Data Preservation Under Spatial Failures in Sensor Networks

Navid Hamed Azimi
Stony Brook University
navid@cs.sunysb.edu

Xiaoxiao Hou
Stony Brook University
xhou@cs.sunysb.edu

Himanshu Gupta
Stony Brook University
hgupta@cs.sunysb.edu

Jie Gao
Stony Brook University
jgao@cs.sunysb.edu

***Abstract*—In this paper, we address the problem of preserving generated data in a sensor network in case of node failures. We focus on the type of node failures that have explicit spatial shapes such as circles or rectangles (e.g., modeling a bomb attack or a river overflow). We consider two different schemes for introducing redundancy in the network, by simply replicating data or by using erasure codes, with the objective to minimize the communication cost incurred to build such data redundancy. We prove that the problem is NP-hard using either replication or coding. We propose $O(\alpha)$-approximation algorithm for each of the schemes, where α is the "fatness" of the potential node failure events. We also design a distributed approximation algorithm using erasure codes. Simulation results show that by exploiting the spatial properties of the node failure patterns, one can substantially reduce the communication cost, compared with resilient data storage schemes in the prior literature.**

I. Introduction

In this paper, we address the problem of data preservation in a wireless sensor network after node failures. We focus on smart dust type networks [13] where the network consists of a large number of cheap unreliable nodes. This design philosophy contrasts with the traditional 'centralized' sensing mechanism in which a small number of powerful sensing stations were used (e.g., the weather stations). Here we use the sheer number of sensors for both wide area coverage and high resolution data collection. Having a large number of nodes also increases the system redundancy and robustness to failures. Since nodes are cheap and unreliable, they are likely to fail for many reasons. Nevertheless we have prepared redundant nodes in the proximity to take over both the sensing tasks and the data that has been generated.

Sensor nodes may fail to operate for many reasons. Since the nodes are inexpensive (thus crappy), they may suddenly stop functioning for no reason. The nodes may also be destroyed by animals, humans, or natural disasters (earthquake, fire, river overflow, etc). They may be destroyed by adversarial attacks (a bomb explosion for example). The nodes may also be temporarily disabled by jamming, traffic congestion, or energy depletion. In case of such unfortunate events we can revoke the replacement sensors to take over the sensing tasks. However the data stored on the nodes that have been destroyed is lost unless we design data storage schemes resilient to node failures, which is the topic of this paper.

We focus in the paper node failures with some spatial patterns, which are arguably the most common type. There is often strong spatial correlation among the failed nodes. The events that destroyed one node may very likely influence a nearby node and destroy it as well. We model such spatial failure patterns by some explicit geometric shapes, where all the nodes contained in the shape fail at the same time. The location and orientation of the shape could be variable or known depending on the scenarios. An example could be a bomb attack with a circular shape and variable location (it can happen anywhere in the network) or break of a dam with fix location and orientation (the location of the dam and area affected are known).

We assume that there are k nodes within the network generating data of interest and some additional n nodes that can be used for extra storage and relay for communication. To preserve data generated in the network, we necessarily need to introduce sufficient redundancy by storing the data at some other node. Due to the small form factor and cheap costs, individual nodes memory cannot scale to the size of the network. More specifically, we assume over time the data generated by one data node would eventually occupy almost all of an individual node's memory. The nodes also have severe communication, computation and memory limitations. Thus our algorithm will try to use as little communication as possible.

Contributions. We address the problem of introducing sufficient redundancy with minimal communication cost to a network such that the entire network data can be retrieved after a failure. We use two approaches to introduce data redundancy in the network.

- **Replication:** Each data packet is copied into multiple storage nodes in the network. The retrieval algorithm is simply pulling the data out of the storage nodes that contains a copy.
- **Erasure codes:** An erasure code of multiple data packets would be computed and stored in the storage node. In particular, we use random linear codes as in [9]. That is, each codeword is a linear combination of the original data (called symbols) with random coefficients. The retrieval process consists of one sensor node locally pulling relevant data from the network so as to solve a system of linear equations to decode the data.

Each method has its own advantages and disadvantages. Generally speaking, data replication allows for straight-forward data recovery from a surviving node holding the data. Using erasure codes, we can potentially use the limited storage nodes in a more efficient and effective manner, since a storage node

can possibly hold information helpful for multiple data nodes. The downside is that data recovery requires the decoding cost of solving a linear system of equations.

Using any of the two approaches, our objective is to minimize the amount of data transmissions for introducing redundancy. We prove that such an optimization problem is NP-hard using any of the two approaches. Therefore, we propose $O(\alpha)$ approximation algorithms, where α is the 'fatness' of the given potential spatial node failures.

II. Problem Formulation and Related Work

In this section, we start by describing our network model. We then give the formal formulation of the problem using each of the redundancy schemes and then, we discuss the related work done in this area.

Network Model. Consider a network of sensor nodes deployed in a plane. In our network models, each node is *either* a data node or a storage node, but never both. Each *data node* generates data of interest, while the *storage nodes* can be used for storage of replicated data. We assume that over time each data node generates data whose size is almost the size of an individual node memory and each data node stores a copy of its own data. Thus, a data node cannot be used for storage of other nodes' data. *Throughout the article, we assume that the total number of storage nodes is more than the total number of data nodes in the network.* Also, each node is aware of its own location and the coordinates of its neighbors relative to itself.

The major focus of the paper is to design a redundant data storage scheme with minimal communication cost such that the network data can be recovered after node failures with spatial patterns. Below, we formally define our model of communication cost and node failure patterns.

Communication Graph and Costs. We use r to denote the uniform *transmission radius* of the sensor nodes, and two nodes can communicate directly with each other if the Euclidean distance between them is less than r. The *communication graph* of the network is defined over the set of all nodes as vertices and has an undirected edge between two nodes i and j if they can communication directly with each other. The *communication distance* between two nodes i and j is the distance between i and j in the communication graph.

