On the Performance of Nash Equilibria for Data Preservation in Base Station-less Sensor Networks

Giovanni Rivera†, Yutian Chen*, Bin Tang†
†Department of Computer Science, California State University Dominguez Hills
grivera64@toromail.csudh.edu, btang@csudh.edu
*Economics Department, California State University, Long Beach, Yutian.Chen@csulb.edu

***Abstract*—Base station-less sensor networks (BSNs) refer to emerging sensing applications deployed in challenging environments (e.g., underwater exploration). As installing a base station in such an environment is not feasible, data generated in the BSN will be preserved in the network before being collected by the uploading opportunities. This process is called *data preservation* in the BSN. Considering that sensor nodes are intelligent and selfish, this paper studies the *Nash Equilibrium* (NE) for data preservation in BSNs. We design a suite of data preservation algorithms, examine whether they achieve NEs, and rigorously analyze the performance of the NEs using existing (i.e., *price of anarchy and price of stability*) and our own designed metrics (i.e., *rate of efficiency loss*). We find that a minimum cost flow-based algorithm produces a NE that achieves the social optimal with minimum energy consumption in the data preservation process. We show the NE from one of our straightforwardly designed greedy algorithms achieves a price of anarchy of 3. On the other hand, we prove that a greedy algorithm always exists (although non-straightforward), achieving the socially optimal NE. Finally, we conduct extensive simulations to investigate the performances of various data preservation NEs and validate our theoretical results under different network parameters.**

Keywords – Nash Equilibria, Efficiency Loss, Price of Anarchy, Price of Stability, Data Preservation, Base Station-less Sensor Networks.

I. Introduction

Background and State-of-the-Arts. In recent years, *base station-less sensor networks* (BSNs) have drawn much attention from the research community [28], [30], [17], [29], [35]. BSNs refer to an array of emerging data-intensive sensing applications with various sensing capabilities, including underwater or ocean exploration [19], [13], [37], volcano eruption monitoring [12], and offshore oil and gas refineries [1], [34]. As the above applications are all deployed in challenging and remote environments, it is not feasible to deploy high-power and high-storage data-collecting base stations in or near the sensing area. The sensory data generated inside the BSN will be collected when uploading opportunities arise (e.g., the periodic visits of autonomous underwater vehicles (AUVs) [6] and robots [33], [18]). One important function of BSNs is thus storing large volumes of sensory data inside the network between two uploading opportunities. BSNs in challenging environments differ significantly from traditional sensor networks and IoT applications in a friendly environment, wherein base stations are always available to collect sensory data.

Our BSN model is as follows. Some sensor nodes (referred to as *source nodes*) are close to the events of interest and constantly generate large amounts of sensory data, thus depleting their storage spaces. The newly generated data that cannot be stored in the source nodes' local storages is called *overflow data*. To avoid data loss, such overflow data must be offloaded to other sensor nodes in the BSN with available storage (referred to as *storage nodes*). We refer to the process of offloading overflow data from source to storage nodes as *data preservation in BSNs*. Various research has been conducted to achieve different objectives for data preservation in the BSN, including energy minimization [29], [35], data aggregation [28], [30], and network lifetime maximization [17].

Motivation and Challenges. In this paper, we consider that sensor nodes could behave selfishly in the data preservation process for the following three reasons. First, with the strides made in sensor technologies over the past decade, sensor systems have become more intelligent [5]. Equipped with artificial intelligence and machine learning technologies, intelligent sensors could actively perceive, learn, and reason on top of the sensing, computing, and communication performed by the conventional sensors [5], [11]. Second, many emerging IoT sensing applications are unrestricted within a building or a small geographical area. They could be on a global scale and distributed in nature, with the sensor nodes being controlled by different entities, each aiming to pursue its self-interest and maximize its benefit. For example, in offshore oil and gas refineries [1], [34], the IoT sensors that detect possible oil and gas leaks could belong to different companies and countries with other goals and incentives [1], [34]. Third, the sensor nodes are usually resource-constrained, with limited processing power, battery energy, and storage spaces. As such, the intelligent sensor nodes in the BSN can behave selfishly, only to conserve their own resources and have little incentive to participate in the data preservation process [10].

