DRE^2: Achieving Data Resilience in Wireless Sensor Networks: A Quadratic Programming Approach

Shanglin Hsu, Yuning Yu, and Bin Tang
Department of Computer Science
California State University Dominguez Hills, Carson, CA 90747, USA
Email: {shsu10,yyu19}@toromail.csudh.edu, btang@csudh.edu

***Abstract*— We focus on sensor networks that are deployed in challenging environments, wherein sensors do not always have connected paths to a base station, and propose a new data resilience problem. We refer to it as DRE^2: *data resiliency in extreme environments*. As there are no connected paths between sensors and the base station, the goal of DRE^2 is to maximize data resilience by preserving the overflow data inside the network for maximum amount of time, considering that sensor nodes have limited storage capacity and unreplenishable battery power. We propose a quadratic programming-based algorithm to solve DRE^2 optimally. As quadratic programming is NP-hard thus not scalable, we design two time efficient heuristics based on different network metrics. We show via extensive experiments that all algorithms can achieve high data resiliences, while a minimum cost flow-based is most energy-efficient. Our algorithms tolerate node failures and network partitions caused by energy depletion of sensor nodes. Underlying our algorithms are flow networks that generalize the edge capacity constraint well-accepted in traditional network flow theory.**

Keywords – Data resilience, integer quadratic and linear programming, network flows, wireless sensor networks.

I. Introduction

Background and Motivation. *Data resilience* refers to the ability of any network to recover quickly and to continue maintaining availability of data despite of disruptions such as equipment failure, power outage, or malicious attack. Due to resource constraint challenges of wireless sensor networks such as unreplenishable battery power and limited storage capacity of sensor nodes [35], link unreliability and scarce bandwidth of wireless medium [41], and the inhospitable and harsh environments in which they are deployed [6, 7], sensor nodes are often prone to failure and vulnerable of data loss. Therefore, how to ensure that collected data reaches the base station reliably has been an active research since the inception of sensor network research. This research is usually named data resilience [20], reliable data transmission [27], or data persistence [23]. We use data resilience throughout the paper.

However, all the existing data resilience research in traditional sensor networks assumes that a base station is always available to collect data, and focuses on how to encode and transmit data to the base station reliably. In this paper, we instead study data resilience from a totally different angle – from emerging sensor network applications wherein a base station is not available to collect the data. Such applications include volcano and seismic sensor networks [26], underground sensor networks [31], underwater or ocean sensor networks [9, 28], and volcano eruption monitoring and glacial melting monitoring [11, 32].

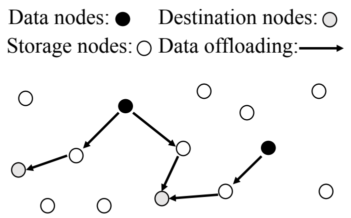


Fig. 1. The network model.

There are two common features of these sensor networks. First, they are all deployed in inaccessible or inhospitable regions, or under extreme weather. Therefore it is not feasible to deploy data-collecting base stations with power outlets in or near the sensor field. Second, as they continuously collect large volumes of data for a long period of time without the base stations, sensory data generated thus has to be stored inside the network first and then being collected when data mules or mobile sinks visit the sensor field periodically [8, 30]. Lacking human intervention and maintenance due to extreme environments, these sensing applications must operate more resiliently and robustly than traditional sensor networks.

Data Resilience Against Sensor Storage Overflow. In this paper, we focus on data resilience against *sensor storage overflow*, wherein some sensor nodes are close to the events of interest and continuously generate large amounts of sensory data, thus depleting their storage spaces and can not store any newly generated data [24, 35]. Our network model is shown in Fig. 1. We refer to the sensor nodes that have exhausted their storage spaces as *data nodes*; the part of data that cannot be stored locally is referred to as *overflow data*. Other sensor nodes that have available storage are *storage nodes* (sensor nodes that generate data but still have available storages are considered as storage nodes).

In order to prevent data loss, the overflow data at the data nodes must be offloaded to the storage nodes to be preserved, waiting to be collected by aforesaid uploading opportunities. Otherwise, data loss occurs and the data resilience is not achieved. The storage nodes that finally store overflow data are *destination nodes*. We refer to the process that overflow data is offloaded from data nodes to destination nodes as *data offloading in sensor networks*. As it is not known beforehand when the next uploading opportunity arrives, it is preferred that the offloaded data being stored in destination nodes for longest period of time (i.e.. survival time) before they run out of battery power. Assume that all the sensor nodes have the same energy depleting rates, data thus should be offloaded to destination nodes with highest battery power. As each sensor node has unreplenishable battery power and limited storage capacity, the challenge is how to design data offloading scheme that maximizes the survival time for the offloaded data packets.

Contributions. Our contributions are twofold. On the practical side, we identify, formulate, and solve a new algorithmic problem in sensor networks called DRE^2: *data resiliency in extreme environments*. We accurately quantify the data resilience and formulate the problem (Section III), and design a quadratic programming (QP)-based algorithm to solve DRE^2 optimally (Section IV). As QP is NP-hard and is not scalable, we design a suite of time-efficient and fault-tolerant heuristic algorithms (Section V). We show that all algorithms achieve high data resiliences while a minimum cost flow (MCF)-based algorithm is most energy efficient (Section VI). MCF is formulated as an integer linear program (ILP).

On the theory side, the underlying enabler of our techniques is flow networks that are delicately converted from the sensor network. These flow networks make possible to identify the convoluted relationship between energy consumption of sensor nodes and the flows of data offloading in DRE^2. We find that in our flow networks, the relationship between flows and capacities on network edges are significantly different from those well established in traditional network flow theory. In particular, we uncover a new relationship wherein the capacity of an edge must be greater than or equal to the *linear combination (i.e., weighted sum) of the flows* on this edge. In contrast, the well-known *edge capacity constraint* of flow networks only mandates that the edge capacity is greater than or equal to the *number of flows* on an edge. Such *generalized edge capacity constraint* uniquely arises from our data resilience problem and to the extend of our knowledge has not been identified in any of the existing literature. With this generalization, we are able to apply QPs and ILPs on the flow networks to solve DRE^2 optimally and energy-efficiently.

II. Related Work

Quadratic programming is the technique of optimizing a quadratic objective function with linear equality and inequality constraints [12, 13], and is one of the simplest forms of non-linear programming. It has been used in sensor network research to solve several important problems such as localization [10, 22], variational data assimilation problem for Lagrangian sensors [10], and optimized transmission for parameter estimation [36]. In contrast, we use this technique to solve a totally different problem. We believe our work is the first one to use quadratic programming to achieve optimal data resilience in sensor networks deployed in challenging environments.