Definition 1. (Communication Cost.) Let d be a data node and S be a set of storage nodes. We use $C(d, S)$ to denote the cost of transmitting one packet of data from d to all the nodes in S. For simplicity, we assume $C(d, S)$ to be equal to the size of S plus the communication distance between d and the nearest destination in S. □

The above cost model is reasonable if S is clustered in a region and we use Geocast [15] like technique to broadcast a message to S. We note that the above communication cost model is only for the sake of simplicity of presentation, and the results of our paper hold even for the most general cost model wherein $C(d, S)$ is the size of the minimum Steiner tree over $d \cup S$ in the communication graph.

Failure Pattern (the geometric definition): A failure pattern on the network is a closed 2-dimensional geometric shape. Both location and orientation of the failure in the plane can be known or unknown. When a failure occurs, the location and orientation of the failure pattern is determined and all the nodes contained in the failure area simultaneously fail (are destroyed).

As an example, a bomb attack can be defined as a failure pattern with circular shape with variable location, since it can happen anywhere in the network domain. Destruction of a dam on the other hand can be defined as a stripe shape failure which can only happen in a predefined fix location.

An important observation is that a spatial failure with fixed location and orientation always destroys a fixed subset of nodes. Furthermore for every given subset of nodes a closed shape can be found to enclose only the nodes in the subset. Following these observations we can present a failure as a subset of nodes it destroys rather than its shape. This gives a combinatorial definition of failures.

Failure Patterns (the combinatorial definition): A failure pattern can be defined as a subset of nodes in the network.

We assume that no two failures happen simultaneously, i.e. after a failure there is sufficient time for the network to reorganize. We also assume that the cost of data recovery is not a concern. Once a failure is sensed a mobile central station with unlimited resources can do the recovery.

Next we present two problem formulations, for the two methods of introducing redundancy.

A. *Redundancy with Replication*

In this subsection, we give the formal definition of the problem when, to allow recovery from failures, we simply replicate data at other storage nodes. We start with defining storing set as follows.

Definition 2. (Storing Set.) For a given data node d, a storing set is a set $S(d)$ of storage node such that no failure destroys the data node and all the nodes in the set $S(d)$. □

Note that in our model the size of data is almost the size of the available storage of a node, and thus, we cannot store more than one data packets in each node. Therefore, the storing sets for different data nodes should be disjoint. We can now formally define the problem as follows.

Minimum Cost Data Replication (MCDR) Problem. Given a network with data and storage nodes and a set of failure patterns, find a set of disjoint storing sets, one for each data node, such that the sum of communication costs from a data node to its storing sets is minimized. Formally, if D is the set of data nodes, then for each $d \in D$, we find a storing set $S(d) \subseteq S$ such that $\sum_{d \in D} C(d, S(d))$ is minimum.

Theorem 1. *The MCDR problem is NP-hard. Moreover, it is also NP-hard to approximate the MCDR problem within any finite approximation ratio.* ■

We defer the proof of the above theorem to Section VI.

B. Redundancy with Erasure Codes

The second scheme to introduce redundancy is to use decentralized erasure codes as introduced in [9]. In this scheme, each of the data nodes pre-routes its packets to specific storage nodes. Each storage node creates and stores a codeword of all the data packets it receives. The codeword is constructed by cascading linear combination of chunks of input data packets and in order to decode the data one should pull the network data and solve a system of linear equations. The details of coding and decoding procedures for decentralized random linear codes can be found in [10].

One can represent this erasure coding scheme by a bipartite graph between data nodes and storage nodes such that there is an edge between a data node d and a storage node s if d pre-routes its packet to s. Since each data node stores a copy of its own data, in the recovery phase, the data of surviving data nodes can be eliminated from the codewords of surviving storage nodes. Hence the necessary condition for recovering data after a failure is to recover the data of destroyed data nodes from the surviving storage nodes. As shown in [9], the necessary condition for successful recovery is the existence of a maximal matching between destroyed data nodes and the surviving storage nodes [5], [6]. We can now formulate our problem as follows.

Minimum Cost Data Coding (MCDC) Problem. Given a network with D and S as the set of data and storage nodes respectively and a set of failure patterns, the MCDC problem is to construct a bipartite graph $G'(D \cup S, E')$ with minimum sum of edge weights where:

- The weight of an edge (s, d) in E' is the cost of communication between s and d, and
- The set of edges E' is such that for any given failure pattern F, the induced subgraph in G' over $(D \cap F) \cup (S - F)$ has a matching of size $|D \cap F|$. Here, we have used F to denote the set of nodes destroyed by F.

Note that for a failure F, $(D \cap F)$ is the set of destroyed data nodes and $(S - F)$ is the set of surviving storage nodes. Thus, the above condition ensures that there is a matching of size equal to the number of destroyed data nodes between the set of destroyed data nodes and the surviving storage nodes.

Theorem 2. *The MCDC problem is NP-hard.* ■

We defer the proof of the above theorem to Section VI.

C. Related Work

Increasing data persistence in the presence of node failure has been a subject of increasing research in recent years. One popular approach to this problem is to use network coding. The usefulness of network coding for data storage was investigated in [17] where authors show a simple distributed scheme using network coding can perform as well as the case when there is complete coordination between nodes.

Dimakis et. al. in [8]–[10] and Lin et. al. in [14], [18] purposed two different schemes for decentralized implementation of Fountain codes in wireless sensor networks. Both algorithms use limited global information such as the total number of nodes and sources. Aly et.al in [2] use simple random walks to implement a decentralized Fountain code without presence of any global information. Kamra et. al. in [1] purposed a different approach with the goal of enhancing data persistence in network in case of node failure. It uses growth codes to maximize the amount of information available for decoding at any chosen moment.

