Computer Communications 49 (2014) 60–68

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Robust LT codes with alternating feedback

Ali Talari [a,*], Nazanin Rahnavard [b]

[a] Oklahoma State University, Stillwater, OK 74078, United States
[b] University of Central Florida, Orlando, FL 32806, United States

ARTICLE INFO

Article history:
Received 24 May 2011
Received in revised form 21 April 2014
Accepted 21 April 2014
Available online 2 May 2014

Keywords:
LT codes
Rateless codes
Forward error correction codes
Erasure channel
Feedback channel

ABSTRACT

In this paper, we propose *robust* LT codes with *alternating feedback* (LT-AF codes), which lightly utilize the feedback channel and surpass the performance of existing LT codes with feedback. In LT-AF codes, we consider a *loss prone* feedback channel for the first time and propose the encoder to generate degree-one output symbols (encoded symbols) only in acknowledgement to the reception of feedbacks. Therefore, LT-AF codes become *robust* against feedback losses meaning that their performance does not deteriorate even at high feedback loss rates in contrast to previous work. To realize this, we design a *new* and *parameterless* coding degree distribution for LT-AF coding based on *Ideal-Soliton* (IS) distribution of LT codes.

In addition, we design a new feedback scheme and use it in conjunction with an existing feedback method. Therefore, in LT-AF codes the decoder can *alternate* between either types of feedback based on its status. To generate our new type of feedback, we propose *three* novel algorithms to analyze the buffered output symbols at the decoder. We will show that LT-AF codes require a significantly lower coding overhead for a successful decoding.

1. Introduction

Rateless codes [1–3] (also called *fountain codes*) are modern and efficient *forward error correction* (FEC) codes with *LT codes* [1] being the first practical realization of such codes. As the name suggests, these codes do not have a fixed coding rate and are *universal* in the sense that they are simultaneously near optimal for every erasure channel with varying or unknown erasure rate $\varepsilon \in [0, 1)$ [1].

In the LT decoding process, when the decoding succeeds, a single-bit feedback is sent to the LT encoder to inform it with the decoding success. Clearly, this feedback channel remains *unused* during the transmission. Consequently, previous studies [4–8] propose to employ the feedback channel during the transmission to further improve the performance of LT codes.

The existing LT codes with feedback are designed such that the decoder informs the encoder with the number of successfully decoded *input symbols* (source symbols) [4–6], a suitable input symbol for decoding [7], or the index of some of the previously recovered input symbols [8]. Further, existing works have considered a lossless feedback channel, which may not always be the case. Therefore, existing codes may experience high performance degradation in the presence of feedback loss.

In this paper, we design *robust* LT codes with *alternating* feedback (*LT-AF* codes), in which we consider a realistic feedback channel with unknown or varying erasure rate $\varepsilon_{fb} \in [0, 1)$. Note that we design LT-AF codes for $\varepsilon_{fb} < 1$; hence, these codes are not defined for $\varepsilon_{fb} = 1$, i.e., 100% feedback loss. We propose to employ the degree-one output symbols (encoded symbols) generated at the encoder as acknowledgment to the reception of feedbacks, which required the design of a new degree distribution. Consequently, we design a *new* and *parameterless* coding degree distribution for LT-AF codes based on Ideal-Soliton (IS) degree distribution of LT codes [1]. We will see that LT-AF codes show only a slight performance degradation even for large values of ε_{fb} in contrast to the existing work; thus, we call them robust. Therefore, LT-AF codes are robust against high feedback channel loss rates.

In addition, we design a new type of feedback for LT codes and propose to combine it with the existing feedback generation technique proposed in [4]. Therefore, the decoder may issue *two types* of feedback based on its *needs*. The decoder generates the first type of feedback based on [4] to inform the encoder with the number of successfully decoded input symbols, and the second type of feedback to request a specific input symbol that makes a significant progress in the decoding of the previously received and *buffered* output symbols. To generate our new proposed type of feedback,

* Corresponding author. Tel.: +1 (405) 744 4669.
E-mail addresses: ali.talari@okstate.edu (A. Talari), nazanin@eecs.ucf.edu (N. Rahnavard).

http://dx.doi.org/10.1016/j.comcom.2014.04.010
0140-3664/© 2014 Elsevier B.V. All rights reserved.

we propose three novel algorithms (with a trade-off in their algorithm complexity and performance) to analyze the decoder's buffer and select suitable input symbols to request.

The paper is organized as follows. In Section 2, we provide a brief review on LT codes. Next, in Section 3 we review the existing rateless codes with feedback. In Section 4, we propose and analyze LT-AF codes. Section 5, reports the performance of LT-AF codes and evaluates its robustness against feedback channel loss. Finally, Section 6 concludes the paper.

2. Overview of LT codes

First, without loss of generality and for simplicity, let us assume that the input and output symbols are binary symbols. LT codes [1] have simple encoding and decoding procedures as follows.

LT encoding: In LT encoding of k input symbols, first an output symbol *degree* d is chosen from the *Robust-Soliton* (RS) degree distribution [1] $\{\mu_1, \mu_2, \ldots, \mu_k\}$, where μ_i is the probability that $d = i$ and $\sum_{i=1}^{k} \mu_i = 1$. This degree distribution can also be specified by its generator polynomial $\mu(x) = \sum_{i=1}^{k} \mu_i x^i$. Next, d input symbols are chosen *uniformly at random* from the k input symbols and are *XOR*ed to generate an output symbol. We refer to the d contributing input symbols in forming an output symbol as its *neighbors*. This procedure can be potentially repeated infinite number of times to generate a *limitless* number of output symbols. However, the encoder stops generating output symbols upon receiving the single feedback indicating the decoding success.

The most important parameter of LT codes is the RS degree distribution designed by Luby [1]. The RS distribution $\mu(.)$ is obtained by combining the *ideal-Soliton* (IS) distribution $\rho(.)$ and distribution $\tau(.)$ given by

$$\rho(i) = \begin{cases} \frac{1}{k} & i = 1, \\ \frac{1}{i(i-1)} & i = 2, \ldots, k, \end{cases} \tag{1}$$

and

$$\tau(i) = \begin{cases} \frac{R}{ik} & i = 1, \ldots, \frac{k}{R} - 1, \\ \frac{R}{k} \ln(\frac{R}{\delta}) & i = \frac{k}{R}, \\ 0 & i = \frac{k}{R} + 1, \ldots, k, \end{cases}$$

respectively, where $R = c \ln\left(\frac{k}{\delta}\right)\sqrt{k}$, and δ and c are two *tuneable* parameters [1]. It is easy to see that the average degree of output symbols with IS distribution is $\rho'(1) = \sum_{i=1}^{k} i\rho(i) = H(k) \approx \ln k$, where $\rho'(x)$ is the first derivative of $\rho(x)$ with respect to its variable x, and $H(k)$ is the kth Harmonic number [1]. Finally, RS degree distribution $\mu(.)$ is obtained by

$$\mu(i) = \frac{\rho(i) + \tau(i)}{\beta}, i = 1, \ldots, k, \tag{2}$$

where $\beta = \sum_{i=1}^{k} \rho(i) + \tau(i)$.

