Achieving Data *K*-Availability in Intermittently Connected Sensor Networks

Bin Tang[1], Neeraj Jaggi[2], Masaaki Takahashi[3]
[1]Department of Computer Science, California State University Dominguez Hills, USA
[2]Department of Electrical Engineering, Columbia University, USA
[3]Department of Electrical Engineering and Computer Science, Wichita State University, USA
Email: btang@csudh.edu, neeraj_jaggi@yahoo.com, aki_28_79@hotmail.com

***Abstract*— We consider the problem of preserving data in *intermittently connected sensor networks* wherein sensor nodes do not always have connected paths to the base stations. The generated data is first stored inside the network before being uploaded to the base station when uploading opportunities arise. Each node has both limited energy level and limited storage space, and is associated with a probability of failure. To guarantee that at any moment, each generated data item is available for being uploaded in the presence of node failure, we propose to replicate K copies of each data item in the network, where K depends upon the node failure probability. We refer to the problem as *Data K-Availability Problem (DKAP)*. *DKAP* is naturally divided into two phases: K-Availability creation and K-Availability maintenance. For K-Availability creation, we show that the problem is NP-hard for arbitrary data sizes, and that it is equivalent to minimum cost flow problem for unit data sizes. For K-Availability maintenance, we show that it is NP-hard even for unit data sizes and design a centralized greedy heuristic. We further design an efficient and low-overhead distributed algorithm, which is applicable to both phases, and show using extensive simulations that it performs close to the heuristic.**

Keywords – Data Availability, Data Replication and (Re)distribution, Intermittently Connected Sensor Networks

I. Background and Motivation

Many of the modern sensor networks applications are deployed in inaccessible and unattended regions, collecting sensory data for a long period of time. They include underwater sensor networks [32], ocean seismic sensor networks [15, 26], and volcanic and glacial monitoring sensor networks [16, 33]. In such environment, since it is not feasible to have a long-term deployment of high power data collection base station (with power outlets) in the field, data generated is first stored inside the network and then uploaded to faraway base stations via different means and opportunities, such as data mules or satellite links [11, 17]. We refer to such sensor networks as *intermittently connected sensor networks*, since the data uploading is possible only when the uploading opportunities become available. Between two uploading opportunities, the generated data needs to be stored inside the network.

There are three main factors contributing to data loss for the stored data in intermittently connected sensor networks: node energy depletion (results in loss of stored data), node storage depletion (nodes can not store newly generated data), and node hardware failure (results in loss of stored data). Overcoming the obstacle of data loss and preserving data until next uploading opportunity is therefore an important problem. In our previous research, we have addressed storage and energy depletion induced data loss [10, 27, 30, 31, 34]. However, unlike storage and energy depletion, sensor node hardware failure is unpredictable and thus can not be alleviated directly using similar techniques.

One way to ensure against data loss due to hardware failure is to maintain multiple copies of each generated data item in the sensor network, such that at any moment, it is with high probability that at least one copy of each data item is available for uploading when uploading opportunity arises. In this paper, we aim to create K copies of each data item and maintain them for maximum amount of time, under the constraints that each node has limited storage space and battery energy. We refer to this problem as *Data K-Availability Problem (DKAP)*, where K depends on the node failure probability.

DKAP is naturally divided into two phases:

- **K-Availability Creation.** In this phase, each source sensor node, which has the original copy of each data item it initially generated, replicates $(K-1)$ additional copies of each data item and $distributes$ the replica copies into the network. The objective of this phase is to create and distribute the desired data redundancy with minimum total energy consumption, while abiding by the storage constraint at each node. At the end of the this phase, each data item has K copies stored in the network, with no two copies of the same data item stored at the same node. Since each node's energy is being constantly drained, all the sensor nodes will exhaust their energy after some time. However, some sensors would get drained out earlier than others due to the differences in the initial energy levels of the sensors, and due to different levels of participation in K-availability Creation phase. This calls for the second phase below.
- **K-Availability Maintenance.** When the energy levels of sensor nodes are different and energy drains continuously at all sensors, there is a need to *redistribute* data replicas from nodes with low energy level to nodes with high energy level, so that the loss of any replica copy of any data item can be delayed. In particular, the goal of this phase is to maximize the minimum remaining energy of all the nodes storing any copies of data items (including replica copies and original copies), post redistribution. In this phase it also needs to guarantee that no two replica copies of each data item be stored at the same sensor node.

Specifically, we formulate both problems as graph-theoretic problems. We show that K-Availability creation is NP-hard for arbitrary data sizes, and is equivalent to minimum cost flow problem [1, 20] for unit data sizes. For K-Availability maintenance, we show it is an NP-hard problem even for unit data sizes and design a centralized greedy redistribution algorithm. The centralized nature of algorithms in both phases makes them unsuitable for distributed sensor network deployment. Therefore, we design an efficient and low-overhead distributed algorithm, which works for both phases and performs close to the optimal. In this paper, we have focused more towards the modeling and formulation of the problems, and towards proposing graph theoretic solution approaches. This paper makes some simplistic assumptions with respect to energy and data storage models in order to keep the solutions tractable. In real world scenarios, where these assumptions may not hold, we believe that the proposed solutions can be easily extended to achieve near optimal performance.

Paper Organization. The rest of the paper is organized as follows. Section II gives an overview of the related literature. In Section III, we introduce the problem models including energy model and cost model, and discuss availability constraint and choice of K. Section IV formalizes the K-Availability creation phase and designs data replication and distribution techniques. Section V proposes data redistribution techniques for the K-Availability maintenance. Section VI presents our distributed algorithm. In Section VII, we compare all the proposed algorithms, discuss the results in details, and present useful insights. Section VIII concludes the paper with a discussion on possible directions for future research.

