LT Codes Based Distributed Coding for Efficient Distributed Storage in Wireless Sensor Networks

Xiucai Ye and Jie Li
University of Tsukuba, Japan

Wen-Tsuen Chen
Academia Sinica, Taiwan

Feilong Tang
Shanghai Jiao Tong University, China

***Abstract*—Fountain codes are linear codes with low complexities. LT (Luby Transform) codes, which are a special class of Fountain codes, are widely used in Wireless Sensor Networks (WSNs) to increase the robustness of data storage and efficiency of data retrieval. In this paper, we propose a novel LT codes based Distributed Coding (LTDC) scheme for efficient distributed storage in WSNs. In the proposed LTDC scheme, we use random walks to disseminate sensed data from a source sensor node to a random subset of sensor nodes by multicast. As long as a data packet stops at an ending sensor node of a random walk, the ending sensor node encodes this data packet in a main packet (an encoded data packet) with a certain probability. By adjusting the main packet with the un-encoded data packets, the number of data packets encoded in the main packet follows the distribution of LT codes. The data collector is able to decode the original data by querying any subset of sensor nodes. The theoretical analysis and simulation results have demonstrated that the proposed LTDC scheme has lower data dissemination cost and lower storage overhead, while maintains the same level of fault tolerance as the original LT codes.**

I. Introduction

Wireless Sensor Networks (WSNs) consist of a large number of sensor nodes, which has low CPU power, small bandwidth, limited battery energy and limited storage space [1]. In many applications, WSNs are usually deployed in a challenging environment where human involvement is limited [2]. In such environment, it is not feasible to set up a powered base station with Internet connection. An approach is to use mobile base station to perform data collection periodically [3]. The sensor nodes collaboratively store the sensed data over a period of time by themselves. The mobile base station collects the sensed data from the sensor nodes.

Due to limited energy and hostile environment, sensor nodes may fail suddenly and unpredictably, resulting in the loss of sensed data. To provide robust data retrieval, it is desirable to distribute the sensed data throughout the network for redundant storage [4], [5]. Thus, the mobile base station can retrieve the sensed data from any subset of sensor nodes, even after some sensor nodes have failed.

Coding is a powerful tool for redundant data storage. Distributed coding, which distributes the encoding operations to multiple nodes, has been widely used in WSNs [6]. Since random linear codes are easily implemented in a distributed way, many distributed coding schemes based on random linear codes are proposed for distributed storage in WSNs (e.g., [7], [8]). Random linear codes have high decoding complexity. Fountain codes are a promising solution to reduce the decoding complexity in WSNs [9]. LT (Luby Transform) codes, which are the first implementation of Fountain codes, are widely used in WSNs.

Dimakis et al. [9] firstly addressed the problem of constructing Fountain codes for distributed storage. The fountain codes are constructed over grid network by using geographic routing. Kamar et al. [10] constructed special fountain codes called growth codes to improve the data recovery at the base station. Similar to the problem in [9], location information was required. Each sensor node sent data to the base station and should know its location.

Lin et al. [11] firstly used random walks to construct Fountain codes in WSNs. Random walk is routing approach without location information [12]. The authors proposed an interesting scheme called Exact Distributed Fountain Codes (EDFC). EDFC has high level of network reliability. However, the storage overhead in EDFC is large for achieving the expected code degree distribution.

Inspired by [11], many distributed schemes applied random walks to construct Fountain codes in WSNs [13], [14], [15]. Kong et al. [13] proposed a LT Codes based Distributed Storage (LTCDS) scheme. LTCDS has low storage overhead. However, each data packet is disseminated to all sensor nodes at least once, which cause high data transmission cost. Jafarizadeh et al. [14] shared the same consideration in [13] and proposed a distributed scheme, which consumed lower data transmission cost than LTCDS. However, similar to [13], the code degree distribution in each sensor node is unstable, which may make the network reliability no so high.

To achieve high network reliability, meanwhile reduce both the data storage overhead and the data dissemination cost, in this paper, we propose a novel LT codes based Distributed Coding (LTDC) scheme. In LTDC, each data packet is disseminated to a random subset of sensor nodes by random walks. The main contributions of this paper are summarized as follows.

- We proposed the LTDC scheme to improve the efficiency of random walks in data dissemination by using a variant of the Metropolis-Hastings algorithm, which only requires local information available in each sensor node.
- The proposed LTDC scheme takes advantage of the shared nature of wireless medium to reduce the data dissemination cost in random walks by multicast.
- The proposed LTDC scheme reduces the storage overhead

ISBN 978-3-901882-68-5 © 2015 IFIP

significantly. Each sensor node encodes the stopping data packets in a main packet with a certain probability, without storing all the original stopping data packets.

- The proposed LTDC scheme has high network reliability, which maintains the same level of fault tolerance as the original LT codes. By adjusting the main packet in each sensor node, the number of data packets encoded in each main packet follows the distribution of LT codes.

Note that EDFC [11] is the first to use random walks to construct LT codes in WSNs. EDFC is influential and typical due to the salient contribution of using Metropolis algorithm in random walks for data dissemination, which not only reduces the length of random walks, but also makes the number of data packets for encoding follows the distribution of LT codes. In the proposed LTDC scheme, we also use random walks to construct LT codes and apply a variant of the Metropolis-Hastings algorithm to reduce the length of random walks. However, the data dissemination method and data encoding method in our LTDC are more efficient than EDFC, which further reduces the data dissemination cost and the storage overhead, meanwhile makes the number of data packets for encoding follow the distribution of LT codes. To the best of our knowledge, no previous scheme is compared with EDFC. We will compare LTDC with EDFC in the evaluation.

The remainder of this paper is organized as follows. The preliminaries and system description are presented in Section II and Section III, respectively. We present the proposed LT Codes based Distributed Coding (LTDC) scheme in Section IV. The performance evaluation is presented in Section V. We conclude the paper in Section VI.

II. Preliminaries

A. Fountain Codes and LT Codes

Fountain codes are linear codes with low encoding and decoding complexities [9]. The encoded data packet is generated by the exclusive-or (XOR) of a subset of data packets. The original data packets can be decoded from any subset of the encoded packets whose size is equal to or only slightly greater than the number of original data packets.

