A Network Coding Based Energy Efficient Data Backup in Survivability-Heterogeneous Sensor Networks

Jie Tian, Tan Yan, and Guiling Wang

Abstract—Sensor nodes deployed outdoors are subject to environmental detriments and often need to cache data for an extended period of time. This paper introduces sensor nodes which are robust to environmental damages, and proposes to utilize Network Coding to back up data in the robust sensors for future data retrieval in an energy efficient way. Our goal is to help regular sensors select robust sensors to back up their data with low energy consumption, such that when needed, all the data can be retrieved by querying only a subset of robust sensors. We formally formulate this backup problem, theoretically prove its NP-Completeness, discover two novel theoretical guidelines for problem solving, and propose two algorithms accordingly to tackle this NP-C problem. The guidelines are based on random linear network coding and provide lower bounds of the number of robust sensors that each regular sensor should choose for data backup, such that the required fault tolerance is provided. A centralized algorithm and a distributed algorithm are developed based on the guidelines such that regular sensors can back up their data efficiently. Both analysis and simulation show our algorithms are effective in achieving fault tolerance, low energy consumption, and high retrieval efficiency.

Index Terms—Heterogeneous sensor networks, data backup, network coding

1 Introduction

Due to the small size and low cost of a sensor node, a wireless sensor network composed of a large number of such nodes can be deployed close to the phenomena or events of interest, monitoring them and generating data. The generated precious data may not be able to be collected constantly and immediately considering many constraints in the physical world, especially in remote and hostile areas. For example, in Great Duck Island, a sensor network has been monitoring the habitat of wild birds [1]. The habitat data can only be collected from the sensors occasionally to minimize the interference on birds' natural life. To let sensors increase the transmission power and remotely send data to human operator drains the battery power quickly or simply is infeasible if the distance between a data collector and the sensor network is too large. Therefore, a sensor network has to act as a distributed data storage before data collection. The duration that data have to be cached in a sensor network varies from minutes to months.

The environments in which a sensor network needs to cache data for an extended period of time before data collection are generally remote or less accessible. Especially in these environments, sensor nodes, which are tiny electronic devices, are subject to environmental damages, such as rain and fire. When a sensor node dies due to the physical damages, the data in the node are lost. Therefore, it is important to back up the data. To simply duplicate data in multiple tiny sensor nodes cannot provide enough fault tolerance because sensors are likely to fail at the same time when the harsh environmental attributes act on them. For example, after a storm, most of these small electronic devices may fail simultaneously. Even though after they are dried in sunshine and are able to work again, it is not likely that the lost data can be recovered. To deal with the problem, we propose to incorporate sensor nodes which are robust to environmental damages. We assume they are water-proof, can tolerate high temperature and withstand other environmental attributes. Considering such robust sensor nodes are of higher cost, we propose to construct heterogeneous sensor networks with both regular and robust sensors. The focus of the paper is to design schemes for regular sensors to back up data in robust sensors.

Our objective is to design energy-efficient data-backup schemes for the proposed heterogeneous sensor networks to achieve high fault tolerance in harsh environment. Considering in harsh environment, all regular sensors may lose data after a storm and some robust sensors may fail due to energy depletion or other reasons, a desired scheme should be able to tolerate the failure of all regular sensors and a portion of the robust sensors. In other words, by accessing any b out of n robust sensors ($b \leqslant n$), all the data stored in the network can be recovered. Existing distributed data storage systems [2], [3], [4], [5] cannot be directly applied to solve our problem either because they cannot achieve the desired fault tolerance or because their system requirement is too high. For example, cluster based storage systems [2], [3] cannot tolerate the failure of all storage nodes in a cluster. Some coding-based data storage systems [4], [5] can tolerate that, but they are under the prerequisite that there are more storage nodes than the data nodes. This means we have to budget more robust sensors

• *The authors are with the Department of Computer Science, New Jersey Institute of Technology. E-mail: {jt66, ty7, gwang}@njit.edu.*

*Manuscript received 28 Apr. 2014; revised 29 Oct. 2014; accepted 3 Nov. 2014. Date of publication 24 Nov. 2014; date of current version 31 Aug. 2015.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2014.2374168*

than regular sensors and the cost of a sensor network is greatly increased. Moreover, the energy consumption of communication for backup is not considered in many existing schemes. This paper aims to develop energy-efficient data backup schemes which can provide required level of fault tolerance.

To achieve the goal, we first theoretically analyze the problem, formulate it as a weighted backup problem, and prove its NP-Complete nature. We also discover two novel theoretical guidelines based on random linear network coding satisfying any of the two can guarantee the desired fault tolerance requirement: all data can be recovered by accessing any b out of the n robust sensors. Based on the two guidelines, two backup schemes are designed to achieve fault tolerance and energy efficiency simultaneously. Theoretical analysis and performance evaluation show that our schemes greatly outperform comparable ones in terms of fault tolerance, energy consumption and retrieval efficiency.

The remainder of the paper is organized as follows. We formulate the problem and prove its NP-Completeness in Section 2. Section 3 presents an overview of our solution. Guideline 1 and the algorithm based on it are presented in Section 4. Section 5 presents Guideline 2 and the algorithm designed on it. The discussion about coefficient matrix is presented in Section 6. Section 7 reports the simulation results. The related work is discussed in Section 8. Finally, we conclude the paper in Section 9.

2 Problem Formulation

In this section, we first introduce the notations and data structure employed. Then we formally define the problem and prove it is an NP-Complete problem.

2.1 Notations and Data Structure

We consider a wireless sensor network composed of k regular sensors and n robust sensors, where $k \geq n$. Without loss of generality, we normalize the data storage of regular sensors to be one unit and robust sensors have s-unit data storage, where $s \geq 1$. In the backup, a regular sensor backs up its one-unit data in some robust sensors. As each robust node can store s-unit data, a data collector has to query at least b robust sensors to recover all the data, where $b = \ulcorner \frac{k}{s} \urcorner \leq n$. To ensure the above inequality, we assume $ns \geq k$; otherwise, the problem becomes unsolvable.

We use three types of graphs to illustrate and analyze the data backup scenario, *Deployment Graph*, *Backup Graph* and *Storage Graph*, based on which, we formulate Weighted Backup Problem and formally prove its NP-Complete nature.

Deployment graph. Given a Bipartite Graph $G = ((V_g + V_r), E)$, which contains two sets of vertices, V_g and V_r. $V_g(= \{u_1, u_2, \ldots, u_k\})$ is the set of regular sensors, and $V_r(= \{v_1, v_2, \ldots, v_n\})$ is the set of robust sensors. An edge $(u, v) \in E$, $u \in V_g$ and $v \in V_r$, if and only if there is a routing path between sensor u and sensor v. W is a set of weights associated to each edge $(u, v) \in E$, which is the routing cost between u and v.

Backup graph. Given a Deployment Graph G, the Backup Graph $G^b = ((V_g + V_r), E^b)$ is a graph such that all the vertices are the same as that in G, and an edge $(u, v) \in E^b$, $u \in V_g$ and $v \in V_r$, if and only if $(u, v) \in E$ is in G and regular sensor u backs up its data in robust sensor v.

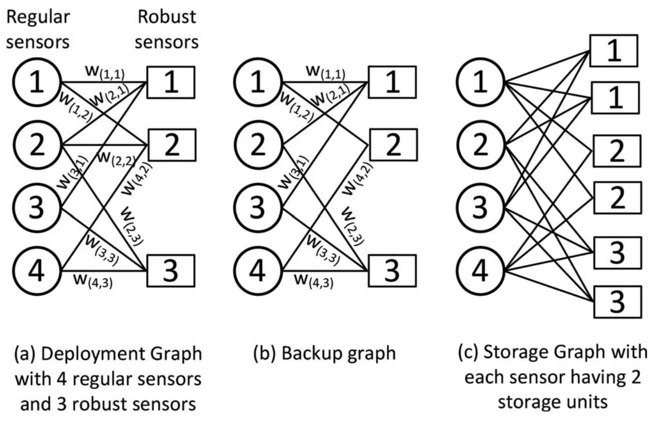

Regular sensors
Robust sensors
W(1,1)
W(1,2)
W(2,1)
W(2,2)
W(3,1)
W(4,2)
W(2,3)
W(3,3)
W(4,3)
(a) Deployment Graph with 4 regular sensors and 3 robust sensors
(b) Backup graph
(c) Storage Graph with each sensor having 2 storage units

Fig. 1. Definition of graphs.

Storage graph. For a Backup Graph G^b, assume each robust sensor in V_r has s units of storage. The Storage Graph G^s is a graph constructed in the way such that, for each vertex in V_r, duplicate it and its connected edges s times.

Fig. 1a is an example of a Deployment Graph with four regular sensors and three robust sensors, where $w_{(u,v)}$ is the routing cost (weight) between vertices u and v. In the network represented by such Deployment Graph, if regular sensors $\{1, 2, 3, 4\}$ back up their data to robust sensors $\{1, 2\}$, $\{1, 3\}$, $\{1, 3\}$, and $\{2, 3\}$, respectively, the constructed Backup Graph is as shown in Fig. 1b. Fig. 1c is the corresponding Storage Graph with each robust sensor having two units of storage.

2.2 Problem Definition

The formal definition of the Weighted Backup Problem is presented as follows.

Definition 2.1 (Weighted Backup Problem). *Given a network represented by Deployment Graph $G = ((V_g + V_r), E)$, we assume each robust sensor in V_r has s ($s \geq |V_g|/|V_r|$) storage units. Let $b = \ulcorner \frac{|V_g|}{s} \urcorner \leq |V_r|$. Our problem is: each regular sensor in V_g forwards its data to some robust sensors in V_r for backup, such that: (1) by picking any arbitrary b robust sensors from V_r, one can recover the data from all the sensors in V_g, and (2) the total forwarding cost is minimized.*

The Weighted Backup Problem is NP-Complete. Before we present the proof, we first present Lemma 2.1 and its corollary, Corollary 2.1. Then we prove the NP-Completeness of the problem.