Data resilience has been an active research since the inception of sensor network research. Ghose et al. [16] achieved resilience by replicating data at strategic locations in the sensor network. Ganesan et al. [14] constructed disjoint multipaths to enable energy efficient recovery from node failures. Recently, network coding techniques are used to recover data from failure-prone sensor networks. Albano et al. [5] proposed in-network erasure coding to improve data resilience to node failures. Kamra et al. [21] proposed to replicate data compactly at neighboring nodes using growth codes that increase in efficiency as data accumulates at the sink. As wireless sensor netowrks utilize sleeping mechanisms to conserve energy, which causes data availability problem, Xu et al. [38] proposed a dataset synchronization protocol in named data networking to achieve data resilience. However, all these research adopts the traditional sensor network model wherein base stations are always available near the networks, therefore do not target the data resilience problem studied in this paper.

In sensor network where the base stations are absent, data resilience is achieved by preserving data inside the network, with two lines of work. The first line is a sequence of system research [25, 37, 40] that designed cooperative sensor storage systems to improve the network storage utilization. The other line of work is our own research. We instead took an algorithmic approach and solving a variety of data offloading problems with different objectives in sensor networks from minimizing the total energy consumption [6, 35], maximizing the total priorities of preserved data [39] or the minimum remaining energy of destination nodes [19], replicating data packets in the events of node failures [6, 34], to overcoming the overall storage overflow [33]. However, none of them attempts to maximize the data resilience defined in this paper.

III. Problem Formulation of DRE^2

Network Model. We model a sensor network as an undirected graph $G(V, E)$, where $V = \{1, 2, ..., n\}$ is the set of n nodes, and E is the set of m edges. Two nodes are connected if their distance is within the sensor nodes' transmission range. There are l data nodes denoted as $V_d = \{1, 2, ..., l\}$. Data node $i \in V_d$ has d_i number of overflow data packets, each has size of k bits. The rest $n - l$ nodes are storage nodes $V - V_d = \{l+1, l+2, ..., n\}$. We denote the total $a = \sum_{i=1}^{l} d_i$ overflow data packets as $D = \{D_1, D_2, ..., D_a\}$. Let the data node of D_j be $dn(j) \in V_d$. Let m_j be the available free storage space at storage node $j \in V - V_d$, meaning that j can further store m_j data packets. We assume that $a \leq \sum_{j=l+1}^{n} m_j$; otherwise, data loss is inevitable and data resilience is not achieved. We leave the more general question of under which conditions that not all the data packets can be offloaded as a future work.

Augmented Energy Model. We augment the well-known first order radio model [18] for wireless energy consumption. When node u sends a k-bit data packet to its one hop neighbor node v over their distance $l_{u,v}$ meters, the *transmission energy* spent by u is $E_u^t(v) = \epsilon_{elec} * k + \epsilon_{amp} * k * l_{u,v}^2$, the *receiving energy* spent by v is $E_v^r = \epsilon_{elec} * k$. Here $\epsilon_{elec} = 100nJ/bit$ is the energy consumption per bit on the transmitter circuit

TABLE I

NOTATION SUMMARY

Notation	Description
V	The set of n data nodes
V_d	$V_d = \{1, ..., l\}$ is the set of l data nodes, and $V - V_d = \{l+1, l+2, ..., n\}$ is the set of storage nodes
d_i	Number of overflow data packets from data node $i \in V_d$
m_j	Storage capacity of storage node $j \in V - V_d$
D	$D = \{D_1, D_2, ..., D_a\}$ is the set of a overflow data packets
$dn(j)$	The data node of $D_j \in D$
E_i	Initial energy level of sensor node i
$E_i^{'}$	Remaining energy level of sensor node i after data offloading
$E_u^t(v)$	Transmission energy spent by u to transmit one packet to v
E_v^r	Receiving energy spent by v to receive one data packet
E_v^s	Storing energy spent by v to store one data packet
r	Data offloading function
P_j	The *offloading path* of data packet $D_j \in D$
$\sigma(i,j)$	Node i's successor node in P_j
$y_{i,j}$	Node i's energy cost of offloading data packet D_j
$\xi(i)$	Number of data packets stored at storage node i
$x_{i,j}$	The amount of flows on edge (i,j) in flow networks for QP and ILP
G'	$G'(V', E')$ is the flow network used for QP
G''	$G''(V'', E'')$ is the flow network used for ILP

and receiver circuit, and $\epsilon_{amp} = 100pJ/bit/m^2$ is the energy consumption per bit on the transmit amplifier.

However, this model does not take into account the energy consumption of storing (or writing) data packets into sensor nodes. As write operation costs 13.2μJ amount of energy in Toshiba 128MB flash [4] and we are dealing with large amounts of sensory data, we assume that energy consumption for storing data is non-negligible. We augment above first order radio model with storing energy cost as follows. When storing a data packet, a storage node $v \in V - V_d$ costs $E_v^s = \epsilon_{store} * k$ amount of *storing energy*. Here we assume that $\epsilon_{store} = 100nJ/bit$ is the energy consumption of storing one bit onto the sensor storage. Let $E_{u,v} = E_u^t(v) + E_v^r$; we have $E_{v,u} = E_{u,v}$. Note a data node not only transmits all of its own data packets, but also can receive and transmit (i.e., relay) data packets for other data nodes. Meanwhile, a storage node can receive data packets from other nodes and then either transmits or stores them. Table I shows all the notations.

Problem Formulation. We define *offloading function* as $r : D \to V - V_d$, indicating that data packet $D_j \in D$ is distributed from its data node $dn(j) \in V_d$ to its destination node $r(j) \in V - V_d$. Let $P_j : dn(j), ..., r(j)$, referred to as the *offloading path* of D_j, be the sequence of distinct sensor nodes along which D_j is offloaded from $dn(j)$ to $r(j)$. Let $\sigma(i,j)$ denote node i's successor node in P_j. Let $y_{i,j}$ be node i's energy cost of offloading data packet D_j, then

$$y_{i,j} = \begin{cases} E_i^t(\sigma(i,j)) & i = dn(j), \\ E_i^r + E_i^s & i = r(j), \\ E_{i,\sigma(i,j)} & i \in P_j - \{dn(j), r(j)\}, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

When i is the data node of D_j, it costs transmission energy $E_i^t(\sigma(i,j))$; when it is the destination node of D_j, it costs both receiving energy E_i^r and storing energy E_i^s; when it is a relaying node of D_j, it costs both receiving and transmission energy, the sum of which is $E_{i,\sigma(i,j)}$. Otherwise, node i is not involved in D_j's offloading thus costs zero amount of energy. Let E_i denote sensor node i's initial energy level and $E_i^{'}$ its remaining energy level after all the a data packets are offloaded, respectively. Then, $E_i^{'} = E_i - \sum_{j=1}^{a} y_{i,j}, \forall\ i \in V$.