All of the previous research on this topic use a probabilistic node failure model without any spatial pattern for node failure. To the best of our knowledge, this is the first paper addressing the problem of preserving the data in case of spatial attacks/-failure on a wireless sensor network. Although any of previous methods can be used for designing failure tolerant network, the present of spatial pattern for node failure allows for reducing the required redundancy and communication significantly.

III. Approximation Algorithm for the MCDR Problem

In this section, we design an algorithm for the MCDR problem which yields a solution with a near-optimal cost on an average, for uniformly random networks.

Survival Matching Algorithm (SMA). Note that in our MCDR problem if we restrict the size of storing sets to be just one storage node, then the MCDR problem can be easily reduced to the minimum-cost 2D-matching problem (in bipartite graphs). For the unrestricted MCDR problem, our approximation algorithm called the *Survival Matching Algorithm (SMA)* essentially solves the restricted MCDR problem optimally (i.e., restricts the storing sets to just one storage node and finds the minimum-cost 2D-matching in an appropriately defined bipartite graph). We will later show that SMA's solution is within a constant factor of the unrestricted optimal solution for uniformly random networks.

SMA Description. Given a network of data and storage nodes, SMA starts with constructing a bipartite graph $G_s(D \cup S, E_s)$ between data and storage nodes wherein there is an edge between a data node d and a storage node s if and only if d and s do not exist together in any particular failure set. Essentially, an edge $(d, s) \in E_s$ signifies that $\{s\}$ can be picked as a storing set for the data node d. Thus, a perfect matching in G_s yields a set of disjoint storing sets of size one each, and hence is a solution (not necessarily optimal) to the MCDR problem. In addition, each edge (d, s) in G_s is assigned a weight equal to the communication cost between d and s, and we actually find a minimum-cost perfect matching in G_s. Note that minimum-cost perfect matching problem can be solved in polynomial time [7].

Approximation Proof. We now show that in uniformly random networks with spatial failures, SMA delivers a solution

whose cost is at most α times the optimal cost with high probability. Here, α is the fatness (as defined below) of the given set of spatial failures.

Fatness of Spatial Failures. Fatness is a well-known concept in geometry which quantifies how a geometric object is spread in all directions [4]. For a given object O, its fatness is defined as follows. Let C and C' be two concentric circles such that C fully contains O and C' is fully contained in O. Then, the fatness of O is defined as the maximum possible value for the ratio $\frac{Radius(C)}{Radius(C')}$ over all possible such concentric circles C and C'. In this paper, we extend the concept of fatness to a set of geometric objects (spatial failures) as below.

Definition 3. (Fatness) For a given set of spatial failures F, let (i) R' be the radius of the largest circle that can be contained in each of the failures, and (ii) R be the radius of smallest circle that can contain each of the failures. The fatness α of the set of spatial failures F is defined as the ratio R/R'. □

Proof Outline. We compute the approximation ratio of SMA by estimating (i) a lower bound on the optimal cost, and (ii) the expected cost of the SMA solution, in the below two lemmas respectively.

Lemma 1. *For any instance of the MCDR problem, the optimal cost of a solution is at least $(|D|R')/(2r)$ where R' (as defined above) is the radius of the largest circle that can be contained in each of the failures, $|D|$ is the number of data nodes, and r is the transmission radius of the nodes.*

Proof: Consider a storing set and two (storage) nodes s_1 and s_2 in it that are farthest from each other. Distance between s_1 and s_2 must be at least R', since otherwise all the nodes in the storing set can be contained in a circle of radius R' and thus in any given spatial failure with appropriately chosen location and orientation (which contradicts the definition of a storing set). Now, the minimum communication cost between a data node to (s_1, s_2) is at least half the communication cost between s_1 and s_2 which is at least R'/r. Thus, the total cost of a set of $|D|$ storing sets is at least $(|D|R')/(2r)$. ■

The proof of the below lemma is rather tedious, and is deferred to Section VI.

Lemma 2. *Consider a uniformly random network, i.e., with uniformly and randomly distributed data and storage nodes, with failures that can destroy at most 1/4 of the network nodes.*

For such networks, SMA delivers a valid solution with a probability of 99.5%, and the expected cost of the delivered solution is $O(|D|.R/r)$, where R (as defined in Definition 3) is the radius of the smallest cirle that can contain each of the given failure patterns, $|D|$ is the total number of data nodes, and r is the transmission radius of the nodes. ■

From the above two lemmas, we get the following theorem.

Theorem 3. *For uniformly random networks with failures that can destroy at most 1/4 of the network nodes, SMA delivers a valid solution with a probability 99.5%, and the expected approximation ratio (over all random networks for which SMA delivers a valid solution) is $O(\alpha)$ where α is the fatness of the given failure patterns.* ■

Significance of Theorem 3. Note that if we allow arbitrarily large failures, then no valid solution may not even exist. Further, since the decision version of MCDR problem is NP-complete (as shown in Section VI-C), it is unlikely that a polynomial algorithm can always return a valid solution if one exist. Thus, an algorithm returning a solution with high probability is the best we can hope for. Finally, since it is NP-hard to approximate the MCDR problem in general (see Theorem 1), the above average approximation-ratio over almost all random networks is of significance.

Distributed Algorithm. To the best of our knowledge, there are no efficient distributed approximation algorithm known for the minimum-cost matching problem, which is a special case of our MCDR problem. We plan to address this direction in our future work.

IV. Approximation Algorithms for the MCDC Problem

In this section, we store erasure codes of data packets to generate data redundancy. We start by showing that SMA of the previous section can be used to also solve the MCDC problem with an approximation ratio for random networks. We also design a distributed approximation algorithm, and prove its performance guarantees.