Lemma 2.1. *Consider a network that can be represented by a Deployment Graph G, the data from all the regular sensors in G can be recovered if and only if the degree of each vertex in V_g in the constructed Storage Graph G^s is no less than s.*

Corollary 2.1. *The data from all the regular sensors in V_g in a Deployment Graph G can be recovered if and only if the constructed Backup Graph $G^b = ((V_g + V_r), E^b)$ has a Hitting Set with size no larger than $|V_r|$. That means, in G^b, every vertex in V_g connects to at least a vertex from V_r.*

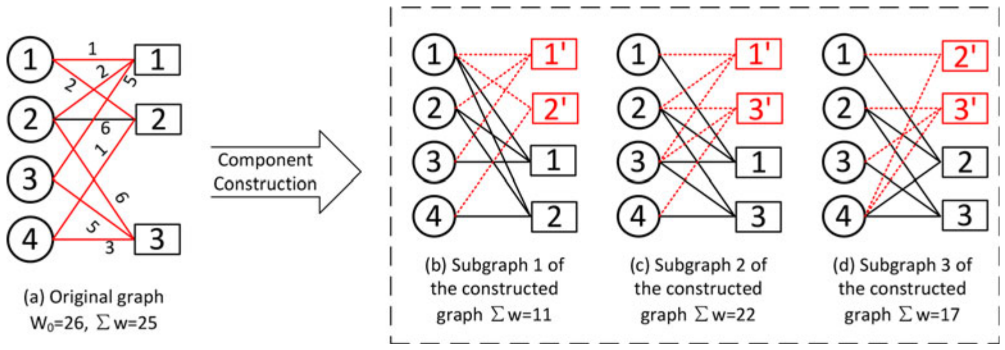

(a) Original graph
W0=26, Σw=25
Component Construction
(b) Subgraph 1 of the constructed graph Σw=11
(c) Subgraph 2 of the constructed graph Σw=22
(d) Subgraph 3 of the constructed graph Σw=17

Fig. 2. Proof of Theorem 2.1.

Lemma 2.1 and Corollary 2.1 can be proved by contradiction. In Lemma 2.1, if in Storage Graph there exists a vertex in V_g that has degree less than s, then this vertex does not have any edge in the corresponding Backup Graph because every edge in Backup Graph should be duplicated s times in the Storage Graph. That means this regular sensor does not backup its data to any robust sensors, and thus its data cannot be recovered. Same for Corollary 2.1, the data of a regular sensor cannot be recovered if it does not connect to at least a vertex from V_r. Assume each regular sensor backs up its data to at least one robust sensor. From Corollary 2.1 we can see that, among all the robust sensors, if we pick up a set of b sensors, the picked sensors can recover the data of all the regular sensors if and only if the size of the Hitting Set of the Backup Graph constructed from the picked robust sensors and all the regular sensors is no larger than b. Thus, the objective of Weighted Backup Problem in Definition 2.1 can be rewritten as follows:

Given a Deployment Graph $G = ((V_g + V_r), E)$, select a set of edges $E' \subseteq E$, which is subjected to:

- *Subj. (1).* Constructing a graph $G^b = ((V_g + V_r'), E^b)$ with: (1) all the vertices from V_g in G, (2) an arbitrary set of b vertices $V_r' \subseteq V_r$, and (3) $E^b \subseteq E'$ such that E^b connects V_g and V_r' in G, one should have the size of the Hitting Set of G^b no larger than b.
- *Subj. (2).* $\sum_{(u,v)\in E'} w_{(u,v)}$ is minimized.

2.3 NPC Proof

Theorem 2.1. *The Weighted Backup Problem is NP-Complete.*

Proof of Theorem 2.1. To facilitate the proof, we formulate the decision version of the Weighted Backup Problem as, given a positive value W_0, is there a set of edges $E' \subseteq E$ that solves the problem with $\sum_{(u,v)\in E'} w_{(u,v)} \le W_0$?

Prove to be in NP. It is obvious to see that Weighted Backup Problem $\in$ NP, since a nondeterministic algorithm only needs to guess a subset of edges and check in polynomial time to determine whether that subset satisfies both Subj. (1) and Subj. (2) and with the sum of the weights less than or equal to W_0.

Component construction. We reduce Weighted Hitting Set Problem [6] to Weighted Backup Problem by applying *component construction* to the problem.

Let an arbitrary instance of the Weighted Backup Problem be given by the graph $G = (V_g + V_r, E)$ as defined in Definition 2.1. We construct a graph instance $\widehat{G} = (\widehat{V}_g + \widehat{V}_r, \widehat{E})$ through the following steps:

1) Duplicate V_g $\binom{|V_r|}{b}$ times and assign them to $\widehat{V}_g$. Thus, $|\widehat{V}_g| = \binom{|V_r|}{b} \times |V_g|$.
2) Among all the vertices in V_r, select $\binom{|V_r|}{b}$ combinations and assign to $\widehat{V}_r$.
3) For each combination in $\widehat{V}_r$, associate them one by one to a set of vertices in $\widehat{V}_g$. Each combination is considered as a subgraph of $\widehat{G}$.
4) In each subgraph generated in Step 3, for each vertex u in $\widehat{V}_g$ and v in $\widehat{V}_r$, connect them and associate them to corresponding weight if there is an edge $(u, v) \in E$ in the original graph G.
5) If there is a solution with a subset of edges $E' \subseteq E$ for the Weighted Backup Problem in original graph G, repeat Step 3; change E to E' in the setting of Step 4 and then repeat Step 4.
6) For each vertex u in $\widehat{G}$, its weight $\widehat{w_u}$ is the sum of all its edges' weights.

Fig. 2 is an example of this construction with four regular sensors, three robust sensors, and $b = 2$. The constructed graph $\widehat{G}$ is divided into $\binom{|V_r|}{b} = \binom{3}{2} = 3$ subgraphs, where the red rectangles and lines are the vertices and edges added in Step 5.

We now claim the Weighted Backup Problem has a solution with $E' \subseteq E$ and $\sum_{(u,v)\in E'} w_{(u,v)} \le W_0$, if and only if the Weighted Hitting Set Problem in the constructed graph $\widehat{G}$ has a solution $\widehat{V} \subseteq (\widehat{V}_g + \widehat{V}_r)$ and $\sum_{u\in\widehat{V}} \widehat{w_u} \le (b-1) \times W_0$.

Reduction. If there is a solution with $E' \subseteq E$ and $\sum_{(u,v)\in S} w_{u,v} \le W_0$ that satisfies Subj. (1), according to Step 5, we add each combination of V_r and the corresponding edges in E' to $\widehat{G}$. Obviously, each combination of V_r is the Hitting Set of each subgraph of $\widehat{G}$. For example, in Fig. 2, the vertices $\{1', 2'\}$ are the Hitting Set for vertex set $\{1, 2, 3, 4\}$ in the left in subgraph 1, and $\{1', 3'\}$ and $\{2', 3'\}$ are for subgraph 2 and 3, respectively. Furthermore, all vertices added in Step 5 together are the Hitting Set of the entire graph $\widehat{G}$. Now we check the weight. In original graph G, to satisfy Subj. (1), the sum of all the weights in S is no larger than W_0, In the constructed graph $\widehat{G}$, according to Step 5, by adding $\binom{|V_r|}{b}$ combinations of sensors from V_r, each vertex in V_r is added exactly $(b-1)$ times in this step, and thus each edge in the solution E' and the corresponding weight are added exactly $(b-1)$ times. Since all vertices added in Step 5 together are the Hitting Set of $\widehat{G}$ and we duplicate E' exactly $(b-1)$ times, in total the sum of the weights of all the vertices are no larger than $(b-1)\times W_0$.

Conversely, if there is a solution $\widehat{V}\subseteq(\widehat{V}_g+\widehat{V}_r)$ to the constructed Weighted Hitting Set Problem with $\sum_{u\in\widehat{V}}\widehat{w_u}\leq(b-1)\times W_0$, simply pick all the unique vertices from $\widehat{V}_r$ and the corresponding edges. It is easy to see the picked edges are the solution to the Weighted Backup Problem. Moreover, since every picked edge appears exactly $(b-1)$ times for $\binom{|V_r|}{b}$ combinations, the sum of the weights of all the picked edges are less than $(b-1)\times W_0/(b-1)=W_0$.

Conclusion. Therefore, the Weighted Hitting Set Problem is reducible to the Weighted Backup Problem. Since all the component construction and reduction are done in polynomial time, the original Weighted Backup Problem is NP-Complete. □

Take Fig. 2 as an example. If $W_0=26$, the figure is a "yes" instance to the Weighted Backup Problem as the sum of weights of all the red edges is $25\leq W_0$.

3 Solution Overview

The objectives of the addressed problem are to meet the required fault tolerance and minimize energy consumption. The problem is NP-Complete and no polynomial-time algorithm can provide an optimal solution. Our realistic goal is to design algorithms which can meet the fault tolerance requirement and have a low energy consumption even though the consumption is not minimized.

We adopt the random linear network coding framework to provide the required fault tolerance. Based on network coding, regular sensors back up data on robust sensors in an encoded format; encoded data on robust sensors are retrieved and original data can be recovered. Inside this framework, our algorithms specify how a regular sensor determines at which robust sensors its data is backed up, such that by querying b out of n robust sensors, all the data generated in the network can be recovered.

Our strategy is to first discover conditions satisfying which the fault tolerance requirement can be met, and then design centralized and distributed algorithms which can satisfy those conditions and have low energy consumption.

In this section, we first introduce the background of random linear network coding. Then we present the assumptions and energy model in data backup and recovery. After that, we present how to use the network coding technique to do data backup and recovery. Finally, we present Lemma 3.1 which is the foundation of our discovered conditions. We name the conditions Guideline 1 and Guideline 2. The two guidelines and the designed algorithms based on them are presented in Sections 4 and 5, respectively.

3.1 Background on Random Linear Network Coding

Random linear network coding is widely used in data storage system [7]. In network coding, each one-unit data d_i is viewed as an element over the finite field $GF(2^q)$. m-unit original data $D(=\{d_1,\ldots,d_m\})$ for a source node can be encoded into s-unit data $X(=\{x_1,\ldots,x_s\})$ and stored in a storage node with at least s-unit space. Here m can be equal to or less than or more than s. To perform the encoding, an $m\times s$ coefficient matrix G is chosen, each element of which is uniformly and independently generated on $GF(2^q)$. The encoded data $X=D\cdot G$. Thus, in addition to storing X, each storage node also needs to store the coefficient matrix G for encoding and decoding. The coefficient matrix occupies msq bits.

To recover n-unit data which originated from one or more source data nodes with a high probability, a data collector needs to retrieve n-unit encoded data along with their coefficients from one or multiple storage node. A linear system of n linear equations and n variables is generated and then solved to retrieve original n-unit data.

A necessary condition to encode and decode successfully is that the coefficient vectors must be linearly independent. As shown in [8], in a large enough field, the probability of linear independency in coefficients is close to 1 and thus the success ratio of decoding is close to 1. For example, the probability is over 99.6 percent when $q=8$. Our work is based on the above result.