II. Related Work

Related Work. Replication has long been a key approach to achieving data availability in World Wide Web [4], peer-to-peer networks [22], data grids [14, 23], and distributed system environments [19, 36]. In all the above, availability is defined as the number of accepted user accesses over total submitted accesses. Data redistribution has been studied extensively in the field of parallel computing [21, 25] and disk storage [13]. It mainly studies how to schedule workload and move associated data from source processors to destination processors, or change one storage configuration into another, to better respond to the data demand changes for the purpose of load balancing of data access. *DKAP* does not consider data *access*, but addresses creating and maintaining data *availability*, in the presence of hardware failures at sensor nodes.

Data replication and (re)distribution have been actively researched in ad hoc and sensor networks. However, most of the techniques focus on reducing the access cost [8, 28, 29, 35] or battery consumption [24], and are therefore not suitable towards achieving K-Availability. Only line of research that focuses on enhancing data availability in sensor networks is called data redistribution or data preservation [10, 27, 30, 31, 34], which address how to preserve data inside sensor networks by tackling either storage- or energy-depletion induced data loss. Tang et al. [30, 31] study energy-efficient data redistribution in sensor networks wherein data is moved from nodes with highly loaded storage space to nodes with surplus storage, while minimizing the total energy consumption. It assumes that each sensor node has infinite amount of initial energy. Takahashi et al. [27] study the data preservation problem in the intermittently connected sensor networks under energy constraints at sensor nodes, without considering storage-depletion. Hou et al. [10] study how to maximize the minimum remaining energy of the nodes that finally store the data, by considering both storage- and energy-depletion of sensor nodes. Xue et al. [34] take a step further, and consider that sensory data from different source nodes have different importance, and study how to preserve data with highest importance. However, non of the work specifically address sensor node hardware failure by replicating data.

The other line of research that focuses on enhancing data availability in sensor networks is called data persistence[12], wherein various network coding techniques are introduced to provide reliable data access in the event of node failure. Our focus is mainly on the hardness and optimality of the problem solutions that provide reliable data access in face of node failure, using data replication technique. To the best of our knowledge, our work is the first to study the hardness and optimality of data availability in sensor networks. Besides, in most of the data persistence research, it is assumed that the base station is still available and the goal is to increase the "persistence" of sensed data, so that data is more likely to reach a data sink when nodes fail. However, in intermittently connected sensor networks studied in this paper, the lack of base station and the lack of frequent data uploading opportunities render storing data and maintaining its availability inside the network imperative.

A recent work that specifically targets data availability in sensor networks is by Montanari et al. [18]. It assumes that each node has some probability of failure due to energy depletion therefore it adaptively creates and maintains a number of replicas of the data. The goal is to minimize number of messages needed to create such replica while guaranteeing that at least one replica copy of each data exists. They solve the problem with a greedy heuristic that uses only information gathered from neighbors. The goal of our work is different. We recognize that different nodes could have different battery energy and that nodes are draining their battery continuously, therefore it is preferred that data and its replica are stored at nodes with higher battery energy, to survive for a longer period of time.

III. Problem Models

Problem Models. We model a sensor network as a general graph $G(V, E)$ where $V = \{1, 2, ..., N\}$ is a set of N sensor nodes distributed uniformly at random in the region, and E is a set of edges. There exist an edge between two nodes if they can communicate directly with each other. Each sensor node i has a finite and unreplenishable initial energy E_i. We adopt the first order radio model [9], which states the following. To send a k-bit data from node i to its neighboring node j over their distance d_{ij}, the sending energy (also called transmission energy) by node i is $E_T(k, i, j) = E_{elec} * k + \epsilon_{amp} * k *$

d_{ij}^2, the receiving energy by node j is $E_R(k) = E_{elec} * k$. Here $E_{elec} = 100nJ/bit$ is the energy consumption per bit on the transmitter circuit and receiver circuit, and $E_{amp} = 100pJ/bit/m^2$ calculates the energy consumption per bit on the transmit amplifier. Note that when d_{ij} is small, the energy spent at sender and receiver nodes is almost equal. This is because, ϵ_{amp} is much smaller than E_{elec}. Thus, for simplicity we can assume that to send a data item from a node to its neighboring node costs 1 unit of energy with 0.5 units spent by each sender and receiver. We use this simplistic model for illustrative purposes later in this paper.

Let c_{ij} denote the minimum energy cost of sending k-bit data from node i to node j, where i and j are not necessarily one-hop neighbors from each other. We adopt unicast network routing model, wherein for a one-hop communication, only the sender and receiver nodes cost energy, and sender's other neighbors do not cost any energy. We also assume that there exists a contention-free MAC protocol (e.g. [2]) that provides channel access to the nodes. In this paper, we assume that all the sensor nodes are awake all the time monitoring while waiting for the data uploading opportunities, draining their battery energy constantly. Duty-cycling, in which individual sensor node activates very briefly for sensing and communication and stays in dormant state for a long period of time, is beyond the scope of this paper.

There are p distinct data items $D = \{D_1, D_2, ..., D_p\}$ generated in the network. D_j has size of s_j units (each unit is of k bits) and is stored in its *source node* $S(j) \in V$ (a source node could have multiple data items). Let m_i be the number of units (each unit is of k bits) of initial free storage for node i, before it generates and stores any data items. For sensor node i that is also a source node, the available space after storing its generated data items is thus $m_i - \sum_{j=1}^{p}\{s_j | S(j) = i\}$.

Availability Constraint. Two requirements must be satisfied in both K-Availability creation and maintenance. First, K copies of each data item should be stored in the network. Second, multiple replica copies of the same data item can not be stored at the same sensor node. These two requirements are collectively referred to as *availability constraint*. Therefore, each node (including the source node) is allowed to store at most p distinct data items, even though its storage capacity could possibly be larger than the total size of the p data items. We therefore define *effective storage capacity* as follows.