LT decoding: Rateless decoding is *iteratively* performed upon arrival of *new* output symbols as follows. The decoder finds an output symbol such that the value of all but one of its neighboring input symbols is known. It recovers the value of the unknown input symbol by simple bitwise XOR operations. This process is repeated until no such an output symbol exists. If all k input symbols are recovered a single-bit feedback indicating the decoding success is issued.

Note that the set of output symbols reduced to degree-one[1] is called the *ripple*. If the ripple becomes empty, the decoding stops and the decoder needs to wait for new output symbols to join the ripple to continue the decoding. In addition, when an output symbol in the ripple decodes an input symbol, its degree reduces to zero and is removed from the decoding process.

Clearly, due to randomness in forming of LT output symbols they are statistically independent. Consequently, as shown in [1] the only condition for a successful LT decoding is the delivery of a *certain number* of output symbols. Let ϵ be the *required coding overhead* to have a successful decoding with high probability (w.h.p.), i.e., $(1 + \epsilon) \cdot k$ coded symbols are enough to decode k input symbols w.h.p. Clearly, it is ideal to have $\epsilon = 0$, i.e., full recovery of k input symbols from k output symbols. Further, let γ denote the *received* coding overhead (meanwhile the transmission is in progress). Therefore, $\gamma \cdot k$ is the number of received output symbols at the receiver, and we have $0 \leqslant \gamma \leqslant (1 + \epsilon)$.

Although the distribution $\mu(.)$ is asymptotically capacity achieving, i.e., $\epsilon \to 0$ as $k \to \infty$ [1], for small values of k (several hundreds to thousands), the value of ϵ becomes significantly larger than 0 [1,9,10]. This results in an inefficient FEC coding. Therefore, we exploit the feedback channel to obtain a much smaller ϵ for a finite k in LT-AF coding.

3. Related work

Authors in [4] proposed *shifted LT* (SLT) codes to exploit the available feedback channel. They have shown that when n input symbols have been recovered at the decoder, the degree of each arriving output symbol decreases by an expected $\frac{k-n}{k}$ fraction (due to earlier recovery of their neighboring input symbol). Therefore, they propose to *shift* the RS distribution such that its average degree $\mu'(1)$ is increased by $\frac{k}{k-n}$, where $\mu'(x)$ is the first derivative of $\mu(x)$ with respect to its variable x. With this setup, arriving output symbols at the decoder *always* maintain an RS degree distribution regardless of the value of n. SLT codes have a considerably improved performance compared to regular LT codes. In this paper, we make some changes to the idea of *distribution shifting* proposed in SLT codes and employ it in the design of the LT-AF codes, while showing that LT-AF codes outperform SLT codes.

In contributions [5,6], *Growth codes* and *RT-oblivious codes* have been proposed, respectively, which have basically the same setup. In these algorithms, as n increases and reaches to certain thresholds a feedback indicating that decoder has achieved the corresponding threshold is issued. Therefore, the encoder gradually increases the degree of output symbols on-the-fly based on the feedbacks such that the *instantaneous* decoding probability of each delivered output symbol is maximized in a greedy fashion. Since Growth and RT-oblivious codes only consider the instantaneous recovery probability of each output symbol upon reception, they may not perform as well as SLT and LT-AF codes.

In [11], authors have proposed to analyze the receiver's buffer and find the current optimal degree of output symbols such that the received output symbols at the decoder can be reduced to degree one or two with the highest probability. The idea proposed in [11], has some similarities to one of the techniques that LT-AF codes employ to generate the second type of feedback.

Authors in [7] propose to employ IS degree distribution $\rho(.)$ for LT coding. They have proposed to start decoding when an overhead of $\gamma = 1$ has been delivered to the decoder. When the decoding cannot be progressed further while some input symbols remain unrecovered, a randomly selected input symbol that is a neighbor of an output symbol of degree two is requested from the encoder. This algorithm is performed iteratively until the decoding completes. Despite the advantages of the algorithm proposed in [7], in this scheme many feedbacks are issued back-to-back as soon as γ exceeds 1. Further, during the iterative request process all degree-two output symbols may be *consumed* (decoded), while more input symbols remain unrecovered.

[1] An output symbol is said to have degree one if all but one of its neighboring input symbols is known.

4. LT-AF codes

Let $\Omega_{k,n}(.)$ denote the degree distribution of LT-AF codes for a data-block of length k when n input symbols are already recovered at the decoder. We adopt the idea of SLT codes [4], and propose to *shift* $\Omega_{k,n}(.)$ based on the knowledge of n (see Section 3). Therefore, we allow the decoder to issue the first type of feedback referred to by fb_1, which is used to keep the encoder updated with the current value of n (number of decoded input symbols at decoder).

Although IS distribution is solely designed for the theoretical analysis of RS distribution, we slightly modify and employ it at the encoding phase of LT-AF codes in combination with two types of feedback. The IS distribution is tuned for $\epsilon = 0$ such that at each decoding iteration in expectation exactly one input symbol is recovered and only one output symbol is reduced to degree 1 and is added to the ripple. The single output symbol in the ripple can decode one input symbol in the next iteration. Since on average only a single degree-one output symbol is generated for k output symbols (note that $\rho(1) = \frac{1}{k}$), the IS distribution would realize an optimal coding/decoding, i.e., complete recovery of k input symbols from k output symbols and $\epsilon = 0$.

However, due to inherent randomness and uncertainties in the output symbol generation there is a high probability that an output symbol does not reduce to degree one when an input symbol is recovered. Consequently, the ripple becomes empty and the decoding stops although undecoded output and unrecovered input symbols are still remaining. Therefore, while the IS distribution shows an ideal behavior in terms of the expected number of encoding symbols needed to recover the data, it is quite fragile and impractical [1]. Despite this, we can easily see that if we exploit the feedback channel and *request a suitable* input symbol (which is an output symbol of degree 1), the decoding may continue and the IS distribution becomes practical. Therefore, we allow the decoder to request its desired input symbols employing the second type of feedback referred to by fb_2.

Moreover, to design $\Omega_{k,n}(.)$ we propose to modify the IS distribution such that the encoder does not generate *any* degree-one output symbol unless in response to a received feedback (fb_1 or fb_2). With this setup we can utilize the reception of a degree-one output symbol at the decoder as its *acknowledgement* for the successful delivery of the feedback. Therefore, the encoder generates a degree-one output symbol if and only if it has received a fb_1 or fb_2, and the lack of the arrival of an output symbol at the decoder with degree-one after issuing a fb_1 or fb_2 clearly indicates a feedback loss. Consequently, all feedback losses can be identified by the decoder and a feedback retransmission is performed. This modification makes LT-AF codes robust against feedback channel loss, and the decoding recovery rate of LT-AF codes does not considerably degrade at *high* feedback channel loss rates $\varepsilon_{fb} \in [0,1)$ in contrast to existing work [4–8].

