

Data Preservation Under Spatial Failures in Sensor Networks

Navid Hamed Azimi Himanshu Gupta Xiaoxiao Hou, Jie Gao

Computer Science Department, Stony Brook University, Stony Brook, NY

{navid,hgupta,xhou,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 design $O(\alpha)$ -approximation centralized and distributed algorithms for the two redundancy schemes, where α is the “fatness” of the potential node failure events. Using erasure codes, data distribution can be handled in an efficient distributed manner. Simulation results show that by exploiting the spatial properties of the node failure patterns, one can substantially reduce the communication cost compared to the resilient data storage schemes in the prior literature.

Categories and Subject Descriptors

C.4 [Performance of Systems]: Fault Tolerance; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Reliability, Theory

Keywords

Data replication, Fault tolerance, Coding, Sensor Networks

1. INTRODUCTION

In this paper, we address the problem of data preservation in a sensor network after node failures. We focus on smart dust type networks [10] where the network consists of a large number of cheap, unreliable nodes. This design

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiHoc '10, September 20–24, 2010, Chicago, Illinois, USA.
Copyright 2010 ACM 978-1-4503-0183-1/10/09 ...\$10.00.

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 the prepared redundant nodes in the proximity can take over both the sensing tasks and the data of the failed nodes.

Sensor nodes may fail to operate for many reasons. They may suddenly stop functioning for no reason. They may 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 when a sensor does not respond, we can invoke 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 on 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 destroy 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 or the break of a dam flooding nodes within a region with fixed location and orientation.

We assume that a certain number of nodes, called the *data* nodes, generate data of interest, while the other network nodes called the *storage nodes* are 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 nodes. The nodes also have severe communication, computation and memory limitations. Thus, our algorithms will find redundant nodes to hold data from each data node, while incurring as little communication as possible.

Contributions. We address the problem of introducing sufficient data redundancy in a network with minimal communication cost, such that entire network data can be re-

tried after node failures. We use two schemes to introduce data redundancy in the network.

- **Replication:** The data from a data node is copied into one or multiple storage nodes in the network. The retrieval algorithm is simply pulling the data out of storage nodes that contain a copy.
- **Erasur codes:** An erasure code of multiple data items would be computed and stored in the storage node. For example, each codeword may be linear combination (with random coefficients) of the original data items [7]. The retrieval process consists 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 communication cost incurred in introducing redundancy. We prove that such an optimization problem is NP-hard using either of the two approaches. Thus, we design $O(\alpha)$ -approximation centralized and distributed algorithms, where α is the ‘fatness’ of the given potential spatial node failures. To the best of our knowledge, ours is the first work that addresses the data preservation problem in the context of *spatial* failures, while minimizing the communication cost incurred in introducing the needed data redundancy.

Paper Organization. The rest of the paper is organized as follows. In Section 2, we present our network model and notations. In Section 3, we discuss the problem with replication as the choice method of generating redundancy. In Section 4, we discuss the problem using erasure codes. We present our simulation results in Section 6, and discuss related works in Section 5. We defer two tedious proofs to the two subsequent sections.

2. NETWORK MODEL AND NOTATIONS

In this section, we present our model of the network and failures, and introduce basic notations.

Network Model. In our sensor network model, each node is aware of its own and neighbors’ location. A node is either a data node or a storage node, but never both. Each data node generates data of interest, and the storage nodes are used by the data nodes for storage of replicated data. For clarity of presentation, we assume that each data node generates exactly one unit of data, and a storage node can be used to store only one unit of data.¹

Communication Model. We assume a quasi-UDG [12] model of parameters r and βr , where r is called the *transmission* radius and $\beta \geq 1$ is called the *quasi-disk* constant. In such a model, two nodes at a Euclidean distance of δ can directly communicate with each other if $\delta \leq r$, may or may not be able to communicate if $r < \delta \leq \beta r$, and can *not* communicate if $\beta r < \delta$.

¹If not, we can make appropriate number of copies of each node, and then, apply our presented techniques.

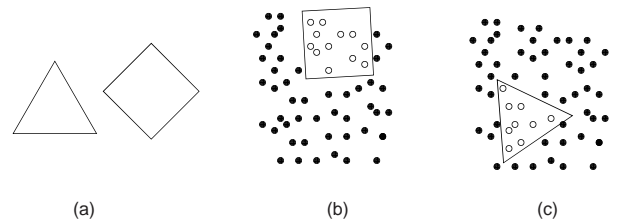


Figure 1: (a) Two given failure-shapes. (b) and (c): Network-failure due to each of the given failure-shapes; hollow nodes are the nodes destroyed.

In this article, we develop data replication schemes in uniformly random sensor networks. Our results in this article hold for any communication cost models that satisfy the following property in uniformly random networks: (i) The *expected* communication cost of sending a data packet between two nodes is $O(\delta/r)$, where δ is the Euclidean distance between them, and r is the transmission radius; (ii) The *expected* communication cost of multicasting a data packet in a region X from a node *inside* X is $O(N_X)$ where N_X is the total number of nodes in X . The above two properties certainly hold for dense networks; we conjecture (and plan to prove in our future work) that they hold in general in uniformly random networks, due to the unlikelihood of large ‘holes’ in a random network.

