Published in IET Communications
Received on 29th September 2013
Revised on 16th December 2013
Accepted on 26th January 2014
doi: 10.1049/iet-com.2013.0864

ISSN 1751-8628

Design of improved Luby transform codes with decreasing ripple size and feedback

Lei Zhang[1,2], Jianxin Liao[1,2], Jingyu Wang[1,2], Tonghong Li[3], Qi Qi[1,2]

[1]*State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, People's Republic of China*
[2]*EBUPT Information Technology Co., Ltd., Beijing 100191, People's Republic of China*
[3]*Department of Computer Science, Technical University of Madrid, Madrid, Spain*
E-mail: liaojx@bupt.edu.cn

Abstract: In this study, the design of improved Luby transform codes with decreasing ripple size (LTC-DRS) with feedback is presented. Under the proposed design, a new degree distribution algorithm named generalised degree distribution algorithm (GDDA) is proposed, which can achieve arbitrary ripple size revolution accurately. On the basis of GDDA, an accurate ripple size revolution based on binomial fitting is proposed, which can keep the ripple size to a suitable value throughout the decoding process. Furthermore, the authors introduce the feedback and propose a shifted ripple size revolution to diversify the degree values. The improved LTC-DRS with feedback is evaluated and compared with the existing schemes. The simulation results demonstrate that it outperforms other existing schemes in terms of average overhead, average degree of encoded symbols, memory usage and energy consumption.

1 Introduction

In the past few years, lots of sensors like global positioning system, accelerometer, camera and microphone have been embedded in mobile devices. Thus, software updates need to be broadcasted to all devices frequently over wireless networks. However, communication over wireless networks suffers from high packet loss rate and long delay. Forward error correction codes such as Turbo codes [1] and low density parity check code codes [2] are employed for communication over wireless networks to recover symbols that are lost during the transmission. However, they cannot guarantee efficient transmission over broadcast networks where there exist a variety of data loss patterns [3]. Fortunately, digital fountain codes, which was originally proposed in 1998 [4], can provide reliable and efficient information delivery over broadcast networks without any knowledge of channel state information at transmitter. Its key property is that the source data can be recovered from any subset of encoded symbols, given that enough symbols are received.

The first universal fountain codes named Luby transfer (LT) codes was proposed in 2002 [5]. However, LT codes have a constant ripple size during the decoding process, which leads to a high overhead, in particular, when the code length is small [6, 7]. Based on LT codes, Raptor codes were proposed in [8], which is the concatenation of systematic pre-codes and LT codes. Several degree distributions were given [8] for particular sizes of input symbols, but no general algorithm was proposed to construct a suitable degree distribution for any size of input symbols. In addition, a novel suboptimal degree distribution algorithm was proposed in [9]. Several approaches were introduced to construct the degree distribution [10–12], but their performances are not optimised. In [13], the variance of the ripple size was analysed throughout the LT decoding process. LT codes with decreasing ripple size (LTC-DRS) was proposed in [14], which uses a decreasing ripple size to reduce the overhead. However, the ripple size revolution does not consider the interaction between the ripple size and the probability that an input symbol is released and added into ripple.

In these codes, encoded symbols are generated without knowledge of feedback, and thus the total number of encoded symbols should be large enough to recover all input symbols. The fixed-rate error control fountain codes was proposed in [15], where a global decoding algorithm, incorporating the feedback between component codes of Raptor codes, is introduced to improve the performance over memoryless and correlated fading channels. Growth codes, employing feedbacks from receiver to keep transmitter aware of decoding process, were proposed in [16]. It increases the degree of encoded symbols to maximise the recovery probability of each encoded symbol. Real time oblivious approach was proposed in [17]. However, it is only suitable for the receiver with limited memory, and requires a large number of encoded symbols for recovering all input symbols. shifted LT (SLT) codes was proposed in [18], which achieves significant communication gain as the feedback information is exploited to better distribute the degree of encoded symbols. However, the diversity of degree values is limited in SLT codes.

In this paper, we first present a new degree distribution algorithm named generalised degree distribution algorithm (GDDA), which can achieve arbitrary ripple size revolution accurately. On the basis of an accurate ripple size revolution, an improved LTC-DRS is proposed, which can keep the ripple size to a suitable value in the decoding process. With the introduction of feedback, our improved LTC-DRS can dynamically adjust the degree distribution according to the number of input symbols that have been recovered. Furthermore, our improved LTC-DRS with feedback diversifies the degree values. Simulation results demonstrate that our improved LTC-DRS with feedback outperforms the original LTC-DRS with feedback in terms of average overhead, average degree of encoded symbols, memory usage and energy consumption.

The rest of the paper is organised as follows. In Section 2, the review of LT codes and SLT codes are given. In Section 3, our GDDA is presented after some preliminaries are demonstrated. In Section 4, our improved LTC-DRS with feedback is presented. The performance of our improved LTC-DRS with feedback is analysed in Section 5. In Section 6, our experimental design is outlined, and the efficiency of our improved LTC-DRS with feedback is illustrated by our experiment results. Finally, our work is summarised.