LT (Luby Transform) codes [16] are the first implementation of Fountain codes, which make Fountain codes work in practice. By LT codes, K data packets can be decoded from any subset of $K + O(\sqrt{K}\ln^2(K/\delta))$ encoded data packets with probability $1-\delta$, where $0 < \delta < 1$. The encoding and decoding complexities are both $O(K\ln(K/\delta))$. The *code degree* which is regarded as an important parameter in LT codes is defined as *the number of data packets to generate an encoded data packet*.

In LT codes, the distribution of code degree follows the *Robust Soliton distribution*. Since the *Robust Soliton distribution* is based on the *Ideal Soliton distribution*, we first introduce the *Ideal Soliton distribution*. Let d denote the code degree of an encoded data packet. The *Ideal Soliton distribution* for K data packets is given by

$$\rho(i) = \begin{cases} 1/K, & i = 1, \\ 1/i(i-1), & i = 2, 3, ..., K. \end{cases}$$

Let $R = c\ln(K/\delta)\sqrt{K}$ for some constant $c > 0$ and

$$\tau(i) = \begin{cases} R/iK, & i = 1, ..., K/R - 1, \\ R\ln(R/\delta)/K, & i = K/R, \\ 0, & i = K/R + 1, ..., K. \end{cases}$$

The *Robust Soliton distribution* for K data packets is defined as follows.

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

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

In the encoding process of LT codes, each encoded data packet is generated independently. To generate an encoded data packet, the encoder first chooses the code degree d from the *Robust Soliton distribution*, and then randomly chooses d distinct data packets from the K data packets. The encoded data packet is the XOR of the d data packets. The decoding process of LT codes is performed by using the Belief Propagation algorithm [17], which is computationally efficient.

B. Random walks on graphs

Random walks have been widely used for data dissemination in WSNs [12]. When using random walks to disseminate a data packet, a source node i first chooses the next hop node j randomly from the neighbors. When the data packet arrives at node j, node j randomly chooses the next hop node u from the neighbors, and so on.

A random walk corresponds to a time-reversible Markov chain, since the choice of the next hop only depends on the current node. If a random walk has sufficient length (i.e., sufficient steps), the distribution of the random walk stopping at a particular node converges to the steady-state distribution $\pi = (\pi_1, \pi_2, ..., \pi_i, ...)$ of the Markov chain, where π_i is the probability that a random walk stops at node i. The length of the random walk should be minimized, since it is proportional to the data dissemination cost. The minimal length of the random walk to approximate the steady-state distribution can be reduced by adjusting the transition matrix $P = [p_{ij}]$ of the Markov chain, where p_{ij} is the probability that node i chooses node j as a next hop. Some algorithms for the design of the transition matrix P have been proposed to reach the steady-state distribution π, while performing as short as possible random walks [18]. The typical algorithms include the maximum-degree algorithm [19] and the Metropolis-Hastings algorithm [20], which are easily adaptable to distributed implementation in WSNs.

In this paper, we use a variant of the Metropolis-Hastings algorithm [20] to compute the transition matrix $P = [p_{ij}]$. Consider a steady-state distribution $\pi = (\pi_1, \pi_2, ..., \pi_i, ...)$, p_{ij} is computed as

$$p_{ij} = \begin{cases} \min(1/D_i, \pi_j/(D_j\pi_i)), & i \neq j \text{ and } j \in N(i), \\ 0, & i \neq j \text{ and } j \notin N(i), \\ 1 - \sum_{i \neq j} p_{ij}, & i = j, \end{cases} \tag{2}$$

where $N(i)$ denotes the set of neighbors of node i, D_i is the node degree of node i. Note that each sensor node only needs the local information (i.e., the steady-state probabilities and the node degrees of its neighbors) to calculate the transition probability p_{ij}. The transition probabilities p_{ij} ($j \in N(i)$) entries in the transition matrix P local to node i are referred to as the probabilistic forwarding table. Each sensor node chooses the next hops according to the probabilistic forwarding table. The length of the random walk to approximate the steady-state distribution π depends on the second largest eigenvalue $|\lambda_2|$ of P as $O(\log N/(1 - |\lambda_2|))$, where $|\lambda_2|$ has been proved to be less than one [21].

III. System Description

We model a WSN as a random geometric graph $G(N, r)$, where N sensor nodes are distributed uniformly at random in a finite region, and two sensor nodes can directly communicate if their distance is within the transmission range r. Each sensor node has B buffers to store data packets, denoted by b_1, b_2,..., b_B (i.e., the buffer size of each sensor node is B). Each buffer can store only one data packet. Consider that a subset of K sensor nodes, referred to as the source nodes, generate sensed data periodically. A time period T consists of three time intervals: data sensing time interval, data dissemination time interval and data collection time interval. At the beginning, each source node generates an equal length data packet containing the sensed data over the sensing time interval. In the data dissemination interval, the data packets are disseminated to a random subset of sensor nodes for encoding by random walks. A data packet stops at an ending sensor node of a random walk. In the data collection time interval, a mobile base station queries any subset of sensor nodes to collect data and performs decoding on the collected data.

Due to the randomization introduced by random walks, the number of distinct data packets stopping at a sensor node is uncertain. This number usually does not equal to the code degree d generated from the *Robust Soliton distribution*. Consider that the expected number of distinct data packets stopping at a sensor node is more than the generated code degree d. Thus, the sensor node can randomly choose d distinct data packets from all the stopping data packets to do encoding.

We define *expected degree* as the expected number of distinct data packets stopping at a sensor node. And, define *actual degree* as the actual number of distinct data packets stopping at a sensor node. We use e_d and a_d to denote the *expected degree* and the *actual degree* in a sensor node with generated code degree d, respectively. Consider that the expected degree is larger than or equals to the generated code degree, i.e., $e_d \geq d$ ($d = 1, ..., K$). Let x_d denote the redundancy for generated code degree d, where $e_d = d + x_d$ and $x_d \geq 0$. By choosing a sufficiently large x_d, the probability that the actual number of stopping distinct data packets being less than d (i.e., $a_d < d$) can be made arbitrarily small. Therefore, the distribution of the number of data packets used in encoding can be arbitrarily close to the Robust Soliton degree distribution.

Note that two critical parameters should be computed before data dissemination, which are the number of random walks launched from each source node and the steady-state distribution. We use the methods in [11] to compute the two parameters.