Failure-Shapes; Network-Failures. We define *failure-shape* as a closed 2D geometric shape, with an undetermined location and orientation. A *network-failure* is modeled as a failure-shape with determined location and orientation – which results in the failure of all nodes in the covered region. See Figure 1. We assume only one network-failure occurs at a time; however, our techniques for the second problem do generalize to multiple network-failures as discussed in Section 4. Below, we also define fatness of a given set of failure-shapes.

Notations. Throughout the article, we use the following notations.

- We use D and S to denote the set of data and storage nodes in the network, r to denote the uniform transmission radius, and β for the quasi-disk constant.
- We use $C(a, B)$ to denote the cost of multicasting a unit of data from the source a to a set B of destination nodes.
- We use \hat{R} to denote the radius of the smallest circle that can contain all the given failure-shapes, and \tilde{R} to denote the radius of the largest circle that can be contained in all the given failure-shapes.

DEFINITION 1. (Fatness) For a given set of failure-shapes \mathcal{F} , we define the fatness α of \mathcal{F} as the ratio \hat{R}/\tilde{R} , where \hat{R} and \tilde{R} are as defined above. \square

DEFINITION 2. (Regular Random Networks.) *Regular random networks* refers to connected sensor networks wherein the set of data nodes *and* storage nodes with transmission radius r have been independently and uniformly distributed in a square region of size $q \times q$, for some q , and the set of given failure-shapes is such that $\hat{R} < q/2$. We assume unit node density (i.e., the total number of nodes to be q^2); this is without loss of generality since the transmission radius is arbitrary. Also, unless mentioned otherwise,

we assume the number of data nodes to be less than storage nodes. \square

3. MCDR PROBLEM

In this section, we consider the scheme of simply replicating data to available storage nodes, to allow recovery from network-failures. In this context, we formally define an appropriately formulated optimization problem, viz., the MCDR problem, and present an approximation algorithm for it. We start with a definition.

DEFINITION 3. (Storing Set.) For a given data node d , a set $S(d)$ of storage nodes is considered a *storing set* if no network-failure (based on the given failure-shapes) can destroy the set of nodes $\{d\} \cup S(d)$. \square

A set of *disjoint* storing sets, one for each data node, can be used for data replication and failure recovery as follows. Each data node can replicate its data to each of the storage nodes in its storing set. By the above definition, such a replication strategy will ensure that no network-failure can destroy all the copies of a data item, and thus, all data items can always be successfully recovered in case of a network-failure. Note that we require the storing sets to be disjoint, since each storage node can only store one unit of data.

Minimum Cost Data Replication (MCDR). Given a network and a set of failure-shapes,² the MCDR problem is to 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, for each $d \in D$, we find a storing set $S(d) \subseteq S$ such that *solution's cost* $\sum_{d \in D} C(d, S(d))$ is minimum.

THEOREM 1. The MCDR problem is NP-hard to approximate within any finite approximation-ratio.

Proof: We first show that our MCDR problem is NP-hard by reducing the well-known 3D-matching (3DM) problem, which is known to be NP-complete [8], to the MCDR-decision problem. The *MCDR-decision* 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 $T' = (X \times Y \times Z) - T$. Below, we construct an instance of MCDR-decision problem from this 3DM instance.

Constructing an MDNR Instance. Consider a network consisting of data nodes X and storage nodes $(Y \cup Z)$. Note that for any arbitrary set of nodes F , a failure-shape can be

²Our designed algorithms only depend on \widehat{R} of the given set of failure-shapes; the \widetilde{R} of the set of failure-shapes is used only to prove the performance guarantees.

constructed such that it only³ covers F . Thus, below we give sets of nodes that can be destroyed by a network-failure, and the corresponding failure-shapes can be easily constructed. In particular, we create two types of network-failures:

- For each data node $x \in X$, we add two network-failures that contains the set of nodes $(x \cup Y)$ and $(x \cup Z)$ respectively.
- For each tuple in T' , we add a network-failure that contains only the nodes in T' .

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

Note that the first type of network-failures dictate that each storing set must contain a node from Y as well as Z . Since $|X| = |Y| = |Z|$ and an MCDR-decision solution must contain $|X|$ disjoint storing sets, each storing set in any MCDR-decision solution must contain *exactly* two storage nodes (one each from Y and Z). Further, for a data node $x \in X$ and its storing set (y, z) , $(x, y, z) \notin T'$ and hence $(x, y, z) \in T$. 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 NP-complete MCDR-decision problem. Thus, MCDR is NP-hard to approximate. \blacksquare

We now design an approximation algorithm for the MCDR problem in regular random networks. The approximation factor of our algorithm is in terms of the fatness of the given failure-shapes, and hence, doesn't contradict the above non-approximability result which involves use of arbitrary failure-shapes.

Survival Matching Algorithm (SMA). If we restrict the *size* of each storing set to one, then the MCDR problem easily reduces to the minimum-cost perfect matching problem in an appropriate bipartite graph (as shown below), and can be solved optimally in polynomial time. For the general MCDR problem, our approximation algorithm (called SMA) essentially restricts the size of storing sets to one and solves the restricted problem optimally. We will show that SMA solution's cost is within a constant-factor of that of the optimal unrestricted solution, in regular random networks.