2 Review of LT codes and SLT codes

2.1 LT codes

In this section, we briefly review LT codes. Suppose bulk data comprising of k input symbols need to be transmitted from transmitter to receiver. Let $\rho(1)\ldots\rho(k)$ be the degree distribution, such that $\rho(d)$ denotes the probability that degree d is chosen. An encoded symbol is generated as follows:

1. A degree (d) is chosen at random according to the distribution $\rho(1)\ldots\rho(k)$.
2. d Distinct input symbols are chosen uniformly at random from k input symbols.
3. An encoded symbol is generated by performing bitwise XOR operations on the selected d input symbols.

Any one of selected d input symbols is called the neighbour of this encoded symbol. Let $(1+\varepsilon)k$ denote the sufficient number of encoded symbols for successful decoding. The process of belief propagation (BP) decoding process is performed through the reverse bitwise XOR operations. Initially, all input symbols are unrecovered. All encoded symbols with degree one are firstly released to recover their unique neighbours. All recovered input symbols that have not been processed are called 'ripple'. Symbols in the ripple are processed one by one until all input symbols are recovered. The processing of an input symbol in ripple is as follows:

1. It is removed from the ripple.
2. It is removed as a neighbour from all encoded symbols that have it as a neighbour.
3. For each encoded symbol with exactly one remaining neighbour, its remaining neighbour is released; this operation is called a symbol 'release'.
4. New released input symbols previously unrecovered are added into the ripple.

For example, in Fig. 1*a*, six encoded symbols are sufficient for successful decoding. In the first step of decoding, shown in Fig. 1*b*, input symbol 1 is added into ripple. In the second step, shown in Fig. 1*c*, input symbol 1 in ripple is processed, that is, input symbols 2 and 3 are released and added into ripple. In the third step, shown in Fig. 1*d*, input symbol 2 is processed, that is, input symbol 3 is released. However, input symbol 3 is already in the ripple, in this case the encoded symbol is redundant.

2.2 SLT codes

The encoding and decoding of SLT codes are similar to these of LT codes, except that SLT codes shift the robust soliton distribution (RSD) according to the number of input symbols received. This feature enables SLT to experience the same theoretical properties as LT codes, and achieve significant throughput gain. In LT codes, encoded symbols are generated according to the coding graph. The edge between an input symbol and an encoded symbol is randomly generated by using RSD. However, when some input symbols have been recovered in receiver, it is inefficient that new encoded symbols are still generated according to RSD. Let k denote the number of input symbols, and 'round' (x) is a function which returns the nearest integer to x. The degree distribution of SLT codes is shown in (1), which is proposed and proved in [18]

$$\gamma_{k,n}(d') = \mu_{(k-n)}(d) \text{ for round}\left(\frac{kd}{k-n}\right) = d' \qquad (1)$$

3 Design of generalised degree distribution algorithm

In this paper, we assume that the BP decoding process is adopted. We first analyse the proposition presented in [5],

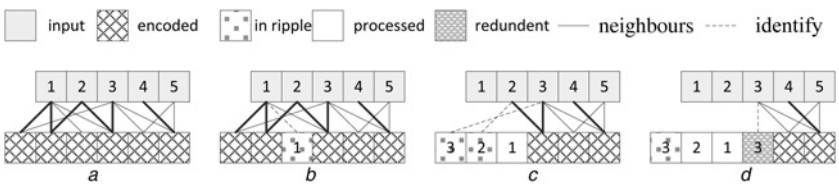

Fig. 1 *Decoding process of LT codes*

a Initial state
b Input symbol 1 is released
c Input symbol 2 and 3 are released
b Input symbol 3 is released again

doi: 10.1049/iet-com.2013.0864

that is, the probability that a particular encoded symbol of degree d is released when L input symbols remain unprocessed. Since an encoded symbol chooses its neighbours independently of all other encoded symbols, the probability that this encoded symbol is released when L input symbols remain unprocessed is independent of the probability that any other encoded symbol is released. The degree release probability is shown in (2) (see (2))

Lemma 1: (probability that an input symbol is released and added into ripple): Consider LT codes with an arbitrary degree distribution $\rho(d)$, $d = 1, \ldots, k$. Given that the ripple size is $R(L+1)$ when L input symbols remain unprocessed, the probability that an encoded symbol is released and added into ripple when L input symbols remain unprocessed is shown in (3)

$$r(L) = \frac{L - R(L+1)}{L} \sum_{d=1}^{k} \rho(d) q(d,\ L) \tag{3}$$

Proof: Referring to [5], the probability that an encoded symbol is released when L input symbols remain unprocessed is $\sum_{d=1}^{k} \rho(d) q(d,\ L)$. On the other hand, the chance that a released encoded symbol is added into the ripple is $L - R(L+1)/L$. Hence, Lemma 1 holds up.