Let w denote the number of random walks launched from each source node. The number of sensor nodes with generated code degree d in the network is $N\mu(d)$, where $\mu(d)$ is the probability that a sensor node has generated code degree d, as defined in equation (1). The expected number of distinct data packets stopping at a sensor node with generated code degree d is e_d. Thus, the total number of distinct data packets stopping at the sensor nodes is $\sum_{d=1}^{K} N\mu(d)e_d$, which equals to the number of data packets disseminated from the K source nodes. That is,

$$\sum_{d=1}^{K} N\mu(d)e_d = wK.$$

Then,

$$w = \frac{\sum_{d=1}^{K} N\mu(d)e_d}{K}. \tag{3}$$

We show how to compute the steady-state distribution $\pi = (\pi_1, \pi_2, ..., \pi_K)$, where π_d is the probability that a data packet stops at a sensor node with generated code degree d. Since the total number of data packets disseminated from all source nodes is wK, the expected number of data packets stopping at this node is $wK\pi_d$. At the same time, the expected number of data packets stopping at this node also equals to e_d. From equation (3), the stopping probability π_d can be computed as

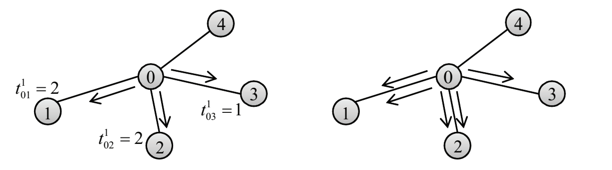
$$\pi_d = \frac{e_d}{\sum_{i=1}^{K} N\mu(i)e_i}. \tag{4}$$

IV. Proposed LT codes based Distributed Coding scheme

In this section, we propose a novel LT codes based Distributed Coding (LTDC) scheme for efficient distributed storage in WSNs. LTDC consists of two processes: *data dissemination process* and *data encoding process*. In the data dissemination process, we use random walks to disseminate a data packet from a source node to a random subset of sensor nodes by multicast. In the data encoding process, each sensor node encodes the stopping data packets in a main packet with a certain probability, without storing all the stopping data packets.

Before the data dissemination and data encoding processes, network initiation is necessary to compute the parameters. The network initiation includes four steps, which are summarized as follows.

Step 1: Degree generation. Each sensor node independently generates a code degree d ($d = 1, 2, ..., K$) from the *Robust Soliton distribution* in equation (1).



(a) Multicast in one transmission (b) Disseminate by five transmissions

Fig. 1. Source node 0 disseminates the data packet to the next hops.

Step 2: Steady-state distribution computation. Each sensor node computes the steady-state distribution by equation (4).

Step 3: Probabilistic forwarding table computation. Each sensor node computes the probabilistic forwarding table by equation (2) based on the steady-state distribution computed in *Step 2*.

Step 4: Number of random walks computation. Each source node computes the number of launched random walks by equation (3).

A. Data dissemination

In the data dissemination process, each source node disseminates its generated data packet to w random selected sensor nodes through w random walks by multicast. Let L denote the length of each random walk. A data packet is transmitted for L steps in a random walk, and finally stops at an ending sensor node at the last step. In each step of the random walks launched by a source node, the sensor node first chooses the next hops, then multicasts the data packet to all the next hops. The next hops are chosen independently according to the probabilistic forwarding table. Let t^k_{ij} denote the number of times that node i chooses node j as a next hop in the k^{th} steps of the random walks. As an example shown in Fig. 1 (a), source node 0 launches five random walks by multicast. In the first step, source node 0 chooses nodes 1 and 2 as the next hops twice respectively (i.e., $t^1_{01} = t^1_{02} = 2$), and chooses node 3 as the next hop once (i.e., $t^1_{03} = 1$). It multicasts the data packet to all the next hops in a single transmission. Multicast reduces the data transmission times significantly. Without multicast, source node 0 disseminates the data packet to the next hops independently. As shown in Fig. 1 (b), source node 0 disseminates the data packet to nodes 1 and 2 twice, and to node 3 once. It needs five transmission times.

Note that t^k_{ij} is also referred to as the number of random walks assigned to node j from node i in the k^{th} step, since t^k_{ij} random walks from node i will go through node j. The value of k in t^k_{ij} is also referred to as the step counter for the assigned random walks. When $k = L$, i.e., the data packet in the assign random walks has been transmitted for L steps, node j will stop the assigned random walks without transmitting the data packet to other nodes, i.e., the data packet in the assigned random walks stops at node j. When sensor node i chooses itself t^k_{ii} times as the next hops, if $k < L$ in t^k_{ii}, before multicasting the data packet, it continues to choose the next hops t^k_{ii} times and increases the step counter by one.

To enjoy the shared nature of wireless medium, we design a new data structure for the data packet generated from the source node to encapsulate the sensed data, as shown in Fig. 2.

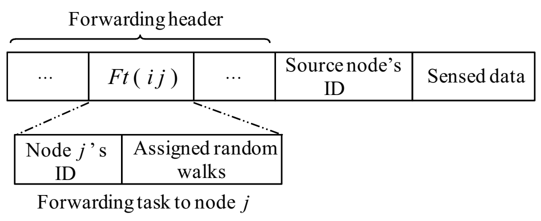


Fig. 2. Data structure of the data packet in node i.

The source node's ID is to identify each data packet. We use a forwarding header to assign the forwarding tasks to the next hops. The forwarding header (in bits) is much smaller than the sensed data (in bits). Let $Ft(ij)$ denote the forwarding task from node i to node j, which contains the next hop's ID (i.e., node j's ID) and the assigned random walks t^k_{ij}.

When node j receives a data packet from node i, it will perform the following steps.

Step 1: Node j checks the assigned random walks in $Ft(ij)$ (e.g., the value of t^k_{ij} and the value of k). If $k = L$ in t^k_{ij}, the t^k_{ij} assigned random walks stop at node j.

Step 2: Node j calculates the number of random walks that can be assigned to the next hops. Let m_{ij} denote the number of random walks in $Ft(ij)$ that can be assigned to the next hops. m_{ij} equals to

$$m_{ij} = \sum_{1 \le k < L} t^k_{ij}. \tag{5}$$

Step 3: Node j chooses m_{ij} next hops. Each next hop is chosen independently according to the probabilistic forwarding table. The step counters are increased by one.