SMA Description. Given a network, SMA starts with constructing a bipartite graph $G_r(D \cup S, E_r)$ between the data and storage nodes, wherein there is an edge (d, s) iff $\{s\}$ can be a storing set for d , i.e., if no single network-failure can destroy the set $\{d, s\}$. Thus, a matching of size $|D|$ in G_r would yield a set of disjoint storing sets of size one each, and hence, a solution (not necessarily optimal) to the MCDR problem. To incorporate communication costs, each edge (d, s) in G_r is assigned a weight equal to $C(d, \{s\})$, and we actually find a minimum-weighted matching of size $|D|$. The above minimum-weighted matching problem can be solved in $O((|D| + |S|)^3)$ time [6]. Note that SMA may not always

³We also need to place the network nodes in such a way that the constructed failure-shape doesn't cover any other non-singleton set of nodes. This can be easily done; we omit the details.

return a solution, since there may not be a matching of size $|D|$ in G_r ; however, as shown later, the probability of such an event is infinitesimally small in regular random networks.

Approximation Ratio of SMA. We now show that in regular random networks, SMA delivers a solution whose cost is within an α -factor of the optimal cost with high probability, where α is the fatness of the given set of failure-shapes. 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. The optimal cost of any MCDR solution is at least $|D|\tilde{R}/(2\beta r)$.

Proof: Consider a data node d and its storing set $S(d)$. In the set $(\{d\} \cup S(d))$, let a and b be the nodes with maximum Euclidean distance between them. Let $\delta(a, b)$ be the Euclidean distance between the nodes a and b . Now, $\delta(a, b)$ must be at least \tilde{R} , since otherwise all the nodes in $(\{d\} \cup S(d))$ could be contained in a circle of radius \tilde{R} and thus be destroyed by any network-failure (contradicting the definition of a storing set). Furthermore, it is easy to see that $C(d, S(d)) \geq \delta(a, b)/(2\beta r)$. Thus, we have $C(d, S(d)) \geq \tilde{R}/(2\beta r)$, and the total cost of any MCDR solution is at least $|D|\tilde{R}/(2\beta r)$. ■

The proof of the below lemma is rather tedious, and is deferred to Section 7.

LEMMA 2. In regular random networks, the SMA solution has an expected cost of $O(|D|\tilde{R}/r)$. □

The above two lemmas yield the following result.

THEOREM 2. In regular random networks, the SMA solution has an expected approximation-ratio of $O(\alpha)$, where α is the fatness of the given set of failure-shapes. □

In the above theorem, in computing the expectation, we have ignored the problem instances wherein SMA fails to return a solution even though one exists. Note that since the MCDR-decision problem is NP-complete (see Theorem 1), there can be no polynomial-time algorithm that always returns a solution if one exists, unless $P=NP$. However, it can be shown (see Corollary 1 in Section 7) that in regular random networks, SMA delivers a solution with a very high probability.

THEOREM 3. For regular random networks, SMA returns a solution with a very high probability (converges to 1 with increase in network size). □

Distributed Algorithm. To the best of our knowledge, there are no efficient distributed approximation algorithm known even for the well-studied minimum-cost matching problem, which is a special case of our MCDR problem. However, in the next section, we present an efficient distributed algorithm for the redundancy scheme based on erasure codes.

4. MCDC PROBLEM

An alternate way to introduce data redundancy is to use decentralized erasure codes as introduced in [7]. In this

scheme, each data node sends its data item to certain storage nodes, and each storage node creates and stores a linear-combination (with random coefficients) of all the data items it receives. For recovery in case of a failure, we retrieve data from a sufficient number of storage nodes and solve a system of linear equations. The main advantage of the above scheme is that it facilitates design of an efficient distributed algorithm, while the main disadvantage is that the data recovery requires the decoding cost of solving a linear system of equations.

Let us represent the above 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 routes its data to s . It can be shown [7] that for successful recovery from a failure, there must be a matching between the set of destroyed data nodes and the set of surviving storage nodes that covers all the destroyed data nodes [3, 4]. We now formulate our MCDC problem based on the above scheme as follows.

Minimum Cost Data Coding (MCDC). Given a network and a set of failure-shapes, the MCDC problem is to construct a bipartite graph $G_c(D \cup S, E_c)$ with minimum sum of edge-weights where:

- The weight of an edge (d, s) in E_c is the communication cost from d to s , and
- The set of edges E_c is such that for any failure, if F is a set of nodes destroyed by a failure, then the induced subgraph in G_c over $(D \cap F) \cup (S - F)$ has a matching of size $|D \cap F|$.

The sum of the edge-weights of a solution G_c is referred to as the *cost of the solution*. Note that above $(D \cap F)$ 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.

We omit the proof of the below theorem.

THEOREM 4. The MCDC problem is NP-hard. □

We now show that SMA of the previous section also yields an approximate solution for the MCDC problem. Then, we also design a distributed approximation algorithm, and prove its performance guarantees.

Using SMA for the MCDC Problem. Note that the output of SMA, viz., a set of disjoint storing sets, can be used to construct a solution G_c for the MCDC problem by connecting each data node to each storage node in its storing set. Now, similar to the arguments in Lemma 1, we can show that the optimal cost of of an MCDC problem is at least $|D|\tilde{R}/(2\beta r)$. Thus, by Lemma 2, we have the following approximation result.