Definition 1: (**Data Resilience Levels (DRLs).**) Given a sensor network G with a data packets to be offloaded, its *data resilience level* (DRL), denoted as $\mathcal{D}(G)$, is defined as the sum of remaining energy of the destination nodes of all the a data packet; i.e., $\mathcal{D}(G) = \sum_{j=1}^{a} E'_{r(j)}$. It is also the case that $\mathcal{D}(G) = \sum_{i=l+1}^{n} \left(E_i^{'} \times \xi(i)\right)$, where $\xi(i)$ is the number of data packets that are finally stored at storage node i. □

$\mathcal{D}(G)$ indicates the network's best achievable effort to preserve data packets, as the more energy of a storage node, the longer time its stored data can survive. The objective of DRE^2 is to find a offloading function r and a set of offloading paths $\mathcal{P} = \{P_1, P_2, ..., P_a\}$, each for one of the a data packets, such that the DRL of the network is maximized, i.e., $\max_{r,\mathcal{P}} \mathcal{D}(G)$, under the energy constraint of sensor nodes: $E_i' \geq 0, \forall\ i \in V$ and the storage capacity constraint of storage nodes: $|\{j \mid r(j) = i, 1 \leq j \leq a\}| \leq m_i, \forall\ i \in V - V_d$. Note that in event-driven or real-time scenarios wherein new data nodes appear periodically (due to new events of interest), we can apply our algorithms in a time-slotted manner to maximize the data resilience of their newly generated data packets.

EXAMPLE 1: Fig. 2(a) shows a linear sensor network with four nodes: 1 and 2 are data nodes, each has two overflow data packets; 3 and 4 are storage nodes, each has storage capacity of four and $E_4 \gg E_3$. To achieve maximum DRL, the optimal solution is to offload all the four data packets to node 4, even though it costs more energy than offloading to node 3. We use this example to illustrate our QP and ILP solutions next. □

IV. QUADRATIC PROGRAMMING (QP) SOLUTION

To maximize the DRL of any given instance of DRE^2, the fundamental challenge is to find each data packet's destination node as well as its offloading path from its data node to this destination node. Then we are able to compute the number of data packets each destination node stores as well as its remaining energy level, therefore calculating the DRL. In particular, we need to represent sensor energy levels, data packets offloaded, and storage capacities utilized such that they can be computed in a holistic manner – network flow modeling [29] is particularly suitable for such purpose.

Graph Conversion. To demonstrate the network flow techniques. we first convert the sensor network graph $G(V,E)$ in Fig. 2(a) to a flow network $G'(V', E')$ in Fig. 2(b).

First, it replaces each undirected edge $(i,j) \in E$ with two directed edges (i,j) and (j,i) of capacities $+\infty$. Second, it splits node $i \in V$ into two nodes *in-node* i' and *out-node* i'' and adds a directed edge (i', i'') with capacity of E_i, the initial energy level of node i. Third, all incoming directed edges of node i are incident on i' and all outgoing directed edges of node i emanate from i''. Therefore the two directed edges (i,j) and (j,i) are now changed to (i'', j') and (j'', i'). Fourth, it connects a super source node s to the in-node i' of the data

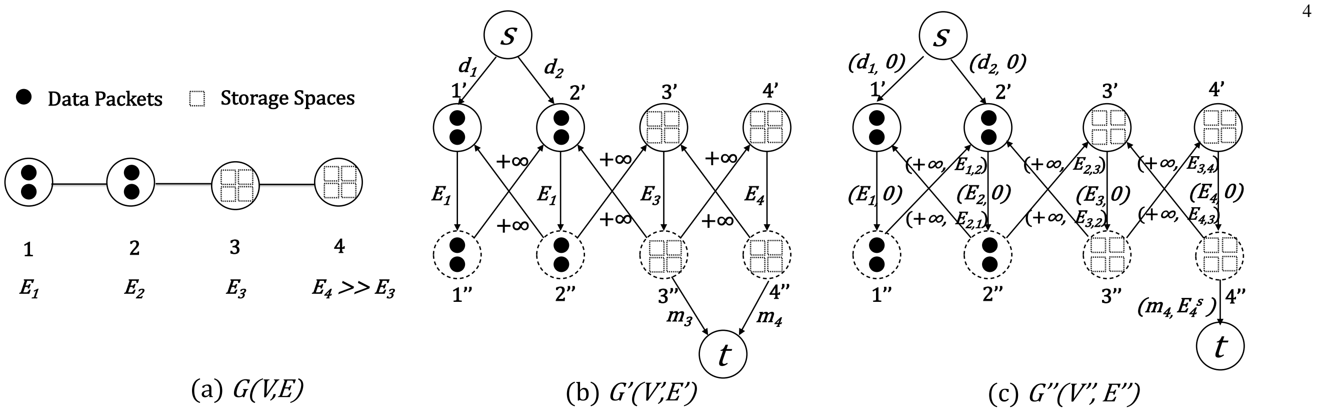


Fig. 2. (a) shows a linear sensor network $G(V,E)$ with two data nodes 1 and 2, each having two data packets to offload, and two storage nodes 3 and 4, each having four storage spaces. (b) shows its converted flow network $G'(V',E')$ for QP (A) that maximizes DRLs. (c) shows its converted flow network $G''(V'',E'')$ for ILP (B) that finds the minimum energy cost of data offloading. As $E_4 > E_3$ and node 4 has enough storage to store all the four data packets, the set of high-energy storage nodes $V_h = \{4\}$.

node $i \in V_d$ with an edge of capacity d_i, the number of data packets at data node i. Finally, it connects the out-node i'' of the storage node $i \in V - V_d$ to a super sink node t with an edge of capacity m_i, the storage capacity of storage node i. We have $|V'| = 2n + 2$ and $|E'| = 2m + 2n$. This graph conversion technique is used in our previous work [19, 39] solving other variants of data offloading in sensor networks.

Rationale of above Conversion. The rationale of above conversion is fourfold. First, as the flows start from s and end at t in flow network G', and s connects to in-nodes of data nodes while t connects to out-nodes of storage nodes, it "forces" the overflow data packets to offload from data nodes to storage nodes. Second, with the node-splitting and the initial energy levels now being capacities of newly created edges, it guarantees that each node cannot exceed its energy capacity. Third, with the d_i and m_i now being "encoded" as the capacities of edges connecting s and i' (for data nodes) and i'' to t (for storage nodes), it makes sure that a data node cannot offload more overflow data packets than it has and a storage node cannot store more overflow data packets than its storage allows. Fourth, and most importantly, the energy consumption of each node and the DRL can now be accurately computed using the network flows in G', as shown below.

Computing the DRL. Let $x_{i,j}$ be the amount of flows on directed edge (i,j) in G'. Recall that although both data nodes and storage nodes can receive overflow data packets from other nodes, a data node must transmit all of them whereas a storage node can store some of them as long as its storage allows and transmit the rest. Besides, a data node must transmit all of its own overflow data packets to others. Fig. 3(a) and (b) visualize the flows in G' that go through a data node and a storage node i respectively. We refer to the edge (i', i'') in G' as sensor node i's *internal edge*, as the amount of flows on (i', i'') represents the amount of "traffic" that goes through i. For data node i, such traffic includes offloading its own packets, receiving packets from other nodes, and transmitting packets to others, as shown in Fig. 3(a). For storage node i, such traffic includes

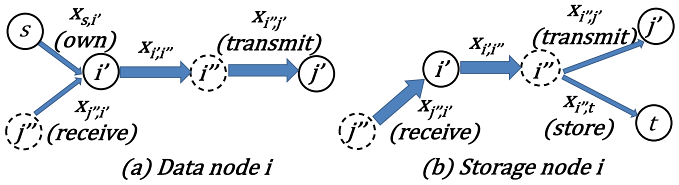


Fig. 3. (a) Data node i transmits its own data packets as well as data packets received from other nodes, shown as two thin traffic converging to one thick traffic at i. (b) Storage node i transmits or stores the data packets received from others, shown as one thick traffic diverging into two thin traffic at i.

receiving packets from others, transmitting packets to others, and storing packets into its local storage, as shown in Fig. 3(b). We have below observations about the flows, their incurred energy cost, and the flow conservations.