Step 4: Node j updates the forwarding header in the data packet by attaching the next hops' ID and the assigned random walks.

Step 5: Node j multicasts the data packet to all the next hops simultaneously in a single transmission.

If node j receives another data packet from node v before *step 5*, it will check that if this data packet has the same source node's ID as the data packet received from node i. If they have the same source node's ID, node j will merge the two received packets by merging the two forwarding tasks $Ft(i, j)$ and $Ft(v, j)$. Node j repeats *step 1*, *step 2* and *step 3* to $Ft(v, j)$ by checking the assigned random walks in $Ft(v, j)$, calculating m_{vj} for $Ft(v, j)$, and independently choosing m_{vj} next hops. Node j merges these next hops'ID and the assigned random walks from $Ft(v, j)$ with that from $Ft(i, j)$ into one forwarding header in one data packet. Then *Step 5* is performed to multicast the data packet to all the next hops.

The data dissemination method in LTDC reduces the data transmission times significantly. As an example shown in Fig. 3, source node 0 launches five random walks. The length of each random walk is two. The transmission times in LTDC is 4. In the data dissemination method of the EDFC scheme [11], each random walk is launched independently. For the same data dissemination case in Fig. 3, the transmission times in EDFC is 10. Note that in both multicast and unicast, the

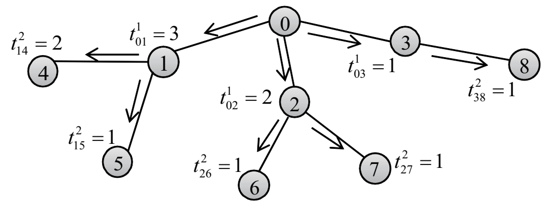


Fig. 3. Source node 0 launches five random walks. The length of each random walk is 2. The transmission times in EDFC is 4

next hops are chosen from the neighbors of the sensor nodes. The next hops are chosen independently according to the probabilistic forwarding table. Thus, the probability that a neighbor is chosen as a next hop by multicast is the same as that by unicast.

B. Data encoding

The stopping data packets in a sensor node for encoding are generated from different source nodes. The j^{th} distinct stopping data packet in a sensor node is denoted as c_j (e.g., c_1 is the first stopping data packet).

1) Encoding method: As long as a data packet stops at an ending sensor node in a random walk (i.e., a data packet is forwarded for L steps in a random walk), the ending sensor node at the last step encodes this data packet in *a main packet* with a certain probability. For a node i with generated code degree d, the main packet is an encoded data packet which is aimed at encoding d data packets randomly chosen from all the distinct stopping data packets. The distinct stopping data packets which are not encoded in the main packet are stored as *the backup packets*. The main packet in node i is denoted as f_i. Each distinct stopping data packet is encoded in the main packet f_i with probability $\frac{d}{e_d}$. If a data packet is encoded in the main packet f_i, the corresponding source node's IDs is also attached in f_i. Let $C(f_i)$ denote the number of data packets encoded in f_i. Since the actual number of distinct stopping data packets in node i is a_d, the number of data packets encoded in the main packet f_i equals to

$$C(f_i) = \frac{d}{e_d} \cdot a_d. \tag{6}$$

Since the actual degree a_d usually does not equal to the expected degree e_d, $C(f_i)$ usually does not equal to d. Each sensor node uses the backup packets to adjust the main packet f_i. Let S_b denote the set of backup packets. The number of backup packets equals to

$$|S_b| = a_d - C(f_i). \tag{7}$$

If the main packet f_i encodes less than d data packets, node i will choose some backup packets to adjust f_i, i.e., node i will choose some backup packets to encode in f_i. Let $\triangle f_i$ denote the number of backup packets for adjusting. After encoding $\triangle f_i$ backup packets in f_i, the number of data packets encoded in f_i equals to d (or is the closest to d). There are three cases about a_d: 1) $e_d \geq a_d \geq d$; 2) $a_d < d$; 3) $a_d > e_d$. Let $\triangle^l f_i$ denote the number of backup packets that should be encoded in f_i in the l^{th} case.

$$S_b = \{c_2, c_4, c_5\}$$

$$f_i = c_1 \oplus c_3 \oplus c_6 \xrightarrow{\text{Adjust}} f_i = c_1 \oplus c_3 \oplus c_6 \oplus c_4 \oplus c_5$$

$$f_i = f_i \oplus c_4 \oplus c_5$$

Fig. 4. In the first case when $d = 5$, $e_d = 8$ and $a_d = 6$, node i chooses some backup packets to encode in f_i.

In the first case $e_d \geq a_d \geq d$, from equation (6), $C(f_i) \leq d$, i.e., the number of data packets encoded in the main packet f_i is less than or equals to d. The number of backup packets that should be encoded in f_i is

$$\triangle^1 f_i = d - C(f_i). \tag{8}$$

Since $a_d \geq d$, from equations (7) and (8), we obtain

$$\triangle^1 f_i \leq |S_b|. \tag{9}$$

That is, the number of backup packets for adjusting is no more than $|S_b|$. Node i will randomly choose $\triangle^1 f_i$ backup packets from S_b to encode in f_i. After encoding $\triangle^1 f_i$ backup packets, f_i encodes d packets.

We take an example to show the data encoding method in the first case when $e_d \geq a_d \geq d$. In node i, the generated code degree is $d = 5$, the expected degree is $e_d = 8$, and the actual degree is $a_d = 6$. The stopping data packets are c_1, c_2, ..., c_7. Each data packet is encoded in f_i with probability $\frac{5}{8}$. As shown in Fig. 4, f_i encodes three data packets c_1, c_3, c_6. The set of backup packets is $S_b = \{c_2, c_4, c_5\}$. Since $C(f_i) = 3$ and $d = 5$, the number of backup packets that should be encoded in f_i is $\triangle^l f_i = 2$. Thus, node i chooses c_4 and c_5 from S_b to encode in f_i. After adjusting, f_i encodes 5 data packets.