3.2 Assumptions and Energy Model

In the paper, we make the following assumptions:

- In the network, robust sensors are assumed to be synchronized since a regular sensor backs up data in multiple robust sensors and the version consistency is an issue. The synchronization can be achieved by many mature techniques with low overheads [9], [10], [11]. The synchronization between regular sensors is not required. The synchronization between a regular sensor and a robust sensor is not required either.
- In terms of the energy consumption, we adopt the energy model proposed in [12]. According to [12], the energy consumed in transmitting and receiving a message with l-bits over a distance d is denoted by $E_{Tx}(l,d)$ and $E_{Rx}(l)$, respectively. The distance d is called transmission distance. The formulas to calculate $E_{Tx}(l,d)$ and $E_{Rx}(l)$ are as follows:

$$E_{Tx}(l,d)=l\times(E_{elec}+\epsilon d^{\alpha}),$$
$$E_{Rx}(l)=l\times E_{elec},$$

  where E_{elec} is the electronics energy depending on factors such as the digital coding, modulation, filtering, and spreading of the signal, $\epsilon\in\{\epsilon_{fs},\epsilon_{mp}\}$ is the transmitter amplifier in the free-space (ϵ_{fs}) model or the multipath (ϵ_{mp}) model, and α is the path-loss exponent, with $2\leq\alpha\leq 4$. The energy model has been widely adopted in many applications and protocol designs [13], [14], [15].

3.3 Data Storing and Recovery

In our heterogeneous sensor network, the data produced by all the k regular sensors are to be backed up in the n robust sensors in an encoded and redundant way, such that by

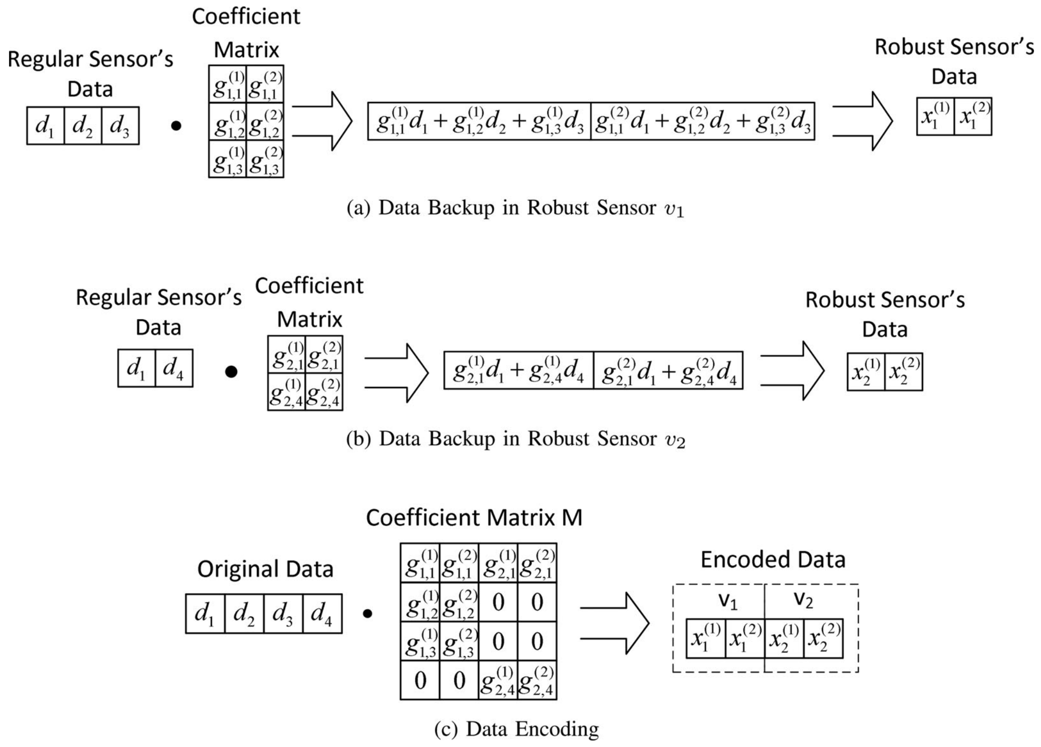

Regular Sensor's Data
Coefficient Matrix
Robust Sensor's Data
d_1 d_2 d_3
$g_{1,1}^{(1)}d_1+g_{1,2}^{(1)}d_2+g_{1,3}^{(1)}d_3$ $g_{1,1}^{(2)}d_1+g_{1,2}^{(2)}d_2+g_{1,3}^{(2)}d_3$
$x_1^{(1)}$ $x_1^{(2)}$
(a) Data Backup in Robust Sensor v_1
$g_{2,1}^{(1)}d_1+g_{2,4}^{(1)}d_4$ $g_{2,1}^{(2)}d_1+g_{2,4}^{(2)}d_4$
$x_2^{(1)}$ $x_2^{(2)}$
(b) Data Backup in Robust Sensor v_2
Original Data
Coefficient Matrix M
Encoded Data
v1 v2
(c) Data Encoding

Fig. 3. A data backup example.

querying any b robust sensors, the original data can be recovered. To be simplicity, we normalize the data generated in each regular sensor to be one unit between two consecutive backups. In the backup, each regular sensor sends this one unit data to a number of robust sensors.[1] Since all the backup processes are the same, in the following, we only focus on one-time backup. In each data backup, a robust sensor stores s-unit encoded data. s is calculated by $\ulcorner \frac{k}{b} \urcorner$. When a robust sensor receives data from m regular sensors, it encodes m-unit data into s-unit data using random linear network coding. It is obvious that $m \leqslant k$. In addition, a robust sensor also stores a coefficient matrix, which is used for the every encoding and decoding.

In the data recovery, a data collector retrieves all the coefficients from b robust sensors, and $k \times f$-unit encoded data. Here f is the backup frequency between two data retrievals.[2] Then a linear system involving k equations and k variables is built and used to decode the original $k \times f$-unit data.

We use the data backup in Fig. 1b as an example to illustrate the data backup and recovery process. In this example, regular sensor u_1 backs its data up at robust sensor v_1 and v_2. u_2 backs its data up at v_1 and v_3. u_3 backs its data up at v_1 and v_3. u_4 backs its data up at v_2 and v_3. Therefore, robust sensor v_1 receives three-unit data in total from three regular sensors, u_1, u_2, and u_3. v_2 receives two-unit data from u_1 and u_4. v_3 receives three-unit data from u_2, u_3, and u_4. (As for why the regular sensors choose these robust sensors to do backup, it is determined by our algorithms, which will be presented in the next two sections.) Given this backup structure, robust sensors v_1 and v_3 will generate six coefficients (a 3×2 coefficient matrix), while v_2 generates four coefficients (a 2×2 coefficient matrix), since v_1 and v_3 need to encode three-unit data into two-unit data while v_2 needs to encode two-unit data into the same size data. Once the coefficient matrices are determined in the network initialization phase, they are used throughout the network lifetime for each data backup.

The above backup plan calculated by our algorithm guarantees that by querying any two robust sensors, all the four unit data can be recovered. Without loss of generality, we assume v_1 and v_2 are queried. In the following, we will show the data encoding and decoding process at v_1 and v_2. Let d_j denote the data from regular sensor u_j, and $x_i^{(l)}$ denote the lth unit encoded data in robust sensor v_i. Let $g_{i,j}^{(l)}$ denote the coefficient of robust sensor v_i for encoding the data received from regular sensor u_j to generate the lth unit encoded data. The encoding process at v_1 and v_2 is illustrated at Figs. 3a and 3b, respectively. After receiving data from regular sensors, the robust sensors multiply them by their coefficient matrices and generate the encoded data.

The encoding process can be viewed from a different perspective. All the data generated by this simple network (d_1, d_2, d_3 and d_4) is a row of data. Based on the coefficient matrices of v_1 and v_2, a global 4×4 matrix M can be generated, as shown in Fig. 3c. In matrix M, certain coefficients are zero because corresponding regular sensor and robust sensor do

1. The size of data generated in each regular sensor can be varied. The backup in each robust sensor is only performed on a predefined unit of data size, which is q bits as mentioned in Section 3.1. When the data size is larger than predefined data size, the data are separated into several blocks, each of which is encoded in the robust sensor separately.

2. The backup in every robust sensor is performed at a frequency predefined by the application, such as once an hour. The frequency can be dynamically adjusted based on data generating speed, weather condition, and failure model of the regular sensors. If a regular sensor fails before the next backup, its data are lost and cannot be recovered. However, since the weather information is known beforehand, this frequency can be adjusted such that data can be backed up before the next storm and such that they will not be lost.

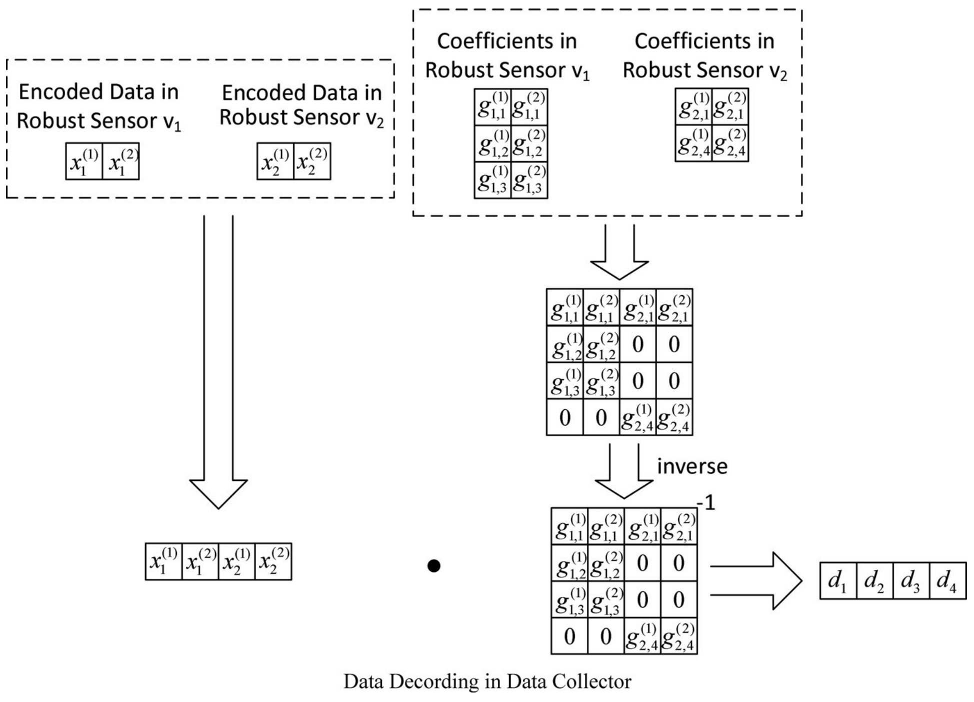

Encoded Data in Robust Sensor v_1
$x_1^{(1)}$ $x_1^{(2)}$
Encoded Data in Robust Sensor v_2
$x_2^{(1)}$ $x_2^{(2)}$
Coefficients in Robust Sensor v_1
Coefficients in Robust Sensor v_2
inverse
-1
d_1 d_2 d_3 d_4
Data Decording in Data Collector

Fig. 4. A data recovery example.

not have this backup relationship. Multiplying the row of data with the coefficient matrix results in the encoded data. Thus, to recover the data, we can simply multiply the encoded data with the inverse of the coefficient matrix.