THEOREM 5. In regular random networks, the MCDC solution delivered by SMA (as described above) has an expected⁴ approximation-ratio of $O(\alpha)$, where α is the fatness of the given failure-shapes. □

⁴As in Theorem 2, in computing the expectation, we ignore the instances wherein SMA fails to deliver a solution even though one exists. However, by virtue of Theorem 3, the probability of such an event is infinitesimally small.

Distributed Storage Algorithm (DSA). The main advantage of storing a linear-combination of data items (as in the MCDC problem) over simple replication (as in the MCDR problem) is that the data nodes don't have to globally compete for exclusive use of storage nodes and thus, the decisions of where to store the data can be made locally. This advantage facilitates design of an efficient distributed algorithm for the MCDC problem. Below, we present a distributed algorithm that guarantees recovery of data with high probability in regular random networks.

DSA Description. Consider a regular random network. Without loss of generality, let us assume the density of nodes 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 $2\hat{R}$ below, as shown in Figure 2. Here, ϕ is a constant (see Equation 1) which depends on $|D|/|S|$ and the desired probability of successful recovery. More formally, if (x, y) are the coordinates of the data node, then its storage region is a rectangle with the left-most top coordinate equal to $(x - 1, y - 2\hat{R})$ and has a length of 2 and a height of ϕ .

The bipartite graph $G_c(D \cup S, E_c)$ 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 (with random coefficients) of all the data items received from various data nodes.

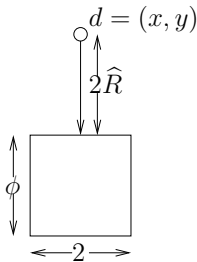


Figure 2: Storage region of a data node in DSA.

THEOREM 6. Given a regular random network with an arbitrary ratio of $|D|$ and $|S|$, the solution returned by DSA allows successful data recovery from a network-failure with a probability of $(1 - \epsilon)$ (for any $0 < \epsilon < 1$), if the value of ϕ is chosen as:

$$\phi = \max((\kappa + 1)c^2 + (\kappa - 1)\hat{R}, c^2(1 + \kappa)), \quad (1)$$

where $\kappa = |D|/|S|$, and c is the smallest real number such that $g(c)$, the Gaussian error function, is greater than $(1 - \epsilon)$. Note that for $|D| < |S|$, the above equation simplifies to $\phi = c^2(1 + \kappa)$. \square

The proof for the above theorem is quite involved and hence, is deferred to Section 8. We now prove the approximation-ratio of DSA.

THEOREM 7. For regular 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 of an MCDC solution is $|D|\hat{R}/(2\beta r)$. Below, we show that the expected cost of DSA's solution is $O(|D|\hat{R}/r)$, proving the theorem.

In DSA, a data node multicasts its data item to a rectangular region of size 2ϕ located at a vertical distance of $2\hat{R}$. The communication cost incurred by each data node is $O(2\hat{R}/r)$ to reach the storage region and $O(\phi)$ to broadcast in the storage region, based on the properties of our communication cost model. Thus, the total expected cost of the DSA solution is $O((|D|\hat{R}/r) + \phi)$ which is $O(|D|\hat{R}/r)$ for a given ϵ , since $|D| < |S|$ for regular random networks. \blacksquare

Multiple Network-Failures. To handle k simultaneous network-failures, we define k storage regions for each node where the i^{th} storage region is a rectangle of size $2 \times \phi$ at a vertical distance of $2\hat{R} + (i - 1)(2\hat{R} + \phi)$. The approximation proofs of DSA can be easily generalized for the above (we omit the details).

5. 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 [1] where the authors show that a simple distributed scheme using network coding can perform as well as the scheme that uses complete coordination between nodes.

Dimakis et al. [7] and Lin et al. [14] used distributed random linear codes for data storage in sensor networks, to ensure data recovery upon arbitrary node failures. These algorithms do not require any global knowledge besides the number of nodes in the network. Lin et al. in [13] and Aly et al. [2] used random walk to construct fountain codes for the same purpose and do not even assume the knowledge of the network size. With both spatial and temporal dimensions, Kamra et al. [11] considered the growth codes, for which the codewords change over time, for the data propagated in the network. This is to ensure that the sink is able to retrieve as much data as possible for sensors deployed in an extremely hostile environment.

All of the previous works on this topic use a probabilistic node failure model without any spatial-pattern for node failures. Furthermore, none of the previous works have tried to address the objective of minimizing communication cost incurred in introducing the required data redundancy. To the best of our knowledge, ours is the first work that addresses the problem of data preservation in case of spatial failures in a sensor network, while minimizing the communication cost in introducing data redundancy. Although any of previous methods can be used to design a failure-tolerant network, the presence of spatial patterns for node failures allows us to design an efficient distributed scheme with significantly less communication cost and required redundancy.

6. SIMULATIONS

In this section, we present our simulation results. We compare our algorithms with the algorithm in [7], referred to as DPR here, in terms of the total communication cost under circular failure-shapes of varying radii. We observe that (i) our algorithms incur much less communication cost than DPR [7], and (ii) for $|D| < |S|$, our algorithms perform within a factor of the lower bound on optimal cost. We start with a brief description of our implementation of DPR. In

DSA, we use Geocast [15] for multicasting in the storage region.