This paper applies game-theoretical techniques to analyze sensor nodes' selfish data preservation behavior in BSNs. Game theory, which studies collaboration and competition among multiple intelligent agents (i.e., players), has increasingly attracted attention from the research community [3]. In particular, we will focus on Nash Equilibrium (NE) [22], [23], a game-theoretical solution that characterizes selfish players' optimal strategies in a non-cooperative game. NEs

have become an essential solution concept for multi-agent reinforcement learning [7], a burgeoning research field studying the interaction of multiple learning agents in applications such as autonomous driving and automated warehousing. Therefore, we focus on studying the NEs of selfish data preservation in BSNs. However, as a NE usually does not yield socially optimal due to the selfish players [21], it needs to study the performance degradation it brings to data preservation in the BSN. In this paper, we attempt to answer the following question: *Can we design data preservation algorithms that achieve NEs with system performance guarantees (i.e., optimal or constant factor performance ratio) in the BSN?*

Our Contributions. We design a suite of data preservation algorithms and examine whether they achieve NEs in the BSN. We rigorously analyze the performance of the achieved NEs and derive their efficiency loss using the *price of anarchy* (PoA) and *price of stability* (PoS) [21], two main concepts in economics and game theory that measure the system degradation due to selfishness. In particular, we show that our minimum cost flow-based algorithm achieves not only NEs but also minimum preservation cost (i.e., social optimal), thus suffering no system degradation that is common for systems with selfish players. In addition, we prove that there always exists a greedy algorithm (albeit non-straightforward) that produces NEs of socially optimal data preservation (i.e., PoS $= 1$). For one of our designed greedy algorithms, we show that the PoA of its NE is guaranteed to be less than 3. As PoA and PoS focus on analyzing the upper- and lower-bound performance of NEs, we further design a metric called *rate of efficiency loss* (REL) to measure the performance loss of any data preservation NEs in our proposed algorithms. Finally, we conduct extensive simulations to investigate the efficiency loss of various NEs and validate our theoretical results under different network parameters using all three metrics.

Paper Organization. Section II reviews the related literatures. Section III formulates the data preservation problem (i.e., DPP) and introduces data preservation NEs. In Section IV, we design a suite of data preservation algorithms, examine if they achieve NEs, and analyze the performance loss of the resultant NEs. In Section V, we conduct extensive simulations to compare different NEs and validate our theoretical results. Section VI concludes the paper with future work.

II. Related Work

Game theory techniques have been extensively applied to solve research problems in computer networks in general and wireless ad hoc and sensor networks, in particular, [25], [26], [3], [8]. Stankovic et al. [27] focused on a coordination problem in mobile sensor networks and studied how to reach a distributed convergence of a NE. Voulkidis et al. [32] proposed a coalitional game-theoretic algorithm that maximizes the lifetime of sensor networks. Niyato et al. [24] studied the solar-powered sensor network that uses a sleep and wakeup strategy for energy conservation. They modeled nodes' sleep and wake-up strategies as a bargaining game and derived the NE as the game's solution. Attiah et al. [4] proposed an evolutionary routing game for energy balance in wireless sensor networks. They derived a mixed strategy NE that reduces the load and avoid collisions on the most used routes in a distributed manner. Kannan et al. [20] studied energy-constrained information routing in sensor networks and derived NE for different payoff models and utility functions.

However, none of the above works tackled data preservation in BSNs. The above results assumed a traditional sensor network scenario where a base station is always available; thus, the generated sensory data packets can be transmitted and uploaded back to the base station. They mainly focused on incentivizing the nodes on the shortest path or minimum spanning tree among the data source and the base station. In contrast, in data preservation in the BSN, the overflow data packets must be stored in the BSN for some time, waiting for the uploading opportunities. Therefore, it needs to find the storage nodes for the data packets and then route them to the storage nodes energy-efficiently. We show that such a unique data preservation model gives rise to a minimum-cost network flow (MCF)-based solution [2]. MCF generalizes the shortest path and minimum spanning tree used in existing works.