From Lemma 1, we can see that $r(L)$ depends on $\rho(d)$. On the other hand, by using traditional degree distribution algorithms such as these proposed in [9, 14], we can calculate $\rho(d)$, which relies on $r(L)$. Thus, $r(L)$ and $\rho(d)$ are interdependent. Owing to this fact, we cannot create an accurate ripple size revolution directly, and instead we first propose a degree distribution algorithm named GDDA that can achieve arbitrary ripple size revolution accurately. By using GDDA, the influence of the ripple size revolution caused by the probability that a new released symbol is already in this ripple is eliminated.

Definition 1 (impractical GDDA): Let $Q(L)$ denote the designed number of input symbols added into ripple in the $(k-L)$th decoding step. Given that the number of encoded symbols required to recover all k input symbols is E, the impractical GDDA is shown in (4)

$$\rho_{\text{impractical}}(d) = \frac{k \times L \times Q(L)}{E \times (L - R(L+1)) \times d(d-1)}, \quad L = k - d + 1, \ldots, 1 \tag{4}$$

Lemma 2: Impractical GDDA can achieve arbitrary designed ripple size revolution $Q(L)$, $L = k, \ldots, 1$.

Proof: As in the proof of Proposition 10 in [5], $\sum_{d=1}^{k} q(d,\ L)/d(d-1) = \frac{1}{k}$ for all $L = k, \ldots, 1$. Hence, the actual number of input symbols added into ripple in the $(k-L)$th decoding step is calculated as follows

$$\begin{aligned} Q_{\text{actual}}(L) &= E \times r(L) = \frac{E \times (L - R(L+1))}{L} \\ &\quad \times \sum_{d=1}^{k} \rho_{\text{impractical}}(d) \times q(d,\ L) \\ &= k \times Q(L) \sum_{d=1}^{k} \frac{q(d,\ L)}{d(d-1)} = Q(L) \end{aligned}$$

$Q_{\text{actual}}(L)$ equals to the designed number $Q(L)$, which means that GDDA can achieve arbitrary designed ripple size revolution $Q(L)$. Hence, Lemma 2 holds up. □

We use the term 'impractical GDDA' because $\rho_{\text{impractical}}(d)$ changes with L, which should be fixed in the encoding process. As impractical GDDA cannot be used in LT codes, we need to find a practical GDDA.

Definition 2 (practical GDDA): To simplify the expression of degree distribution, we define the coefficient of degree d, which is shown in (5). Thus, the practical GDDA is given by (6)

$$\tau(d) = \frac{\sum_{L=1}^{k-d+1} Q(L) \times q(d,\ L)}{\sum_{L=1}^{k-d+1} \big(q(d,L)(L - R(L+1))\big)/L}, \quad d = 1, \ldots, k \tag{5}$$

$$\rho(d) = \frac{k \times \tau(d)}{E \times d(d-1)}, \quad d = 1, \ldots, k \tag{6}$$

We have: $\sum_{d=1}^{k} \rho(d) = 1$. Thus, E acts as the normalisation factor.

Lemma 3: Practical GDDA achieves the same performance as impractical GDDA.

Proof: Consider the number of encoded symbols with degree d released and added into ripple in the $(k-L)$th decoding step. If the degree distribution is calculated by impractical GDDA, the number of input symbols added into ripple is $((L - R(L+1))/L)\rho(d) \times q(d, L)$. On the other hand, if the degree distribution is calculated by practical GDDA, the number of input symbols added into ripple is $((L - R(L+1))/L)\rho_{\text{impractical}}(d) \times q(d, L)$. We can see that the total number of encoded symbols with degree d added into ripple from $L = 1$ to $L = k - d + 1$ in practical GDDA is equal to the one in impractical GDDA, which is shown in (7).

$$q(d,L) = \begin{cases} 1, & L = k,\ d = 1 \\ \dfrac{d(d-1)L \prod_{j=0}^{d-3} \big(k - (L+1) - j\big)}{\prod_{j=0}^{d-1} (k - j)}, & L = 1, \ldots, k - d + 1,\ d = 2, \ldots, k \\ 0, & \text{for all other } d \text{ and } L \end{cases} \tag{2}$$

Hence, Lemma 3 holds up.

$$\sum_{L=1}^{k-d+1}\left(\frac{L-R(L+1)}{L}\rho(d)\times q(d,L)\right)$$
$$=\frac{k\times\tau(d)}{E\times d(d-1)}\sum_{L=1}^{k-d+1}\left(\frac{L-R(L+1)}{L}q(d,L)\right) \quad (7)$$
$$=\sum_{L=1}^{k-d+1}\left(\frac{L-R(L+1)}{L}\rho_{\text{impractical}}(d)\times q(d,L)\right)$$

4 Design of improved LTC-DRS with feedback

For a good degree distribution, the ripple size throughout the decoding process should be small and the ripple should not become empty before the successful decoding. The relationship between the degree of an encoded symbol and the point of release has been derived in [5]. In this section, we first present the ripple size revolution without feedback and propose the design of our improved LTC-DRS. Afterwards, we discuss the ripple size revolution with feedback and propose the design of our improved LTC-DRS with feedback.