DPR Algorithm [7]. In DPR, each data node sends its data item to randomly chosen $\log |D|$ storage nodes. In our implementation of DPR, we compute the communication cost by computing the 2-approximation of minimum-size Steiner tree connecting the data node and the randomly chosen storage nodes in the network’s communication graph; the communication cost incurred is the size of the computed tree. Note that the communication cost incurred in our implementation of DPR is essentially a *lower bound* on the communication cost of any DPR’s implementation; we chose the above implementation to make our performance comparison independent of the communication cost model.

Comparison of Communication Costs. We conduct three sets of experiments to compare communication costs of various algorithms: (i) Varying radius of the failure-shape, (ii) Varying the network size, and (iii) Varying the ratio of number of data nodes to storage nodes. In all the below experiments, we use nodes with a uniform transmission radius of 2.5 units (the minimum required to ensure connectivity).

Varying Failure-Shape Radius. For the first experiment (see Figure 3(a)), we use a relatively small network of size 10,000 nodes uniformly distributed in a region of size 100×100 units. We use 20% data nodes, and vary the radius of the circular failure-shape from 5 to 15 units. We compare the communication cost of four algorithms, viz., DPR, SMA, and DSA for two values of $dpsr$ (desired probability of successful recovery) viz. 98.93% and 99.999998% (corresponding to $c = 3$ and $c = 6$ respectively). We also plot the lower bound $|D|\tilde{R}/(2r)$ on the optimal cost. We observe that the communication cost of SMA is very close to the lower bound, and SMA and DSA both outperform DPR by a large margin. Note that DPR does not adjust to the radius of failure-shape, and hence, its plot is a straight line. For the above setting, all three algorithms have a theoretical $dpsr$ more than 99.99%, however, in our experiments, all algorithms were always able to successfully recover data.

Varying Network Size. In Figure 3(b), we vary the network size from 10,000 to 1M nodes in a square region of proportional size to maintain a constant density of 1 node per unit area. As before, we use 20% data nodes. We vary the radius of the failure-shape in proportion to the size of the region, i.e., we use the radius of the failure-shape as $l/10$ where l is the length of the square region. In this (and next) set of experiments, we could not run SMA due to the large size of networks; note that SMA’s time-complexity is cubic in network size. We observe that our DSA again outperforms DPR by an order of magnitude while closely following the lower bound, which demonstrates the scalability of our distributed algorithm DSA.

Varying $|D|/|S|$. In Figure 3(c), we vary the ratio $|D|/|S|$. Here, we use a network of 250,000 nodes distributed uniformly and randomly in a region of size 500×500 . We fix the radius of the failure-shape to be 50. Note that both SMA and DPR require the ratio $|D|/|S|$ to be less than 0.5. We do not plot SMA in Figure 3(c) due to the large size of the network, but plot DPR for lower values of $|D|/|S|$. In the graph, we observe that the communication cost of DSA increases quadratically. This is expected, since the total cost of DSA is proportional to $|D|\phi$, and along the x-axis, D as

well as $|D|/|S|$ (which causes a proportional increase in ϕ) increase linearly.

Varying $dpsr$ for DSA. In this last set of experiments, we vary the value of $dpsr$, and observe its effect on the total communication cost incurred in DSA. Note that for increasing $dpsr$, the value of ϕ (and thus, the storage region’s area) increases, which causes an increase in the communication cost. Figure 3(d) shows how the communication cost increases with increase in $dpsr$. We use a uniformly random network of size 250,000 nodes in a region of 500×500 , wherein 20% nodes are data nodes, the radius of the failure-shape is 50. From the graph, we observe that for values of $dpsr$ up to 99.98%, the communication cost shows only a slow increase, but increases drastically with further increase in $dpsr$ value.

7. PROOF OF LEMMA 2

Proof of Lemma 2. Recall that the MCDR problem is to find disjoint storing sets of unrestricted size. In contrast, SMA finds singleton storing sets (i.e., a matching of size $|D|$) which it can compute 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 in uniformly random networks, there exists a $|D|$ -size matching of cost $O(|D|\hat{R}/r)$ with a high probability.

We show the existence of such a matching by introducing an algorithm (called CMA) that finds a matching of expected cost $O(|D|\hat{R}/r)$ with a high probability for large networks. Note that unlike SMA, CMA doesn’t always find a matching when a matching exist. However, the number of such instances converges to 0 with the in size of the network, and the cost of matching in these instances is bounded by the network size. Thus, the expected cost of SMA is bounded by that of CMA.

Cell Matching Algorithm (CMA). CMA works by creating a grid in the network with unit square cells. Then, at each stage, if there exist an unmatched data node, CMA tries to find a matching-node for it in the cell assigned to it. At each stage, the size of the cells and the cost of matching increase, but the number of nodes that needs to be matched decrease with a faster rate – yielding the desired expected cost.

As mentioned above, 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 with some storage node.

- If all the data nodes in a cell are already matched (in a previous step), the cell is considered *complete*. Remaining cells are considered *incomplete* in this stage. In the initial stage, all cells are incomplete.
- For each incomplete cell, we assign the cell below at a vertical distance of $2\hat{R}$ to be its *storage cell*.
- For each yet-unmatched data node in an incomplete cell C , find an unmatched storage node in C ’s storage cell.
- Merge pairs of all (complete as well as incomplete) horizontally-adjointing cells to construct new cells of double the width (but the height remains unit).