The only works that applied game theory to analyze data preservation in the BSN are [36], [9]. They used algorithmic mechanism design techniques to incentivize selfish sensor nodes to participate in the data preservation. However, they did not study NEs in data preservation. NE is the most fundamental game theory concept that determines the optimal strategies for selfish players who are not incentivized. We design a suite of data preservation algorithms that reach NEs. In particular, we show that our MCF-based algorithm achieves both NEs and minimum preservation cost, thus suffering no system degradation expected in NEs. We also prove that the PoA is no more than 3 for one greedy algorithm and that a greedy (although not straightforward) algorithm always exists that achieves NEs with PoS $= 1$.

III. Data Preservation Nash Equilibrium in the BSN

In this section, we formulate the data preservation problem in the BSN and define its data preservation Nash Equilibrium.

A. Data Preservation Problem (DPP)

Network Model. We model a BSN as an undirected connected graph $G(V, E)$, where $V = V_s \cup V_r$ includes a set of source nodes V_s and a set of storage nodes V_r. Assume $|V| = n$, $|V_s| = k$, $|V_r| = q$, where $k + q = n$, and denote $V_s = \{S_1, S_2, ..., S_k\}$, and $V_r = \{R_1, R_2, ..., R_q\}$. Let $d_i > 0$ denote the number of overflow data packets generated at source node $S_i \in V_s$. Let $d = \sum_{i=1}^{k} d_i$ be the total number of overflow data packets and let $D = \{D_1, D_2, ..., D_d\}$ denote the set of d data packets. Let $S_{s(j)} \in V_s$, $1 \leq j \leq d$, denote the source node where D_j is generated. Let $m_j > 0$ be the available free storage space (regarding the number of data packets) at storage node $R_j \in V_r$. We assume that $\sum_{j=1}^{q} m_j \geq d$; otherwise, the BSN has insufficient space to store all the overflow data packets. Each packet is a bits.

TABLE I
NOTATION SUMMARY

Notation	Description
$G(V, E)$	A BSN graph, $V = V_s \cup V_r$, $\|V\| = n$
V_s	Set of $\|V_s\| = k$ source nodes, $V_s = \{S_1, S_2, ..., S_k\}$
V_r	Set of $\|V_r\| = q$ storage nodes, $V_r = \{R_1, R_2, ..., R_q\}$
d_i	The number of overflow packets in source node $S_i \in V_s$
d	Total number of overflow data packets, $d = \sum_{i=1}^{k} d_i$
m_j	Storage capacity of storage node $R_j \in V_r$
D	Set of d overflow data packets, $\|D\| = d$
$s(j)$	Source node of data packet D_j, $1 \leq j \leq d$
$E_u^t(v)$	Transmission energy of u to transmit one packet to v
E_u^r	Receiving energy of u to receive one data packet
f	Data preservation function
c_i	Preservation costs of all packets at source node S_i
$c(i, j)$	Shortest path cost between nodes S_i and R_j
A_i	Set of data preservation strategies of source node S_i
$s_i \in A_i$	A data preservation strategy of S_i
A	Set of data preservation strategies of all source nodes V_s
$s \in A$	A data preservation strategy profile
u_i	Payoff received by S_i under strategy profile s, $u_i = -c_i$
$c(s)$	Total data preservation cost of the strategy profile s
$Eq \subseteq A$	The set of data preservation NEs
$s^* \in Eq$	A data preservation NE
REL(s^*)	Rate of efficiency loss of a data preservation NE s^*

Following the first-order radio model [16], when node u sends a data packet to its neighbor v over their distance $l_{u,v}$, the amount of *transmitting energy* spent by u is $E_u^t(v) = a \cdot \epsilon^a \cdot l_{u,v}^2 + a \cdot \epsilon^e$ and the amount of *receiving energy* spent by v is $E_v^r = a \cdot \epsilon^e$. Here, ϵ^a and ϵ^e are the energy consumption of transmitting one bit on a node's transmit amplifier and sending or receiving one bit on its circuit, respectively; their values are 100 $pJ/bit/m^2$ and 100 nJ/bit respectively, following [16]. Given an edge $(u, v) \in E$, we define its weight $w(u, v)$ as the total energy consumption of sending and receiving one packet from u to v; that is, $w(u, v) = E_u^t(v) + E_v^r$.