4.1 Construction of ripple size evolution without feedback

In [14], Sorensen *et al.* argued that the ripple size should be larger than c_1L^{1/c_2}, for the suitable constant c_1 and c_2, which is justified by viewing the ripple evolution as a simple random walk with variable bias, that is, each time an encoded symbol is processed, the probability that the ripple size is increased by one is less than the probability that the ripple size is decreased by one. However, it is difficult to create an accurate random walk model for the ripple [14]. In our design of GDDA, the number of input symbols released at each step is extended to cover the probability that a new released symbol is already in the ripple. Hence, we expect that an input symbol is added into the ripple at each decoding step. For example, if the probability that an input symbol is added into the ripple is λ in the $(k-L)$th decoding step, then $Q(L)/\lambda$ input symbols are released in this step. We choose the ripple size evolution based on a random walk model for the ripple. This model is a simple symmetric one-dimensional random walk model, that is, the ripple size either increases with one or decreases with one. We use the binomial fitting method to capture the dynamics of the root mean square (RMS) distance. The ripple size is shown in (8)

$$R(L)=b_1L^2+b_2L+b_3,\quad L=k,\ldots,1 \quad (8)$$

For example, the expected ripple size for $k=256$ is shown in Fig. 2, which shows that the binomial fitting method is better than the method proposed in [5, 14].

After choosing the designed size of the ripple evolution, the number of symbols that must be released in the $(k-L)$th decoding step is shown in (9), where $R(L+1)$ is the ripple size when L input symbols remain unprocessed. By substituting the designed ripple size revolution $R(L)$ and

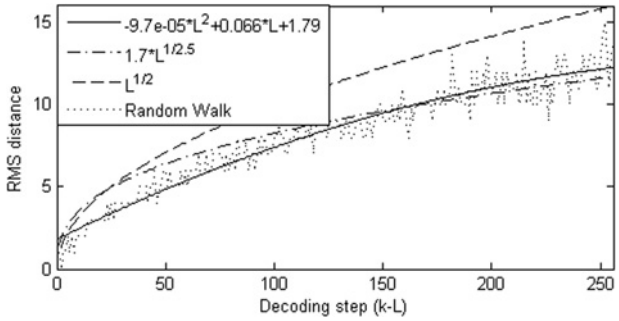

Fig. 2 *RMS distance as a function of decoding step*

$Q(L)$ into (5) and (6), the degree distribution of our improved LTC-DRS is constructed

$$Q(L)=\begin{cases}R(L), & L=k\\ R(L)-R(L+1)+1, & k>L>0\end{cases} \quad (9)$$

4.2 Construction of ripple size evolution with feedback

The RSD of LT codes is too sparse when n input symbols have been recovered by receiver. These n recovered input symbols are essentially the encoded symbols with degree one. Thus, the degree distribution should be changed for the future encoding. However, the diversity of degree values generated by the method proposed in [18] is limited. For example, assuming that five input symbols (the number of all input symbols is ten) have been recovered in receiver. The set of degree values is shown in Table 1. Only a subset of degree is used to generate encoded symbols. However, the diversity of degrees reduces the probability that encoded symbols are linear dependent. In this section, a method of diversifying the degree values is proposed.

Lemma 4: (degree release probability): The probability that an encoded symbol of degree d is released when n input symbols have been recovered is shown in (10)

$$\theta(d,\ k,\ n)=\frac{C_{k-n}^{1}C_{n}^{d-1}}{C_{k}^{d}},\quad d=1,\ 2,\ \ldots,n+1 \quad (10)$$

Proof: The analysis of the classic balls and bins process can be applied here. This is the probability that one neighbour of the encoded symbol is among $k-n$ input symbols that are not recovered, and the other $d-1$ neighbours are among n recovered input symbols. □

Lemma 5: For k input symbols, the minimum degree of degree distribution is given by $d_0=k/k-n$ when n input symbols have been recovered.

Table 1 Set of degree values

LT codes	SLT codes	Improved LTC-DRS with feedback
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	{2, 4, 6, 8, 10}	{2, 3, 4, 5, 6, 7, 8, 9, 10}

doi: 10.1049/iet-com.2013.0864

Proof: Let $P(X=i)$ denote the probability that i unrecovered symbols are chosen when d input symbols are chosen from k input symbols. It conforms to a hypergeometric distribution, and its probability is shown in (11). Thus, $\theta(d, k, n)$ is interpreted as the probability that only one unrecovered symbol is chosen. The mathematical expectation of $P(X=i)$ is shown in (12). Substituting $E(X)=1$, we yield the statement of Lemma 5

$$P(X=i)=\frac{C_{k-n}^{i}C_{n}^{d-i}}{C_{k}^{d}} \quad d=1,\ 2,\ \ldots,n+1 \tag{11}$$

$$E(X)=\frac{d(k-n)}{k} \tag{12}$$

Lemma 6: When L input symbols have not been processed and n input symbols have been added into the ripple before the start of the decoding process, the ripple size is shown in (13)

$$R'(L)=\begin{cases} R'(L+1)-1, & R'(L+1)>R(L)+1 \\ R(L), & R'(L+1)\le R(L)+1 \end{cases} \tag{13}$$