When a data collector needs to retrieve data, it requests the encoded data and coefficient matrices from robust sensors v_1 and v_2, as illustrated in Fig. 4. Note that it only needs to request the coefficient matrices once and stores them to avoid unnecessary communication overhead. Then the collector can build the 4×4 coefficient matrix M out of the received matrices from v_1 and v_2 and calculate its inverse. Multiplying the four-unit row of encoded data and the inverse can recover the original four-unit data. Note that the matrix inverse must exist in order to recover the original data. Our algorithms calculate the backup schedule, following the schedule can guarantee the existence of M's inverse with a high probability. In the paper, the detailed process of data collection from b robust sensors to a certain data collector to recover original data is not considered. There are several existing methods [16], [17] about how to collect the data in an energy efficient way.

In the following, we provide a preliminary analysis. Then in the next two sections, we present our algorithms and prove they can provide such a guarantee.

3.4 Preliminary Analysis

Our objective is that a data collector can recover all the original k-unit data after it queries arbitrary b robust sensors. By querying arbitrary b robust sensors, the $k \times k$ coefficient matrix M can be obtained from the constructed system of linear equations. Obviously, whether there is a solution in the linear system and the original k unit data can be recovered depends on whether coefficient matrix M is nonsingular, which means the determinant of matrix M is not 0. Before presenting the necessary and sufficient condition we have derived to guarantee M is nonsingular, we first define two graphs.

Backup SubGraph. Given a Backup Graph $G^b = ((V_g + V_r), E^b)$, a corresponding Backup SubGraph $G^b_{sub} = ((V_g + V'_r), E^b_{sub})$ is a graph in which V'_r is an arbitrary subset of V_r containing b vertices, and E^b_{sub} is the corresponding subset of E^b connecting V'_r and V_g. Given a data backup scheme expressed by a Backup Graph, querying b arbitrary robust sensors can be expressed by the correspondingly generated Backup SubGraph.

Storage SubGraph. The Storage SubGraph $G^s_{sub} = ((V_g + V^+_r), E^s_{sub})$ is a bipartite graph constructed from a Backup SubGraph $G^b_{sub} = ((V_g + V'_r), E^b_{sub})$ by the following two operations: (1) for each vertex in V'_r, duplicate it and its connected edges s times to construct V^+_r and E^s_{sub}; (2) identify the smallest degree vertex in V'_r and remove $b \times s - k$ vertices in V^+_r which are duplicated from this vertex and their corresponding edges.

Note that after the first operation in Storage SubGraph construction, V^+_r has $b \times s$ vertices, which can be greater than or equal to k. If $bs > k$, $bs - k$ vertices need to be removed from V^+_r to construct a bipartite graph. Since $s > bs - k$, we can choose one vertex in V'_r and remove $bs - k$ vertices which are duplicated from this vertex. To maintain most of graph information, we choose the vertex with smallest degree in V'_r and remove $bs - k$ vertices duplicated from this vertex. Among all vertices in V^+_r, these $bs - k$ vertices must have smallest number of edges as well and removing them results in the least information loss. After the second operation, both V^+_r and V_g have k vertices in the graph. So in $G^s_{sub} = ((V_g + V^+_r), E^s_{sub})$, $|V^+_r| = |V_g| = k$.

Considering a Backup SubGraph representing a data collector's querying b arbitrary robust sensors, a $k \times k$ coefficient matrix M can be obtained from the corresponding Storage SubGraph. We discover that to check the singularity of M is equivalent to check whether there exists a perfect matching in the Storage SubGraph, motivated by Edmond's

Theorem [18], which states a connection between the determinant of a matrix and graph matchings in a bipartite graph. We express the discovery in Lemma 3.1.

Lemma 3.1. *The $k \times k$ coefficient matrix M is nonsingular with a high probability if and only if there exists a perfect matching in G^s_{sub}.*

Lemma 3.1 can be easily proved based on Edmond's Theorem. A key thing in Lemma 3.1 is perfect matching. In a bipartite graph, a perfect matching is a set of edges such that no two edges share a common vertex and no vertex is isolated. Based on Lemma 3.1, to develop a data backup scheme satisfying our objectives reduces to select edges from Deployment Graph G such that *there always exists a perfect matching in any arbitrarily generated G^s_{sub}*. Based on the theoretical foundation, we derive Guideline 1 and Guideline 2.

4 Guideline 1 and Associate Algorithm

4.1 Guideline 1

Guideline 1 specifies the minimum number of robust sensors that a regular sensor should randomly choose to back up its data such that the required level of fault tolerance can be provided without imposing any other conditions.

Guideline 1. It is sufficient to guarantee that a data collector can decode all the data with a high probability by querying any arbitrary b robust sensors, if every regular sensor randomly chooses at least $\lceil 5\frac{ns}{k}\ln(k)\rceil$ robust sensors to back up its data.

Guideline 1 is derived from the following two theorems: Theorem 4.1 and Theorem 4.2. Theorem 4.1 indicates that $\Omega(\ln(k))$ is the minimum magnitude of the number of robust sensors at which a regular sensor should back up its data to achieve the required level of fault tolerance. We use $c_1\ln(k)$ to denote the lower bound. Theorem 4.2 presents the value of c_1.

Theorem 4.1. *If every vertex in V_g (a regular sensor) selects vertices in V_r (robust sensors) independently and randomly, it must select $\Omega(\ln(k))$ robust sensors to ensure $det(M) \neq 0$ with a high probability.*

Proof of Theorem 4.1. To ensure $det(M) \neq 0$ with a high probability, every vertex in V_r^+ is at least covered by one vertex in V_g in G^s_{sub}. Otherwise, if there is even one vertex in V_r^+ with no edge, there is 1 column of zeros in the constructed coefficient matrix M, resulting in a singular M. In other words, all vertices in V_g can be viewed as a big vertex, and the big vertex needs to cover every vertex in V_r^+. Then it becomes to a classic problem—coupon collector's problem.[3] The big vertex acts as the collector, which collects k different vertices randomly. According to [18], the big vertex needs to at least randomly connect vertex in V_r^+ for $\beta k\ln(k)$ times to cover all k vertices with a high probability under some β. (More details are presented in the proof of Theorem 4.2.) Then for each vertex in V_g, the number of random connections is $\beta k\ln(k)/k = \beta\ln(k)$, which is at the magnitude of $\ln(k)$. Since G^s_{sub} is linearly transformed from G^b_{sub} and G^b_{sub} is a subset of G^b, the number of random connections for every regular sensor is at the magnitude of $\ln(k)$ in G^b. Therefore, each regular sensor node must connect $\Omega(\ln(k))$ robust sensors when the connections are made independently and uniformly. □

3. In the coupon collector's problem, there are n types of coupons and at each trial a coupon is chosen randomly. Each randomly selected coupon can be one of the n types at equal probability. The random selections are mutually independent. Let m be the number of trials. The goal is to study the relationship between m and the probability of collecting at least one coupon of each of the n types.

Theorem 4.2. *When $c_1 \geq 5\frac{ns}{k}$, there exists a perfect matching in any arbitrarily generated G^s_{sub} with a high probability.*

Before theoretically deriving the value of c_1, we first present a lemma about perfect matchings in bipartite graphs[19].

Lemma 4.1. *Let G_{bi} be a bipartite graph with vertex classes V_g and V_r^+, where $|V_g| = |V_r^+| = k$. Suppose G_{bi} has no isolated vertices and it does not have a perfect matching. Then there is a set $A \subset V_g$ or V_r^+ such that:*

i) $\Gamma(A) = \{v_j : (v_i, v_j) \in E(G_{bi})$ *for some* $v_i \in A\}$ *has* $|A| - 1$ *elements,*

ii) the subgraph spanned by $A \cup \Gamma(A)$ *is connected and*

iii) $2 \leqslant |A| \leqslant (k+1)/2$.

Lemma 4.1 is used to analyze the scenario that a perfect matching exists in G^s_{sub} with no isolated vertices in V_g and V_r^+.

Proof of Theorem 4.2. Based on Lemma 4.1, G^s_{sub} has no perfect matching only in the following two cases:

- Case I: there exists a set A satisfying Lemma 4.1.
- Case II: G^s_{sub} has one or more isolated vertices, denoted as I.

Thus, the probability that G^s_{sub} has no perfect matching is $P(\exists I \bigcup \exists A) \leq P(\exists I) + P(\exists A)$.

We first analyze $P(\exists A)$ in case I. From Lemma 4.1, the size of A varies from 2 to $(k+1)/2$. Therefore,

$$P(\exists A) = P\left(\bigcup_{a=2}^{(k+1)/2}(\exists A, |A| = a)\right) \leq \sum_{a=2}^{(k+1)/2} P(\exists A, |A| = a). \tag{1}$$

Furthermore, set A can be the subset of V_g (case I.a) or the subset of V_r^+ (case I.b). In case I.a, $A \subset V_g$. In case I.b, $A \subset V_r^+$. Then we have:

$$\begin{aligned} P(\exists A) \leq \sum_{a=2}^{(k+1)/2} &(P(\exists A \subset V_g, |A| = a) \\ &+ P(\exists A \subset V_r^+, |A| = a)). \end{aligned} \tag{2}$$

In the following, we will calculate $P(\exists A \subset V_g, |A| = a)$ and $P(\exists A \subset V_r^+, |A| = a)$, respectively.

Case I.a. $A \subset V_g$. When set A exists in G^s_{sub}, there must be a set A' in G^b_{sub}, which contains the same elements as set A. One example is shown in Fig. 5, in which $k = 6$, $s = 2$ and $b = 3$. Then set $A = \{1, 2, 3\}$ in G^s_{sub} is illustrated in Fig. 5a, and set $A' = \{1, 2, 3\}$ in G^b_{sub} is illustrated in Fig. 5b. Since G^s_{sub} is equivalently converted from G^b_{sub}, then we have $P(\exists A'_g, |A'| = a) = P(\exists A \subset V_g, |A| = a)$.