Using SMA for the MCDC Problem. Here, we show that SMA for the MCDR problem can also be used to yield an approximation solution for the MCDC problem in random networks. Firstly, note that the output of SMA, viz., a set of disjoint storing sets, can be used to construct a valid solution G' for the MCDC problem by connecting each data node to each storage node in its storing set. Similar to the arguments in Lemma 1, we can show that the optimal cost of any instance of an MCDC problem is at least $|D|R'/2r$. Thus, by Lemma 2, which also applies to the MCDC solution yielded by SMA, we have the following approximation result.

Theorem 4. *For uniformly random networks, the MCDC solution delivered by SMA (as described above) has an average approximation ratio of $O(\alpha)$, where α is the fatness of the given spatial failures.* ■

Distributed Storage Algorithm (DSA). The main advantage of storing linear combination of data (as in the MCDC problem) over simple replication (as in the MCDR problem) is that the decisions of where to store the data can be made locally by the data nodes, i.e. the data nodes don't have to globally compete for exclusive use of storage nodes. However, the drawback of this linear combination approach is that recovery of data cannot be guaranteed for all failures, since in case of some failure the relevant remaining linear equations may not yield a full rank system. Below, we present a distributed algorithm that guarantees recovery of data with a high probability in random networks.

DSA Description. Consider a uniformly random network of data nodes and storage nodes. Without loss of generality, we assume the density of the network to be one unit. For each data node, we define its ***storage region*** to be the rectangle of size $2 \times \phi$ at a vertical distance of $2R$ below, as shown in Figure 1. Here, R is the radius of the smallest circle that can contain all failures, and ϕ is a constant (see Equation 1) which depends on desired probability of recovery and ratio of number of data to storage nodes and is defined later (see Equation 1). More formally, if (x, y) are the coordinates of the data node, then its storage area is a rectangle with the left-most top coordinate equal to $(x - 1, y - 2R)$ and has a length of 2 and a height of ϕ.

The bipartite graph $G'(D \cup S, E')$ returned by DSA consists of edges that connect a data node to each storage node in its storage region. The implementation of DSA entails each data node broadcasting its data to all the storage nodes in its storage region, and each storage node stores a linear combination of all the data packets received from various data nodes.

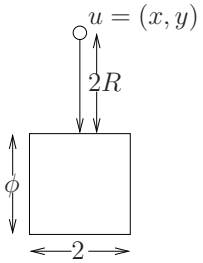

Figure 1. Storage area of a data node in DSA.

Theorem 5. *Given a uniformly random network and a set of failure patterns, the solution returned by DSA allows successful recovery of data with a probability of $(1-\varepsilon)$, when there is a single failure, if the value of ϕ is chosen as:*

$$\phi = \frac{1}{n}\mathbf{max}((n+k)c^2 + (k-n)R, c^2(n+k)), \qquad (1)$$

where k and n are the number of data and storage nodes respectively, R is the radius of the smallest circle that can contain each failure, and c is the smallest real number such that $g(c) > (1-\varepsilon)$ where $g(x)$ is the Gaussian error function. Note that when $k < n$, the above equation simplifies to $\phi = c^2(n+k)/n$. ■

The proof for the above theorem is quite involved and hence, is deferred to Section VI. We now prove the average approximation ratio of DSA.

Theorem 6. *For uniformly random networks, DSA returns a solution with an expected approximation-ratio of $O(\alpha)$.*

Proof: Using arguments similar to Lemma 1, we can show that the minimum cost for a solution to MCDC is $(|D|R')/(2r)$. Below, we show that the expected cost of DSA's solution is $O(|D|R/r)$ which will prove the theorem.

In DSA, a data node broadcasts its data to a rectangular region of size $r\phi$ located at a vertical distance of $2R$. Since our choice of ϕ value is $O(1)$ (see Equation 1) when number of data nodes is less than the number of storage nodes, the communication cost incurred by each data node is $O(2R/r) = O(R/r)$. Thus, the total expected cost of the DSA solution is $O(|D|R)$. ■

V. Simulations

In this section, we provide simulation results of our algorithm. We compare our algorithms with Dimakis' algorithm [8] in terms of the total communication cost under spatial failures with different radii in networks of different sizes. We show that our algorithm incurs much less communication cost than Dimakis' algorithm, and achieves very similar successful recovery probability.

Comparison of Communication Costs. Figure 2(a) compares the cost of SMA and DSA to the decentralized erasure codes as used in [8]. Our experiment is performed on a network with $10,000$ sensor nodes, 20% of them are data nodes. All nodes are uniformly distributed in a 100×100 rectangle area. The communicate radius is 2.5. The X-axis is the radius of circular potential failure, varying from 5 to 15. The Y-axis is the total communication cost. For the algorithm in [8], each data node sends its data to exactly $\log k$ storage nodes. Since the algorithm in [8] does not adjust to different failure size, the curve of [8] on the figure is a line. From the figure, we can find SMA and DesSMA need much fewer messages than the algorithm in [8]. And the recovery probability is very similar, the theoretical recovery probabilities are all more than 99.99%. In our simulations, all three algorithms can successfully recover data.

Fig 2(b) is the comparison of total communication costs when the network size is increasing. The X-axis is the network size. In particular, x^2 nodes are uniformly randomly distributed in an $x \times x$ area. x varies from 100 to 1000. For each network, 20% nodes are data nodes, the communication radius is 2.5, and the potential failure radius $0.1x$. The Y-axis is the total message costs, on a log scale. We can find, in every size of the network, our DSA algorithm is one order of magnitude better.

We are also concerned about the communication costs for networks with different fractions of data nodes. Fig 2(c) shows the result. This time we fix the network size. We have $250,000$ nodes uniformly randomly distributed in a 500×500 area, but the ratio of data nodes changes. The X-axis is the ratio of the data nodes, varying from 10% to 80%. The communication radius is 2.5, and the potential failure size is 50. The Y-axis shows the total cost of messages to distribute data. We can find that when the ratio of data nodes increases, the cost is increasing fast. Especially when the radio is bigger than 0.5, which means there are more data nodes than storage nodes. However, for the type of network we are considering there are enough number of redundant (i.e., storage) nodes, so this situation is rare.