Clearly, the degree-one output symbol generated following the arrival of a fb_2 at the encoder contains the requested input symbol. However, after a fb_1 this output symbol contains a randomly selected input symbol since fb_2 is intended to inform the encoder with n. Let $\Omega_{k,n}(x) = \sum_{j=1}^{k} \Omega_{k,n,j} x^j$, where $\Omega_{k,n,j}$ is the probability of selecting degree j to generate an LT-AF output symbol when n input symbols are decoded at the receiver. Inspired by the distribution shifting idea from [4] and the setting of our problem we define $\Omega_{k,n,j}$ as

$$\Omega_{k,n,j} = \begin{cases} 0 & j = 1, \\ \frac{k-\alpha_n}{k-1}\rho_{k-n}(i) & j = \left\lceil \frac{i}{1-\frac{n}{k}} \right\rceil, 2-\alpha_n \leqslant i \leqslant k-n, \end{cases} \tag{3}$$

where $\lceil . \rceil$ is the ceiling function, $\rho_k(.)$ is the IS distribution for a data-block of length k, and α_n is given by

$$\alpha_n = \begin{cases} 0 & n = 0, \\ 1 & n > 0. \end{cases} \tag{4}$$

Note that for $n = k-1$, (3) results in $\Omega_{k,n,k} = 1$, i.e, at the end all the k input symbols are added to form an output symbol.

Lemma 1. *For any* $0 \leqslant n < k, \Omega_{k,n,j}, j = 2, \ldots, k$ *is a probability distribution.*

Proof. First, we discuss $1 \leqslant n < k$, and investigate the special case of $n = 0$ next. We need to prove that mapping $j = \left\lceil \frac{i}{1-\frac{n}{k}} \right\rceil$ is a one-to-one mapping from $1 \leqslant i \leqslant k-n$ to $2 \leqslant j \leqslant k$ for $1 \leqslant n < k$. In other words, each integer i in the range $1 \leqslant i \leqslant k-n$ is mapped to a distinct j in $\{2,3,\ldots,k\}$. We have $\frac{1}{1-\frac{n}{k}} \geqslant 1$ for $1 \leqslant n < k$. Therefore, $\frac{i}{1-\frac{n}{k}}$ and $\frac{i'}{1-\frac{n}{k}}$ will differ by at least one for any two different integers i and i'. Accordingly, $j = \left\lceil \frac{i}{1-\frac{n}{k}} \right\rceil$ and $j' = \left\lceil \frac{i'}{1-\frac{n}{k}} \right\rceil$ will also differ by at least one.

Next, we need to show that $1 \leqslant i \leqslant k-n$ results in $2 \leqslant j \leqslant k$. We note that j is a non-decreasing function of i. Therefore, it is sufficient to show that $j \geqslant 2$ when $i = 1$ and $j \leqslant k$ when $i = k-n$. The former is clear considering that $\frac{1}{1-\frac{n}{k}} \geqslant 1$ we have $j = \left\lceil \frac{1}{1-\frac{n}{k}} \right\rceil \geqslant 2$. For the latter, we note that $j = \left\lceil \frac{k-n}{1-\frac{n}{k}} \right\rceil = k$.

Based on the above discussions and having $\alpha_n = 1$, we conclude that $\sum_{j=2}^{k} \Omega_{k,n,j} = \sum_{i=1}^{k-n} \rho_{k-n}(i) = 1$.

In the special case of $n = 0$, we have $j = \left\lceil \frac{i}{1-\frac{n}{k}} \right\rceil = i$. However to avoid the generation of degree one symbols, we have set $\Omega_{k,0,j} = 0$ for $j = 1$ in (3) (which has been equal to $\rho_k(1) = \frac{1}{k}$ in the IS distribution). To compensation for this, we scale $\Omega_{k,0,j}$ for $2 \leqslant j \leqslant k$ by a factor $\frac{k}{k-1}$. Consequently, $\sum_{j=2}^{k} \Omega_{k,0,j} = \frac{k}{k-1} \sum_{i=2}^{k} \rho_k(i) = 1$. □

Lemma 2. *The average degree of a check node generated employing* $\Omega_{k,n}(.)$ *distribution is*

$$\Omega'_{k,n}(1) = \sum_j j\Omega_{k,n,j} \approx \frac{k}{k-n} \ln(k-n), \tag{5}$$

where $\Omega'_{k,n}(x)$ *is the first derivative of* $\Omega_{k,n}(x)$ *with respect to its variable* x.

Proof. The average degree of LT-AF distribution is

$$\Omega'_{k,n}(1) = \sum_{j=2}^{k} j\Omega_{k,n,j} \approx \sum_{i=2-\alpha_n}^{k-n} \left\lceil \frac{i}{1-\frac{n}{k}} \right\rceil \frac{k-\alpha_n}{k-1} \rho_{k-n}(i),$$
$$= \frac{k}{k-n} \frac{k-\alpha_n}{k-1} \sum_{i=2-\alpha_n}^{k-n} i\rho_{k-n}(i). \tag{6}$$

For $n = 0$, we have $\alpha_n = 0$; which gives

$$\sum_{i=2}^{k-n} i\rho_{k-n}(i) = \sum_{i=1}^{k-n-1} \frac{1}{i} = H(k-n-1). \tag{7}$$

For $n > 0$, we have $\alpha_n = 1$, which gives

$$\sum_{i=1}^{k-n} i\rho_{k-n}(i) = \frac{1}{k-n} + \sum_{i=2}^{k-n} \frac{1}{i-1} = H(k-n). \tag{8}$$

Given $k \gg 1$, we have $k-1 \approx k$. Consequently, using (6) and (7), or (6) and (8), we have $\Omega'_{k,n}(1) \approx \frac{k}{k-n} H(k-n) \approx \frac{k}{k-n} \ln(k-n)$. □

4.1. Generating fb_1

Obviously, the encoder is not always aware of the current value of n at the decoder unless its knowledge about n is updated by a

fb_1. Initially, the encoder assumes $n = 0$ and employs the degree distribution $\Omega_{k,0}(.)$ to generate output symbols. Let n_r denote the most recent reported value of n using a fb_1. Similar to [4], we propose the decoder to generate a fb_1 when $\Omega'_{k,n}(1) - \Omega'_{k,n_r}(1) \geqslant \sqrt{\ln k}$, i.e., average degree of $\Omega_{k,n}(.)$ increases by at least $\sqrt{\ln k}$. Let n_i be the *threshold* that for $n \geqslant n_i$ the ith fb_1 is generated. In the following lemma we give the expression for n_i.

Lemma 3. *In LT-AF codes with data-block length k, n_i the threshold of n for which ith fb_1 is issued is recursively obtained as follows.*

$$n_0 = 0$$
$$n_i = \left\lceil k + \frac{W_{-1}(-A_i(k))}{A_i(k)} \right\rceil, i > 0, \tag{9}$$

where $A_i(k) = \frac{1}{k}\left(\sqrt{\ln k} + \frac{k}{k-n_{i-1}} \ln(k - n_{i-1})\right)$ and $W_m(.)$ is the mth root of Lambert W-Function (the Lambert W-Function is defined as the inverse function of $f(x) = x \exp x$ [12]).