Definition 1: (Effective Storage Capacity.) The *effective storage capacity* of node i, denoted as $m_i^{'}$, is the maximum storage capacity of i that can be used to store data items. It is the minimum of its initial storage m_i and the total size of the p data items: $m_i^{'} = \min\{m_i, \sum_{j=1}^{p} s_j\}$. □

Choice of K. Next let's decide the number of replicas of each data item, K. Assume that the independent node failure probability of each node is p_f. Given that K copies of a particular data item are maintained at different nodes in the network, let us denote the number of available copies of the data item at an arbitrary time instant as C. We have,

$$P[C = 0] = p_f^K, \quad \text{and} \quad P[C = K] = (1 - p_f)^K. \tag{1}$$

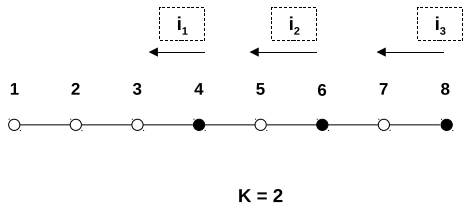

Fig. 1. An example of 2-Availability creation in a linear network of 8 nodes, where 4, 6, and 8 are source nodes, each has one data item. The storage capacity of each node is 1.

Also,

$$P[C = i] = \binom{K}{i}(1 - p_f)^i p_f^{(K-i)}, \quad \forall i \in \{0 \dots K\}. \tag{2}$$

The expected number of available copies is given by

$$E[C] = \sum_{i=0}^{K} i \cdot \binom{K}{i}(1 - p_f)^i p_f^{(K-i)} = K\,(1 - p_f)\,. \tag{3}$$

In general, it would be desirable to have at least one copy available in the network. Therefore, we need,

$$K\,(1 - p_f) \geq 1, \quad \text{or} \quad K \geq \frac{1}{1 - p_f}.$$

Thus, K would be chosen adaptively to satisfy the above condition, given the failure probability p_f. For the rest of the paper, K is fixed and equals $1/(1 - p_f)$.

IV. K-Availability Creation

Problem Formulation. We define *distribution function* $r : D \times \mathcal{K} \rightarrow V$, indicating that the k^{th} replica copy of data item $D_j \in D$, where $1 \leq j \leq p$ and $k \in \mathcal{K} = \{1, 2, ..., K\}$, is distributed from $S(j)$ to $r(j,k) \in V$ via the shortest path between them. Here we assume that the original copy of D_j in $S(j)$ is its first replica copy and it is not distributed, i.e. $r(j,1) = S(j)$. The data replicas from the same node are distributed one by one in a unicast fashion. For one data item, its *distribution cost* is equal to the sum of the minimum energy cost of distributing each of its K replica copies into the network. The K-Availability creation problem is to find such a distribution function r to minimize the *total distribution cost*, which is the sum of the distribution costs of all the data items in the network:

$$\sum_{j=1}^{p}\sum_{k=1}^{K} s_j \times c_{S(j)r(j,k)}, \tag{4}$$

under the storage capacity constraint of each node: $\sum_{j=1}^{p}\{s_j | r(j,k) = i\} \leq m_i^{'}, \forall i \in V, 1 \leq k \leq K$, and the availability constraint: $r(j,k) \neq r(j,k^{'}), 1 \leq j \leq p, k \neq k^{'}$.

EXAMPLE 1: Fig. 1 illustrates the K-Availability creation problem in a linear sensor network of 8 nodes, where $K = 2$. There are three source nodes: 4, 6, 8, initially having data item i_1, i_2, and i_3 respectively. The storage capacity of each node is one. The minimum cost solution for 2-Availability creation is given by: node 4 distributes a replica of i_1 to node 3, node 6 distributes a replica of i_2 to node 5, while node 8 distributes a replica of i_3 to node 7. Assuming sending

or receiving a data item each costs 0.5 energy, the total distribution cost is 3. □

Theorem 1: The K-Availability creation is NP-hard.

Proof: The K-Availability creation problem can be proved to be NP-hard by a reduction from the Multiple Knapsack Problem (MKP), which is known to be NP-hard [3]. We show that MKP is a special case of K-Availability creation problem. MKP is defined as follows. Given is a set of m bins (knapsacks) and a set of n items. Each knapsack has a capacity c_j $(1 \leq j \leq m)$, and each item i has a size s_i and a profit p_i $(1 \leq i \leq n)$. The objective is to find a subset of data items of maximum profit such that they have a feasible packing in the bins.

To show MKP is a special case of K-Availability creation problem, we consider each of $K-1$ copies of each of the p original data items as one "item" in MKP, and each sensor node (including the source node) as a "bin" in MKP; all the replica copies of data item i has the same size s_i. Let $\mathcal{D} = \max_{i,j} d_{ij}$, where i is a source node and $j \in V$, i.e. $\mathcal{D}$ is the maximum distance between any source node and any sensor node. Let $\mathcal{D}$ minus the shortest distance between a data item's source node and a sensor node be the "profit" of the data item if distributed to that sensor node. With this, the objective of K-Availability creation problem is rephrased appropriately to maximizing profit, to be consistent with the maximization objective of MKP.

MKP is a special case of K-Availability creation problem in two aspects.[1] First, the profit of an item is a constant in MKP, while in K-Availability creation problem, the profit of an item can vary based upon the specific bin that it is put in. Second, in K-Availability creation problem, the $K-1$ copies of the same data item must be distributed to different sensor nodes (bins) due to the availability constraint. Therefore, if we relax the availability constraint and associate a constant profit with each data item in K-Availability creation, the problem becomes a MKP. Thus, K-Availability creation is at least as hard as MKP, and hence is NP-hard. ∎

Below we show that when each data item is of unit size, the K-Availability Creation problem is polynomially solvable.