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

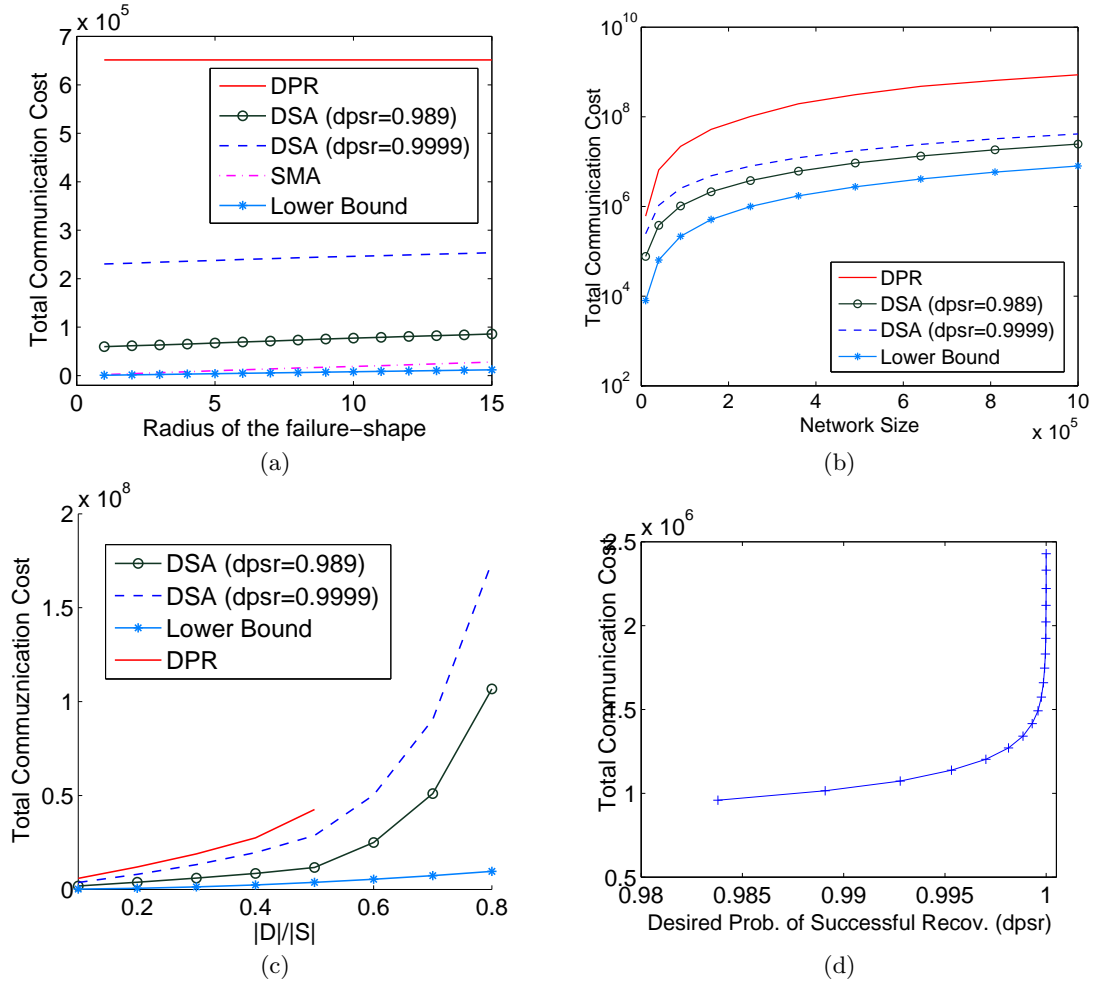


Figure 3: Total communication cost of applicable algorithms for increasing (a) failure-shape radius, (b) network size, (c) $|D|/|S|$, and (d) $dpsr$ (for DSA), values.

See Figure 4 for a brief illustration of CMA. In the first step, cell 3 is the storage cell of cell 1. In the second step, cells 1 and 2 are merged to form a cell (1,2). Similarly, cells 3 and 4 are merged to get a cell (3,4). In the next step, the storage cell of (1, 2) is (3, 4).

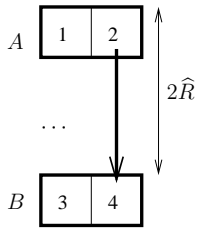


Figure 4: Cell Matching Algorithm (CMA).

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 (for the first step, we use $i = 0$), we have the following.

- The expected number of nodes in each cell is 2^i , since the size of a cell is $2^i \times 1$.
- The maximum communication cost⁵ between two nodes that can be matched is

$$O\left(\frac{1}{r}\sqrt{\widehat{R}^2 + 2^{2i}}\right) = O(2^i \widehat{R}/r).$$

To bound the cost incurred in matching 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 $O(2^{2i} \widehat{R}/r)$ in the i^{th} step.

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

⁵Here, we have used the assumption from Section 2 that the communication cost between two nodes is $O(d/r)$, where d is the Euclidean distance between them.

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

Finally, from Lemma 3 (see below), we get that the expected number of incomplete cells in the i^{th} step is ((Number of Cells) $\times O(e^{-2^{(i-1)}})$). Now, since the number of non-empty cells (i.e., cells containing some data nodes) is bounded by $|D|$ at each step, we get the overall cost of CMA solution as:

$$\sum_i |D|O(e^{-2^{(i-1)}})O(2^{2i}\widehat{R}/r) = O(|D|\widehat{R}/r).$$

■

LEMMA 3. In CMA (as described above), the probability of any particular cell being incomplete in the i^{th} step is $O(e^{-2^{(i-1)}})$.