Proof. Let us first analyze n_1 the value of n that initiates the first fb_1. Since before the first fb_1 no distribution shifting occurs we have $\Omega'_{k,n_0}(1) = \Omega'_{k,0}(1) = \ln k$. Therefore, the first fb_1 is issued for a value of n_1 that $\Omega'_{k,n_1}(1) - \Omega'_{k,0}(1) = \sqrt{\ln k}$. Using Lemma 2 we have

$$\frac{k}{k-n_1}\ln(k-n_1) - \frac{k}{k-n_0}\ln(k-n_0) = \sqrt{\ln k}, \tag{10}$$

which gives

$$\frac{\ln(k-n_1)}{k-n_1} = \frac{1}{k}\left(\sqrt{\ln k} + \frac{k}{k-n_0}\ln(k-n_0)\right). \tag{11}$$

Next, let $A_i(k) = \frac{1}{k}\left(\sqrt{\ln k} + \frac{k}{k-n_{i-1}}\ln(k-n_{i-1})\right)$. Employing Lambert's W function, we have

$$k - n_1 = -\frac{W_{-1}(-A_1(k))}{A_1(k)}, \tag{12}$$

which gives

$$n_1 = \left\lceil k + \frac{W_{-1}(-A_1(k))}{A_1(k)} \right\rceil. \tag{13}$$

Further, we can easily see that n_2 can be obtained from $\Omega'_{k,n_2}(1) - \Omega'_{k,n_1}(1) = \sqrt{\ln k}$, which in the same way gives

$$n_2 = \left\lceil k + \frac{W_{-1}(-A_2(k))}{A_2(k)} \right\rceil. \tag{14}$$

Finally, we have $\Omega'_{k,n_i}(1) - \Omega'_{k,n_{i-1}}(1) = \sqrt{\ln k}$ that proves the lemma. □

Lemma 3 gives the value of n for which fb_1's are generated. In Fig. 1, we have depicted $\frac{n_i}{k}, i \in \{1, 2, \ldots, 5\}$ versus k. From Fig. 1, we can see that $\frac{n_i}{k}$ decreases as k increases. As an example, we can see that at $k = 10^2$ the first and the second fb_1's are issued at $n \geqslant 39$ and $n \geqslant 58$, respectively. Further, for $k = 10^4$ the first and the second fb_1's are issued at $n \geqslant 2740$ and $n \geqslant 4346$, respectively.

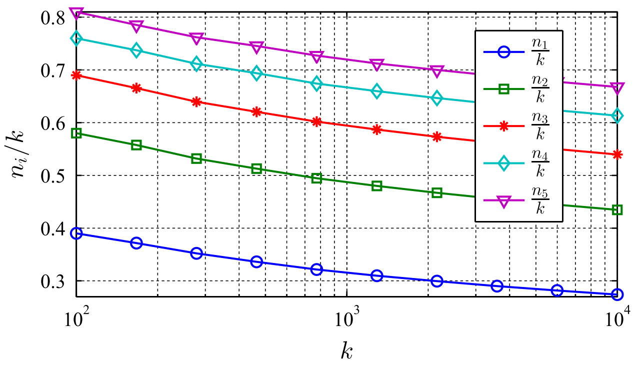


Fig. 1. Values of $\frac{n_i}{k}, i \in \{1, 2, \ldots, 5\}$ versus k.

4.2. Generating fb_2

Since in LT-AF coding no degree-one output symbol is generated, no decoding is performed and we have $n = 0$ until some degree-one output symbols are requested using fb_2's. The idea to generate fb_2 is to *smartly* and *greedily* choose and request an input symbol, whose reception at the decoder will result in the recovery of the maximum number of input symbols.

It is well-known that LT codes have *all-or-nothing* decoding property (also called waterfall phenomenon) [1], where an abrupt jump in the ratio of decoded input symbols occurs at a γ close to $1 + \epsilon$. Therefore, transmission of fb_2's before $\gamma = 1$ does not considerably contribute to decoding progress. Therefore, we propose to generate fb_2's only when γ exceeds 1. Note that authors in [7] have employed the same idea to determine the start point of their single type of feedback.

To uniformly distribute fb_2's and to avoid feedback channel congestion, an LT-AF decoder issues a fb_2 on the reception of every Dth output symbol (starting from kth received output symbols, or equivalently $\gamma = 1$). In other words, a fb_2 is generated when the number of received output symbols at the decoder is equal to $k, k + D, k + 2D, \ldots$. Clearly, a smaller D results in generating fb_2 more frequently by the decoder.

The performance of LT-AF codes greatly varies with D. Clearly, setting $D = 1$, i.e., one feedback per every received output symbol, results in the optimal LT-AF coding with the lowest coding overhead. However, such a feedback scheme is not desirable due to huge number of generated feedbacks and possibly congesting the feedback channel. To have a fair comparison with SLT codes, we experimentally set $D = \ln k$ so that the number of generated feedbacks in LT-AF codes becomes less or equal to the number of feedback in SLT codes as we later see.

Let us first describe the structure of input and output symbols in the buffer of a decoder. Input and received output symbols of an LT code at a decoder can be viewed as vertices of a *bipartite* graph G. The input symbols are the *variable nodes* $v_i, i \in \{1, \ldots k\}$ and the output symbols are the *check nodes* $c_j, j \in \{1, 2, \ldots, \gamma k\}$ [3,13], and they are connected to their neighbors denoted by $\mathcal{N}(v_i)$ and $\mathcal{N}(c_j)$, respectively, with undirected edges.

During data transmission some variable nodes $v_i, i \in \{1, \ldots k\}$ are decoded and some check nodes $c_j, j \in \{1, 2, \ldots, \gamma k\}$ are reduced to degree zero and are both removed from the decoding graph G. Let us refer to the set of remaining <u>u</u>n*decoded* <u>v</u>*ariable* nodes by V_{un} and the set of <u>*buff*</u>*ered* <u>*c*</u>*heck* nodes with a degree higher than one by C_{buff}. We remind that the set of check nodes with degree 1 is called the ripple. Fig. 2 illustrates such a graph G at a decoder at $\gamma = 1$ for $k = 7$. Note that we interchangeably employ the terms variable and check nodes for input and output symbols, respectively, in the rest of the paper.

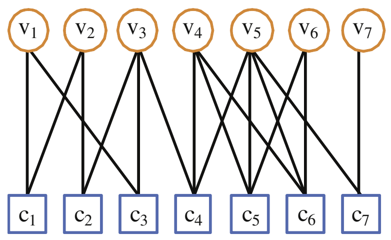


Fig. 2. The bipartite graph representing the input and the output symbols of a LT-AF code at the buffer of a decoder.

It is important to note that the design of fb_2 is to greedily decode as many as possible input symbols so that decoding succeeds at a smaller ϵ. However, as discussed earlier as n increases (where the decoder is closer to the end of decoding) the average degree of check nodes should be increased to decrease the probability of a check node being useless (due to earlier recovery of all of their neighboring variable nodes). This is the rationale to employ the distribution shifting and fb_1 along with fb_2. In the next sections, we devise three algorithms to analyze the graph G at the decoder and greedily select a suitable variable node to generate fb_2's.