- *Observation 1 (for both data and storage nodes).* Given any node $i \in V$ and any of its neighboring node j (i.e., $(i,j) \in E$)[1], the number of data packets i receives from j is $x_{j'',i'}$, the number of data packets i transmits to j is $x_{i'',j'}$. Thus, node i's total receiving energy cost is $E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'}$ and its total transmission energy cost is $\sum_{j:(i,j)\in E}(E_i^t(j) \times x_{i'',j'})$.
- *Observation 2 (for data nodes).* For any data node $i \in V_d$, the number of its own data packets that it transmits is $x_{s,i'}$. As the data packets i transmits are either its own or received from others, we have $x_{s,i'} + \sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'} = x_{i',i''}$, where $x_{i',i''}$ is the amount of flow on edge (i', i'').
- *Observation 3 (for storage nodes).* For any storage node $i \in V - V_d$, the number of data packets it stores is $x_{i'',t} = \xi(i)$ (recall $\xi(i)$ is the number of data packets that are finally stored at storage node i). As the data packets i receives are either transmitted to other nodes or stored at i, we have $\sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'} + x_{i'',t} = x_{i',i''}$. The total storing energy cost of i is $E_i^s \times x_{i'',t}$.

Therefore the DRL can be represented as below:

[1]Note that it is not that $(i,j) \in E'$, as all the data nodes, storage nodes, and neighboring nodes are sensor nodes in G, not in G'.

$$\mathcal{D}(G) = \sum_{i=l+1}^{n} \left(\xi(i) \times E_i^{'}\right)$$
$$= \sum_{i \in V-V_d} \Bigg(x_{i'',t} \cdot \Big(E_i - E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} - \sum_{j:(i,j)\in E} \left(E_i^t(j) \times x_{i'',j'}\right) - E_i^s \times x_{i'',t} \Big) \Bigg). \quad (2)$$

QP Formulation. As $\mathcal{D}(G)$ is a concave quadratic expression, DRE^2 can be represented using below QP formulation (A):

$$(A) \qquad \max\ \mathcal{D}(G) \quad (3)$$
s.t.
$$x_{s,i'} = d_i, \quad \forall i \in V_d \quad (4)$$
$$x_{i'',t} \le m_i, \quad \forall i \in V - V_d \quad (5)$$
$$x_{s,i'} + \sum_{j:(i,j)\in E} x_{j'',i'} = x_{i',i''}, \quad \forall i \in V_d \quad (6)$$
$$x_{i',i''} = \sum_{j:(i,j)\in E} x_{i'',j'}, \quad \forall i \in V_d \quad (7)$$
$$\sum_{j:(i,j)\in E} x_{j'',i'} = x_{i',i''}, \quad \forall i \in V - V_d \quad (8)$$
$$x_{i',i''} = \sum_{j:(i,j)\in E} x_{i'',j'} + x_{i'',t}, \quad \forall i \in V - V_d \quad (9)$$
$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} \left(E_i^t(j) \times x_{i'',j'}\right) \le E_i, \quad \forall i \in V_d \quad (10)$$
$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} \left(E_i^t(j) \times x_{i'',j'}\right) + E_i^s \times x_{i'',t} \le E_i, \quad \forall i \in V - V_d \quad (11)$$

Eqn. 4 mandates that to achieve data resilience, data node i must be able to offload all its d_i number of data packets into the network. Inequality 5 shows the storage constraint of storage nodes. Eqns. 6 and 7 show the flow conservation for in-node and out-node of each data node, respectively. The combination of these two equations shows that the total number of packets transmitted by a data node equals the number of its own offloaded packets plus the number of its received packets (Observation 2). Eqns. 8 and 9 show the flow conservation for in-node and out-node of each storage node, respectively. The combination of them shows that the total number of packets received by a storage node equals the number of packets transmitted by it plus the number of packets stored locally by it (Observation 3). Inequalities 10 and 11 represent the energy constraint of data nodes and storage nodes respectively. Note that data nodes consume energy when receiving and transmitting data packets while storage nodes consume energy when receiving, transmitting, and saving data packets (Observations 1 and 3).

Generalized Edge Capacity Constraint. For either a data node or a storage node i and its corresponding internal edge (i', i''), although Inequalities 10 and 11 enforce energy constraint on i, they do not reveal the relationship between $x_{i',i''}$, the amount of flows on edge (i', i''), and E_i, the capacity of this edge. Their relationship does not simply follow the traditional edge capacity constraint $x_{i',i''} \le E_i$, that the amount of flows on an edge is less than or equal to the capacity on that edge. Before exploring such relationship for data nodes and storage nodes, we first define *generalized edge capacity constraint* as below. We will then show another forms of Inequalities 10 and 11 indeed exhibit such generalized edge capacity constraint.

Definition 2: (**Generalized Edge Capacity Constraint.**) In a flow network, given any edge (u, v), let $f(u, v)$ and $cap(u, v)$ denote its amount of flows and capacity respectively. The generalized capacity constraint is defined as $\sum_{k=1}^{f(u,v)} a_k \le cap(u, v)$, where a_k is the weight for the k^{th} flow on the edge. When $a_k = 1$ for all the flows, it becomes $f(u, v) \le cap(u, v)$, the traditional edge capacity constraint. □

For a data node i with $x_{i',i''}$ flows going through it, we need to figure out which node each flow goes to in order to compute the incurred energy of this flow (note that energy cost of receiving a packet does not depend on where the packet is from while energy cost of transmitting a packet depends on where the packet goes to). As the number of i's own offloaded data packet is $x_{s,i'}$, the number of data packets it relays (i.e., receives and then transmits) is $x_{i',i''} - x_{s,i'}$. Assume the QP (A) has computed that the k^{th} flow, $1 \le k \le x_{i',i''}$, goes to node $\gamma(k)$. Then the energy cost of i to offload its own $x_{s,i'}$ data packets is $\sum_{k=1}^{x_{s,i'}} E_i^t\big(\gamma(k)\big)$, and the energy cost of i to relay the rest $x_{i',i''} - x_{s,i'}$ packets is $\sum_{k=x_{s,i'}+1}^{x_{i',i''}} \big(E_i^r + E_i^t(\gamma(k))\big)$. Therefore Inequalities 10 is changed to:

$$\sum_{k=1}^{x_{s,i'}} E_i^t\big(\gamma(k)\big) + \sum_{k=x_{s,i'}+1}^{x_{i',i''}} \big(E_i^r + E_i^t\big(\gamma(k)\big)\big) \le E_i. \quad (12)$$

Similarly, Inequalities 11, which is for a storage node i with $x_{i',i''}$ flows going through it and $x_{i'',t}$ of them going to t (i.e., i stores $x_{i'',t}$ data packets), is changed to

$$\sum_{k=1}^{x_{i'',t}} (E_i^r + E_i^s) + \sum_{k=x_{i'',t}+1}^{x_{i',i''}} \big(E_i^r + E_i^t\big(\gamma(k)\big)\big) \le E_i. \quad (13)$$

Theorem 1: Eqns. 12 and 13 show that edges (i', i'') in G' follow generalized edge capacity constraint.