Proof: When $R'(L+1)>R(L)+1$, the ripple size is too large and thus the possibility that the released symbol is redundant is high. The ripple size should be decreased quickly. However, the ripple size can be decreased by one at most each time a symbol is processed. Thus, we yield the statement of Lemma 6. □

Definition 3: Let $Q'(L)$ denote the ripple size evolution when feedback is introduced, which is shown in (14)

$$Q'(L)=\begin{cases} R'(L), & L=k \\ R'(L)-R'(L+1)+1, & k>L>0 \end{cases} \tag{14}$$

4.3 Improved LTC-DRS with feedback

Lemma 7: (degree distribution of improved LTC-DRS with feedback): Let n denote the number of input symbols that have been added into the ripple before the start of the decoding process, and E' denotes the number of encoded symbols required to recover all k input symbols. To simplify the expression of degree distribution, we define coefficient of degree d, which is shown in (15). Thus, the degree distribution of improved LTC-DRS with feedback is given by (16)

$$\tau'(d)=\frac{\sum_{L=1}^{k-d+1}Q'(L)\times q(d,\ L)}{\sum_{L=1}^{k-d+1}\left(q(d,\ L)(L-R'(L+1))\right)/L}, \quad d=d_0,\ \ldots,k,\ d_0=\frac{k}{k-n} \tag{15}$$

$$\rho'(d)=\frac{k\times\tau'(d)}{E'\times d(d-1)}, \quad d=d_0,\ \ldots,k,\ d_0=\frac{k}{k-n} \tag{16}$$

Proof: The minimum degree in this degree distribution is $d_0=k/k-n$, thus, we set $\rho'(d)=0$, $d=1,\ldots,d_0-1$. By substituting the designed ripple size revolution $R'(L)$ and $Q'(L)$ into (5) and (6), we yield the statement of Lemma 7. □

5 Analysis of improved LTC-DRS with feedback

In this section, we theoretically analyse the properties of improved LTC-DRS with feedback. The following proposition shows that the decoding graph of a reliable decoding algorithm has at least an order of E' encoded symbols and the average degree of an encoded symbol is relatively low.

Lemma 8: A decoder requires E' encoded symbols to recover $k-n$ input symbols

$$E'=k\sum_{d=1}^{k}\left(\frac{\tau'(d)}{d(d-1)}\right) \tag{17}$$

Proof: Note that Lemma 7 represents a degree distribution algorithm, where E' measures how many encoded symbols must be collected in order to achieve the designed ripple size. We sum the degree distribution of all degrees d, as shown in (18). Thus, we yield the statement of Lemma 8 by transforming (18)

$$1=\sum_{d=1}^{k}\rho'(d)=\frac{k}{E'}\sum_{d=1}^{k}\left(\frac{\tau'(d)}{d(d-1)}\right) \tag{18}$$

Lemma 9: The average degree of an encoded symbol under the proposed degree distribution of improved LTC-DRS with feedback is given in (19)

$$\bar{d}=\sum_{d=1}^{k}\left(\frac{\tau'(d)}{(d-1)\sum_{d=1}^{k}\left(\tau'(d)/d(d-1)\right)}\right) \tag{19}$$

Proof: The proof is obtained from the definition, as shown in (20). By substituting (15) and (16), we yield the statement of Lemma 9

$$\bar{d}=\sum_{d=1}^{k}d\times\rho'(d) \tag{20}$$

Definition 4: The overhead $(1+\varepsilon)$ in improved LTC-DRS with feedback is given by (21)

$$(1+\varepsilon)=\frac{E'}{k-n} \tag{21}$$

Table 2 shows the expected number of encoded symbols and the average degree of encoded symbols required for successful decoding. The expected overhead required for successful decoding in improved LTC-DRS is smaller than that in LTC-DRS. With 1024 input symbols, improved LTC-DRS requires 1024×1.055 encoded packets, whereas LTC-DRS requires 1024×1.087 [14]. When $n=k/2$, we

Table 2 Expected number of encoded symbols and the average degree of encoded symbols for a variety of k

k		256	512	768	1024
b_1		-9.749×10^{-5}	-3.647×10^{-5}	-2.182×10^{-5}	-1.431×10^{-5}
b_2		0.0658	0.0469	0.0403	0.0346
b_3		1.791	2.741	3.177	3.910
$n=0$	$(1+\varepsilon)$	1.108	1.078	1.062	1.055
	$\bar{d}$	7.017	7.715	7.911	8.186
$n=$ k/2	$(1+\varepsilon)$	1.268	1.227	1.203	1.194
	$\bar{d}$	10.629	11.844	12.232	12.715

assume that $k/2$ input symbols have been recovered, the degree distribution is constructed to recover $k-n$ input symbols that have not been recovered. Interestingly, the overhead of $n=k/2$ is larger than that of $n=0$. This is because many encoded symbols are redundant to recover the input symbols that have not been recovered.