Theorem 2: When each data item is of unit size (i.e. $\forall j, s_j = 1$) and the total size of the data items after replication is less than or equal to the total size of the effective storage space in the network (i.e. $K \times p \leq \sum_{i \in V} m_i^{'}$), the K-Availability Creation problem is equivalent to minimum cost flow problem.

Proof: When each data item is of unit size, the storage capacity of each node is multiple of data item size. For clarity of illustration, we assume that each data item is stored at a different source node, and the source node of D_i is node i. For more general case where a source node has multiple data items, subdivide each source node into same number of sub-nodes, each of which stores one data item. We transform

[1]Note that in MKP, the total size of the items could be larger than the total capacities of the knapsacks, i.e. $\sum_{i=1}^{n} s_i > \sum_{j=1}^{m} c_j$. Similarly, in K-Availability creation, the total size of the data items after replication could be larger than the total size of the effective storage space in the network, i.e. $K \times \sum_{i=1}^{p} s_i > \sum_{j \in V} m_j^{'}$. In this case, some data items can not have K replica copies.

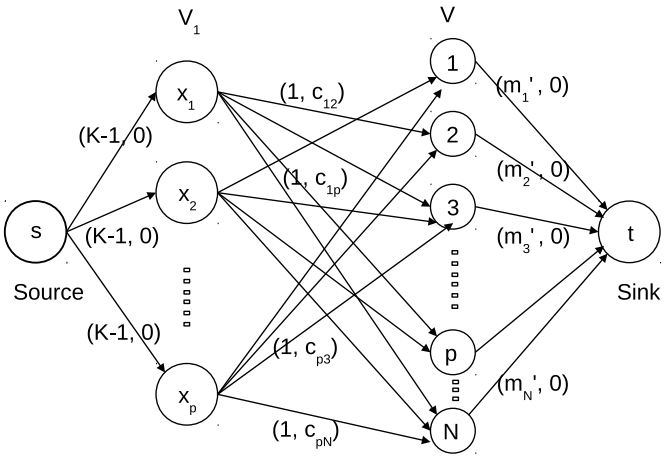

Fig. 2. K-Availability creation problem with unit data item size is equivalent to minimum cost flow problem. In each parenthesis, the first value is the capacity of the edge and the second the cost of the edge. Note that there are no edges: $(x_1, 1), (x_2, 2), ..., (x_p, p)$.

the K-Availability creation problem with unit data size to the minimum cost flow problem by changing $G(V, E)$ into a new graph $G'(V', E')$, as shown in Fig. 2:

1. $V' = V \cup \{s\} \cup \{t\} \cup V_1$, where s is the new source node, t is the new sink node, and $V_1 = \{x_1, x_2, ..., x_p\}$ is a set of p new nodes. Here x_i represents node i, the source node of D_i.
2. $E' = \{(s, i) : i \in V_1\} \cup \{(j, t) : j \in V\} \cup \{(i, j) : i \in V_1, j \in V\} - \{(x_1, 1), (x_2, 2), ..., (x_p, p)\}$.
3. For each edge (s, i), set its capacity as $K-1$ and its cost 0. For each edge (j, t), set its capacity as m_j' and its cost 0.
4. For all other edges (x_i, j), $x_i \in V_1, j \in V$, set its capacity as 1, and its cost as c_{ij}, the minimum energy cost of sending a unit-size data from i to j. Together with 2, it guarantees that the $K-1$ copies of each data item are distributed to $K-1$ different nodes other than its source node.
5. Set the supply of each node in $V_1 \cup V$ as 0. Set both the supply at s and the demand at t as $p \times (K-1)$.

Since all the inputs are integer, the minimum cost flow problem always has an integer minimum cost flow [1]. Now a valid flow of amount $p \times (K-1)$ from s to t includes $K-1$ amount on edge (s, x_1), $K-1$ amount on (s, x_2), ..., and $K-1$ amount on (s, x_p). This is the maximum possible flow. Therefore solving the minimum cost flow problem on $G'(V', E')$ gives the minimum distribution cost in the original graph $G(V, E)$. ∎

The minimum cost flow problem can be solved efficiently in polynomial time using well-known algorithms [1]. In this paper, we use the algorithm and implementation by Goldberg [6, 7] due to its practical nature. This algorithm has the time complexity of $O(N^2 M log(NC))$, where N, M, and C are the number of nodes, the number of edges, and the maximum capacity of an edge in graph G'. In our case, $C = K-1$.

V. K-Availability Maintenance

There are now K copies of each data item in this phase. Let D_{jk} denote the k^{th} copy of D_j. Let $S(j,k)$ be the node that stores D_{jk}. Due to availability constraint, $S(j,k) \neq S(j,k')$ if $k \neq k'$. The goal of the maintenance phase is to redistribute the data copies (including both replica copies and original copies of data items) to maximize the minimum remaining energy of the nodes that store any data copy. We assume that the battery power drainage rate at each sensor is a constant c (but our algorithm can be extended to varying drainage rate case). In K-Availability maintenance, even the original copies of the data items could be redistributed. For simplicity of formulation, we assume that each data item (and its replica) is of unit size. The formulation and proposed solution approaches can easily be extended to the scenario where data items have different sizes. The energy level of sensor node i is $E_i^{'}$, its remaining energy at the end of K-Availability creation. In this section, any node storing a data item (original or replica) is referred to as a *source node*.