Probability of Successful Recovery for DSA. For DSA, the size of the storage rectangle would affect the probability of

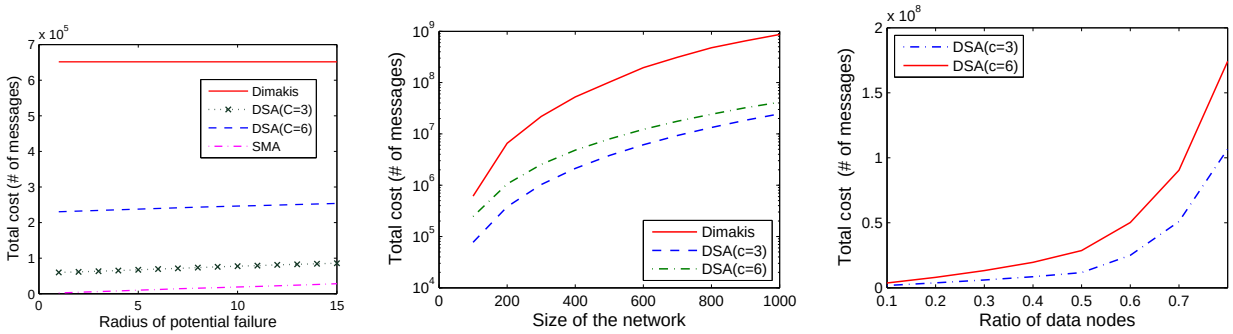

Figure 2. The communication cost of various algorithms for increasing (a) potential failure sizes, (b) network size, and (c) ratio of data nodes to storage nodes.

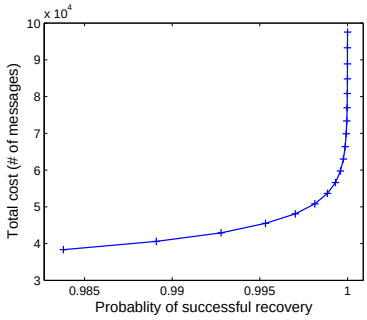

Figure 3. The total communication cost for different desired probability of successful recovery.

successful recovery. To achieve higher probability, we need a larger storage rectangle, which means higher cost during the data distributing phase. Fig 3 shows the relationship between the communication cost and the probability of successful recovery. X-axis is the recovery probability, varying from 0.9837 to $1 - 7 \times 10^{-7}$. Y-axis is the total communication cost. All simulations are ran on a sensor network with 10,000 nodes uniformly randomly distributed in a 100×100 area. 20% nodes are data nodes, and the potential failure size is 10, and communication radius is 2.5. Our result shows that the communication cost grows slowly if we keep the success probability to be lower than 0.9998 (where $C = 2.6$).

In the last two sets of simulations, we did not include SMA algorithm. SMA is a centralized algorithm with running time $O(n^3)$. It is computationally too heavy for the large network that we are testing.

VI. PROOFS

In this section, we present proofs of Lemma 2, Theorem 5, and the NP-hardness results in the three subsections.

A. *Proof of Lemma 2*

Proof of Lemma 2. Due to lack of space and clarify, here we prove only the second claim of the lemma about the expected cost of the SMA solution. We refer the reader to our full version of the paper [3] for the proof of the first claim of 99.5% probability.

Recall that the MCDR problem is to find disjoint storing sets of arbitrary size. In contrast, SMA finds singleton storing sets (i.e., a maximal matching) which it can do it optimally since min-cost matching problem can be solved in polynomial time. Thus, in order to bound the cost of the SMA solution, it is sufficient to show that there exist some matching of cost $O(|D|R/r)$ in uniformly random networks. We show the above by introducing an algorithm that finds a matching if a matching exists and show that its expected cost is $O(k.R)$.

Cell Matching Algorithm (CMA). CMA works by creating a grid in the network with initial unit square cells. Then at each if there exist unmatched data nodes CMA try to find a matching for it in the cell assigned to it. At each stage the size of the cells and cost of matching increase however the number of nodes needs to be matched decrease with a faster rate, which result in a expected cost of $O(k.R)$.

CMA starts by dividing the network region into cells of unit square size, and then *repeats* the following steps until all the data nodes have been paired/matched to some storage node.

- For each cell, we assign the cell at vertical distance of $2R$ below to be its "storage cell".
- For each yet-unmatched data node, try to find an unmatched storage node in the storage cell of its cell. If all the data nodes of a cell are matched, mark the cell as `complete`.
- Merge pairs of horizontally-adjoining cells to construct new cells of double the width (but the height remains unit).

In the end, when each cell has become a complete row, we match the remaining data nodes to the remaining unmatched (possibly, very far away) storage nodes in the network.

Estimating the Cost of CMA Matching. To estimate the cost of the matching delivered by CMA, let us first compute an upper bound on the cost of matching nodes in each cell at each step. At the i^{th} step, we have the following.

- The expected number of nodes in each cell is 2^i, since the size of a cell is $2^i \times 1$ and we assume unit density.
- The maximum communication distance between two nodes that can be matched is $O(\sqrt{R^2/r^2 + 2^{2i}}) = O(2^i R/r)$.

To bound the cost incurred in matching/pairing nodes in the i^{th} step, let us assume the worst case scenario that each node in each *incomplete* cell actually gets matched. In such a case, an incomplete cell incurs a maximum cost of $O(2^{2i}R)$ in the i^{th} step.

In the very last stage (after each cell is a complete row), all the yet-unmatched nodes are matched wherever possible. But,

since at this stage, 2^i is equal to length of the network, the cost is still bounded by $O(2^{2i}R)$.