In the second case $a_d < d$, the number of all distinct stopping data packets in node i is less than d. To make $C(f_i)$ the closest to d, all the backup packets are encoded in f_i. The number of backup packets for adjusting is

$$\triangle^2 f_i = |S_b|. \tag{10}$$

After adjusting, f_i encodes all the distinct stopping data packets. $C(f_i)$ is the closest to d.

In the third case $a_d > e_d$, from equation (6), $C(f_i) > d$, i.e., the number of data packets encoded in f_i will be larger than d. Thus, node i stops to encode any data packets in f_i when f_i has encoded d data packets. In this case, no backup packets are encoded in f_i. That is, the number of backup packets for adjusting is

$$\triangle^3 f_i = 0. \tag{11}$$

During data collection, the sensor nodes queried by the mobile base station will upload the main packets. The data packets encoded in the main packets affect the decoding of the original data packets. In EDFC [11], the d data packets for encoding are randomly chosen from all the distinct stopping data packets. We show that the probability of a stopping data packet being encoded in the main packet in LTDC is the same as that in EDFC through the following theorem.

Theorem 1: The probability of a stopping data packet being encoded in the main packet in LTDC is the same as that in EDFC.

Proof. Let P_E and P_L denote the probability of a stopping data packet c_j being encoded in the main packet f_i in EDFC and LTDC, respectively. We prove that $P_E = P_L$.

In EDFC, each sensor node randomly chooses d data packets from the a_d distinct stopping data packets to encode in the main packet. If $a_d \geq d$, $P_E = \frac{d}{a_d}$. If $a_d < d$, all the distinct stopping data packets are encoded in the main packet, i.e., $P_E = 1$.

In LTDC, a stopping data packet c_j may be encoded in the main packet f_i directly after being received, or may be stored as a backup packet and then chosen to encode in f_i. Let P_{L_d} denote the probability that c_j is encoded in f_i directly and P_{L_b} denote the probability that c_j is encoded in f_i after being stored as a backup packet. Note that $P_{L_d} = \frac{d}{e_d}$ and c_j is stored as a backup packet with probability $1 - \frac{d}{e_d}$. If $e_d \geq a_d \geq d$ (i.e., the first case), $\triangle^1 f_i$ backup packets are chosen from the $|S_b|$ backup packets to encode in f_i. Then,

$$P_{L_b} = (1 - \frac{d}{e_d})(\frac{\triangle^1 f_i}{|S_b|}). \tag{12}$$

Form equations (6), (7), (8) and (12), we can obtain

$$P_L = P_{L_d} + P_{L_b} = \frac{d}{a_d}. \tag{13}$$

If $a_d > e_d$ (i.e., the third case), none of the backup packets is chosen to encode in f_i (i.e., $P_{L_b} = 0$). The probability that c_j is encoded in f_i directly is $P_{L_d} = \frac{d}{e_d}$. If e_d stopping data packets have been received, c_j is not encoded in f_i since f_i has encoded d packets. c_j is encoded in f_i only if c_j is received before the sensor node has received e_d distinct stopping data packets. The probability that c_j is the first received e_d distinct stopping data packets is $\frac{e_d}{e_a}$. Thus,

$$P_L = \frac{e_d}{e_a} \cdot P_{L_d} = \frac{d}{a_d}. \tag{14}$$

From equations (13) and (14), we obtain $P_L = P_E$ if $a_d \geq d$ (i.e., the first and third cases). If $a_d < d$ (i.e., the second case), all the distinct stopping data packets are encoded in f_i, i.e., $P_L = 1$. That is $P_L = P_E$. $\square$

Since the mobile base station decodes the data packets based on the collected main packets, Theorem 1 implies that the data decoding ratios in LTDC and EDFC are almost the same. We will further evaluate the decoding ratios of LTDC and EDFC in the simulation.

2) Bounds of the number of backup packets for adjusting: We show the bounds of the number of backup packets for adjusting in LTDC.

From equations (9), (10) and (11), the number of backup packets that should be encoded in f_i is

$$0 \leq \triangle f_i \leq |S_b|. \tag{15}$$

Note that the sensor nodes choose backup packets to adjust f_i in the first and second cases when $a_d \leq e_d$. When $a_d \leq e_d$, from equation (6), equation (7) can be converted into

$$|S_b| = \frac{a_d}{e_d} \cdot (e_d - d). \tag{16}$$

Since $a_d \leq e_d$ and $e_d - d = x_d$, from equation (16), we obtain

$$|S_b| \leq x_d. \tag{17}$$

From equations (15) and (17), we obtain the bound of $\triangle f_i$ as

$$0 \leq \triangle f_i \leq x_d. \tag{18}$$

That is, in LTDC, the number of backup packets for adjusting is an integer varying from 0 to x_d.

3) Storage of the backup packets: Note that only in the first and second cases when $a_d \leq e_d$, the sensor nodes should choose the backup packets to encode in f_i. We consider the backup packets storage in the first and second cases when $a_d \leq e_d$.

Without encoding the backup packets, each buffer can store only one backup packet. Node i needs $|S_b|$ buffers to store the set of backup packets S_b. To adjust f_i, node i randomly chooses $\triangle f_i$ backup packets to encode in f_i. $\triangle f_i$ is an integer from 1 to $|S_b|$.

The storage overhead can be further reduced by encoding the backup packets as encoded data packets. Let f'_k denote the encoded backup packet that are stored in buffer b_k, $k = 1, 2, ..., B-1$. The main packet f_i is stored in buffer b_B. The number of backup packets encoded in f'_k $(1 \leq k \leq B-1)$ should satisfy

$$C(f'_k) \leq 2^{k-1}. \tag{19}$$

Let c'_j denote the j^{th} received backup packet in S_b, $j = 1, 2, ..., |S_b|$. The backup packets are encoded in the encoded packets from f'_1 to f'_{B-1}. That is, the first backup packet is encoded in f'_1, then the next received backup packets are encoded in f'_2 until $C(f'_k) = 2$ and so on.

Node i chooses some encoded backup packets which encode $\triangle f_i$ backup packets to encode in f_i. After adjusting, the main packet is

$$f_i = f_i \oplus f'_{p_1} \cdots \oplus f'_{p_r}, \tag{20}$$

where

$$C(f'_{p_1}) + \cdots C(f'_{p_r}) = \triangle f_i, \tag{21}$$

f'_{p_j} is the chosen encoded backup packet, p_j is the sequence number of f'_{p_j}, r is the total number of chosen encoded backup packets, $j = 1, ..., r$.