Problem Formulation. A *redistribution function* is defined as $r : D \times \mathcal{K} \rightarrow V$, indicating that D_{jk} is redistributed from node $S(j,k)$ to node $r(j,k) \in V$ where $1 \leq j \leq p$ and $1 \leq k \leq K$. Let $P_{jk} : S(j,k), ..., r(j,k)$, referred to as *redistribution path* of D_{jk}, be the sequence of distinct sensor nodes along which D_{jk} is distributed from $S(j,k)$ to $r(j,k)$ (if $r(j,k) = S(j,k)$, it means that D_{jk} is kept in $S(j,k)$). Note that P_{jk} is not necessarily one of the shortest paths between node $S(j,k)$ and $r(j,k)$. Let R be the set of the destination nodes of the redistribution, i.e. $R = \bigcup_{1 \leq i \leq p, 1 \leq j \leq K} \{r(i,j)\}$. Let $E_i^{''}$ denote node i's energy level after the data redistribution, and let x_{ijk} be the energy cost incurred at node i in distributing D_{jk} from node $S(j,k)$ to $r(j,k)$,

$$E_i^{''} = E_i^{'} - \sum_{j=1}^{p} \sum_{k=1}^{K} x_{ijk},$$

where $x_{ijk} = E_T(k, i, \sigma(i,j,k))$ if $i = S(j,k)$, $x_{ijk} = E_R(k)$ if $i = r(j,k)$, $x_{ijk} = E_T(k, i, \sigma(i,j,k)) + E_R(k)$ if $i \in P_{jk} - \{S(j,k), r(j,k)\}$, and $x_{ijk} = 0$ otherwise. Here $\sigma(i,j,k)$ indicates the successor node of i on P_{jk}.

K-availability maintenance problem is to find a redistribution function r and a corresponding set of paths $\mathcal{P} = \{P_{ij}\}$, $1 \leq i \leq p$ and $1 \leq j \leq K$, to redistribute each of the $p \times K$ data item copies, such that the minimum remaining energy among all the destination nodes R is maximized post redistribution, i.e.

$$\max_{r,\mathcal{P}} \min_{1 \leq j \leq p, 1 \leq k \leq K} E_{r(j,k)}^{''}, \tag{5}$$

under the energy constraint that $E_i^{''} \geq 0, \forall i \in V$, the availability constraint: $r(j,k) \neq r(j,k^{'})$, when $k \neq k^{'}$, and the storage constraint: $|\{j | r(j,k) = i\}| \leq m_i^{'} = \min\{m_i, p\}, 1 \leq j \leq p, 1 \leq k \leq K$, which means that node i stores at most p or m_i distinct data items, whichever is smaller. Now with constant draining rate c of each sensor node's battery[2], the time that takes for the first data loss to occur is therefore $\frac{\max_{r,\mathcal{P}} \min_{1 \leq j \leq p, 1 \leq k \leq K} E_{r(j,k)}^{''}}{c}$. We refer to this as the *data preservation time* for the rest of the paper.

[2]If energy draining rates are varying for different nodes, instead of a constant, Equation 5 can be adjusted to account for the drainage rate n_j at node j as follows: $E_{r(j,k)/n_{r(j,k)}}^{''}$.

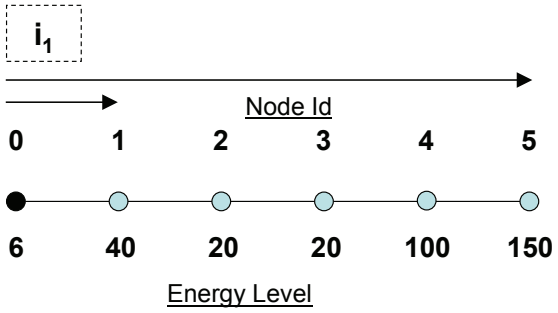

Fig. 3. An example of 2-Availability maintenance in a linear network of 6 nodes. The storage capacity of each node is 1.

EXAMPLE 2: Fig. 3 illustrates the K-Availability maintenance phase in a linear sensor network with $K = 2$. Each node's current energy is indicated in the graph, and each node has storage capacity of 1. Node 0 stores the original copy of data item i_1, node 4 stores the other copy of i_1. To preserve both copies as long as possible under energy and availability constraints, node 0 redistributes i_1 to node 5. This example illustrates that the node with maximum remaining energy level is a good choice for redistributing the data during K-Availability maintenance. □

Theorem 3: K-Availability maintenance problem is NP-hard even for unit data sizes.

Proof: It can be shown that the disjoint connecting paths (DCP) problem [5], which is known to be NP-hard, is a special case of the decision version of the K-Availability maintenance problem.

It suffices to show that $K = 1$ case is NP-hard (thus distribution path P_{ij} can be simply written as P_i). Without loss of generality, let D_i be stored at node i ($1 \leq i \leq p$). We further simplify the problem by assuming that $m_i = 1$ for sensor node i (including the source nodes). We show that the disjoint connecting paths (DCP) problem [5], which is known to be NP-hard, is a special case of the decision version of above simplified data preservation problem. The DCP problem is as follows. Given a graph $G(V,E)$ and a set of p disjoint source and destination vertex pairs (s_i, t_i), where $s_i, t_i \in V$ for $1 \leq i \leq p$, the goal is to find whether there are p vertex-disjoint paths $P_1, P_2, ..., P_p$ in G where P_i connects s_i to t_i.

In K-Availability maintenance problem, let p source nodes be $S = \{s_1, s_2, ..., s_p\}$, and let $T = \{t_1, t_2, ..., t_p\}$ with $T \cap S = \phi$ (empty set). Then assign the energy level of each node in T as $E \gg 1$, and the energy level for other nodes in $(V - T)$ (including S) as 1. We claim that to determine whether the maximum data preservation time of the network is $(E - 0.5)/c$ is the same as solving DCP problem, whether there exist p vertex-disjoint paths connecting the source and destination vertex pairs.

On one hand, if the maximum data preservation time equals $(E - 0.5)/c$, it must be the case that data item in one of the nodes in S is distributed to one of the nodes in T. Since

sending and receiving any data item for each node costs 0.5 units of energy, and all the nodes in $(V-T)$ have energy level of 1, the p distribution paths $P_1, P_2, ..., P_p$ must be mutually vertex-disjoint. On the other hand, if there exist p vertex-disjoint paths connecting p source and destination vertex pairs, these p paths are the distribution paths we use to distribute the p data items. In this case, the energy level of each node in T after distribution is $E - 0.5$, therefore maximum data preservation time of $(E - 0.5)/c$ can be reached. ∎

The proof technique used here is similar to that used in [27] for a closely related energy depletion induced data preservation problem in intermittently connected sensor networks.