Proof: For Eqn. 12, let $a_k = E_i^t\big(\gamma(k)\big)$ when $1 \le k \le x_{s,i'}$, and $a_k = E_i^r + E_i^t\big(\gamma(k)\big)$ when $x_{s,i'} + 1 \le k \le x_{i',i''}$, then Eqn. 12 becomes $\sum_{k=1}^{x_{i',i''}} a_k \le E_i$, which is the generalized edge capacity constraint following Definition 2. Similarly, for Eqn. 13, let $a_k = E_i^r + E_i^s$ when $1 \le k \le x_{i'',t}$, and $a_k = E_i^r + E_i^t\big(\gamma(k)\big)$ when $x_{i'',t} + 1 \le k \le x_{i',i''}$, then Eqn. 13 becomes $\sum_{k=1}^{x_{i',i''}} a_k \le E_i$. ■

Corollary 1: There exists a special energy model under which Eqns. 12 and 13 demonstrate traditional edge capacity constraint where the number of flows on an edge is less than or equal to the edge capacity.

Proof: Let $a_k = 1, \forall k$ for Eqns. 12 and 13, we get $E_i^t\big(\gamma(k)\big) = E_i^r + E_i^t\big(\gamma(k)\big) = E_i^r + E_i^s = E_i^r + E_i^t\big(\gamma(k)\big)$. Thus $E_i^r = 0$ and $E_i^t\big(\gamma(k)\big) = E_i^s = 1$; that is, receiving a packet costs zero energy while transmitting and saving a packet each costs one unit of energy. ∎

Intuitively, Corollary 1 gives a special and simplified energy model where every node on a data packet's offloading path (including the data node, any intermediate relaying node, and the destination node) costs one unit of energy. This corresponds to the traditional edge capacity constraint where one amount of flow consumes one unit of edge capacity. By generalizing this widely used constraint in flow network, our work augments the existing network flow model and could have a potential impact on its related theory.

Solving QP. QP can be solved by the classic Wolfe's modified simplex method [13], which is based on solving a system of linear relations subject to complementarity conditions. There are many production QP solvers such as CGAL [1] and CPLEX [2]. We adopt CPLEX due to its performances. Besides, CPLEX can improve the efficiency of QP by allowing *gap tolerance* to find a feasible solution quickly (more in Section VI). As QP is NP-hard [15], we next design two time-efficient heuristic algorithms below and show via experiments that they perform close to the optimal QP solution.

V. Heuristic Algorithms

To maximize DRLs, an intuitive solution is to offload data packets to nodes with initial high energy levels, defined below.

Definition 3: (**High-Energy Storage Nodes.**) High-energy storage nodes, denoted as V_h, are the set of storages nodes with the highest initial energy levels that can store all the a data packets. More formally, we sort the $n-l$ storage nodes $V - V_d$ in non-ascending order of their initial energy level: $E_{v_1} \geq E_{v_2} \geq ... \geq E_{v_{n-l}}$. Then the top $k+1$ nodes $\{v_1, ..., v_k, v_{k+1}\}$ where $\sum_{i=1}^{k} m_{v_i} < a \leq \sum_{i=1}^{k+1} m_{v_i}$ is V_h. □

We design two algorithms below that center around how to offload data packets to V_h in an energy-efficient manner.

Network-Based Algorithm. For each high-energy storage node $i \in V_h$, Algo. 1 finds m_i data packets that are closest to i (in terms of energy consumption) and offloads them to i via the currently available shortest path. This takes place until all the a data packets are offloaded. It takes $O(n^2)$.

Algorithm 1: Network-Based Algorithm.

```
Input: A sensor network $G$ with $m_i$, $E_i$, data packets $D$;
Output: $\mathcal{D}(G)$;
1. Compute $V_h = \{v_1, ..., v_k, v_{k+1}\}$;
2. for $(1 \leq i \leq k)$
3.     Find the $m_i$ data packets that are closest to $v_i$ and
       offload them to $v_i$ via the current shortest path
       between each data packet and $v_i$;
4.     Update the energy levels of all the nodes on the path;
5. end for;
6. Offload each of the $a - \sum_{i=1}^{k} m_{v_i}$ data packets to $v_{k+1}$
   via the current shortest path and update the energy levels;
7. Compute $\mathcal{D}(G) = \sum_{i=l+1}^{n} \big(E_i^{'} \times \xi(i)\big)$;
8. RETURN $\mathcal{D}(G)$.
```

Minimum-Cost-Flow (MCF)-Based Algorithm. Although Algo. 1 saves energy by offloading data packets to their closest nodes in V_h, it does not consider global energy minimization in data offloading. Below we design Algo. 2 that minimizes total energy consumption in data offloading. It is a MCF-based algorithm [29] applied on another properly converted flow network $G''(V'', E'')$ from the sensor network $G(V, E)$. In MCF, each edge in the flow network has a capacity and a cost and the goal is to minimize the total cost of the flows.

Graph Conversion. The conversion from G to G'' is similar to the one of converting G to G' in Section IV including the node splitting, the edge connectivity, and the edge capacities, however with two differences. First, the super sink t only connects to the out-nodes i'' of the high-energy storage node $i \in V_h$, as we focus on offloading data packets to V_h. Second, we assign the costs of all directed edge (i'', j') as $E_{i,j} = E_i^t(j) + E_j^r$, the sum of node i's transmitting energy to j and node j's receiving energy, the costs of directed edges (j'', i') as $E_{j,i} = E_j^t(i) + E_i^r$, the sum of node j's transmitting energy to i and node i's receiving energy, and the costs of all other edges as zeros.