Note that unless the last encoded backup packet f'_{B-1} which may encode less than 2^{B-1} backup packets, each f'_k encodes 2^{k-1} backup packets, $k = 1, 2, ..., B-2$. For the sake of convenience, we consider that if there is $C(f'_k) = C(f'_{B-1})$ (or $\sum\limits_{k \in \{1,..,B-2\}} C(f'_k) = C(f'_{B-1})$), f'_k (or the combination of $C(f'_k)$, $k \in \{1, .., B-2\}$) has the priority to be encoded in f_i. That is f'_{B-1} is encoded in f_i only if it encodes 2^{B-1} backup packets. Thus, equation (21) can be represented as

$$\triangle f_i = v_1 2^0 + v_2 2^1 + \cdots + v_{B-1} 2^{B-2}, \tag{22}$$

where

$$v_k = \begin{cases} 1, & f'_k \text{ is encoded in } f_i, \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

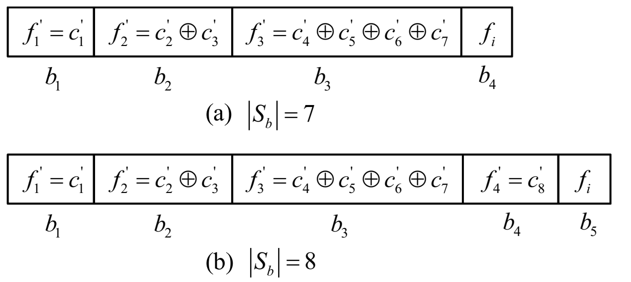

$f'_1 = c'_1$
$f'_2 = c'_2 \oplus c'_3$
$f'_3 = c'_4 \oplus c'_5 \oplus c'_6 \oplus c'_7$
f_i
b_1 b_2 b_3 b_4
(a) $|S_b| = 7$
$f'_1 = c'_1$
$f'_2 = c'_2 \oplus c'_3$
$f'_3 = c'_4 \oplus c'_5 \oplus c'_6 \oplus c'_7$
$f'_4 = c'_8$
f_i
b_1 b_2 b_3 b_4 b_5
(b) $|S_b| = 8$

Fig. 5. The backup packets are stored as encoded data packets in node i.

We show that $\triangle f_i$ in equation (22) can represent the integers from 1 to $|S_b|$. The $B-1$ encoded backup packets encode $|S_b|$ backup packets. That is

$$C(f'_1) + \cdots + C(f'_{B-1}) = |S_b|. \tag{24}$$

Since $C(f'_k) \leq 2^{k-1}$ $(k = 1, ..., B-1)$, we have

$$C(f'_1) + \cdots + C(f'_{B-1}) \leq 2^0 + \cdots + 2^{B-2}. \tag{25}$$

From equations (24) and (25), we obtain

$$|S_b| \leq 2^{B-1} - 1. \tag{26}$$

From equation (26), $|S_b| < 2^{B-1}$. Thus, each of the integers from 1 to $|S_b|$ can be represented by the $(B-1)$-bit binary. If 2^k is converted to binary digit, $\triangle f_i$ in equation (22) is the $(B-1)$-bit binary, in which v_k is the value in the k^{th} place of the $(B-1)$-bit binary. Therefore, $\triangle f_i$ in equation (22) can represent the integers from 1 to $|S_b|$.

Fig 5 shows how to encode the sets of backup packets $S_b = \{c'_1, c'_2, ..., c'_7\}$ and $S_b = \{c'_1, c'_2, ..., c'_8\}$. We can see that in Fig 5 (a) (or Fig 5 (b)), each of the integers from 1 to 7 (or 8) can be represented by the combination of $C(f'_i)$ $(i \in \{1, ..., 7$ (or 8)$\}$.

4) Storage overhead: We consider the storage overhead as the number of buffers for storing the data packets. If the backup packets are not encoded, node i should store $|S_b| + 1$ packets, which include $|S_b|$ backup packets and one main packet. Node i needs $B = |S_b| + 1$ buffers for storage. From equation (17), since $|S_b| \leq x_d$, the upper bound of storage overhead in node i is $x_d + 1$. The buffer size of each sensor node is set to maximum$\{x_d\} + 1$.

By encoding the backup packets, node i stores $B-1$ encoded backup packets and one main packet. Node i needs B buffers for storage. Since B is an integer, from equation (26) we obtain

$$B = \lceil \log_2(|S_b| + 1) \rceil + 1. \tag{27}$$

That is, node i needs $\lceil \log_2(|S_b| + 1) \rceil + 1$ buffers to store the encoded backup packets and main packet. Since $|S_b| \leq x_d$, the upper bound of storage overhead in node i is $\lceil \log_2(x_d + 1) \rceil + 1$. The buffer size of each sensor node is set to $\lceil \log_2(\text{maximum}\{x_d\} + 1) \rceil + 1$. Thus, encoding the backup packets further reduces the buffer size of sensor nodes.

Note that the storage overhead in LTDC (both the cases of un-encoding and encoding the backup packets) are less than that in EDFC [11], since each sensor node does not store all the original distinct stopping data packets in LTDC. In EDFC [11], each sensor node should first store all the distinct stopping data packets, then randomly chooses d data packets to do encoding. For a sensor node with code degree d, the number of all the distinct stopping data packets is a_d. Including the encoded data packet, the storage overhead in EDFC is $a_d + 1$. The buffer size of each sensor node in EDFC should be maximum$\{a_d\} + 1 =$ maximum$\{x_d + d\} + 1 = K + 1$.

5) Effect of the redundancy of code degree: We have shown that the upper bound of storage overhead in node i depends on the redundancy x_d. From equation (3), since $e_d = d + x_d$, the number of data packets disseminated from the source nodes also depends on the redundancy x_d. Thus, the upper bound of storage overhead and the number of disseminated data packets can be reduced by reducing the redundancy x_d. However, a small value of x_d may result in that the actual number of distinct stopping data packets in many sensor nodes are less than the generated code degree (i.e., $a_d < d$).