Since the K-Availability maintenance problem is NP-hard, we propose a centralized heuristic as follows.

Centralized Max-Min Heuristic. First, the destination node is chosen such that it has the maximum energy level among all nodes in the network, and with available storage. Then find the source node that is closest to this destination node. We redistribute a data item from this source node to this destination node, using a shortest path between them, so as to maximize the minimum remaining energy of any node along the path (after redistribution is complete). If the chosen destination is unable to store all the data (due to availability or storage constraint), the next best destination is chosen to redistribute the remaining data in a similar fashion. The time complexity of this algorithm is $O(qKN^3)$, here q is total number of data items in the network. The detailed analysis is omitted here due to space constraint.

VI. Distributed algorithm for K-Availability Problem

In this section, we design a distributed algorithm, which could be used in both K-Availability creation and maintenance phases. The algorithm works in iterations. Each iteration has the following stages:

1. **Advertisement Stage.** The source node i broadcasts an advertisement message containing its ID i, its current energy E_i, and the IDs of all the data items stored in i. Assume that there are $n \leq p$ data items stored at i: $\{I_1, ..., I_n\}$.
2. **Storage Commitment Stage.** Each sensor node j, upon receiving the advertisement message from node i, performs the following steps[3]:
   a. If $E_j \leq E_i$, or $n = 0$, or $m'_j = 0$, go to Step d.
   b. Computes $\phi(i,j) = E_j \times q_j / d_{ij}$. E_j is the current energy level of node j. d_{ij} is the shortest path distance from node i to j, which can be obtained by incrementing an integer stored in advertisement message every time a node receives it the first time. q_j is the number of i's data items that could be stored at node j according to availability constraint. It is computed as follows:
      – $q_j = 0$.
      – $\forall y \in \{I_1, ...I_n\}$, if a copy of y is not stored at node j, increment q_j.
      – $q_j = \min\{q_j, m'_j\}$.
   c. If $\phi(i,j) = 0$, go to Step d. Else, sends a commitment message to node i, along with $\phi(i,j)$ and the IDs of the data items it can store for i: $\{I_{q_1} ... I_{q_j}\}$. When no node sends out storage commitment during a complete round of advertisements, network lifetime is reached.
   d. Forwards the advertisement message.
3. **Data Offloading Stage.** Once source node i has received all commitment message (this can be done by i to wait for enough time), it performs the following steps (it does nothing if it does not receive any commitment message):
   a. Selects the node j with maximum $\phi(i,j)$ (ties are broken randomly), and sends these q_j data items to node j following the reverse of the path from which it receives the commitment message from j.
   b. Removes the q_j data items from its list of data items to be redistributed and sets $n = n - q_j$. Removes j from its list of committed nodes. Removes these q_j data items from the list of committed data items sent by other nodes. Let q^*_k denote the size of new list for committed node k. For each committed node k, node i recomputes $\phi^*(i,k) = \phi(i,k) \times q^*_k / q_k$.
   c. If $n = 0$, stop. Else, $\phi(i,k) = \phi^*(i,k)$. If $\phi(i,k) = 0$, remove node k from the list of committed nodes. Go back to Step a.

[3]Note that the advertisement message from the same node is processed and forwarded only once by node j.

Discussion and Message Complexity. $\phi(i,j) = E_j \times q_j / d_{ij}$ is used in both Stage 2 and Stage 3 to facilitate the communication among node i and j. The rationale is that a node with high energy should be an ideal storage node, and a node with many replica copies should offload more copies to nodes with high energy if its own energy is low. Meanwhile, a nearby node with less energy level is preferred over a far away node with similar or slightly higher energy level. Over time, as the network dynamics change, this nearby node can always redistribute the data item before its energy runs out. Thus, it is still possible that the data item will reach the far away node later on. We also calculate the number of transmissions in the distributed algorithm without going through the detailed analysis – the total number of transmissions needed is $O((q + \bar{m})KN^{\frac{3}{2}})$, where q is total number of data items, $\bar{m}$ is the average storage space of each node.

VII. Performance Evaluation

In this section, we present simulation results and analysis for K-Availability creation and K-Availability maintenance phase, respectively. In both cases, we adopt grid-like sensor network topology (note that our proposed algorithms are applicable to other topologies). In all cases, the transmission range of the sensor is one unit, the length of each grid edge.

K-Availability Creation. Fig. 4 shows the optimal energy consumption obtained by minimum cost flow algorithm, by varying the network size from 15×15, 20×20, 25×25, ..., to 50×50. For each network size, we randomly choose the source nodes in the network with number varying as the ratio of the network size, from 1%, 6%, 10%, 30%, to 50%. Each source initially has one data item of unit size. The storage capacity of each node is 100 units. We let $K = 200$, which corresponds to

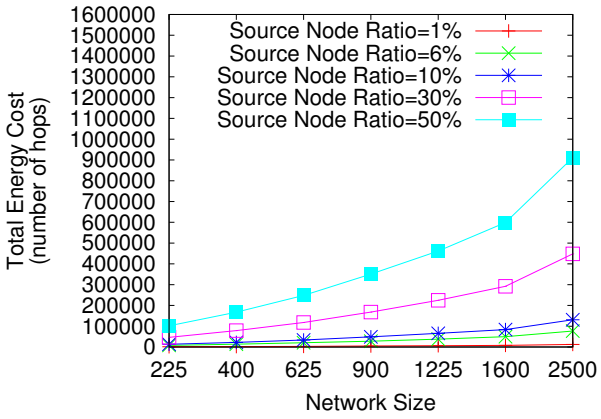

Fig. 4. Minimum cost flow algorithm for K-Availability creation, $K = 200$.