MCF ILP Formulation. With above transformation, the sensor network $G(V, E)$ in Fig. 2(a) is now converted to a flow network $G''(V'', E'')$ in Fig. 2(c). Let $x_{i,j}$ and $c_{i,j}$ be the amount of flows and cost on edge $(i, j) \in E''$, respectively. Below shows the MCF ILP formulation (B) that minimizes the total cost in $G''(V'', E'')$.

$$(B) \qquad \min \sum_{(i,j) \in E''} x_{i,j} \times c_{i,j} \tag{14}$$

s.t.

$$x_{s,i'} = d_i, \quad \forall i \in V_d \tag{15}$$

$$x_{i'',t} \leq m_i, \quad \forall i \in V_h \tag{16}$$

$$x_{s,i'} + \sum_{j:(i,j) \in E} x_{j'',i'} = \sum_{j:(i,j) \in E} x_{i'',j'}, \quad \forall i \in V_d \tag{17}$$

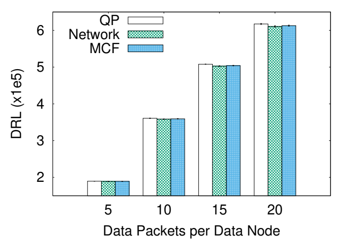
$$\sum_{j:(i,j) \in E} x_{j'',i'} = \sum_{j:(i,j) \in E} x_{i'',j'} + x_{i'',t}, \quad \forall i \in V_h \tag{18}$$

$$E_i^r \times \sum_{j:(i,j) \in E} x_{j'',i'} + \sum_{j:(i,j) \in E} \big(E_i^t(j) \times x_{i'',j'}\big) \leq E_i, \quad \forall i \in V_d \tag{19}$$

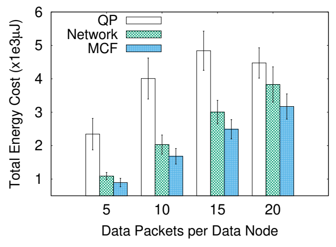
$$E_i^r \times \sum_{j:(i,j) \in E} x_{j'',i'} + \sum_{j:(i,j) \in E} E_i^t(j) \times x_{i'',j'} + E_i^s \times x_{i'',t} \leq E_i, \quad \forall i \in V_h \tag{20}$$

$$\sum_{j:(i,j) \in E} x_{j'',i'} = \sum_{j:(i,j) \in E} x_{i'',j'}, \quad \forall i \in V - V_d - V_h \tag{21}$$

$$E_i^r \times \sum_{j:(i,j) \in E} x_{j'',i'} + \sum_{j:(i,j) \in E} E_i^t(j) \times x_{i'',j'} \leq E_i, \quad \forall i \in V - V_d - V_h \tag{22}$$

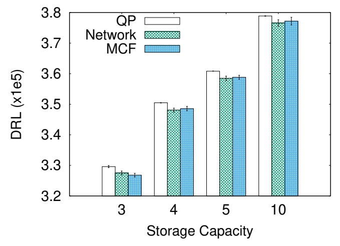


(a) Data resilience level.

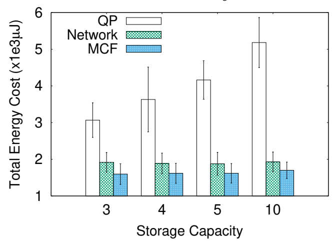


(b) Total Energy Cost.

Fig. 4. Small-scale comparison by varying d_i. $m_j = 5$.



(a) Data resilience level.



(b) Total Energy Cost.

Fig. 5. Small-scale comparison by varying m_j. $d_i = 10$.

The Constraints 15-20 are similar to those in QP (A), except that Constraints 16, 18, and 20 are now applied on V_h, as only high-energy storage nodes V_h can store data packets. Finally, we add two more constraints viz. Equation 21 and Inequality 22 to respectively address the flow conservation and energy constraint of all the storage nodes that are not in V_h. Algo. 2 below calls ILP (B) as a subroutine:

Algorithm 2: MCF-Based Algorithm.

```
Input: A sensor network $G$ with $m_i$, $E_i$, and $D$;
Output: $\mathcal{D}(G)$;
1. Compute $V_h = \{v_1, ..., v_k, v_{k+1}\}$;
2. Convert $G(V, E)$ to flow network $G''(V'', E'')$;
3. Compute ILP (B) on $G''$;
4. Compute $\mathcal{D}(G) = \sum_{i=l+1}^{n} \left(E_i^{'} \times \xi(i)\right)$;
5. RETURN $\mathcal{D}(G)$.
```

Theorem 2: Algo. 2 achieves minimum energy consumption in offloading a data packets to nodes in V_h.

Proof: We give a proof sketch due to space constraint. By applying MCF algorithm on G'', it guarantees that all the a overflow data packets are offloaded to V_h with minimum total energy cost while respecting the storage and energy constraints of sensor nodes. ∎

Solving MCF. We implement MCF ILP using CPLEX [2]. MCF can also be solved efficiently and optimally by combinatorial algorithms such as scaling push-relabel proposed by Goldberg [17]. Its time complexity is $O(a^2 \cdot b \cdot \log(a \cdot c))$, where a, b, and c are number of nodes, number of edges, and maximum edge capacity in the flow network, respectively.

All QP, Network- and MCF-based algorithms are fault-tolerant as they find the offloading paths in spite of node failures and network partitions caused by energy depletion of sensor nodes. We show that our fault-tolerant algorithms still achieve high DRLs and energy-efficiency in Section VI.

VI. Performance Evaluation

We compare the performance of different algorithms viz. QP-based (referred to as **QP**), Network-Based (referred to as **Network**), and MCF-Based (referred to as **MCF**). We write our own simulator in Java that takes as input the sensor network instance including the network topology, initial energy levels of each sensor nodes E_i, the data nodes and their overflow data packets d_i, the storage nodes and their storage capacities m_j, and generates various output files that conform to the format needed for CPLEX execution. We consider both small scale networks of 50 sensor nodes (10 of them are data nodes) and large scale of 100 nodes (20 of them are data nodes). The sensor nodes are uniformly distributed in a region of 1000m $\times$ 1000m. Each data node generates some number of overflow data packets, each of 512B, that to be offloaded into the network. For initial energy levels, we consider both *varying model*, where different sensors have different initial energy levels, and *uniform model*, where all sensor nodes have the same initial energy levels. Transmission range is 250m. In all plots, each data point is an average over ten runs, in each of which a different sensor network instance is generated. For fair comparisons of different algorithms, however, we use the same sensor network instance in the same run. The error bars indicate 95% confidence intervals.

Small-scale Comparison. As QP takes time to run, we compare them in small scale of 50 nodes with 10 data nodes. The initial energy levels are random numbers in $[2000\mu J, 4000\mu J]$. Fig. 4(a) varies d_i from 5, 10, 15, to 20 while setting m_j as 5. It shows that with the increase of d_i, the DRLs achieved by all algorithms increase, as more data packets are now stored in storage nodes. QP always achieves slightly higher DRLs than MCF, while MCF higher than Network most of the time. As QP is optimal and all three perform close, it demonstrates the efficacy of all three algorithms in achieving data resilience. Fig. 4(b) shows the total energy consumptions of three algorithms, among which MCF has the smallest. QP costs the most energy, though, as it sometimes detours (instead of following the shortest energy path) from sources to destinations in order to achieve high DRLs. Fig. 5 varies m_j while fixing $d_i = 10$. It shows again that QP achieves highest DRLs while MCF consumes the least amount of energy.

Gap Tolerance (%)	2	3	5	10	30
Execution Time (sec)	2014.91	826.3	11.98	13.44	15.92
	788.8	182.53	8.27	7.89	7.98
	1034.27	160.98	22.64	22.8	28.32

TABLE II
INVESTIGATING TOLERANCE GAP OF THE QP IN CPLEX.