Now, let us consider a general case. Suppose there are a nodes in a set $A_1 \subset V_g$ and $a-1$ nodes in a set $A_2 \subset V_r^+$ in G^s_{sub}, where $\Gamma(A_1) = A_2$. Then there must be $\ulcorner\frac{a-1}{s}\urcorner$

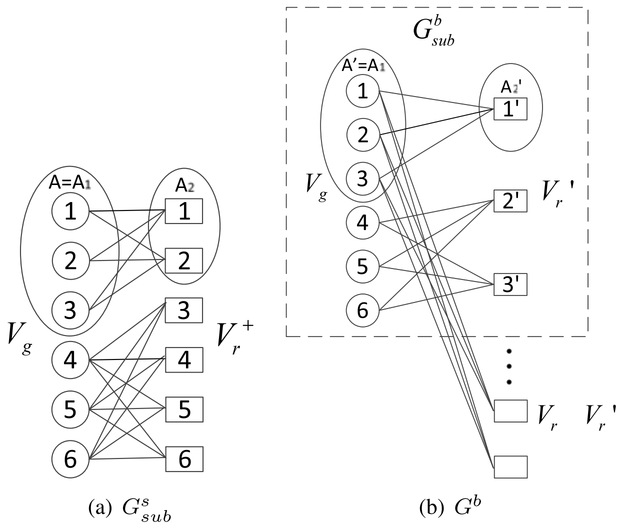


Fig. 5. One example of Case I.a.

nodes $A'_2 \subset V'_r$ in G^b_{sub}. Such value can be proved by contradiction.

Proof by contradiction. Let Q denote the quotient and R denote the remainder of $\frac{a-1}{s}$, respectively. Then we have $\ulcorner\frac{a-1}{s}\urcorner = Q$ if $R = 0$ and $\ulcorner\frac{a-1}{s}\urcorner = Q + 1$ if $R > 0$. Assume $|A'_2| = D < \ulcorner\frac{a-1}{s}\urcorner$ in G^b_{sub}. Then after constructing a Storage Subgraph G^s_{sub}, there must be $D \times s$ nodes in A_1 in G^s_{sub}. If $R = 0$, $|A'_2| = D < Q$ and then $|A_1| = D \times s < Q \times s = a - 1$. If $R > 0$, $|A'_2| = D < Q + 1$ and then $D \leq Q$. Then we have $|A_1| = D \times s \leq Q \times s$. Since $R > 0$, $Q \times s < Q \times s + R = a - 1$ and thus $|A_1| < a - 1$. Therefore both cases conflict the assumption that $|A_1| = a - 1$. Similarly, we can prove that $|A'_2|$ can not be greater than $\ulcorner\frac{a-1}{s}\urcorner$. Thus, the number of nodes in set A'_2 in G^b_{sub} is $\ulcorner\frac{a-1}{s}\urcorner$.

The probability that set $A = A_1$ with $\Gamma(A) = A_2$ satisfying Lemma 4.1 in G^s_{sub} is equal to the probability that all the edges starting from A_1 connect A'_2 in G^b_{sub}. Note that every node in V_g picks $c_1\ln(k)$ neighbors from the set V_r in G according to Theorem 4.1. We calculate the probability $P(\exists A' \subset V_g, |A'| = a)$ by allowing $c_1 a\ln(k)$ edges starting from A_1 to land in $A'_2 \bigcup (V_r - V'_r)$ shown in Fig. 5b. Since $|A'_2| = \ulcorner\frac{a-1}{s}\urcorner$ and $|(V_r - V'_r)| = n - \ulcorner\frac{k}{s}\urcorner$, we have $|A'_2 \bigcup (V_r - V'_r)| = n - \ulcorner\frac{k}{s}\urcorner + \ulcorner\frac{a-1}{s}\urcorner$. The the probability that a edge starting from A_1 to land in $A'_2 \bigcup (V_r - V'_r)$ is $\frac{n - \ulcorner\frac{k}{s}\urcorner + \ulcorner\frac{a-1}{s}\urcorner}{n}$. There are $\binom{k}{a}$ choices for A_1 and $\binom{\ulcorner\frac{k}{s}\urcorner}{\ulcorner\frac{a-1}{s}\urcorner}$ choices for A'_2. Then we have:

$$\begin{aligned} &P(\exists A \subset V_g) \\ &= P(\exists A' \subset V_g) \\ &\leq \sum_{a=2}^{(k+1)/2} \binom{k}{a} \binom{\ulcorner\frac{k}{s}\urcorner}{\ulcorner\frac{a-1}{s}\urcorner} \left(1 - \frac{\ulcorner\frac{k}{s}\urcorner - \ulcorner\frac{a-1}{s}\urcorner}{n}\right)^{c_1 a\ln(k)} \\ &\approx \sum_{a=2}^{(k+1)/2} \binom{k}{a} \binom{\frac{k}{s}}{\frac{a-1}{s}} \left(1 - \frac{k-a+1}{ns}\right)^{c_1 a\ln(k)}. \end{aligned} \tag{3}$$

We can always bound the above summation by the maximum value of $\binom{k}{a}\binom{\frac{k}{s}}{\frac{a-1}{s}}(1 - \frac{k-a+1}{ns})^{c_1 a\ln(k)}$ times k. Therefore, in order to let the probability to be close to 0, it needs to show that:

$$kP(\exists A \subset V_g, |A| = a) = o(1), \quad \forall a \in [2, (k+1)/2] \tag{4}$$

as $k \to \infty$.

From Stirling's approximation, we obtain the bound $\binom{k}{a} \leq (\frac{ek}{a})^a$. Let $\mathcal{Y} = 1 - \frac{k-a+1}{ns}$, we have

$$\begin{aligned} &P(\exists A \subset V_g) \leq k\left(\frac{ek}{a}\right)^a \left(\frac{ek}{a-1}\right)^{\frac{a-1}{s}} \mathcal{Y}^{c_1 a\ln(k)} = e^{\mathcal{F}} \\ &\mathcal{F} = \ln(k) + a\ln\left(\frac{ek}{a}\right) + \frac{a-1}{s}\ln\left(\frac{ek}{a-1}\right) + c_1 a\ln(k)\ln(\mathcal{Y}). \end{aligned} \tag{5}$$

When $e^{\mathcal{F}} = o(1)$, it is sufficient to have $\mathcal{F} < 0$ and thus the coefficient of $\ln(k)$ be negative. Since k tends to $+\infty$, we need to have:

$$a + \frac{a-1}{s} + ac_1\ln(\mathcal{Y}) + 1 < 0, \tag{6}$$

which gives us a bound for c_1:

$$c_1 > -\frac{1 + a + \frac{a-1}{s}}{a\ln(\mathcal{Y})}. \tag{7}$$

Notice that $\mathcal{Y} < 1$, since $ns \geq k$ and $k > a$. It is possible to satisfy this inequality for a positive c_1. This bound should be true for every $a \in [2, (k+1)/2]$. So we have:

$$\frac{1 + a + \frac{a-1}{s}}{a} \leq \frac{5}{2} \tag{8}$$

and

$$\mathcal{Y} \leq 1 - \frac{k+1}{2ns}. \tag{9}$$

Then according to Taylor Approximation, we have

$$-\frac{1}{\ln(\mathcal{Y})} < \frac{2ns}{k+1} \tag{10}$$

and

$$-\frac{1 + a + \frac{a-1}{s}}{a\ln(\mathcal{Y})} < \frac{2ns}{k+1} \cdot \frac{5}{2} < \frac{5ns}{k}. \tag{11}$$

Therefore, a sufficient condition for $\exists A \subset V_g$ is

$$c_1 \geq \frac{5ns}{k} \tag{12}$$

Case I.b. $A \subset V_r^+$. With similar analysis, we can obtain a bound if $A \subset V_r^+$. When set A exists in G^s_{sub}, there must be a set A' in G^b_{sub}. Set A' contains $\ulcorner\frac{|A|}{s}\urcorner$ elements, which can also be proved by the similar contradiction presented in case I.a. As the example shown in Fig. 6, in which $k = 6$, $s = 2$ and $b = 3$, set $A = \{1, 2, 3\}$ in G^s_{sub} is illustrated in Fig. 6a, and set $A' = \{1', 2'\}$ in G^b_{sub} is illustrated in Fig. 6b. Then we have $P(\exists A' \subset V'_r, |A'| = a) = P(\exists A \subset V_r^+, |A| = a)$.

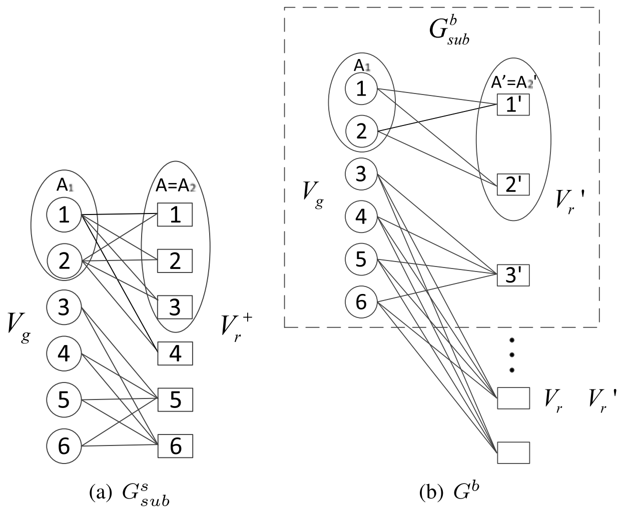


Fig. 6. One example of Case I.b.

We assume a set $A_2 \subset V_r^+$ with a nodes and a set $A_1 \subset V_g$ with $a-1$ nodes in G^s_{sub}. Then we have $\ulcorner \frac{a}{s} \urcorner$ nodes in set $A'_2 \subset V'_r$ in G^b_{sub}. To satisfy LEMMA 4.1 with $A = A_2$ in G^s_{sub}, we require that all edges that link to A_2 land in set A_1. To have $\Gamma(A_2) = A_1$ in G^s_{sub}, all the edges starting from $V_g - A_1$ must land outside A'_2 of G^b, which is $V_r - A'_2$ as shown in Fig. 6b. There are $c_1 k\ln(k) - c_1(a-1)\ln(k)$ such edges and every edge has a probability $1 - \frac{\ulcorner \frac{a}{s} \urcorner}{n}$ to land outside A'_2 of G^b. Thus we have:

$$P(\exists A \subset V_r^+) = P(\exists A' \subset V'_r) \\ \leq \sum_{a=2}^{(k+1)/2} \binom{k}{a-1} \binom{\ulcorner \frac{k}{s} \urcorner}{\ulcorner \frac{a}{s} \urcorner} (1 - \frac{\ulcorner \frac{a}{s} \urcorner}{n})^{c_1 k\ln(k) - c_1(a-1)\ln(k)}. \tag{13}$$

After similar calculations, it can be seen that when $c_1 > \frac{ns+n}{k-a+1}$, $P(\exists A \subset V_r^+) = o(1)$ as $k \to \infty$. Notice that $\frac{ns+n}{k-a+1}$ is an increasing function as a increases. So the maximum value of $\binom{k}{a-1}\binom{\ulcorner \frac{k}{s} \urcorner}{\ulcorner \frac{a}{s} \urcorner}(1 - \frac{\ulcorner \frac{a}{s} \urcorner}{n})^{c_1 k\ln(k) - c_1(a-1)\ln(k)}$ is obtained at $a = 2$ or $a = \frac{k+1}{2}$, whichever is larger. After we examine the extreme cases, it can be shown that $c_1 \geq 5\frac{ns}{k}$ is required to satisfy both extreme cases.