Finally, Lemma 3 shows that the expected number of incomplete cells at each step is ((Number of Cells) $\times O(e^{(-2^{i/2})})$). Now, since the number of cells containing a data node is bounded by $|D|$ at each step, we get the overall cost of CMA solution as:

$$\sum_i |D|O(e^{(-2^{i/2})})O(2^{2i}R/r) = O(|D|R/r).$$

■

The proof of the below lemma is omitted (see [3]).

Lemma 3. *In CMA (as described in the above lemma), the probability that a cell remains uncomplete in the i^{th} step is $O(e^{-(2^i/2)})$.* ■

B. *Proof of Theorem 5*

In this subsection, we present proof of Theorem 5. We use the following two basic notations throughout this subsection. Let $G'(D \cup S, E')$ be the solution returned by DSA for a network with D and S as the set of data and storage nodes.

- We use D_X and S_X to denote the set of data and storage nodes respectively in X, where X is a geographic region in the network or a set of network nodes.
- For a set of data nodes δ, we use $N(\delta)$ to denote the set of storage nodes that are connected by an edge in G' to some data node in δ. In other words, $N(\delta)$ is the set of storage nodes in the storage region of one of the data nodes in δ.

Proof of Theorem 5. As discussed in Section II-B, for successful recovery of data when a failure F occurs, the induced subgraph of G' over $(D \cap F) \cup (S - F)$ must have a matching of size $|D \cap F|$. Here, we are using F to also denote the set of nodes destroyed by F. Thus, to prove the theorem, we need to show that such a matching exists with a probability of $(1 - \varepsilon)$.

Without loss of generality, let us assume the given failure F to be a square region of size $2R \times 2R$. Recall that R is radius of the smallest circle that can contain each given failure. We will prove the theorem using the following sequence of claims.

- First, note that F does not destroy any node in $N(D_F)$, since the storage region of a data node is more than a distance of $2R$. Thus, it suffices to show that with a probablity of $(1 - \varepsilon)$, there is a matching of size $|D_F|$ in the induced subgraph in G' over $(D_F \cup N(D_F)$; note that $D_F = (D \cap F)$ and that only the nodes in $N(D_F)$ are useful in finding a matching.
- By Hall's Theorem [12], the above desired matching does not exist iff there is $\delta \subset D_F$ such that $|\delta| > |N(\delta)|$. We show that the probability of this event is at most ε using the following two steps.
  - First, we show in Lemma 4 that if there exist a set $\delta \subset D_F$ such that $|\delta| > |N(\delta)|$ then there is *rectangular* region L in the region F such that $|D_L| > |N(D_L)|$.
  - Then, in Lemma 5, we show that the probability of such a rectangle L existing is less than ε. Intuitively, this is true due to proportionally smaller size of L in comparison to the union of the storage regions of the data nodes in L.

■

We now prove the two lemmas used in the above proof.

Lemma 4. *If there exists a set $\delta \subset D_F$ such that $|\delta| > |N(\delta)|$, then there is a rectangular region L in the failure region F such that $|D_L| > |N(D_L)|$.*

Proof: Consider $\delta \subset D_F$ such that $|\delta| > |N(\delta)|$. Let B_δ be the smallest axis-aligned rectangle that contains δ. Without loss of generality, let us assume that $\delta \subset D_F$ is the set with *smallest* B_δ that satisfies $|\delta| > |N(\delta)|$.

Since δ is contained in B_δ, we have $|D_{B_\delta}| > |\delta|$. Since, $|\delta| > |N(\delta)|$, we get $|D_{B_\delta}| > |N(\delta)|$. Below, we show that $N(\delta) = N(D_{B_\delta})$, which will imply that $|D_{B_\delta}| > |N(D_{B_\delta})|$ and thus, showing the B_δ is the desired rectangle L.

Showing $N(\delta) = N(D_{B_\delta})$. Essentially, we wish to show that expanding the set of data nodes from δ to D_{B_δ} doesn't necessarily increase their total storage region.

Let us assume that there is a storage node z such that z is in $N(D_{B_\delta})$ but not in $N(\delta)$; let z be the highest (with largest y-coordinate) such storage node. As shown in Figure 4, consider the four rectangles Z_1, Z_2, Z_3, and Z_4. Here, Z_1 and Z_2 partition the rectangle B_δ at a horizontal line at a vertical distance of $2R + \phi$ from z, and Z_3 and Z_4 partition $N(D_{B_\delta})$ based on z.[1] We claim the following.

1) Since z is the highest storage node that is in $N(D_{B_\delta})$ but not in $N(\delta)$, each storage node in Z_3 is in $N(\delta)$.
2) Each data node is Z_1 stores its data only in the storage nodes in Z_3, since DSA stores data in storage nodes that are at a vertical distance of at most $2R + \phi$.
3) Number of data nodes in Z_1 that are in δ must be less than the number of storage node in Z_3 that are in $N(\delta)$, since otherwise δ wouldn't be the data set with smallest enclosing rectangle that satisfies $|\delta| > |N(\delta)|$.

The above three claims imply that if we omit the set of data nodes in Z_1 from δ, we get a set of data nodes δ' with a smaller enclosing rectangle (Z_2) that satisfies $|\delta'| > |N(\delta')|$ — which is a contradiction to our original premise. Thus, no such storage node z exists, and hence, $N(D_{B_\delta}) \subseteq N(\delta)$ which implies $N(D_{B_\delta}) = N(\delta)$. ■

Lemma 5. *For a given failure region F, the probability of existence of a rectangle L in F such that $|D_L| > |N(D_L)|$ is less than ε.*

Proof: For an arbitrary rectangle L over a uniformly random network, both the number of data nodes and number

[1]For simplicity, we have used $N(D_{B_\delta})$ to denote the rectangular *region* corresponding to the union of the storage regions of nodes in D_{B_δ}.