Large-scale Comparison. Next we compare the algorithms in larger scale of 100 sensors, 20 of them are data nodes. We find that CPLEX cannot finish computing an instance of QP after more than 13 hours, thus decide to resort to its gap tolerance technique to improve its execution time.

Gap Tolerance of the QP. CPLEX [2] can be parameterized using a percentage value called *gap tolerance*, wherein

CPLEX stops once it finds a feasible solution within this percent of optimal. We first investigate the tradeoff between the execution time and solution quality of different gap tolerance values. Table II records the CPLEX execution time for different gap tolerances between 2% and 30%, for three randomly generated networks. As 2% can take more than half an hour, we choose the next value of 3% as the gap tolerance for the QP; i.e., for the rest comparisons the QP always achieves at least 97% of the optimal DRLs.

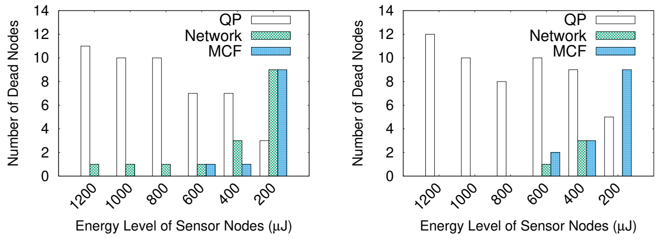
Fig. 6 compare the three algorithms by varying d_i from 50, 75, 100, to 125 with $m_j = 50$ and $E_i = 2500\mu J$. We observe that even with gap tolerance, QP still achieves slightly larger DRLs with much higher energy cost than the Network and MCF do, as it focuses on high DRLs and not on the energy efficiency to achieve them. It also shows that the energy cost of QP decreases when d_i increases from 100 to 125. This is rather counter-intuitive, as offloading more data packets should cost more energy. Our conjecture is that when there are multiple routes to offload a data packet without affecting the DRL achieved, the QP randomly chooses one to save execution time. When the network has more data packets to offload, however, choosing such a path randomly could negatively affect the DRL maximization. As such, the QP chooses more energy-efficient offloading paths thus decreasing the energy cost. Fig. 7 varies m_j and shows that the difference of DRLs by different algorithms seem to increase when increasing the storage capacity. As high energy nodes have more spaces to store data packets with increasing of m_j, the QP, being optimal, does a better job of utilizing the available spaces in order to maximize the DRL.

Investigating Fault-Tolerance. Finally we investigate the fault-tolerance of the three algorithms by finding the resultant dead nodes (i.e., nodes with depleted energy). We randomly generate one sensor network of 100 nodes, 20 of them are data nodes with $d_i = 50$. It sets the E_i of all the nodes as $1200\mu J$ and gradually decreases it while recording the number of dead nodes along the way. It stops until at least one of the algorithms fails to offload all the data packets. Fig. 8(a) sets m_j as 50 and shows that the algorithms can tolerate up to 11 dead nodes. However, as QP focuses more on DRLs and less on energy costs, it has more dead nodes than the other two most of the time. There is an interesting observation that when decreasing E_i, number of dead nodes increases in Network and MCF while decreases in QP. This is understandable for Network and MCF, as less initial energy levels contribute to more dead nodes in offloading the same amount of data packets. For QP, however, when E_i gets low, in order to maximize the DRL, it distributes the data more evenly into the network, thus resulting in lesser number of dead nodes. Fig 8(b) decreases m_j to 13, at which point the network is almost full after data offloading. In contrast to Fig 8(a), there is no dead nodes for Network when E_i is between $1200\mu J$ and $800\mu J$. When decreasing m_j, more storage nodes participate in the data offloading process, thus energy consumptions per node gets smaller, resulting in smaller number of dead nodes. Note that when $E_i = 200\mu J$, Network cannot offload all the data packets thus does not appear in the plot.

Last, Fig. 9(a) and (b) study the fault-tolerance of algorithms at $E_i = 1200\mu J$ by varying d_i and m_j respectively. It is interesting to notice that with the increase of m_j, the number of dead nodes increases for both Network and MCF while decreasing for QP. For Network and MCF, as the number of destination nodes gets smaller with increase of m_j, less number of them participate in the data offloading process, depleting their energies more quickly. For QP, with the increase of m_j it can distribute data packets to nodes more evenly (by taking longer time), thus reducing the number of dead nodes.

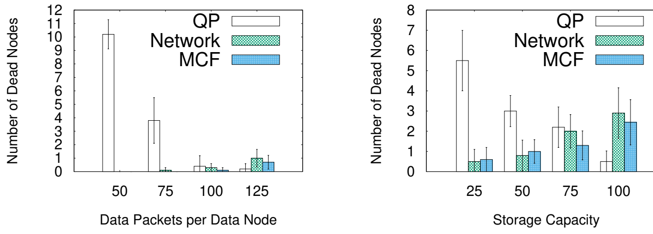
VII. Conclusion and Future Work

We solved a new data resilience problem that uniquely arises from emerging sensor network applications deployed in the



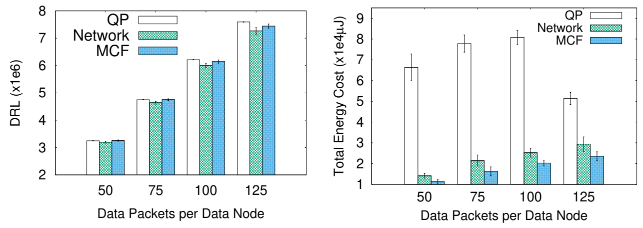
(a) $m_j = 50$. (b) $m_j = 13$.

Fig. 8. Fault-tolerance of three algorithms by varying E_i.



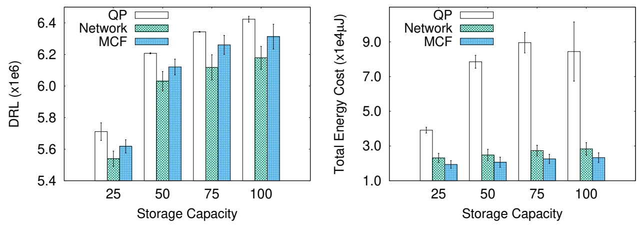
(a) Varying d_i. $m_j = 50$. (b) Varying m_j. $d_i = 100$.

Fig. 9. Fault-tolerance of three algorithms at $E_i = 1200\mu J$.



(a) Data resilience level. (b) Total Energy Cost.

Fig. 6. Large-scale comparison by varying d_i. $m_j = 50$.



(a) Data resilience level. (b) Total Energy Cost.