Let Pr$(a_d < d)$ denote the probability that the sensor nodes with generated code degree d receive less than d distinct stopping data packets. Lin et al. [11] have proved that Pr$(a_d < d)$ decreases exponential as $O((d + x_d)/\sqrt{K})$ and

$$\text{Pr}(a_d < d) = \sum_{j=0}^{d-1} \binom{K}{j} p^j (1-p)^{K-j}, \tag{28}$$

where $p = 1 - e^{-(d+x_d)/K}$ when $N \to \infty$.

Our objective is to minimize the upper bound of storage overhead in each sensor node, and also minimize the data dissemination cost which is governed by the number of disseminated data packets. Therefore, the optimization objective is to minimize x_d, subject to the constrains that the probability $Pr(a_d < d)$ should be sufficiently low for $d = 1, ..., K$. The optimization problem is formulated as follows.

$$\begin{aligned} &\text{minimize} \quad x_d \\ &\text{subject to} \quad \text{Pr}(a_d < d) \leq \delta_d \\ &\qquad\qquad x_d \geq 0 \\ &\qquad \text{for} \quad d = 1, ..., K, \end{aligned} \tag{29}$$

where δ_d is a small constant and Pr$(a_d < d)$ is given in equation (28).

The optimization problem in equation (29) can be solved off-line before network deployment [11]. Therefore, its complexity is not a main concern. Each generated code degree d has a optimal expected degree e_d. The network initiation can be set according to the suitable optimal values of expected degrees.

V. Performance Evaluation

We evaluate the performance of the proposed LTDC scheme by simulations. Since EDFC [11] is well-known and typical, we compare LTDC with EDFC in the simulations. We compare LTDC with EDFC on data dissemination cost, storage overhead and decoding ratio.

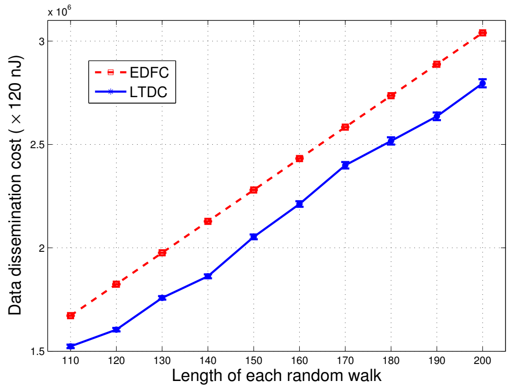


Fig. 6. Data dissemination cost vs. length of each random walk.

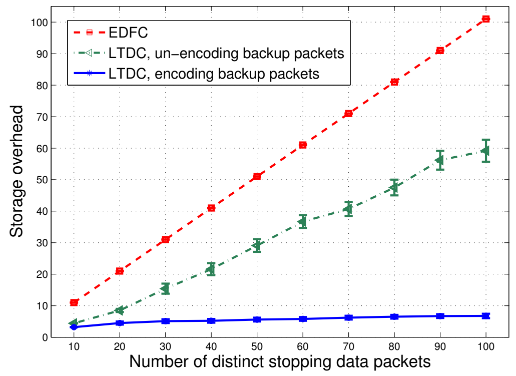


Fig. 7. Storage overhead vs. number of distinct stopping data packets.

We distribute the sensor nodes uniformly on a unit square. We set the number of sensor nodes as $N = 1000$ and the number of source nodes as $K = 100$. The transmission range $r = 0.08$, and LTDC is in the case of encoding the backup packets in most experiments with exceptions explicitly stated. The parameters in equation (1) for the generated degree are set as $\delta = 0.05$ and $c = 0.02$. When the mobile base station performs data collection, the queried sensor nodes upload the main packets. The mobile base station decodes the collected data by using the Belief Propagation algorithm [17]. To mitigate randomness, for each data point in all figures, we show the average and the 95% confidence interval from 100 independent experiments.

A. Data dissemination cost

We consider the data dissemination cost as the energy consumption for disseminating the data packets from the source nodes to a random subset of sensor nodes. The data dissemination cost mainly depends on the energy consumption for data sending and data receiving in each data transmission. Since the energy consumption for data receiving are the same in LTDC and EDFC, we compare LTDC with EDFC on energy consumption for data sending by varying the length of random walks. The energy consumption for sending a data packet is set as 120 nJ. As shown in Fig. 6, the data dissemination cost in the proposed LTDC scheme is lower than that in EDFC. This is because in EDFC each random walk is launched independently from a source node, while in the proposed LTDC scheme all the random walks from a source node is launched simultaneously by multicast. Multicast in LTDC reduces the data transmission times significantly, which results in the reduction of energy consumption.

B. Storage overhead

We compare LTDC with EDFC on storage overhead in the sensor nodes. We consider the storage overhead as the number of buffers for storing the data packets. The storage overhead in EDFC depends on the actual number of distinct stopping data packets (i.e., a_d). The storage overhead in LTDC depends on the number of backup packets. In LTDC, we show the storage overhead both in the cases of encoding and un-encoding the backup packets. Fig. 7 shows the storage overhead by varying the actual number of distinct stopping data packets in the sensor nodes. The storage overheads in LTDC (both the cases of encoding and un-encoding the backup packets) are much lower than that in EDFC, since in LTDC the sensor nodes do not store all the distinct stopping data packets while in EDFC they have to. Specially, encoding the backup packets in LTDC reduces the storage overhead much more significantly. The storage overhead in the case of encoding the backup packets in LTDC maintains low as the number of distinct stopping data packets increases.

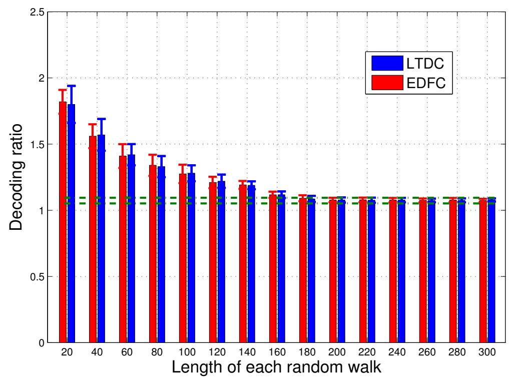


Fig. 8. Decoding ratio vs. length of each random walk.