Case II. There exists isolated vertices in G^s_{sub}. A vertex in G^s_{sub} is isolated only when it has no neighbors. It is obvious that if there are isolated nodes in G^s_{sub}, they can only be in V_r^+. It also means that isolated nodes can only be in V_r of G. In other words, we need to show that each vertex in V_r is at least covered by one vertex in V_g with a high probability. The problem also becomes a coupon collector's problem as discussed in Theorem 4.1. All regular sensors acting as a collector to randomly collect n different robust sensors.

Let C denote the number of total edges required to cover all n vertices in V_r. According to the analysis results of the coupon collector's problem in [18], we have:

$$P[C > \beta n\ln(n)] \leq n^{-(\beta-1)}. \tag{14}$$

To satisfy such condition, we need to have $\beta \geq 2$ such that the probability tends to 0. From above calculation for c_1, it suffices to show that

$$\frac{kc_1\ln(k)}{\beta n\ln(n)} > 1 \tag{15}$$

Thus,

$$\beta < \frac{5s\ln(k)}{\ln(n)}. \tag{16}$$

For a reasonable deployment, k is always larger than n, which means the number of robust sensors is less than the regular sensors, and s is always larger than 1. Thus we can always find some $\beta \geq 2$ with $c_1 \geq 5\frac{ns}{k}$ to satisfy the condition that each vertex in V_r is covered by at least one vertex in V_g with a high probability.

Conclusion. Since $c_1 \geq 5\frac{ns}{k}$ is sufficient for a perfect matching existing in any arbitrary G^s_{sub} with a high probability, we prove the theorem. $\square$

As $k \to \infty$, the probability of the existence of a perfect matching approaches 1. Since generally a sensor network is composed of a large number of regular sensors, the probability is almost 100 percent.

4.2 A Robust Randomized Algorithm

Following Guideline 1, a robust randomized algorithm is developed: every regular sensor randomly chooses $\lceil 5\frac{ns}{k}\ln(k) \rceil$ robust sensors to back up its data. The algorithm is simple and robust.

In the algorithm, the random selection of robust sensors through the network is a must. This means when we choose $\lceil 5\frac{ns}{k}\ln(k) \rceil$ *closest* robust sensors to back up data, the randomness is destroyed and the conditions of previous theorems are violated. (In the future work, we will study whether we can sacrifice certain randomness to incorporate selection rules based on backup cost.)

After the sensors' deployment, energy-efficient routes between any two sensors can be established by existing routing protocols [13], [14], [15], which construct routes based on sensors' locations and the same energy model presented in Section 3.2. When the regular sensors start data backup process, they select robust sensors for $\lceil 5\frac{ns}{k}\ln(k) \rceil$ times randomly and independently. All robust sensors' IDs are preloaded in regular sensors before network deployment. The randomness of the selection can be easily achieved by randomly selecting robust sensors' IDs. Then every regular sensor decides the robust sensors that it backs up its data and sends its data to such robust sensors via established routes. When a robust sensor is selected more than once by the same regular sensor, the algorithm still works effectively according to previous analysis.

5 Guideline 2 and Associate Algorithm

5.1 Guideline 2

Guideline 2 specifies two conditions that must be satisfied simultaneously such that the required level of fault tolerance can be achieved.

Guideline 2. It is sufficient to guarantee that a data collector can decode all the data with a high probability by querying any arbitrary b robust sensors, if the following two conditions are satisfied simultaneously:

- Every regular sensor backs up its data at at least $c_2 = n - b + 1$ different robust sensors.
- Every robust sensor receives data from at least s different regular sensors.

Guideline 2 is derived from following theorem.

Theorem 5.1. *Given a Deployment Graph $G = ((V_g, V_r), E)$, in which the degree of each vertex in V_g is no less than $c_2 = n - b + 1$ and the degree of each vertex in V_r is no less than s, it is sufficient to guarantee the existence of a perfect matching in any G^s_{sub} generated from G.*

Proof of Theorem 5.1. We prove the theorem by contradiction. Assume a vertex u_i in V_g is connected to c' vertices in V_r, where $c' \le n - b < n - b + 1$. There are always at least $b(= n - c')$ vertices in V_r, which are not connected to u_i in V_g. When G^s_{sub} is constructed from these b vertices, there is always a row of 0 in matrix M. Thus M is singular. According to Lemma 3.1, there doesn't exist a perfect matching.

Assume a vertex v_j in V_r is connected to c'' vertices in V_g, where $c'' \le s - 1 < s$. When G^s_{sub} is constructed from such vertex v_j, s vertices duplicated from v_j in V^+_r must connect $c'' < s$ vertices in V_g. Then we can always find a subset A of the set containing such s vertices satisfying Lemma 4.1. Thus, there doesn't exist a perfect matching.

Therefore, we prove the theorem. □

5.2 A Centralized Algorithm

Based on Guideline 2, we develop a centralized algorithm to determine which regular sensor backs up its data in which robust sensor such that the energy consumption in communication is minimized. In the algorithm, a central server first performs a one-time network discovery to obtain the location of each sensor and then determines routes and calculates energy costs between regular sensors and robust sensors based on the location information. Then it calculates the backup schedule at the network initialization phase.

Consider that any arbitrary selection of robust sensors satisfying the conditions of Guideline 2 can guarantee the successful decoding of all the data. Thus we aim to select robust sensors which can minimize energy consumption in data backup. Note that in Section 2, we have proved the NP-Completeness of the Weighted Backup Problem and we cannot have an optimal solution. Based on the random linear network coding technique and Guideline 2, a centralized algorithm can be designed to tackle the problem. Its general idea is to let a robust sensor choose s regular sensors with least energy cost to receive data from and let each regular sensor choose $c_2(= n - b + 1)$ robust sensors with least energy cost to send data to. In the following, the process of the centralized algorithm is presented in details.

Same as the randomized algorithm, after the sensors' deployment, energy-efficient routes are constructed in the network by existing routing protocols. The central server establishes a cost matrix to store the energy cost of the route between every regular sensor and every robust sensor. Then the algorithm runs the following three steps to choose robust sensors for each regular sensor to back up its data.

- Step 1: each robust sensor selects s regular sensors with the least energy cost from the cost matrix to receive backup data.
- Step 2: each regular sensor selects c_2 robust sensors with the least energy cost from the cost matrix to back up its data.
- Step 3: remove the redundant edges, the removal of which will not violate the two conditions in the Guideline 2.

After the first two steps, both two conditions in the Guideline 2 can definitely be satisfied. Note that, the minimum units of data that have to be sent by all regular sensors, which is $c_2 \times k$, is generally larger than the minimum units of data that have to be received by robust sensors, which is $n \times s$. Thus in the Backup Graph, only $c_2 \times k$ edges are required ideally. However, since the backup selections between regular sensors and robust sensors may not be overlapped, the edges selected after the first two steps can be at most $c_2 \times k + n \times s$. This means there are redundant selections. The final step is to remove the redundant edges in the graph. The algorithm examines every edge from highest cost to lowest cost. If the removal of a edge does not violate the two conditions in the Guideline 2, then this edge is a redundant backup edge and is removed from the Backup Graph. The algorithm is formally presented in Algorithm 1.

Algorithm 1. A Centralized Algorithm

Output:

B: a $k \times n$ backup matrix. $b_{ij} = 1$ indicates regular sensor u_i backs up data in robust sensor v_j and 0 indicates otherwise. $b_{i\cdot}$ and $b_{\cdot j}$ represent the corresponding row/vector of B.

W: a $k \times n$ energy cost matrix. w_{ij} indicates the energy cost from regular sensor u_i to robust sensor v_j. $w_{i\cdot}$ and $w_{\cdot j}$ represent the corresponding row/vector of W.

```
1:  Calculate routes between any two sensors.
2:  for all $v_j \in V_r$ do
3:      for all $u_i \in V_g$ do
4:          Put route cost from $u_i$ to $v_j$ in $w_{ij}$.
5:      end for
6:  end for
7:  for all $v_j \in V_r$ do
8:      Choose $s$ regular sensors with least energy cost in $W$ for
        backup.
9:      Update $b_{\cdot j}$ and $w_{\cdot j}$.
10: end for
11: for all $u_i \in V_g$ do
12:     Choose $c_2$ robust sensors with least energy cost in $W$.
13:     Update $b_{i\cdot}$ and $w_{i\cdot}$.
14: end for
15: Find a $w_{ij}$ with the largest value in $W$.
16: while $w_{ij}$ is nonzero do
17:     Remove $b_{ij}$ from $B$ and $w_{ij}$ from $W$, the removal of which
        will not make regular sensor $u_i$ back up data at less than
        $c_2$ robust sensors, and robust sensor $v_j$ receive less than $s$
        units of data.
18:     Go to find $w_{ij}$ with the next highest value in $W$.
19: end while
```

6 Discussion About Coefficient Matrix

The coefficient matrix is dynamically determined after the deployment. The determination process is as follows: Before the deployment, each robust sensor is preloaded with a large matrix which works for all the regular sensors. After the deployment and the backup relationship is determined by our algorithms, only a subset of this large matrix is needed and thus the unused columns/rows are deleted to save space, resulting in the coefficient matrix that we need.

7 Performance Evaluation

7.1 Simulation Methodology and Setting

In this section, we evaluate the randomized algorithm based on Guideline 1 and the centralized algorithm based on Guideline 2. Our objective in conducting the evaluation study is two-fold: (1) Evaluating the efficiency of our algorithms in minimizing energy consumption while providing required fault tolerance; (2) Testing the performance of our algorithms under different system parameters.

To evaluate our algorithms, four metrics are employed: (1) total energy consumption per backup, (2) energy consumption per node, (3) number of storage units needed for data backup per robust node and (4) whether the required level fault tolerance is achieved.

Since the centralized algorithm requires cost information about the communication between a regular sensor and a robust sensor, we evaluate the centralized algorithm under the situation when the cost can be obtained from other applications running in the sensor network, and the situation when the cost is not available and needs to be discovered by a flooding.

We compare our algorithms with DISC protocol [3], which has the same objective as ours. In DISC protocol, the network is composed of data nodes and storage nodes, and the network is divided into clusters. In each cluster, one of the storage nodes is randomly selected as the cluster head for data backup. In one backup, all data nodes back up their data in their cluster, and every cluster head then backs up its data in a neighbor cluster. In our implementation, every robust sensor is viewed as a storage node and every regular sensor is viewed as a data node in DISC.