4.2.1. Generating fb_2 based on Variable Node with Maximum Degree (VMD)

One insight in choosing a suitable variable node is requesting the variable node $v_i \in V_{\text{un}}$ with the *maximum degree*. Such a selection greedily removes the largest number of edges in the first step of decoding after the delivery of the respective input symbol. Based on this idea we propose an algorithm called "*Variable Node with Maximum Degree*" (VMD), where the decoder requests the variable node with the highest degree in its current decoding graph to issue a fb_2. For instance, in Fig. 2 VMD would choose and request v_5. On the arrival of c_8 containing only v_5, the decoding graph reduces to the graph shown in Fig. 3, where the dashed nodes and edges are removed from graph G. We can see that c_7 is added to the ripple, which recovers v_7 in the next decoding iteration. Note that at this step the ripple becomes empty and decoding stalls; hence, we have $C_{\text{buff}} = \{c_1, c_2, \ldots, c_6\}$ and $V_{\text{un}} = \{v_1, v_2, v_3, v_4, v_6\}$. We can see that VMD greedily removes the largest possible number of edges from G and decreases the degree of many check nodes.

Although VMD seems naïve, we will later see that it greatly improves the performance of LT codes with feedback, while it has a low computational complexity.

4.2.2. Generating fb_2 based on Longest Degree-Two Chain (LDC)

Although VMD's complexity is suitably low, it aims for the recovery of as many as possible input symbols by removing the largest number of edges from the decoding graph only in the first step of iterative decoding. However, removing the largest number of edges in the first decoding iteration does not guarantee decoding of the highest number of variable nodes. Therefore, we propose a second algorithm "*Longest Degree-2 Chain*" (LDC), that considers the subsequent decoding iterations as well.

From LT-AF distribution, we observe that at $\gamma = 1$ no decoding can be performed; hence, we have $C_{\text{buff}} = \{c_1, c_2, \ldots, c_k\}$ and $V_{\text{un}} = \{v_1, v_2, \ldots, v_k\}$. In such a decoding graph, on average 50% of the check nodes are of degree-two since $\Omega_{k,0,2} = 0.5$. Consider a decoding graph G_2, which is formed by check nodes of degree-two and their respective neighbors, i.e., $G_2 = \{(v_i \bigcup c_j) | c_j \in C_{\text{buff}}, |\mathcal{N}(c_j)| = 2, v_i \in V_{\text{un}}, v_i \in \mathcal{N}(c_j)\}$. Indeed, G_2 is the decoding graph *induced* by degree-two check nodes.

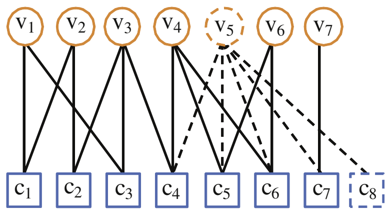


Fig. 3. The decoding bipartite graph G after the reception of the requested variable node v_5 employing VMD. Dashed nodes and edges will be removed from G after the reception of c_8.

Definition. By investigating the decoding graph G_2 we observe that some check nodes along with $n_v > 1$ variable nodes form structures that the delivery of the *any* of n_v variable nodes results in the decoding of all other $n_v - 1$ variable nodes. We call such a structure decoding *chain* of length n_v.

For instance, a single degree 2 check node forms a chain of length $n_v = 2$ since knowing the value of either of its neighboring variable node results in the decoding of the other one. Fig. 4 shows two chains of length $n_v = 4$ with different structures. The graph G_2 obtained from G in Fig. 2 has a chain of length $n_v = 3$ including v_1, v_2, and v_3 and a chain of length $n_v = 2$ including v_5 and v_7. We emphasize that this rule only holds for check nodes of degree-two as we discuss further in the next section.

From Fig. 4, we can see that the degree of variable nodes does not affect the length of chains, and the chains extend as far as the variable nodes are connected to degree-two check nodes. Based on our discussion, the decoder finds all chains of degree 2 and randomly selects a variable node from the *longest* chain. Next, this variable node is requested employing a fb_2. To give an example of the size of the chain, for $k = 10^4$ and $\gamma = 1$ we empirically find $n_v \approx 250$. Hence, on average the first fb_2 generated employing LDC decodes 250 variable nodes out of $k = 10^4$. Later, we will see that LDC has higher complexity compared to VMD while it surpasses VMD's performance.

In [11], authors have analyzed the degree-2 chain of decoder's buffer and analytically found the optimal instantaneous degree that maximizes the probability that output symbols are reduced to degree one or two. The proposed method in [11] has many similarities to LDC except that LDC uses the information from the longest degree-2 chain to request an appropriate input symbol using feedback.

4.2.3. Generating fb_2 based on Full Variable Node Decoding (FVD)

LDC is designed considering the graph induced by degree-two check nodes only. However, higher degree check nodes are also present in the decoding graph, which may form more complex decoding chains. When higher degree check nodes are also considered, the main rule of the decoding chains is violated. That is, if recovery of a particular variable node v_i results in the recovery of a set of variable nodes $v_j \in V_i$, the delivery of any of $v_j \in V_i$ does not necessarily guarantee the decoding of v_i. Therefore, no decoding chain can be defined in this case. Consequently, for all $v_i \in V_{\text{un}}$ we need to find V_i the set of variable nodes that are decoded as a result of v_i's delivery.

In Fig. 5, we have illustrated a part of a decoding graph of a LT-AF code. We can observe that the delivery of v_2 results in the decoding of v_1 and v_3, while delivery of v_1 does not decode v_2 and v_3. This is clearly due to considering c_2 that is a degree-three check node in the decoding chain.

Therefore, we propose "*Full Variable Node Decoding*" (FVD) that considers all check nodes with any degree, and provides the *optimal* selection of variable nodes to issue fb_2's. FVD is performed once when a fb_2 is to be issued as follows.

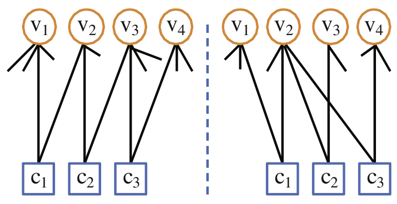


Fig. 4. Two decoding chains with $n_v = 4$.

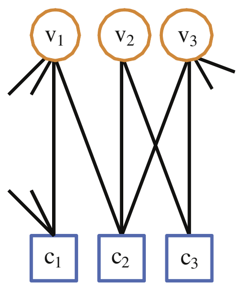


Fig. 5. Chain of decoding considering check nodes with degrees higher than two. Delivery of v_1 does not necessarily decode v_2 and v_3 while delivery of v_2 results in decoding of v_1 and v_3.

1. For all $v_i \in V_{\text{un}}$ find V_i by running *dummy* decodings.
2. Find $i^* = \text{argmax}_i|V_i|$ and generate a fb_2 containing i^*.