6 Numerical results

We have presented our improved LTC-DRS with feedback and outlined its properties for the practical implementation. In this section, we evaluate its performance by comparing it with our improved LTC-DRS, LTC-DRS with feedback and LTC-DRS through simulations. LTC-DRS with feedback is constructed by combining LTC-DRS with SLT codes.

6.1 Comparison of improved LTC-DRS and LTC-DRS

We first compare improved LTC-DRS with feedback, against improved LTC-DRS, LTC-DRS with feedback and LTC-DRS in a single unicast stream. In each round of simulation an encoded symbol is generated and transmitted, until all input symbols are recovered by receiver. In the scheme with feedback, there is a feedback channel between transmitter and receiver, and the feedback is sent once every $\sqrt{k}$ input symbols are successfully recovered by receiver, following the configuration in a related literature [18]. Encoded symbols are stored in the memory if they have not been successfully recovered by the decoding process. We assume that the channel between transmitter and receiver is an erasure one, and loss rate $r=10\%$. The metrics used in our evaluations are: (i) the memory usage; (ii) the average overhead required for successful decoding, that is, when all k input symbols are recovered; (iii) the average degree of encoded symbols required for successful decoding; and (iv) the energy consumption.

We set the number of input symbols k to 256. Fig. 3 shows the number of encoded symbols cached in memory as a function of the number of encoded symbols transmitted from transmitter to receiver. We can see that the schemes with feedback outperform the schemes without feedback. In addition, we can see that improved LTC-DRS needs less memory than LTC-DRS. This is because of the more accurate degree distribution used, which releases the encoded symbol with high probability. The maximum number of encoded symbols cached in memory as a function of the number of input symbols is shown in Fig. 4 (on average, over 100 000 trials). Improved LTC-DRS with feedback saves more memory with the increase of k, compared with other schemes. This is because in improved LTC-DRS with feedback, the probability that an encoded

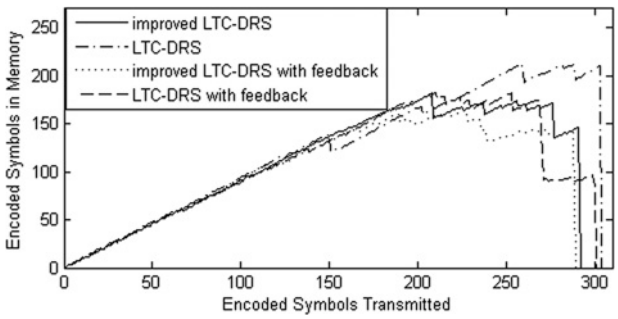

Fig. 3 *Number of encoded symbols in memory as a function of the number of encoded symbols transmitted*

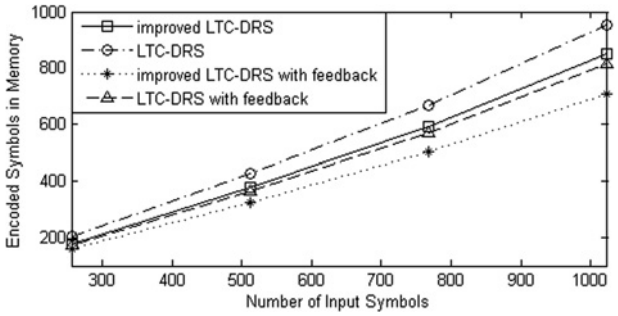

Fig. 4 *Number of encoded symbols in memory as a function of the number of input symbols*

symbol is released becomes smaller when k increases. The average overhead of encoded symbols as a function of the number of input symbols is shown in Fig. 5. When k is small, all schemes show similar performances. However, as the number of input symbols increases, improved LTC-DRS with feedback performs best. For example, with 1024 input symbols, improved LTC-DRS with feedback requires 1110 encoded symbols in the average compared with 1128 encoded symbols needed in LTC-DRS. Fig. 6 shows the average degree of encoded symbols as a function of the number of input symbols. Note that the average degree of encoded symbols for improved LTC-DRS without feedback and improved LTC-DRS with feedback increases more slowly than the other two schemes. This is because of the accurate degree distribution used, which approximates to the ideal degree distribution. Fig. 7 shows the energy consumption of different schemes by varying k. The energy consumption is measured by using the PowerTOSSIM simulator. Obviously, improved LTC-DRS without