C. Decoding ratio

To evaluate the fault tolerance of the proposed LTDC scheme, similar to that considered in [11], we define the decoding ratio R_d as

$$R_d = \frac{K_d}{K},$$

where K_d is the number of sensor nodes that should be queried by the mobile base station for successful decoding, K is the number of source nodes. The fewer sensor nodes that are required for querying to decode all the data packets, a higher percentage of sensor nodes are allowed to fail.

We compare LTDC with EDFC on the decoding ratio. Fig. 8 shows the decoding ratio by varying the length of each random walk. Since the decoding ratios in the two schemes are very close, we use histogram to show them clearly. To show the degree of fault tolerance, we also show the decoding ratio of the centralized LT codes. The two dashed lines represent the 95% confidence interval for the decoding ratio of the centralized LT codes. For both LTDC and EDFC, the decoding ratios decrease as the length of each random walk increases. The decoding ratios stay stationary on a certain value if the length of random walk exceeds a threshold. This is because the random walk approaches the steady-state distribution when the length increases. In Fig. 8, when the length of random walk is larger than 200, the decoding ratio stays stationary around 1.07, which implies that LTDC and EDFC achieve the same degree of fault tolerance as the original centralized LT codes. We also can see from Fig. 8, the decoding ratios in LTDC and EDFC are almost the same, since the probabilities of a stopping data packet being encoded in the main packet in the two schemes are similar. This fact is also stated in the proof of Theorem 1.

If some sensor nodes fail, the total number of sensor nodes N and the number of source nodes K decrease. It is not feasible to update K and N to all sensor nodes whenever they change. The sensor nodes will overestimate the values of N and K. We also evaluate the decoding ratio of LTDC under this condition. Successful decoding is still achieved when K and N decrease. The decoding ratio maintains low (around 1.07) when only N decreases, while it increases as K decreases. Due to the space limitation, we do not show the results in detail. To maintain a low decoding ratio, we can let the sensor nodes take turns to act as the source nodes, which can guarantee that the number K is not too small. Update N and K periodically is also a reasonable method to maintain a low decoding ratio [11], [13].

VI. Conclusions

In this paper, we propose a novel LT codes based Distributed Coding (LTDC) scheme for efficient distributed storage in WSNs. In LTDC, we apply random walks by multicast to improve the efficiency of data dissemination. During data encoding, the sensor node encodes the data packet in a main packet with a certain probability, without storing all the original stopping data packets. The number of encoded data packets in the main packet follows the distribution of LT codes. The mobile base station is able to decode the original data by querying any subset of sensor nodes (at least K sensor nodes). The comprehensive performance evaluation has been conducted through computer simulation. It is shown that the proposed LTDC scheme has lower dissemination cost and lower storage overhead while maintains the same level of fault tolerance as the original LT codes.

Acknowledgments

This work is partially supported by Grant-in-Aid for Scientific Research of Japan Society for Promotion of Science (JSPS), Research Collaboration Grant from NII, National Natural Science Foundation of China (NSFC) project (Nos. 91438121 and 61373156), Key Basic Research Project (No.12JC1405400) and Shanghai Pujiang Program (No. 13PJ1404600) of the Shanghai Municipality, and Shanghai Branch of Southwest Electron and Telecom Technology Research Institute Project (No. 2013008).

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comm. ACM*, vol. 38, no. 4, pp. 393–422, 2002.

[2] Y. Ren, V. A. Oleshchuk, and F. Y. Li, "Optimized secure and reliable distributed data storage scheme and performance evaluation in unattended wsns," *Computer Communications*, vol. 36, pp. 1067 – 1077, 2013.

[3] G. Maia, D. Guidonia, A. Vianab, A. Aquinoc, R. Minid, and A. Loureiroa, "A distributed data storage protocol for heterogeneous wireless sensor networks with mobile sinks," *Ad Hoc Networks*, vol. 11, pp. 1588 – 1602, 2013.

[4] D. Leong, A. Dimakis, and T. Ho, "Distributed storage allocations," *IEEE Transactions on Information Theory*, vol. 58, pp. 4733 – 4752, 2012.

[5] M. Gerami, X. Ming, C. Fischione, and M. Skoglund, "Decentralized minimum-cost repair for distributed storage systems," in *Proceedings of IEEE ICC 2013*. IEEE, 2013.

[6] A. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2809–2816, 2006.

[7] D. Wang, Q. Zhang, and J. Liu, "Partial network coding: Concept, performance, and application for continuous data collection in sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 3, pp. 1–22, 2008.

[8] X. Ye, J. Li, and L. Xu, "Group data collection in wireless sensor networks with a mobile base station," in *Proceedings of IEEE WCNC 2013*. IEEE, 2013.

[9] A. Dimakis, V. Prabhakarna, and K. Ramchandran, "Distributed fountain codes for networked storage," in *Proceeding of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2006.

[10] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth codes: Maximizing sensor network data persistence," in *Proceedings of ACM Sigcom 06*, 2006.

[11] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *Proceedings of IEEE INFOCOM 2007*. IEEE, 2007.

[12] R. Beraldi, R. Baldoni, and R. Prakash, "A biased random walk routing protocol for wireless sensor networks: The lukewarm potato protocol," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 1649 – 1661, 2010.

[13] Z. Kong, S. Aly, and E. Soljanin, "Decentralized coding algorithms for distributed storage in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 261 – 267, 2010.

[14] S. Jafarizadeh and A. Jamalipour, "Data persistency in wireless sensor networks using distributed luby transform codes," *IEEE Sensors Journal*, vol. 13, no. 12, pp. 4880 – 4890, 2013.

[15] D. Vukobratovic, C. Stefanovic, V. Crnojevic, F. Chiti, and R. Fantacci, "Rateless packet approach for data gathering in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 1169 – 1179, 2010.

[16] M. Luby, "Lt codes," in *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, 2002.

[17] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Info. Theory*, vol. 47, pp. 569 – 584, 2001.

[18] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Mixing times for random walks on geometric random graphs," in *Proceedings of SIAM Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, 2005.

[19] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equations of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, pp. 1087 – 1092, 1953.

[20] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, vol. 46, pp. 667 – 689, 2004.

[21] A. Sinclair and M. Jerrum, "Approximate counting, uniform generation and rapidly mixing markov chains," *Information and Computation*, vol. 82, pp. 93 – 133, 1989.