FVD finds the variable node v_{i^*} for fb_2 that results in the *largest* number of decodings considering the full graph G. As we later see, FVD has a much higher complexity than LDC and VMD.

4.3. Combining fb_1 and fb_2

Assume that in LT-AF coding, no fb_2 is generated. An fb_1 is generated if and only if n (the number of decoded input symbols) surpasses certain thresholds $n_1k, n_2k, \ldots$ (see Lemma 3). Since no degree-one is ever generated using the degree distribution (3), n does not increase due to lack of degree-one output symbols. Therefore, no fb_1 is issued as well. Consequently, acknowledgments to fb_1 in form of degree-one are also not generated. We can see that without fb_2 the decoding is stalled no matter how large γ grows. Clearly, this is resolved by simply adding fb_2's that are generated at certain γ's. This is the reason why feedbacks in LT-AF codes starts with a fb_2, and as we discussed in Section 4.2, we propose to start their transmission at $\gamma = 1$.

In Fig. 6, we have shown the average values of γ for which fb_1's and fb_2's are generated in LT-AF codes with VMD for $k = 500$ and $k = 1000$. The values of γ have been found by averaging values of γ for which each feedback is generated over a large enough number of numerical simulations. We can observe that in LT-AF codes feedbacks start with a fb_2 and the few first feedbacks are fb_2's. After several fb_2's, n increases such that it surpasses n_i's and fb_1's are generated accordingly.

5. Performance evaluation

In this section, we evaluate the performance of LT-AF codes employing numerical simulations. Our results are obtained employing Monte-Carlo method by averaging over the results of at least 10^7 numerical simulations.

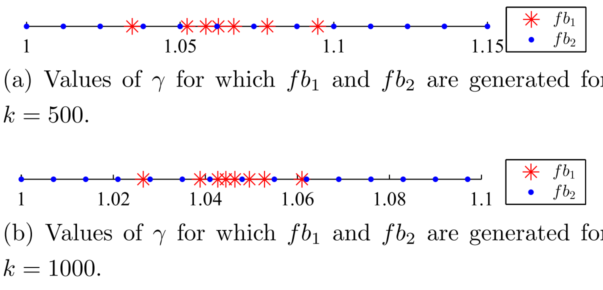


Fig. 6. The order and the frequency of generating fb_1 and fb_2 versus γ in LT-AF codes with VMD.

5.1. LT-AF decoding error rate

We plot the decoding *bit-error-rate* (BER) (average ratio of unrecovered input symbols to the total number of input symbols $1 - E[\frac{n}{k}]$) and the *ratio of successful decodings* versus received overhead γ in Figs. 7 and 8, respectively, for $k = 500$ and $k = 1000$ with $D = \ln k$ (see Section 4.2 for the definition of D). Note that we set $c = 0.9$ and $\delta = 0.1$ for SLT codes as proposed in [4]. We also plot the BER of Growth codes [5] in Fig. 7.

Figs. 7 and 8 show that LT-AF codes significantly surpass SLT codes. We can see that the required coding overhead ϵ (to achieve BER $\leqslant 10^{-6}$) for $k = 1000$ has decreased from 0.3 for SLT codes to 0.09, 0.12, and 0.14 for LT-AF coding using FVD, LDC, and VMD algorithms, respectively. This is respectively equal to **70**%, **60**%, and **53**% reduction in codings overhead, ϵ, for full decoding compared to SLT codes.

Next, we summarize the average runtime of a full round of decoding including the time that decoder need to generate fb_2's for LT-AF codes employing FVD, LDC, and VMD along with SLT codes in Table 1.

Table 1 shows that the complexity of FVD is way higher than the other two proposed algorithms. However, we can see that LDC and VMD have close complexities. Therefore, LDC may be the *best* option that provides a low complexity besides improved coding performance. Further, we can see that LT-AF codes employing VMD and LDC have lower complexities compared to SLT codes. The reason for this lower complexity is that in LT-AF codes for $\gamma < 1$ no decoding is performed and no feedback is generated, and when the decoding starts the full recovery is obtained at a smaller γ resulting in less number of decoding iterations. Therefore, LT-AF codes employing VMD and LDC outperform SLT

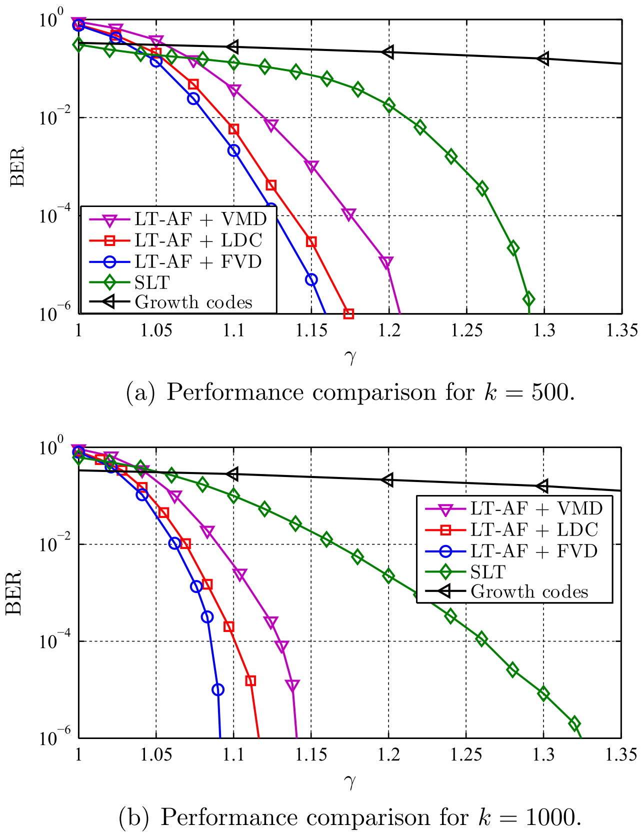
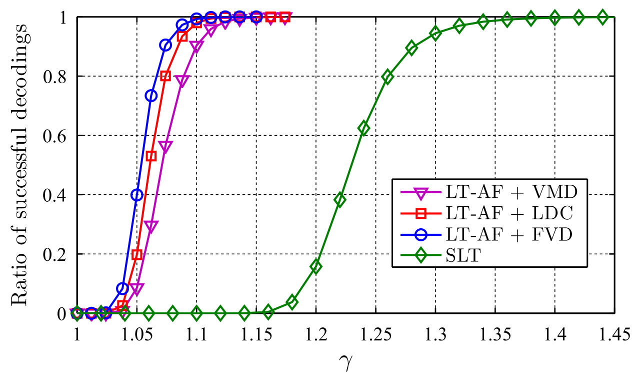
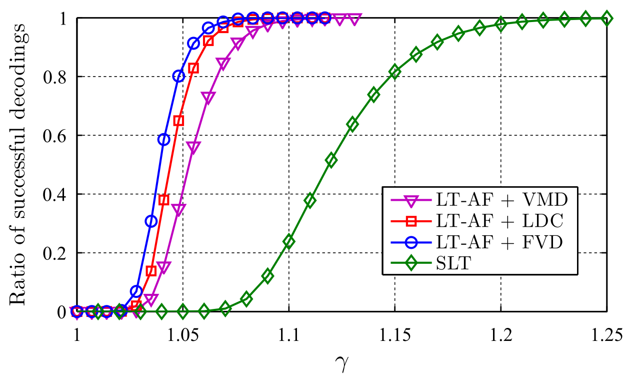


Fig. 7. The BER of SLT codes, Growth codes, and various setups of LT-AF codes versus received overhead γ for $k = 500$ and $k = 1000$.