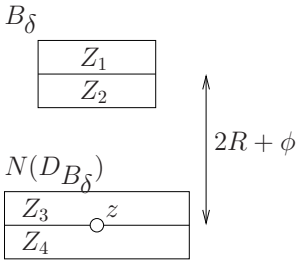

Figure 4. Partitioning of rectangles B_δ and $N(D_{B_\delta})$ based on z.

of storage nodes in X are random variables. Below, we start by computing the probability distribution functions (pdf) of these two random variables. We can then compute the pdf of the difference between the number of data nodes in L and the number of storage nodes in the storage region of L. We then show that if ϕ is chosen as defined in Equation 1, the required probability is smaller than ε.

As mentioned before, for sake of simplicity, we assume the network density to be one unit. We use n and k to denote the total number of data and storage nodes in the network. Since the density of the network is 1, the total area of the network is $n+k$.

Computing distribution of D_X and S_X. Recall that for a region/set X, D_X and S_X are the number of data and storage nodes respectively in X. For a randomly selected rectangle X of length l and height w, D_X and S_X are random variables with a binomial distribution of number of experiments equal to k and n respectively and a success probability of $lw/(n+k)$.[2] Since both n and k are large numbers, we can use the normal approximation [16] for the above binomial distributions. Thus, we have

$$D_X \sim \mathcal{N}\left(\begin{array}{l}\mu = \frac{klw}{(n+k)},\\ \sigma^2 = \frac{klw}{(n+k)}\left(1-\frac{lw}{(n+k)}\right)\end{array}\right)$$

$$S_X \sim \mathcal{N}\left(\begin{array}{l}\mu = \frac{nlw}{(n+k)},\\ \sigma^2 = \frac{klw}{(n+k)}\left(1-\frac{lw}{(n+k)}\right)\end{array}\right)$$

Distribution of the difference (U). Consider a random rectangle X of size $l \times w$, and let Y be its storage region (i.e., union of the storage regions of the data nodes in X). Note that Y is a rectangle of size $(l+2)\times\phi$. We now wish to compute the pdf of the random variable $U = S_Y - D_X$. Our goal is to bound the probability of $U < 0$.

Now, given two independent random variables V_1 and V_2 with normal approximations $V_1 = \mathcal{N}(\mu_1, {\sigma_1}^2)$ and $V_2 = \mathcal{N}(\mu_2, {\sigma_2}^2)$ respectively, the probability distribution of $(V_1 - V_2)$ is given by $\mathcal{N}(\mu_0 + \mu_1, {\sigma_1}^2 + {\sigma_2}^2))$. Since the random variables D_X and S_Y are independent, we get the below as the distribution for $U = S_Y - D_X$. Recall that Y is of size $(l+2)\times\phi$.

$$U \sim \mathcal{N}\left(\begin{array}{l}\mu = \frac{klw}{(n+k)} + \frac{n(l+2)(w+\phi)}{(n+k)},\\ \sigma^2 = \frac{klw}{(n+k)}\left(1-\frac{lw}{(n+k)}\right) +\\ \qquad \frac{k(l+2)(w+\phi)}{(n+k)}\left(1-\frac{(l+2)(w+\phi)}{(n+k)}\right)\end{array}\right)$$

Simplifications. In order to estimate the probability of $U > 0$, the above pdf must be significantly simplified. We use the following tactics to simplification.

- The average of U is greater than zero and we want to upper-bound the probability of $U > 0$. Thus, we can use a higher-value estimate for σ and a lower value for μ. Thus, we omit the terms $(1-\frac{(l+2)(w+\phi)}{(n+k)})$ and $(1-\frac{(l+2)(w+\phi)}{(n+k)})$, and use $l+2$ instead of l.
- We can multiply both the average and standard deviation with a constant number without changing the probability. We multiple both by $\frac{1}{l+2}$.

Applying the above simplifications, we get:

$$U \sim \mathcal{N}\left(\begin{array}{l}\mu = \frac{kw}{(n+k)} + \frac{n(w+\phi)}{(n+k)},\\ \sigma^2 = \frac{kw}{(n+k)} + \frac{n(w+\phi)}{(n+k)}\end{array}\right)$$

Bounding Pr($U > 0$). Now we want to bound the probability of $U > 0$ by ε, over all possible values of w, by choosing an appropriate value for ϕ. Since U has a normal distribution, the probability of $U < 0$ is less than ε if ((average of U) $>$ ($c\times$ standard deviation of U)) where $c = g(1-\varepsilon)$ where $g()$ is the Guassian error function.[3] Since both average and standard deviation of U are positive numbers, we can write the inequality in the form (average)$^2 >$ ($c\times$ standard deviation)^2. Thus, we get the following equation that ensure the desired upper bound on $Pr(U > 0)$

$$\left(\frac{kw}{(n+k)} + \frac{n(w+\phi)}{(n+k)}\right)^2 > c^2.\left(\frac{kw}{(n+k)} + \frac{n(w+\phi)}{(n+k)}\right)$$

For given values of n and k, we want to choose ϕ such that the above equation holds for all values of $0 \le w \le R$. Solving the above (omitting details), we get

$$\phi = (1/n)\mathbf{max}\left((n+k)c^2 + (k-n)R, c^2(n+k)\right)$$

■

C. *NP-Hardness Proofs*

Proof of Theorem 1. We show that our MCDR problem is NP-hard by reducing the well-known 3D-matching (3DM) problem, which is known to be NP-complete [11], to the decision version of the MCDR problem. The decision version

[2]The probability of a particular node i to be in X is equal to (Area of X)/(Total Area of the Network). The number of data nodes in X is the repeat of this single trial k times.

[3]This is a known characteristic of normal distributions.

of MCDR problem is to check if there is a set of disjoint storing sets (irrespective of the cost), one for each of the data nodes. We start with defining the 3DM problem.