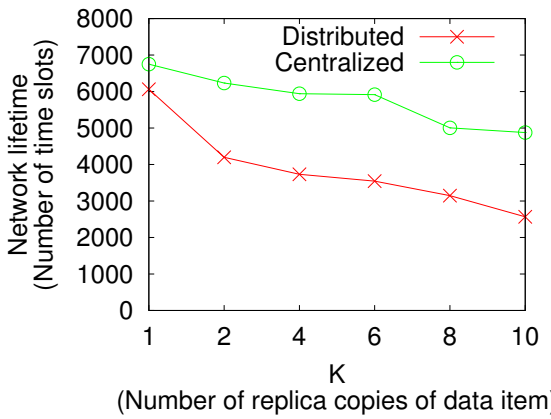

Fig. 6. Network lifetime with respect to number of replica copies in K-Availability maintenance.

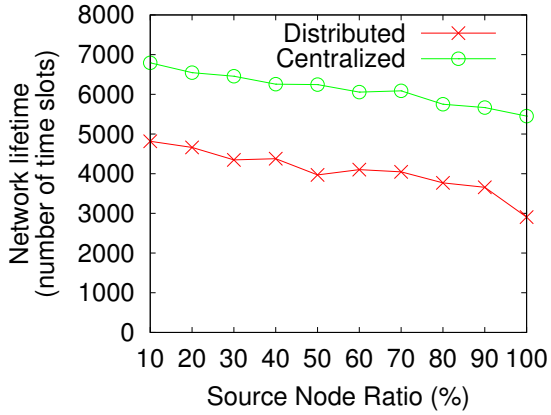

Fig. 5. Network lifetime with respect to number of source nodes in K-Availability maintenance.

a very large probability of failure, as $p_f = 1-1/200 = 0.995$. This represents the case where sensor nodes are quite cheap and unreliable, or the environment where they are deployed is harsh, making replication necessary in order to maintain data availability. It suggests that when the total size of the replicated data is much less than the total effective storage capacity, the total energy consumption increases slowly with an increase in the network size (for example, when the source node ratio is 1%, the total storage capacity in the network is 50 times the total storage demand.). However, when the total size of the replicated data is comparable to the total effective storage capacity, the total energy cost increases exponentially with the network size. For example, when the source node ratio is 50%, the total storage capacity in the network equals the total storage demand, for all network sizes. Therefore, the cost of redistribution is much higher, particularly for larger network size, as the average path length increases causing heavy energy drainage at intermediate relaying nodes.

K-Availability Maintenance. For K-Availability maintenance, we compare the distributed algorithm with the centralized heuristic described in Section V. Here, the network size is 5×5, $K = 3$. The storage capacity of each node equals 100, and initial energy of each node is a random number between 1000 and 10000. We assume the energy depletion rate of each node is $c = 1$, and the network lifetime equals to the maximum remaining energy of the destination nodes divided by c. Fig. 5 depicts this comparison in terms of the network lifetime achieved, for a range of source node ratios. It shows that the lifetime decreases almost linearly with an increase in storage demand. The centralized heuristic performs better in all cases. This is because the centralized algorithm does not incur the energy costs incurred by the distributed algorithm during the advertisement and storage commitment stages. Nevertheless, our proposed distributed algorithm's performance $\geq 60\%$ of the centralized algorithm in all cases.

Fig. 6 studies the effect of K upon system performance, with source node ratio 20%. We observe that the performance decrease is drastic when K is increased from 1 to 2, but is more gradual thereafter. Similar performance trends are depicted in Fig. 7 when $K = 3$, for various choices of energy depletion rate parameter c. Note that, when c is large, the communication cost is relatively small compared to the constant energy drainage at all nodes. This accounts for the smaller difference in performance between the centralized and distributed algorithms, since the major contributor to the difference is the communication and relaying energy costs. Also, the network lifetime decreases drastically as c is increased initially from 1 to 2. Thereafter, the network lifetime decreases gradually with increase in c, similar to the trend observed with K in Fig. 6.

VIII. Conclusion and Future Work

Data collection and distributed storage using sensor networks is important in many scientific applications. Data K-Availability problem endeavors to preserve data in the face of energy and storage space constraints in sensor networks, as well as node failure, by utilizing data replication, distribution, and redistribution. When data loss is a critical concern, K-Availability is a feasible solution towards preserving the complete data. We formulated the K-Availability problem in sensor networks, and showed that both K-availability creation and maintenance are NP-hard in the general case. K-Availability becomes particularly important when a network could become

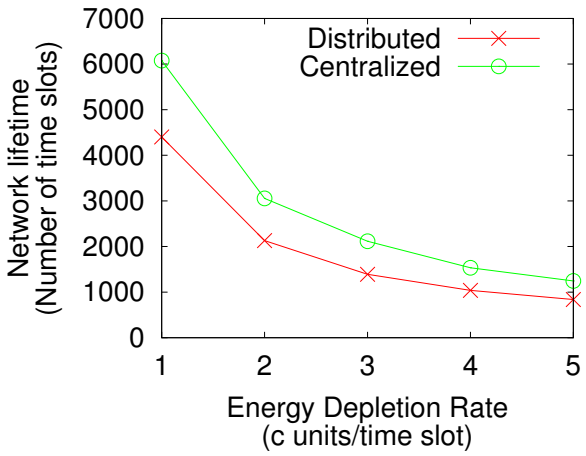

Fig. 7. Network lifetime with respect to energy depletion rate in K-Availability maintenance.

disconnected from the base station for considerably larger periods of time. We designed a distributed algorithm for this problem and discussed its performance trends and comparison with an appropriate centralized heuristic. For the future work, on the theory side, we will consider varying energy consumption (i.e., energy cost depends on the distance of two communicating nodes) and varying data sizes, and investigate the hardness of the $DKAP$ problem. On the experiment side, the simulations in this paper are only limited to grid networks. We would like to experiment on randomly generated sensor networks, and investigate how network lifetime is affected in different network scenarios.