(a) Performance comparison for $k = 500$.



(b) Performance comparison for $k = 1000$.

Fig. 8. The ratio of successful decodings for SLT codes and various setups of LT-AF codes versus received overhead γ.

Table 1
Runtime comparison of LT-AF and SLT codes on the same platform in seconds.

Algorithm	$N = 500$	$N = 1000$
LT-AF + VMD	0.122	0.683
LT-AF + LDC	0.181	1.050
LT-AF + FVD	11.000	122.270
SLT	0.535	1.550

codes both in the number of required output symbols and complexity.

5.2. Number of feedbacks

In this section, we compare the total number of feedbacks issued by LT-AF codes and compare it to that of SLT codes for $k = 500$ and $k = 1000$ with $D = \ln k$ (see Section 4.2 for the definition of D). As we mentioned earlier, we aim to have equal or smaller number of feedbacks compared to SLT codes. The expected number of feedbacks for LT-AF and SLT codes are summarized in

Table 2
The average number of feedbacks issued in LT-AF and SLT codes for full decoding of data block.

Algorithm	$N = 500$			$N = 1000$		
	fb_1	fb_2	Total	fb_1	fb_2	Total
LT-AF + VMD	2.68	7.37	10.05	2.68	9.29	11.97
LT-AF + LDC	3.15	6.49	9.64	3.90	8.00	11.90
LT-AF + FVD	2.75	6.01	8.76	3.58	6.92	10.5
SLT	–	–	10.43	–	–	12.27

Table 2 for $k = 500$ and $k = 1000$. From Table 2, we can verify that LT-AF codes can decrease the required coding overhead for a successful decoding ϵ with slightly smaller number of feedbacks, which is achieved with $D = \ln k$.

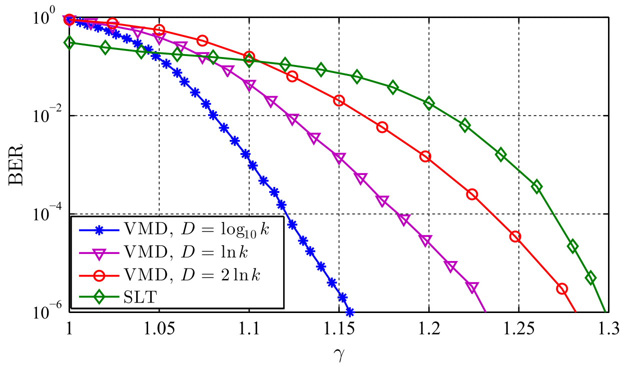
5.3. Effect of distance of fb_2's, D, on the performance

As described in Section 4.2, we set $D = \ln k$ to achieve the same or smaller number of feedbacks in LT-AF codes compared to SLT codes to realize a fair comparison. In this section, we investigate how variations in D affects the performance of LT-AF codes. We set $D = 2\ln k$ and $D = 2\log_{10} k$ and plot the BER of LT-AF codes using VMD versus γ in Fig. 9 for $k = 500$ and $k = 1000$.

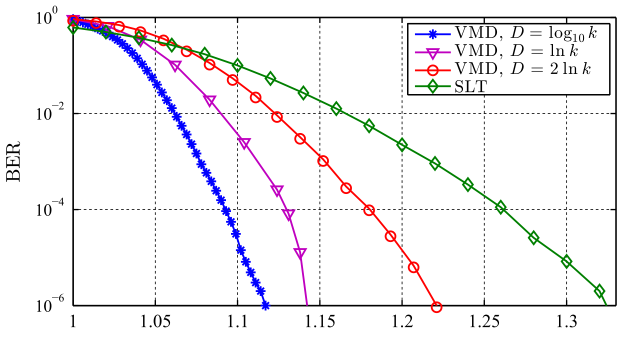
Fig. 9 shows that the BER of LT-AF codes improve if the number of fb_2 increases and vice versa. However, we can see that even with $D = 2\log k$ LT-AF codes still surpass SLT codes. It is worth noting that with $D = 2\ln k$ and $D = 2\log_{10} k$ the total number of feedbacks are 6.67 and 15.68, respectively.

5.4. Robustness to erasure in feedback channel

We mentioned that LT-AF codes are designed to be resilient to loss in feedback channel and their decoding recovery rate does not considerably deteriorate for $\varepsilon_{fb} \in [0, 1)$. In contrast, the existing codes with feedback [4–8] may show some levels of degradation in the presence of loss in feedback channel. We evaluate the effect of feedback loss on the performance of LT-AF codes and SLT codes. Assume that the loss rate of the feedback channel is $\varepsilon_{fb} = 0.9$ (which is not known to encoder and decoder), hence 90% of the feedbacks are lost in transmission. Note, that in a lossy forward



(a) BER of LT-AF codes for $D = \log_{10} k, \ln k, 2\log k$ for $k = 500$.



(b) BER of LT-AF codes for $D = \log_{10} k, \ln k$, and $2\log k$ for $k = 1000$.

Fig. 9. The BER of SLT codes and LT-AF codes with VMD for various D's versus received overhead γ for $k = 500$ and $k = 1000$. Markers show the values of γ where fb_2's are issued.

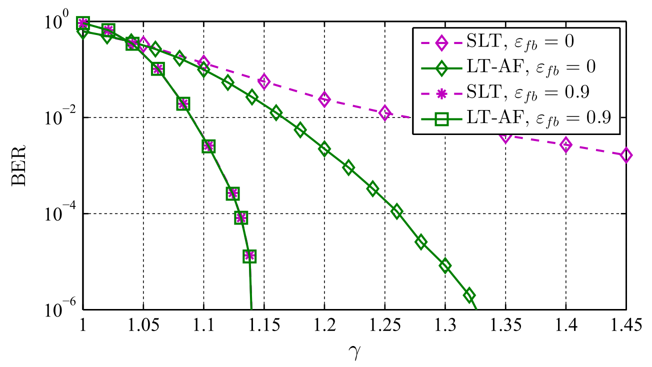


Fig. 10. Effect of 90% feedback loss on the performance of SLT and LT-AF codes employing VMD. The curves representing LT-AF codes for $\varepsilon = 0$ and $\varepsilon = 0.9$ are almost overlapping.

channel the degree-one acknowledgements may also be dropped while fb_1 or fb_2 may have already been delivered. In the case of fb_2 loss, the retransmission compensates this loss. However, in case of fb_1 loss, the encoder shifts the degree distribution accordingly while the decoder remains unaware of this shift. In this case, feedback retransmission is not even required since the degree distribution shift has already occurred. Therefore, we consider the worst case in our simulations and assume that if an acknowledgement is lost the distribution shifting does not occur as well. Fig. 10 shows the performance of LT-AF codes and SLT codes for $k = 1000$ and $\varepsilon_{fb} = 0.9$.