To have a fair comparison, we let the cluster heads in DISC back up data at neighboring clusters multiple times following their selection algorithm to achieve the comparable fault tolerance as our designed algorithms. If we back up data only once strictly following DISC, one cluster's data are stored in only two robust sensors in different clusters. Then the data recovery will fail with a high probability, since a large amount of robust sensors don't have any data. According to THEOREM 5.1, at least $n-b+1$ different robust sensors need to store data. Then at least $(n-b+1)/2$ different robust sensors should be selected in both current cluster and neighbor cluster to achieve required fault tolerance. Since the robust sensors for backup are randomly selected, it turns to be a version of the coupon collector's problem. According to [18], the expected value of backups that need to be performed to cover m different robust sensors is $m\ln(m)$. Thus, $m\ln(m)$ $(m=(n-b+1)/2)$ backups are performed to achieve the required fault tolerance, which is recovering all data by accessing arbitrary b storage sensors in DISC. We divide the network into four clusters and ensure $n/4 \geq m$ in the evaluation such that enough robust sensors for backup are provided in each cluster.

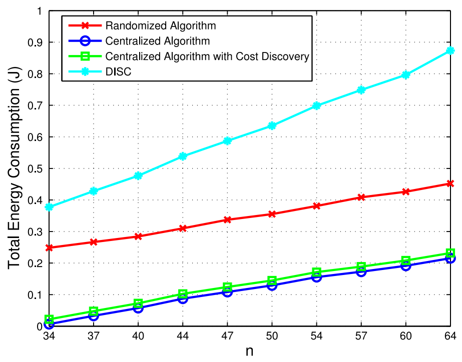


Fig. 7. Total energy consumption under different n (s = 6).

We evaluate the performance through simulation. In the simulation, 200 regular sensors are randomly deployed in a $40\,\mathrm{m}\times 40\,\mathrm{m}$ area. Each sensor's communication range is 10 m. The energy model is the free space mode presented in Section 3.2. According to [12], the communication energy parameters are set as: $E_{elec}=50nJ/bit$, $\epsilon_{fs}=10pJ/bit/m^2$ and $\alpha=2$. The central server broadcasts its location information, which are 64 bits including longitude and latitude in the data type of float. Sensing data include a timestamp and a measurement, which are 64 and 32 bits respectively. We adopt the routing algorithm proposed in [15] to establish routes. The communication overhead in route construction is not considered. We set s to be 6 unless we evaluate the impact of robust sensor's storage s. We also set $ns/k=1.5$. Correspondingly, the number of robust sensors is 50, which is $k/4$, unless we evaluate the impact of number of robust sensors. In each simulation, the network topology is different by randomly redeploying all sensor nodes in the network. All simulation results are the average of 100 time simulations.

7.2 Evaluation of Total Energy Consumption

We evaluate total energy consumption under different s and n. The number of robust sensors n ranges from 34 to 64, which is from $1/6$ to $1/3$ of the number of regular sensors k. Fig. 7 shows the total energy consumption of the three algorithms. The green line is the energy consumption of the centralized algorithm with cost discovery overhead. From the figure, we can see that our schemes outperform DISC. The cost discovery does not introduce a big overhead. When n is 34, the total energy consumption under randomized algorithm is 0.2485 J. It is 34.2 percent less than that under DISC, which is 0.3776 J. The total total energy consumption under centralized algorithm is 0.0067 J, which is about 98.2 percent less than that under DISC.

As n increases, total energy consumption of all the algorithms increases. The performance under DISC increases much faster than under our algorithms. The reason is that the number of backup times increases as n increases. Among the two of our algorithms, the performance under the centralized algorithm is better than that under the randomized algorithm. This shows that the centralized algorithm performs better with a smaller n.

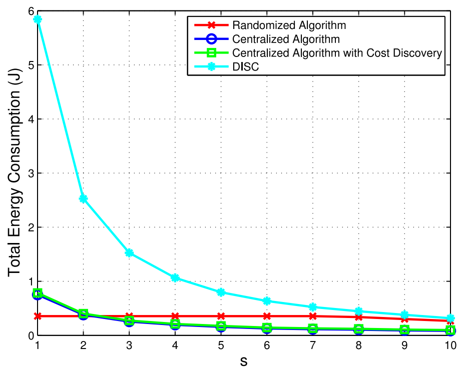


Fig. 8. Total energy consumption under different s (ns/k = 1.5).

We vary the number of storage units in a robust sensor from 1 to 10. The results are shown in Fig. 8. From the figure, we can see that our schemes outperform DISC under different s. When s is 1, the total energy consumption under randomized algorithm is 0.3547 J. It is 93.9 percent less than that under DISC, which is 5.8425 J. The total energy consumption under centralized algorithm is 0.7512 J, which is 87.1 percent less than that under DISC. As s increases, the randomized algorithm keeps an almost constant energy consumption, but the centralized algorithm consumes less. The reason is that under given ns/k and k, c_1 doesn't change while c_2 decreases with less number of robust sensors, which reduces the number of backup messages. This shows that the randomized algorithm performs better with a smaller s.

7.3 Energy Consumption per Node

We evaluate the energy consumption per node under different n and s. n ranges from 34 to 64. Fig. 9 shows the energy consumption per node under three algorithms. From the figure, we can see that our algorithms outperform DISC. For example, when n is 34, the energy consumption per node under randomized algorithm is 1.06×10^{-3} J, which is 35.0 percent less than that under DISC, which is 1.63×10^{-3} J. The energy consumption per node under centralized algorithm is 2.85×10^{-5} J, which is 98.3 percent less than that under DISC. The reason is that the data backup between cluster heads consumes a lot of energy in DISC. It can also be observed that as n increases, energy consumption per node increases under all three algorithms. The reason is that as n increases, the total number of messages transmitted from every regular sensor increases faster than the number of robust sensors, and thus the energy consumption per node increases.

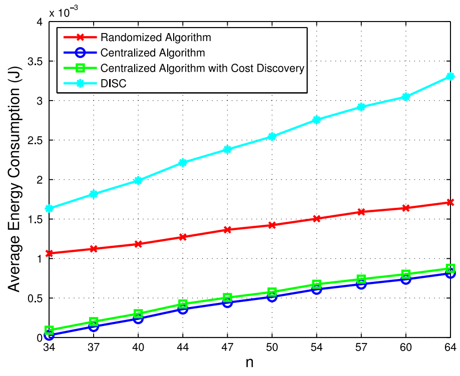


Fig. 9. Energy consumption per node under different n (s = 6).

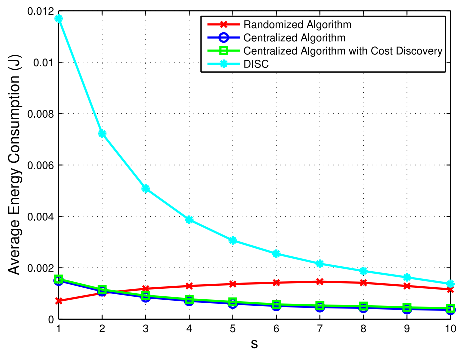


Fig. 10. Energy consumption per node under different s (ns/k = 1.5).

Fig. 10 shows the energy consumption per node under different s. From the results, our algorithms have better performances than DISC. For example, when s is 1, the energy consumption per node under randomized algorithm and centralized algorithm is 7.09×10^{-4} J and 1.50×10^{-3} J, which are 93.9 and 87.2 percent less than that under DISC, which is 1.17×10^{-2} J. From the results, we can also see that energy consumption per node under both centralized algorithm and DISC decreases as s increases, and the energy consumption per node under randomized algorithm increases as s increases. The reason is that as s increases, the number of robust sensors decreases with given ns/k and s. In centralized algorithm, the total number of messages transmitted from every regular sensor decreases faster than the total number of sensors. Thus the energy consumption per node decreases under centralized algorithm. In randomized algorithm, the total number of messages transmitted from every regular sensor is a constant value under given ns/k. Thus the energy consumption per node increases under randomized algorithm as total number of sensors decreases.

7.4 Storage Size Required for One Backup in a Robust Node

We evaluate the size of storage units required for one backup in a robust sensor on average under different s and n. A small variance on such metric indicates that the storage requirement of the algorithm is stable.

n ranges from 34 to 64. Fig. 11 shows the size of storage units used in a robust sensor under three algorithms. From the result, we can see that the size of storage units taken on a robust sensor is much lower under our algorithms than that

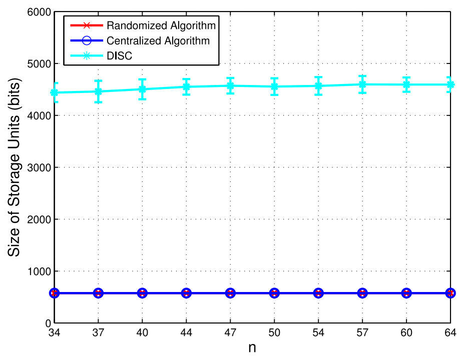


Fig. 11. Storage units under different n (s = 6).

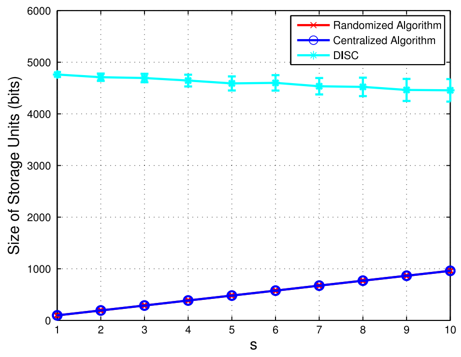


Fig. 12. Storage units under different s (ns/k = 2).

under DISC. For example, when $n = 34$, the size of storage units required on a robust sensor under our algorithms is 576 bits on average, which is 87.0 percent less than that under DISC, which is 4439.1 bits on average. Moreover, the storage units required in a robust sensor under our algorithms are constant. The storage units required are not stable under DISC. The reason is that by using network coding, our algorithms only need to store s units data in each robust sensor. In DISC, since each cluster head may need to store original data from other cluster heads, the difference of number of storage units between robust sensors is large.

Fig. 12 shows the the size of storage units required in a robust sensor under different s. From the results, we can see that the value increases under our algorithms as s increases. The value decreases and becomes more unstable under DISC as s increases. Our algorithms only need s units storage in a robust sensor. Then as s increases, the number of storage units on every robust sensor increases. Moreover, the requirement of storage units under our algorithms is still much lower than that under DISC. It is obvious that DISC takes lots of storage in robust sensors to meet the required level of backup fault tolerance.

7.5 Evaluation of Fault Tolerance

In all the above simulation settings, both our centralized and randomized algorithms can achieve the required fault tolerance, which is, querying any b robust nodes can recover all the data generated by the sensor networks. For DISC, if a cluster head backs up data multiple times, the required fault tolerance can also be achieved; however, the energy consumption on data backup is high, which has been shown in previous sections. If we strictly follow DISC protocol, which is a cluster head backs up data on one neighbor storage node, only 6.06 percent times, the required fault tolerance is achieved.