Fig. 7. Large-scale comparison by varying m_j. $d_i = 100$.

extreme environment. We designed a QP-based optimal algorithm and two time- and energy-efficient heuristic algorithms. Although sensor network research has been around for more than two decades, data resilience in our network setup has not been studied in any existing literature. We uncovered a generalized edge capacity constraint, wherein the consumed capacity on an edge is the linear combination of its flows. This generalizes the well-accepted edge capacity constraint in traditional network flows. One limitation of our approach seems to be that the network scenario including the data nodes and their overflow data packets are known beforehand. We believe this can be overcome by periodically executing our algorithms whenever new data nodes arise, thus solving the real-time version of this data resilience problem. As a future work, we plan to focus on a few specific topologies such as stars and trees, and investigate if optimal or approximate time-efficient algorithms exist. Finally, under which conditions such that not all the data packets can be offloaded and then how to achieve data resilience for such a fraction of data packets (i.e., event goodput [3]) remain two relevant new problems.

Acknowledgment

This work was supported by NSF Grant CNS-1911191.

References

[1] Cgal 5.0.2 - linear and quadratic programming solver. https://doc.cgal.org/latest/QP_solver/index.html.

[2] Ibm cplex optimizer. https://www.ibm.com/analytics/cplex-optimizer.

[3] Models and performance evaluation of event goodput in sensor platforms. *Computer Networks*, 123:119 – 136, 2017.

[4] G. Aathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 5(4), 2009.

[5] M. Albano and J. Gao. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proc. of ALGOSENSOR*, 2010.

[6] B. Alhakami, B. Tang, J. Han, and M. Beheshti. Dao-r: Integrating data aggregation and offloading in sensor networks via data replication. *International Journal of Sensor Networks*, 29(2):134 – 146, 2019.

[7] Y. Chen and B. Tang. Data preservation in base station-less sensor networks: A game theoretic approach. In *Proc. of the 6th EAI International Conference on Game Theory for Networks (GameNets 2016).*

[8] G. Citovsky, J. Gao, J. S. B. Mitchell, and J. Zeng. Exact and approximation algorithms for data mule scheduling in a sensor network. In *Proc. of ALGOSENSORS 2015.*

[9] R. W. L. Coutinho, A. Boukerche, and S. Guercin. Performance evaluation of candidate set selection procedures in underwater sensor networks. In *Proc. of IEEE ICC 2019.*

[10] L. Doherty, K. S. J. pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *Proc. of IEEE INFOCOM 2001*, 2001.

[11] S. Edwards, T. Murray, T. O'Farrell, I. C. Rutt, P. Loskot, I. Martin, N. Selmes, R. Aspey, T. James, S. L. Bevan, and T. Baugé. A High-Resolution Sensor Network for Monitoring Glacier Dynamics. *IEEE SENSORS JOURNAL*, 14(11):3926–3931, 2013.

[12] C. A. Floudas and V. Visweswaran. Quadratic optimization. *Nonconvex Optimization and Its Applications*, 2:217–269, 1995.

[13] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956. Web. 4 June 2015.

[14] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):11–25, October 2001.

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

[16] A. Ghose, J. Grossklags, and J. Chuang. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proc. of MDM*, 2003.

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

[18] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of HICSS 2000.*

[19] X. Hou, Z. Sumpter, L. Burson, X., and B. Tang. Maximizing data preservation in intermittently connected sensor networks. In *Proc. of IEEE MASS 2012*, pages 448–452.

[20] Y. Huang, J. F. Martínez, J. Sendra, and L. López. Resilient wireless sensor networks using topology control: A review. *Sensors (Basel)*, 15(10):24735–24770, 2015.

[21] A. Kamra, J. Feldman, V. Misra, and D. Rubenstein. Growth codes: Maximizing sensor network data persistence. In *Proc. of SIGCOMM*, 2006.

[22] J. Lee, W. Chung, and E. Kim. A new range-free localization method using quadratic programming. *Computer Communications*, 34(8):998–1010, 2011.

[23] F. Liu, M. Lin, Y. Hu, C. Luo, and F. Wu. Design and analysis of compressive data persistence in large-scale wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(10):2685–2698, 2015.

[24] L. Liu and R. Wang. Message dissemination for throughput optimization in storage-limited opportunistic underwater sensor networks. In *Proc. of SECON 2016.*

[25] L. Luo, C. Huang, T. Abdelzaher, and J. Stankovic. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *Proc. of INFOCOM 2007.*

[26] D. E. Phillips, M. Moazzami, G. Xing, and J. M. Lees. A sensor network for real-time volcano tomography: System design and deployment. In *Proc. of IEEE ICCCN 2017.*

[27] D. Puccinelli and M. Haenggi. Reliable data delivery in large-scale low-power sensor networks. *ACM Trans. Sen. Netw.*, 6(4):28:1–28:41, 2010.

[28] M. Rahmati and D. Pompili. Uwsvc: Scalable video coding transmission for in-network underwater imagery analysis. In *Proc. of IEEE MASS 2019.*

[29] K. A. Ravindra, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[30] R. Sugihara and R. K. Gupta. Path planning of data mules in sensor networks. *ACM Trans. Sen. Netw.*, 8(1):1:1–1:27, 2011.

[31] Z. Sun and I. F. Akyildiz. On capacity of magnetic induction-based wireless underground sensor networks. In *Proc. of INFOCOM*, 2012.

[32] Rui Tan, Guoliang Xin, Jinzhu Chen, Wen-Zhan Song, and Renjie Huang. Fusion-based volcanic earthquake detection and timing in wireless sensor networks. *ACM Transaction on Sensor Networks (ACM TOSN)*, 9, 2013.

[33] B. Tang. dao^2: Overcoming overall storage overflow in intermittently connected sensor networks. In *Proc. of IEEE INFOCOM 2018.*

[34] B. Tang, N. Jaggi, and M. Takahashi. Achieving data k-availability in intermittently connected sensor networks. In *Proc. of IEEE ICCCN 2014.*

[35] B. Tang, N. Jaggi, H. Wu, and R. Kurkal. Energy efficient data redistribution in sensor networks. *ACM Transactions on Sensor Networks*, 9(2):1–28, May 2013.

[36] A. Tinka, I. Strub, Q. Wu, and A. M. Bayen. Quadratic programming based data assimilation with passive drifting sensors for shallow water flows. In *Proc. of IEEE Conference on Decision and Control (CDC)*, 2009.

[37] L. Wang, Y. Yang, D. K. Noh, H. L., T. Abdelzaher, M. Ward, and J. Liu. Adaptsens: An adaptive data collection and storage service for solar-powered sensor networks. In *Proc. of the 30th IEEE Real-Time Systems Symposium (RTSS 2009).*

[38] X. Xu, H. Zhang, T. Li, and L. Zhang. Achieving resilient data availability in wireless sensor networks. In *Proc. of IEEE ICC 2019 Workshops.*

[39] X. Xue, X. Hou, B. Tang, and R. Bagai. Data preservation in intermittently connected sensor networks with data priorities. In *Proc. of IEEE SECON 2013*, pages 65–73.

[40] Yong Yang, Lili Wang, Dong Kun Noh, Hieu Khac Le, and Tarek F. Abdelzaher. Solarstore: enhancing data reliability in solar-powered storage-centric sensor networks. In *Proc. of MobiSys 2009*, pages 333–346, 2009.

[41] M. Z. Zamalloa and B. Krishnamachari. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Transactions on Sensor Networks*, 3(2):1277–1280, 2007.