Proof: For a cell to be incomplete in step i , it should contain an unmatched data node. This means that the number of data nodes in at least one of its two constructing cell at step $i-1$ should be more than the number of the storage node in its storage area. For step i , let U_i be the difference between number of data nodes of a cell and the number of storage nodes in its storing cell. The probability of a cell being incomplete at step i is at most twice the probability of $U_{i-1} < 0$. Below, we compute the *probability distribution function (pdf)* of U_i and show that it decreases with a super-exponential rate with increase in i . To be more precise, we show that the probability of $U_i < 0$ is $O(e^{-2^i})$ which would imply the probability of a cell being incomplete at step i to be $O(e^{-2^{(i-1)}})$.

Computing pdf of U_i . For a rectangle X in the network, let D_X and S_X denote the number of data and storage nodes respectively in X . In a regular random network, both D_X and S_X are independent random variables. By computing the pdf of D_X and S_X , we can compute the pdf of the number of data and storage nodes (denoted as D_i and S_i respectively) in a cell at step i of CMA. We can compute the pdf of $U_i (= S_i - D_i)$.

Computing pdf of D_X and S_X . For a randomly selected rectangle X of size $l \times w$, D_X and S_X are random variables with a binomial distribution of number of experiments equal to $|D|$ and $|S|$ respectively and a success probability of $lw/(|D|+|S|)$, since the total area of the unit-density network is $(|D|+|S|)$. Since both $|D|$ and $|S|$ are large numbers, we can use the normal approximation [9] for the above binomial distributions. Thus, using $k = |D|$ and $n = |S|$ for clarity, we get:

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

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

where μ is the mean and σ is the variance of the distribution.

Probability distribution of U_i . In step i of our algorithm,

each cell has a size of $(l = 1, w = 2^i)$. We have,

$$D_i \sim \mathcal{N} \left(\begin{array}{l} \mu = \frac{k2^i}{(n+k)}, \\ \sigma^2 = \frac{k2^i}{(n+k)} \left(1 - \frac{2^i}{(n+k)}\right) \end{array} \right)$$

$$S_i \sim \mathcal{N} \left(\begin{array}{l} \mu = \frac{n2^i}{(n+k)}, \\ \sigma^2 = \frac{n2^i}{(n+k)} \left(1 - \frac{2^i}{(n+k)}\right) \end{array} \right)$$

And the probability distribution of $U_i = S_i - D_i$ is

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

Computing $\Pr(U_i < 0)$. Since we want to compute an upper-bound for the probability of $U_i < 0$ we can replace the variance with a higher value. This allow us to omit the term $1 - \frac{2^i}{(n+k)}$. We get,

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

Base on normal distribution properties [9] the probability of $U_i < 0$ is $O(1 - g(c_i))$ where $g(x)$ is the Gaussian error function and

$$c_i = \frac{\mu}{\sigma} = \frac{n-k}{n+k} 2^{i/2}.$$

We have $(1 - g(c_i)) < e^{-c_i^2}$ [9]. Thus, the $\Pr(U_i < 0) = O(e^{-c_i^2}) = O(e^{-2^i})$, for a fixed k/n . ■

LEMMA 4. CMA (as described above) finds a matching with a very high probability, which converges to 1 for large networks.

Proof: For a network of size $q \times q$, CMA performs $\log q$ steps. By Lemma 3, the probability of a cell containing unmatched data nodes at the end of the final step is $O(e^{-2^{(q-1)}}) = O(e^{-q})$. Since the total number of cells at the final step is q , the probability of existence of an incomplete cell at the end of the final step is at most $O(q \cdot e^{-q})$ which converges to 0 as q goes to infinity. Thus, the probability of CMA *not* finding a matching converges to zero for large networks. ■

COROLLARY 1. For large regular random networks, the MCDR problem has a solution (and hence, SMA delivers one) with a very high probability (converges to 1 with increase in network size). □

8. PROOF OF THEOREM 6

Notations $D_X, S_X, N(\delta)$. We use D_X and S_X to denote the number of data and storage nodes respectively in X , where X is a subregion in the network or a set of nodes. For a set of

data nodes δ , we use $N(\delta)$ to denote the set of storage nodes that lie in the storage region of some data node in δ .

Proof of Theorem 6. As discussed in Section 4, for successful recovery of data when a network-failure F occurs, the induced subgraph of G_c 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 network-failure F to be a square region of size $2\widehat{R} \times 2\widehat{R}$. 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 $2R$ away. Thus, it suffices to show that with a probability of $(1 - \varepsilon)$, there is a matching of size $|D_F|$ in the induced subgraph in G_c 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 [5], 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 5 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 6, we show that the probability of such a rectangle L existing is less than ε . Intuitively, this is true due to size of L being much smaller in comparison to the union of the storage regions of the data nodes in L .

We now prove the two lemmas used above. ■

LEMMA 5. If there exists a set $\delta \subset D_F$ such that $|\delta| > |N(\delta)|$, then there is a rectangular region L in the network-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 that 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 5, 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 .⁶ We claim the following.

⁶For 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} .