8 Related Work

Networked data storage and backup have been intensively studied in the past. Depending on methodology, existing works can be classified into two categories: schemes using network coding and schemes not. Mechanisms based on network coding are studied in [4], [5], [8], [20], [21], [22], [23], [24], [25], [26], [27], [28]. Albano and Chessa provide an abstract model of in-network storage by using erasure codes in [4]. A decentralized erasure code for data storage is proposed in [5], in which data sources are distributed in the network. Wang et al. in [8] present a partial network coding (PNC) scheme for collecting recent data in wireless sensor networks. Storing large files in a distributed manner is studied in [20]. Dimakis et al. investigate the problem of constructing fountain codes for distributed storage in sensor networks in [21]. Regenerating codes with existing available nodes to repair failure nodes in distributed storage systems is studied in [22]. Liu et al. in [23] use Slepian-Wolf Code to minimize the communication cost in a network with a single sink. Hu et al. propose a mutually cooperative recovery (MCR) mechanism for multiple node failures in distributed storage systems in [24]. A network coding scheme based on sociality of wireless sensor networks is proposed in [25]. Yang et al. propose a compressed network coding based distributed data storage scheme based on compressed sensing and network coding theories in [26]. To guarantee data integrity and availability, Zeng et al. in [27] propose a distributed fault/intrusion-tolerant data storage scheme based on network coding and homomorphic fingerprinting. In [28], Wu studies the problem of constructing network codes to achieve an optimal tradeoff between storage efficiency and network bandwidth.

Without using network coding, data storage using other techniques are presented in [2], [3], [29], [30], [31], [32], [33]. A Geographic Hash Table system for data-centric storage is proposed by Ratnasamy et al. in [29], which hashes keys into geographic coordinates, and stores a key-value pair at the sensor node geographically nearest the hash of its key. Tanushetty et al. present a concept of multiple hash locations for storing sensed data to provide efficient resiliency for data centric storage in [30]. A distributed data storage protocol for large-scale heterogeneous WSNs with mobile sinks is proposed in [31], which guarantees robustness in data collection by intelligently managing data replication among selected storage nodes in the network. Liao and Yang propose a power-saving data storage scheme for WSN based on grid architecture in [32]. A distributed data storage algorithm that generates redundant data based on Luby transform coding is proposed in [34]. Sheng et al. in [33] study the deployment problem of storage nodes to minimize the total energy cost in data storage system, and present an optimal algorithm based on dynamic programming. A line based data dissemination protocol in wireless sensor networks with mobile sinks is studied in [35]. DISC in [3] randomly chooses cluster heads in neighboring clusters to conduct data backup, and uses a Bloom filter based search engine to retrieve data and minimize energy consumption. Hashmi et al. in [2] propose to rotate cluster heads to do backup to balance network energy consumption.

None of the existing works focuses on designing a data backup scheme in heterogenous sensor networks to maximize the fault tolerance and minimize the communication cost, and cannot be directly applied to solve the problem in this paper.

9 Conclusion

In this paper, a weighted backup problem in heterogeneous wireless sensor networks is formulated and studied. We formally prove its NP-completeness. Leveraging random linear network coding, we derive two design guidelines, satisfying which, the required fault tolerance can be provided. Based on the two guidelines, we design two data backup schemes, which outperform existing solutions.

ACKNOWLEDGMENTS

This work is supported by the US National Science Foundation grants NSF-1128369. G. Wang is the corresponding author.

REFERENCES

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM Int. Workshop Wireless Sens. Netw. Appl.*, 2002, pp. 88–97.

[2] S. Hashmi, H. Mouftah, and N. D. Georganas, "Achieving reliability over cluster-based wireless sensor networks using backup cluster heads," in *Proc. IEEE Global Telecommun. Conf.*, 2007, pp. 1149–1153.

[3] C. Jardak, E. Osipov, and P. Mahonen, "Distributed information storage and collection for wsns," in *Proc. IEEE Int. Conf. Mobile Adhoc Sens. Syst.*, 2007, pp. 1–10.

[4] M. Albano and S. Chessa, "Distributed erasure coding in data centric storage for wireless sensor networks," in *Proc. IEEE Symp. Comput. Commun.*, 2009, pp. 22–27.

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

[6] D. West, *Introduction to Graph Theory*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2001.

[7] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[8] D. Wang, Q. Zhang, and J. Liu, "Partial network coding: Concept, performance, and application for continuous data collection in sensor networks," *ACM Trans. Sens. Netw.*, vol. 4, no. 3, pp. 14:1–14:22, May 2008.

[9] Z. Yang, L. Cai, Y. Liu, and J. Pan, "Environment-aware clock skew estimation and synchronization for wireless sensor networks," in *Proc. IEEE INFOCOM*, 2012, pp. 1017–1025.

[10] Z. Zhong, P. Chen, and T. He, "On-demand time synchronization with predictable accuracy," in *Proc. IEEE INFOCOM*, 2011, pp. 2480–2488.

[11] Y. Chen, Q. Wang, M. Chang, and A. Terzis, "Ultra-low power time synchronization using passive radio receivers," in *Proc. 10th Int. Conf. Inform. Process. Sens. Netw.*, 2011, pp. 235–245.

[12] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[13] A. E. Abdulla, H. Nishiyama, and N. Kato, "Extending the lifetime of wireless sensor networks: A hybrid routing algorithm," *Comput. Commun.*, vol. 35, no. 9, pp. 1056–1063, May 2012.

[14] A. Papadopoulos, A. Navarra, J. A. McCann, and C. M. Pinotti, "Vibe: An energy efficient routing protocol for dense and mobile sensor networks," *J. Netw. Comput. Appl.*, vol. 35, no. 4, pp. 1177–1190, Jul. 2012.

[15] D. Zhang, G. Li, K. Zheng, X. Ming, and Z.-H. Pan, "An energy-balanced routing method based on forward-aware factor for wireless sensor networks," *IEEE Trans. Ind. Inform.*, vol. 10, no. 1, pp. 766–773, Feb. 2014.

[16] C. Konstantopoulos, G. Pantziou, D. Gavalas, A. Mpitziopoulos, and B. Mamalis, "A rendezvous-based approach enabling energy-efficient sensory data collection with mobile sinks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 5, pp. 809–817, May 2012.

[17] R. Sugihara and R. Gupta, "Optimal speed control of mobile node for data collection in sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 1, pp. 127–139, Jan. 2010.

[18] R. Motwani and P. Raghavan, *Random Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[19] B. Bollóbas, *Random Graphs*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2001.

[20] S. Acedanski, S. Deb, M. Médard, and R. Koetter, "How good is random linear coding based distributed networked storage," in *Workshop Netw. Coding, Theory Appl.*, 2005, pp. 1–6.

[21] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Distributed fountain codes for networked storage," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, vol. 5, 2006, pp. V–V.

[22] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inform. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[23] J. Liu, M. Adler, D. Towsley, and C. Zhang, "On optimal communication cost for gathering correlated data through wireless sensor networks," in *Proc. 12th Annu. Int. Conf. Mobile Comput. Netw.*, 2006, pp. 310–321.

[24] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, "Cooperative recovery of distributed storage systems from multiple losses with network coding," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 268–276, Feb. 2010.

[25] W. Ou, Z. Yang, L. Tang, and G. Zhongyang, "Design of wireless sensor network based on random linear network coding," in *Proc. Int. Conf. Comput. Sci. Serv. Syst.*, 2012, pp. 986–990.

[26] X. Yang, X. Tao, E. Dutkiewicz, X. Huang, Y. Guo, and Q. Cui, "Energy-efficient distributed data storage for wireless sensor networks based on compressed sensing and network coding," *IEEE Trans. Wireless Commun.*, vol. 12, no. 10, pp. 5087–5099, Oct. 2013.

[27] R. Zeng, Y. Jiang, C. Lin, Y. Fan, and X. Shen, "A distributed fault/intrusion-tolerant sensor data storage scheme based on network coding and homomorphic fingerprinting," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 10, pp. 1819–1830, Oct. 2012.

[28] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 277–288, Feb. 2010.

[29] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: A geographic hash table for data-centric storage," in *Proc. 1st ACM Int. Workshop Wireless Sen. Netw. Appl.*, 2002, pp. 78–87.

[30] R. Tanushetty, L. H. Ngoh, and P. H. Keng, "An efficient resiliency scheme for data centric storage in wireless sensor networks," in *Proc. IEEE 60th Veh. Technol. Conf.*, vol. 4, 2004, pp. 2936–2940.

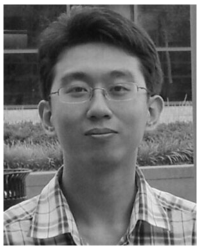
[31] G. Maia, D. L. Guidoni, A. C. Viana, A. L. Aquino, R. A. Mini, and A. A. Loureiro, "A distributed data storage protocol for heterogeneous wireless sensor networks with mobile sinks," *Ad Hoc Netw.*, vol. 11, no. 5, pp. 1588–1602, Jul. 2013.

[32] W.-H. Liao and H.-C. Yang, "A power-saving data storage scheme for wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 35, no. 2, pp. 818–825, Mar. 2012.

[33] B. Sheng, Q. Li, and W. Mao, "Data storage placement in sensor networks," in *Proc. 7th ACM Int. Symp. Mobile ad hoc Netw. Comput.*, 2006, pp. 344–355.

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

[35] E. Hamida and G. Chelius, "A line-based data dissemination protocol for wireless sensor networks with mobile sink," in *Proc. IEEE Int. Conf. Commun.*, 2008, pp. 2201–2205.



Jie Tian received the BS degree in computer science from Tianjin University, Tianjin, China, in 2005, and the MS degree in computer science at Nankai University, Tianjin, China, in 2008. He is working towards the PhD degree in the Department of Computer Science at the New Jersey Institute of Technology. His research includes wireless networks, ad hoc/sensor network and mobile computing.

Tan Yan received the BE degree in 2007 from the School of Information Science and Technology, Southeast University, Nanjing, China, the MS degree from the Department of Electrical & Computer Engineering, NJIT, in 2009, and the PhD degree, under the supervision of Dr. Guiling Wang, from the Department of Computer Science, New Jersey Institute of Technology. He joined NEC Laboratories America in 2014, after completing his PhD. His research includes network analytics, time series mining, mobile ad hoc data management and dissemination, and graph theory.

Guiling Wang received the BS degree in software from Nankai University, Tianjin, China, and the PhD degree in computer science and engineering with a minor in statistics from The Pennsylvania State University, State College, PA, in 2006. She joined the New Jersey Institute of Technology, Newark, NJ, in the fall of 2006 and was promoted to an associate professor with tenure in June 2011.