3D-matching. Given three disjoint (unordered) sets X, Y, and Z where $|X| = |Y| = |Z|$, and a relation $T \subseteq X \times Y \times Z$, is there a subrelation $M \subseteq T$ of size $|X|$ (called a maximal 3D-matching) such that for all pairs of elements (x_i, y_i, z_i) and (x_j, y_j, z_j) in M we have $x_i \neq x_j$, $y_i \neq y_j$, and $z_i \neq z_j$. Note that M must contain an element (x_i, y_i, z_i) for each element $x_i \in X$.

Now, consider an instance (i.e., the sets X, Y, Z, and the relation T) of the 3DM problem. Let $T' = X \times Y \times Z - T$. We now construct an instance of our MCDR decision problem from the above as follows.

Constructing an MDNR Instance. Consider a network consisting of data nodes X and storage nodes $Y \cup Z$. We create two types of failure sets:

- For each data node $x \in X$, we add two failures viz., $x \cup Y$ and $x \cup Z$.
- We add a failure set corresponding to each tuple in T'.

Now, we show that the above instance of MCDR has a solution if and only if the 3DM instance has a maximal matching. First, its easy to see that any maximal matching of the 3DM instance gives a solution to the MCDR problem. Below, we show that a solution to the MCDR problem gives a maximal matching to the 3DM instance.

Note that the first type of failure sets dictate that each storing set must contain a node from Y as well as Z. Since $|X| = |Y| = |Z|$ and an MCDR solution must contain $|X|$ disjoint storing sets, each storing set in any MCDR solution must contain *exactly* two storage nodes (one for each Y and Z). Further, for a data node $x \in X$, if its storing set is (y, z), then (x, y, z) must not be in T' and hence must be in T. Thus, the set of storing sets yields a maximal matching.

To prove that MCDR is NP-hard to approximate, note that an approximate algorithm for MCDR can be also used to solve the above defined decision version of the MCDR problem which is NP-complete. Thus, MCDR is NP-hard to approximate. ■

Proof of Theorem 2. We prove MCDC is NP-Hard by reducing the GRAPH-COLORING problem to the decision version of the MCDC problem. In the decision version of the MCDC problem, the objective is to determine if there is a solution G' that has a total edge weight of at most a given quantity. Given an instance (G, k) of GRAPH-COLORING, we construct an instance of the MCDC problem as follows.

- The set of data nodes D is V, the set of vertices of G.
- The set of storage nodes are the k colors plus an additional special node s'.
- For each edge $(i, j) \in E$, we construct a failure $\{i, j, s'\}$.
- For each pair of data and storage nodes, the communication cost is one.
- The objective is to find a solution G' of total cost at most $|V| = |D|$.

Note that the above instance of MCDC has a solution of cost at least $|D|$, since in the solution G' each data node must be connected to at least storage node. Also, if there is a solution of cost $|V$, then each data node graph must be connected to *exactly* one storage node in G'. In such a case, the MCDC solution G' yields a k-coloring of the graph G since (i) each data node is connected to exactly one storage node in G', and (ii) if (i, j) is an edge in G, then data ndoes i and j cannot be connected to the same storage nodes in G' because otherwise the failure $\{i, j, s'\}$ would not allow recovery of data at i or j.

On the other hand, if graph G has a valid k-coloring then we can construct a graph $G'(D \cup S, E')$ of total edge weight $|D|$ by connecting each data node to the storage node corresponding to its assigned color. It is easy to verify that G' is a valid solution to the MCDC problem instance. ■

VII. Conclusions

In this paper, we address the problem of introducing data redundancy in sensor networks with minimum communication cost so as to survive node failures. We presented a distributed approximation algorithm for uniform random networks. For the future work we would like to pursue two directions. First we would like to examine the decoding cost of storage schemes with network coding. Second, we will consider the problem when the network data is correlated.

References

[1] *Growth codes: maximizing sensor network data persistence*, volume 36, 2006.

[2] S. A. Aly, Zhenning Kong, and E. Soljanin. Fountain codes based distributed storage algorithms for large-scale wireless sensor networks. In *Proc.IPSN '08*, 22–24 April 2008.

[3] N. Azimi, X. Hou, H. Gupta, and J. Gao. Data preservation under spatial failures in sensor networks. Technical report, BBU, 2009. http://cs.sunysb.edu/~navid/spf.pdf.

[4] Mark De Berg. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2008.

[5] Bla Bollobs. *Modern graph theory*. Graduate texts in mathematics 184. Springer, New York, 1998.

[6] Bla Bollobs. *Random graphs*. Cambridge studies in advanced mathematics 73. Cambridge University Press, 2nd edition, 2001.

[7] William Cook and Andre Rohe. Computing minimum-weight perfect matchings. *INFORMS J. on Computing*, 11(2):138–148, 1999.

[8] Dimakis et al. Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes. In *IPSN '05*, 2005.

[9] Dimakis et al. Decentralized erasure codes for distributed networked storage. *IEEE/ACM Trans. Netw.*, 14(SI), 2006.

[10] Dimakis et al. *Network Coding for Distributed Storage in Wireless Networks*. Springer, 2008.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. Freeman & Co., 1990.

[12] U. S. R. Murty John Adrian Bondy. *Graph theory*. Springer, 2007.

[13] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". In *MobiCom '99*. ACM, 1999.

[14] Y. Lin, B. Liang, and B. Li. Data persistence in large-scale sensor networks with decentralized fountain codes. In *INFOCOM 2007*, 2007.

[15] Julio C. Navas and Tomasz Imielinski. Geocast—geographic addressing and routing. In *MobiCom '97*, 1997.

[16] Fundamentals of applied probability and random processes. *Oliver Chukwudi Ibe*. Academic Press, 2005.

[17] M. Mdard S. Acedanski, S. Deb and R. Koetter. How good is random linear coding based distributed networked storage. In *In NetCod*, 2005.

[18] B. Liang Y. Lin and B. Li. Geometric random linear codes in sensor networks. In *ICC '08*, 2008.