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. Storage region of each data node in Z_1 is contained in Z_3 , since storage region of a data nodes is at a vertical distance of $2\widehat{R} + \phi$.
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 $|\delta'| > |N(\delta')|$. This yields a contradiction to our original premise. Thus, no such storage node z exists. Hence, $N(D_{B_\delta}) \subseteq N(\delta)$ and $N(D_{B_\delta}) = N(\delta)$. ■

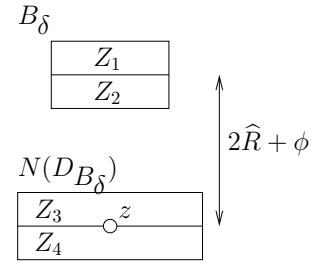


Figure 5: Partitioning of rectangles B_δ and $N(D_{B_\delta})$ based on z .

LEMMA 6. For a given network-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: 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 $(l + 2) \times (\phi + w)$. Let U be the random variable $U = S_Y - D_X$. We will show that the probability of $U < 0$ is less than ε , proving the lemma.

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 pdf 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 pdf for $U = S_Y - D_X$, using the pdf expressions for D_X and S_Y from Lemma 3. Recall that Y is of size $(l + 2) \times (\phi + w)$.

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

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 mean of U is greater than zero and we want to compute the upper-bound the probability of $U < 0$. Thus, we can consider a higher value for σ and a lower value for μ . Thus, we omit the terms $(1 - \frac{lw}{(n+k)})$ and $(1 - \frac{(l+2)(w+\phi)}{(n+k)})$ in σ^2 . We also can use $(l+2)$ instead of l in both μ and σ^2 .
- We can multiply both the mean and standard deviation with a positive number without changing the probability. So, we multiple both by $(1/(l+2))$.

Applying the above simplifications, we get:

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

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 $\mu > c\sigma$ where $c = g(1-\varepsilon)$ and $g(x)$ is the Gaussian error function [9]. Since both mean and standard deviation of U are positive numbers, we may rewrite the inequality as $\mu^2 > (c\sigma)^2$. Thus, the following equation ensures the desired upper bound on $\Pr(U < 0)$

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

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

$$\begin{aligned} \phi &= (1/n) \max((n+k)c^2 + (k-n)R, c^2(n+k)) \\ &= \max((1+\kappa)c^2 + (\kappa-1)R, c^2(1+\kappa)) \end{aligned}$$

where $\kappa = |D|/|S| = k/n$. ■

9. CONCLUSIONS

In this paper, we addressed the problem of introducing data redundancy in sensor networks with minimum communication cost so as to survive node failures. We presented centralized and distributed approximation algorithms for two redundancy schemes, in uniformly random networks. Our simulation results show that our schemes incur near-optimal communication cost, and easily outperform the previous approach. To the best of our knowledge, ours is the first work that addresses the data preservation problem in the context of spatial failures, while minimizing the communication cost incurred in introducing data redundancy. It would be interesting to extend our approaches to exploit spatial and temporal data correlations in conjunction with an appropriate data expiry mechanism. We plan to address this in our future work.

10. REFERENCES

- [1] ACEDANSKI, S., DEB, S., MEDARD, M., AND KOETTER, R. How good is random linear coding based distributed networked storage. In *Proceedings of the IEEE International Symposium on Network Coding (NetCod)* (2005).
- [2] ALY, S. A., KONG, Z., AND SOLJANIN, E. Fountain codes based distributed storage algorithms for large-scale wireless sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)* (2008).
- [3] BOLLOBAS, B. *Modern graph theory*. Springer, 1998.
- [4] BOLLOBAS, B. *Random graphs*. Cambridge University Press, 2001.
- [5] BONDY, J., AND MURTY, U. *Graph theory*. Springer, 2007.
- [6] COOK, W., AND ROHE, A. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing (JOC)* (1999).
- [7] DIMAKIS, A., PRABHAKARAN, V., AND RAMCHANDRAN, K. Decentralized erasure codes for distributed networked storage. *IEEE/ACM Transactions on Networking* (2006).
- [8] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
- [9] IBE, O. C. *Fundamentals of applied probability and random processes*. Academic Press, 2005.
- [10] KAHN, J., KATZ, R., AND PISTER, K. Next century challenges: Mobile networking for smart dust. In *Proceedings of The International Conference on Mobile Computing and Networking (MobiCom)* (1999).
- [11] KAMRA, A., MISRA, V., FELDMAN, J., AND RUBENSTEIN, D. Growth codes: maximizing sensor network data persistence. *SIGCOMM Comput. Commun. Rev.* 36, 4 (2006).
- [12] KUHN, F., WATTENHOFER, R., AND ZOLLINGER, A. Ad hoc networks beyond unit disk graphs. *Wireless Networks* 14, 5 (2008).
- [13] LIN, Y., LIANG, B., AND LI, B. Data persistence in large-scale sensor networks with decentralized fountain codes. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)* (2007).
- [14] LIN, Y., LIANG, B., AND LI, B. Geometric random linear codes in sensor networks. In *Proceedings of the International Communications Conference (ICC)* (2008).
- [15] NAVAS, J. C., AND IMIELINSKI, T. Geocast: geographic addressing and routing. In *Proceedings of The International Conference on Mobile Computing and Networking (MobiCom)* (1997).