Fig. 10 shows the excellent resilience of LT-AF codes to feedback loss in contrast to SLT codes. In practice, the performance of SLT codes approach that of regular LT codes as the feedback loss ratio increases. It is worth mentioning that at the loss rate of $\varepsilon_{fb} = 0.9$, the decoder generates about 10 times the number of feedbacks it generates with $\varepsilon_{fb} = 0$. However, due to loss in feedback channel, equal number of feedbacks are received at the encoder in both cases. To the best of our knowledge the existing work has not designed a facility to detect a feedback loss and are not resilient to feedback loss as a consequence. This significantly distinguishes the LT-AF codes.

5.5. Alternating feedbacks for Raptor codes

Raptor codes [2] are extension to LT codes to realize rateless codes with linear time encoding and decoding. Raptor encoding includes two steps. In the first step, the data of length k is *precoded* with a high rate linear code (*inner code*) to generate $\tilde{k}$ intermediate symbols. Next, the intermediate symbols are encoded using an LT code (*outer code*) to generate the final output symbols. In this section, we show that alternating feedbacks can also greatly decrease the required amount of Raptor coding overhead.

We choose an optimized degree distribution from [2] given by (15), and employ it instead of IS distribution $\rho_{k,n}(.)$ in LT-AF coding along with VMD. To performs the precoding, we employ an LDPC inner code with parameters $P(x^4, 1100, 50)$, where in LDPC ensemble $P(\Lambda(x), \tilde{k}, r)$, $\Lambda(x)$ is the degree distribution of *variable nodes* and r is the number of check nodes (see [2, Sec. VII-C] for more information). In Fig. 11, we compare the performance of regular Raptor codes, Raptor codes without precoding, and Raptor codes with VMD (without precoding) for $k = 10^3$.

$$\begin{aligned}\Omega(x) = {} & 0.00797x + 0.49357x^2 + 0.16622x^3 + 0.07265x^4 \\ & + 0.08256x^5 + 0.05606x^8 + 0.03723x^9 + 0.05559x^{19} \\ & + 0.02502x^{65} + 0.00314x^{66} \qquad (15)\end{aligned}$$

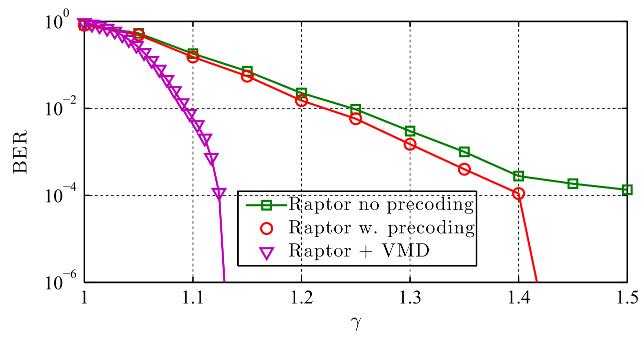


Fig. 11. Improving the performance of Raptor codes using the proposed alternating feedbacks technique in LT-AF codes.

Fig. 11 shows that our proposed alternating feedback generation method can greatly decrease the amount of required overhead in Raptor codes. We can see that to achieve the error rates of smaller than 10^{-6}, regular Raptor codes require coding overhead of $\epsilon \approx 0.41$, while they need an overhead of $\epsilon \approx 0.13$ when VMD is employed. As expected, Raptor codes without precoding have a high error floor, and do not have a comparable performance.

6. Conclusion

In this paper, we proposed LT-AF codes that are LT codes with *alternating* feedback, which reduce the amount of required overhead for successful decoding by lightly utilizing the feedback channel. We designed a new coding degree distribution to employ degree-one output symbols as acknowledgement to the reception of feedbacks. This made LT-AF code resilient against high loss rates in the feedback channel. The designed degree distribution of LT-AF codes is parameterless in contrast to all previous work.

Next, we proposed a new method to generate feedbacks for LT codes and combined it with an existing method in the design of LT-AF codes. Therefore, in LT-AF coding, the decoder can employ two types of feedbacks according to its status. To design our coding scheme we devised three algorithms to analyze the decoder's buffer and request a suitable input symbol for decoding progress.

We showed that our contribution in the design of LT-AF codes compared to existing work is threefold. LT-AF codes reduce the coding overhead for a successful decoding. Further, we observed that LT-AF codes have no parameter to tune in contrast to regular LT codes. Finally and most importantly, LT-AF codes' performance does not considerably degrade at large loss rates in the feedback channel.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. They brought it to our attentions that when this manuscript was under review, paper [11], which shares some ideas with LT-AF codes has been published. This material is based upon work supported by the National Science Foundation under Grants ECCS-1418710 and CCF-1439182.

References

[1] M. Luby, LT codes, in: The 43rd Annual Proceedings IEEE Symposium on Foundations of Computer Science, 2002, 2002, pp. 271–280.
[2] A. Shokrollahi, Raptor codes, IEEE Trans. Inf. Theory 52 (6) (2006) 2551–2567.
[3] P. Maymounkov, Online codes, NYU Technical, Report TR2003-883.
[4] A. Hagedorn, S. Agarwal, D. Starobinski, A. Trachtenberg, Rateless coding with feedback, INFOCOM 2009 (2009) 1791–1799.

[5] A. Kamra, V. Misra, J. Feldman, D. Rubenstein, Growth codes: maximizing sensor network data persistence, SIGCOMM Comput. Commun. Rev. 36 (4) (2006) 255–266.

[6] A. Beimel, S. Dolev, N. Singer, RT oblivious erasure correcting, IEEE/ACM Trans. Network. 15 (6) (2007) 1321–1332.

[7] S. Kokalj-Filipovic, P. Spasojevic, E. Soljanin, R. Yates, ARQ with doped fountain decoding, in: IEEE 10th International Symposium on Spread Spectrum Techniques and Applications, ISSSTA, 2008, pp. 780–784.

[8] J. Srensen, P. Popovski, J. stergaard, On the Role of Feedback in LT Codes, in press, arXiv:1012.2673.

[9] E. Bodine, M. Cheng, Characterization of Luby transform codes with small message size for low-latency decoding, in: IEEE International Conference on Communications, ICC, 2008, pp. 1195–1199.

[10] E. Hyytia, T. Tirronen, J. Virtamo, Optimal degree distribution for LT codes with small message length, in: 26th IEEE International Conference on Computer Communications, INFOCOM, 2007, pp. 2576–2580.

[11] Y. Cassuto, A. Shokrollahi, On-line fountain codes for semi-random loss channels, IEEE Information Theory Workshop (ITW) (2011) 262–266.

[12] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, D. Knuth, On the lambert *W* function, Adv. Comput. Math. 5 (1) (1996) 329–359.

[13] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman, V. Stemann, Practical loss-resilient codes, in: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, 1997, pp. 150–159.