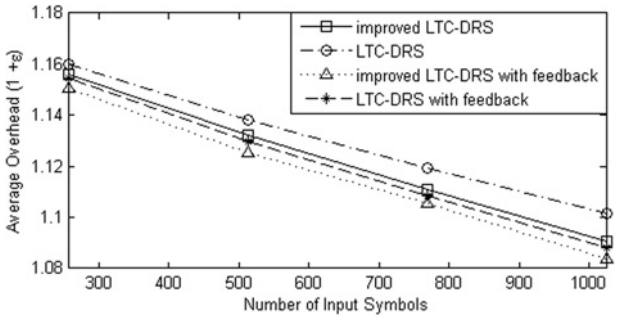

Fig. 5 *Average overhead as a function of number of input symbols*

doi: 10.1049/iet-com.2013.0864

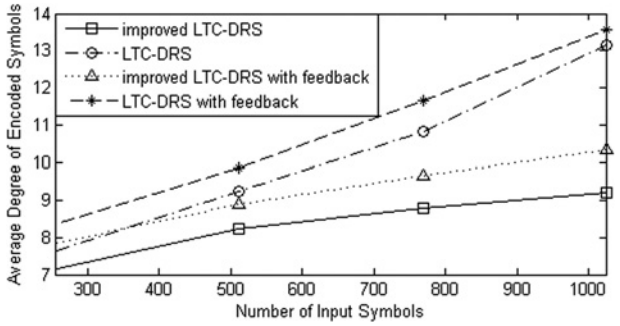

Fig. 6 *Average degree of encoded symbols as a function of number of input symbols*

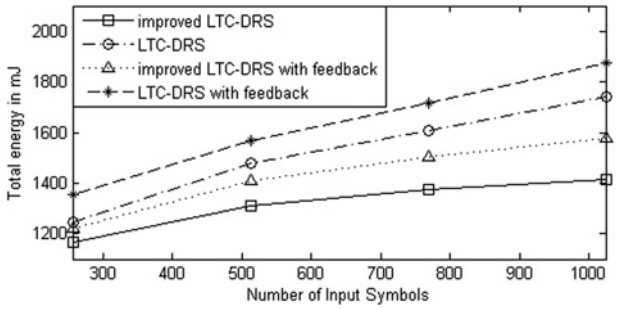

Fig. 7 *Total energy used by all nodes as a function of number of input symbols*

feedback requires less energy than improved LTC-DRS with feedback, which needs the energy to send feedback to receiver, and receive feedback and shift the degree distribution in transmitter.

6.2 Communication of broadcast

In this section, we study the communication cost in broadcast. Our experiment consists of six sensor nodes, in which one node serves as the broadcaster and others serve as receivers. All feedback channels from five nodes to the broadcaster are set with 5% loss rate, and the loss rate of forward channel are varied from 0 to 10%. The degree distributions are pre-computed at the expense of increased memory usage in order to save energy. Our results shown in Fig. 8 are the averaged values over 100 trials. We can see that improved LTC-DRS with feedback transmits fewer symbols than

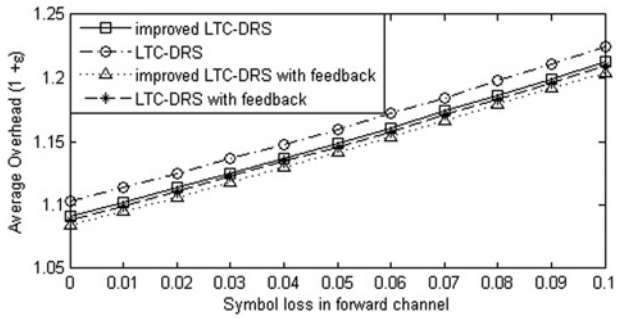

Fig. 8 *Average overhead of encoded symbols transmitted as a function of symbol loss in forward channel*

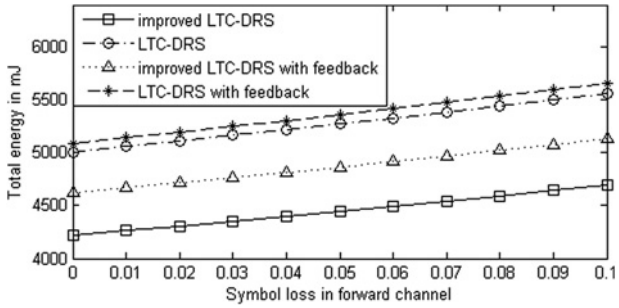

Fig. 9 *Total energy used by all nodes as a function of symbol loss in forward channel*

others for the complete dissemination of the data, thus improved LTC-DRS with feedback is significantly more efficient than others. Fig. 9 shows the energy measurements of different schemes by using the Power TOSSIM simulator. In this figure, we can see that improved LTC-DRS without feedback achieves a roughly 17% improvement in terms of energy savings compared with LTC-DRS with feedback.

7 Conclusions

In this paper, we first propose our improved LTC-DRS, in which a new degree distribution is designed based on our GDDA, random walk and the binomial fitting method. When the feedback is introduced, we note that the degree diversity of SLT codes is limited, which means that the probability that encoded symbols are linear dependent is high. After analysing how the number of recovered symbols impacts the minimum degree of encoded symbols, we propose a new degree distribution with feedback to diversify the degree value.