IX. Acknowledgement

This work was supported in part by NSF Grants CNS-1116849, CNS-1248315 and EPS-0903806.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] Costas Busch, Malik Magdon-Ismail, Fikret Sivrikaya, and Bulent Yener. Contention-free mac protocols for wireless sensor networks. In *Proc. of DISC*, pages 245–259, 2004.

[3] Chandra Chekuri and Sanjeev Khanna. A ptas for the multiple knapsack problem. *SIAM J. Comput.*, 35(3):713–728, 2005.

[4] Lei Gao, Mike Dahlin, Amol Nayate, Jiandan Zheng, and Arun Iyengar. Improving availability and performance with application-specific data replication. *IEEE Transactions on Knowledge and Data Engineering*, 17:106–120, 2005.

[5] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[6] A. V. Goldberg. Andrew goldberg's network optimization library. http://www.avglab.com/andrew/soft.html.

[7] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29, 1997.

[8] Takahiro Hara and Sanjay K. Madria. Data replication for improving data accessibility in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11):1515–1532, 2006.

[9] Wendy Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of HICSS 2000*.

[10] Xiang Hou, Zane Sumpter, Lucas Burson, Xinyu Xue, and Bin Tang. Maximizing data preservation in intermittently connected sensor networks. In *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2012)*. Short Paper.

[11] Sushant Jain, Rahul Shah, Waylon Brunette, Gaetano Borriello, and Sumit Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *MONET*, 11(3):327–339, 2006.

[12] Abhinav Kamra, Vishal Misra, Jon Feldman, and Dan Rubenstein. Growth codes: maximizing sensor network data persistence. In *Proc. SIGCOMM*, 2006.

[13] Samir Khuller and Yoo ah Kim. Algorithms for data migration with cloning. In *SIAM Journal on Computing*, pages 27–36. ACM Press, 2003.

[14] Ming Lei, Susan V. Vrbsky, and Xiaoyan Hong. An on-line replication strategy to increase availability in data grids. *Future Generation Computer Systems*, 24:85–98, 2008.

[15] Shi Li, Yunhao Liu, and Xiang-Yang Li. Capacity of large scale wireless networks under gaussian channel model. In *Proc. of MOBICOM 2008*.

[16] Kirk Martinez, Royan Ong, and Jane Hart. Glacsweb: a sensor network for hostile environments. In *Proc. of SECON 2004*.

[17] Ioannis Mathioudakis, Neil M. White, and Nick R. Harris. Wireless sensor networks: Applications utilizing satellite links. In *Proc. of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007)*, pages 1–5, 2007.

[18] M. Montanari, R. Crepaldi, I. Gupta, and R. Kravets. Using failure models for controlling data availability in wireless sensor networks. In *Proc. of Infocom 2009 Miniconference*.

[19] Giwon On, Jens Schmitt, and Ralf Steinmetz. Quality of availability: Replica placement for widely distributed systems. In *Proc. of IWQos 2003*.

[20] Christos Papadimitriou and Kenneth Steiglitz. Combinatorial optimization: Algorithms and complexities. *Prentice Hall*, 1982.

[21] A. Pinar and B. Hendrickson. Interprocessor communication with limited memory. *IEEE Transactions on Parallel and Distributed Systems*, 15:606–616, 2004.

[22] Kavitha Ranganathan, Adriana Iamnitchi, and Ian Foster. Improving data availability through dynamic model-driven replication in large peer-to-peer communities.

[23] Florian Schintke and Alexander Reinefeld. Modeling replica availability in large data grids. *Journal of Grid Computing*, 2(1):219–227, 2003.

[24] Masako Shinohara, Takahiro Hara, and Shojiro Nishio. Data replication considering power consumption in ad hoc networks. In *Proc. of the International Conference on Mobile Data Management (MDM 2007)*, pages 118–125.

[25] Stephen F. Siegel and Andrew R. Siegel. Madre: The memory-aware data redistribution engine. In *Proc. of PVM/MPI 2008*.

[26] Affan A. Syed, Wei Ye, and John Heidemann. T-lohi: A new class of mac protocols for underwater acoustic sensor networks. In *Proc. of INFOCOM 2008*.

[27] Masaaki Takahashi, Bin Tang, and Neeraj Jaggi. Energy-efficient data preservation in intermittently connected sensor networks. In *Proc. of the International Workshop on Wireless Sensor, Actuator and Robot Networks (WiSARN), in conjunction with IEEE INFOCOM 2011*.

[28] Bin Tang and Himanshu Gupta. Caching placement in sensor networks under update cost constraint. *Journal of Discrete Algorithms*, 5(3):422–435, September 2007.

[29] Bin Tang, Himanshu Gupta, and Samir Das. Benefit-based data caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 7(3), March 2008.

[30] Bin Tang, Neeraj Jaggi, Haijie Wu, and Rohini Kurkal. Energy efficient data redistribution in sensor networks. In *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2010)*.

[31] Bin Tang, Neeraj Jaggi, Haijie Wu, and Rohini Kurkal. Energy efficient data redistribution in sensor networks. *ACM Transactions on Sensor Networks*, 9(2), march 2013.

[32] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proc. of SenSys 2005*.

[33] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of OSDI 2006*.

[34] Xinyu Xue, Xiang Hou, Bin Tang, and Rajiv Bagai. Data preservation in intermittently connected sensor networks with data priority. In *Proc. of SECON*, 2013.

[35] Liangzhong Yin and Guohong Cao. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, 2006.

[36] Haifeng Yu and Amin Vahdat. Minimal replication cost for availability. In *Proc. of the twenty-first annual symposium on Principles of distributed computing (PODC 2002)*, pages 98–107.