers, M. Luby, and M. Mitzenmacher, “A digital fountain approach to asynchronous reliable multicast,” *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1528–1540, 2002.

[8] R. G. Gallager, “Low density parity check codes,” Ph.D. dissertation, Massachusetts Institute of Technology, 1960.

[9] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near shannon limit error-correcting coding and decoding: Turbo-codes,” in *IEEE International Conference on Communications*, vol. 2, 1993, pp. 1064–1070.

[10] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A digital fountain approach to reliable distribution of bulk data,” in *ACM SIGCOMM*, vol. 28, no. 4, 1998, pp. 56–67.

[11] D. J. MacKay, “Fountain codes,” in *IEE Proceedings-Communications*, vol. 152, no. 6, 2005, pp. 1062–1068.

[12] J. H. Sørensen, T. Koike-Akino, and P. Orlik, “Rateless feedback codes,” in *IEEE International Symposium on Information Theory*, 2012, pp. 1767–1771.

[13] E. Hyytia, T. Tirronen, and J. Virtamo, “Optimal degree distribution for LT codes with small message length,” in *IEEE INFOCOM*, 2007, pp. 2576–2580.

[14] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, “Growth codes: Maximizing sensor network data persistence,” in *ACM SIGCOMM*, vol. 36, no. 4, 2006, pp. 255–266.

[15] Y. Cassuto and A. Shokrollahi, “On-line fountain codes for semi-random loss channels,” in *IEEE Information Theory Workshop (ITW)*, 2011, pp. 262–266.

[16] A. Talari and N. Rahnavard, “Robust LT codes with alternating feedback,” *Computer Communications*, 2014.

[17] J. Dai, F. Zesong, S. Chenglin, C. Congzhe, and K. Jingming, “LT codes with limited feedback,” in *Computer and Information Technology (CIT), 2014 IEEE International Conference on*. IEEE, 2014, pp. 669–673.

[18] S. Sanghavi, “Intermediate performance of rateless codes,” *arXiv preprint cs/0612075*, 2006.

[19] A. Talari and N. Rahnavard, “Rateless codes with optimum intermediate performance,” in *IEEE Global Telecommunications Conference*, 2009, pp. 1–6.

[20] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, “An optimal partial decoding algorithm for rateless codes,” in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 2731–2735.

[21] N. Rahnavard, B. N. Vellambi, and F. Fekri, “Rateless codes with unequal error protection property,” *IEEE Transactions on Information Theory*, vol. 53, no. 4, pp. 1521–1532, 2007.

[22] H. Lu, J. Cai, and C. H. Foh, “Joint unequal loss protection and LT coding for layer-coded media delivery,” in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–5.

[23] J. H. Sørensen, P. Popovski, and J. Østergaard, “On the role of feedback in LT codes,” *arXiv preprint arXiv:1012.2673*, 2010.

[24] M. Hashemi and A. Trachtenberg, “CDP: a coded datagram transport protocol bridging UDP and TCP,” in *Proceedings of the 8th ACM International Systems and Storage Conference*. ACM, 2015, p. 11.

[25] S. M. Ross, *Introduction to probability models*. Access Online via Elsevier, 2006.