Through simulations, we demonstrate that improved LTC-DRS with feedback outperforms improved LTC-DRS, LTC-DRS and LTC-DRS with feedback in terms of average overhead, average degree of encoded symbols, memory usage. This is because improved LTC-DRS with feedback uses a new shifted degree distribution, which can keep the ripple size to a suitable value to reduce the redundancy of encoded symbols. We also note that improved LTC-DRS outperforms improved LTC-DRS with feedback in terms of energy consumption. Note that our GDDA can achieve arbitrary ripple size revolution in LT codes. Hence, different types of LT codes can be constructed for different scenarios. For example, if the ripple size revolution keeps the ripple to a large size at the beginning of the decoding process, the average degree of encoded symbol will be decreased, and thus these LT codes are more suitable for resource limited devices. Our future work plans to apply this technology to the broadcast communication in wireless relay networks.

8 Acknowledgments

This work was jointly supported by: (i) the National Basic Research Program of China (no. 2013CB329102); (ii) the National Natural Science Foundation of China (no. 61372120, 61271019, 61101119, 61121001, 61072057 and 60902051); (iii) PCSIRT (no. IRT1049); (iv) MICINN (no. TIN2010-19077); and (v) CAM (no. S2009TIC-1692).

9 References

1 Berrou, C., Glavieux, A.: 'Near optimum error correcting coding and decoding: Turbo-codes', *IEEE Trans. Commun.*, 1996, **44**, (10), pp. 1261–1271
2 Gallager, R.G.: 'Low-density parity-check codes', PhD thesis, Department of Electrical Engineering, MIT, Cambridge, MA, MIT Press, 1963
3 Huang, W.Z.: 'Investigation on digital fountain codes over erasure channels and additive white Gaussian noise channels', PhD thesis, Department of Electrical Engineering, Ohio University, 2012
4 Byers, J., Luby, M., Mitzenmacher, M., Rege, A.: 'A digital fountain approach to reliable distribution of bulk data'. Proc. SIGCOMM, Vancouver, BC, CA, September 1998, pp. 56–67
5 Luby, M.: 'LT Codes'. Proc. ACM Symp. Foundations of Computer Science, Vancouver, BC, CA, November 2002, pp. 37–47
6 Sejdinovic, D., Piechocki, R., Doufexi, A.: 'AND-OR tree analysis of distributed LT codes'. Proc. Information Theory Workshop on Networking and Information Theory, 2009, pp. 261–265
7 Bodine, E.A., Cheng, M.K.: 'Characterization of Luby transform codes with small message size for low-latency decoding'. Proc. Int. Conf. Communications, 2008, pp. 1195–1199
8 Shokrollahi, A.: 'Raptor codes', *Inf. Theory*, 2006, **52**, (6), pp. 2551–2567
9 Zhu, H.P., Zhang, G.X., Li, G.X.: 'A novel degree distribution algorithm of LT codes'. Proc. Int. Conf. Communication Technology Proc., 2008, pp. 221–224
10 Bodine, E.A., Cheng, M.K.: 'Characterization of Luby transform codes with small message size for low-latency decoding'. Proc. ICC, 2008, pp. 1195–1199
11 Hyytia, E., Tirronen, T., Virtamo, J.: 'Optimal degree distribution for LT codes with small message length'. Proc. INFOCOM, 2007, pp. 2576–2580
12 Chen, C.M., Chen, Y.P., Shen, T.C., Zao, J.K.: 'Optimizing degree distributions in LT codes by using the multiobjective evolutionary algorithm based on decomposition'. Proc. Congress on Evolutionary Computation, 2010, pp. 1–8
13 Maatouk, G., Shokrollahi, A.: 'Analysis of the second moment of the LT decoder', *IEEE Trans. Inf. Theory*, 2012, **58**, (5), pp. 2558–2569
14 Sorensen, J.H., Popovski, P., Ostergaard, J.: 'Design and analysis of LT codes with decreasing ripple size', *IEEE Trans. Commun.*, 2012, **60**, (11), pp. 1–7
15 Sivasubramanian, B., Leib, H.: 'Fixed-rate Raptor codes over Rician fading channels', *IEEE Trans. Veh. Technol.*, 2008, **57**, (6), pp. 3905–3911
16 Kamra, A., Misra, V., Feldman, J., Rubenstein, D.: 'Growth codes: maximizing sensor network data persistence'. Proc. Applications, Technologies, Architectures, Protocols Computer Communications, 2006, pp. 255–266
17 Beimel, A., Dolev, S., Singer, N.: 'RT oblivious erasure correcting', *IEEE Trans. Netw.*, 2007, **15**, (6), pp. 1321–1332
18 Hagedorn, A., Agarwal, S., Starobinski, D., Trachtenberg, A.: 'Rateless coding with feedback'. Proc. INFOCOM, Rio de Janeiro, Brazil, April 2009

doi: 10.1049/iet-com.2013.0864