Problem Formulation of DPP. We define a *preservation function* as $f : D \to V_r$, showing $D_j \in D$ is offloaded from its source node $S_{s(j)} \in V_s$ to a storage node $R_{f(j)} \in V_r$ along the shortest path between them (referred to as *data preservation path*). Let $c(i, j)$ be the cost of the data preservation path between source node S_i and storage node R_j and c_i be the energy cost of offloading all the d_i data packets at source node S_i. The goal of the DPP is to find an f to offload all the overflow packet D, such that the *total preservation cost* $C = \sum_{i=1}^{k} c_i = \sum_{j=1}^{d} c\big(s(j), f(j)\big)$ is minimized under the storage constraint of storage nodes: $|\{j | 1 \leq j \leq d, f(j) = i\}| \leq m_i$, $\forall R_i \in V_r$. Table I shows all the notations.

EXAMPLE 1: Fig. 1 shows a linear BSN with 6 nodes viz. A to F. The energy cost on each edge is 1. Nodes B, D, and F are source nodes; each has one overflow data packet; nodes A, C, and E are storage nodes, each with a storage capacity of 1. The optimal data preservation solution is to offload B's packet to A, D's to C, and F's to E, resulting in a minimum preservation cost of 3. Any other solution is not optimal. □

B. Data Preservation Nash Equilibrium

In our data preservation game, all the k source nodes $\{S_1, S_2, \ldots, S_k\}$ are the players, as they wish to offload their overflow data packets into the BSN for preservation. Each source node will compensate all other nodes, including storage and other source nodes, for their incurred energy costs participating in the data preservation. Source node S_i has a set of *data preservation strategies* A_i, each indicating how many of its d_i data packets are offloaded to which storage node following the shortest path between them. Let $s_i \in A_i$ denote a particular strategy chosen by source node S_i and s_{-i} the set of strategies chosen by all other source nodes in the game. Let $A = A_1 \times A_2 ... \times A_k$ denote the set of *data preservation strategy profiles* of all the source nodes and $s = \{s_i, s_{-i}\} \in A$ one such strategy profile.

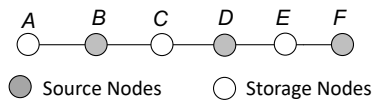


Fig. 1. Illustrating the DPP.

The *utility function* u_i of S_i is defined as $u_i : A \to \mathbb{R}^+$. Given a strategy profile $s \in A$, S_i receives a corresponding utility of $u_i = -c_i$, where c_i is the incurred preservation cost for all its d_i data packets under s. Note that the utilities of storage nodes and other source nodes (if any) participating in source node S_i's data preservation are always zeros as they are compensated by S_i with the same amount as their energy costs. Let $c(s)$ denote the total preservation cost of all the data packets under s, $c(s) = \sum_{i=1}^{k} c_i$.

We are interested in a special data preservation strategy profile $s^* = \{s_i^*, s_{-i}^*\} \in A$ where no source node S_i, $1 \leq i \leq k$, has any incentive to deviate from its chosen strategy s_i^* given the strategies s_{-i}^* of other source nodes; i.e., $u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$ for all $s_i \in A_i$. We refer to this strategy profile as a *data preservation NE*. It indicates a stable state condition wherein any source node perceives its data preservation strategy as optimal based on others' strategies and thus does not want to switch its strategy. Let $Eq \subseteq A$ denote all the data preservation NEs. In a data preservation game, each player S_i acts selfishly to maximize its utility u_i, equivalent to minimizing its data preservation cost c_i. We quantify the performance of a data preservation NE as follows.

Definition 1: (Price of Anarchy (PoA) and Price of Stability (PoS) in Data Preservation NEs) Giving a DPP instance with all the players' strategy profiles A and NEs Eq, its PoA (and PoS) is defined as the ratio between the total preservation cost of its *best (and worst) NE* and the social optimal (i.e., minimum) data preservation cost. That is, $\text{PoA} = \frac{\max_{s \in Eq} c(s)}{\min_{s \in A} c(s)}$ and $\text{PoS} = \frac{\min_{s \in Eq} c(s)}{\min_{s \in A} c(s)}$. □

PoA and PoS [21] are well-known metrics used in economics and game theory to measure how a system's efficiency degrades due to its agents' selfish behavior. However, as PoA and PoS focus on the upper- and lower-bound performance of NEs, they cannot be used to measure the performance loss of specific data preservation NEs from our algorithms proposed in Section IV. It is thus necessary to design a more relevant

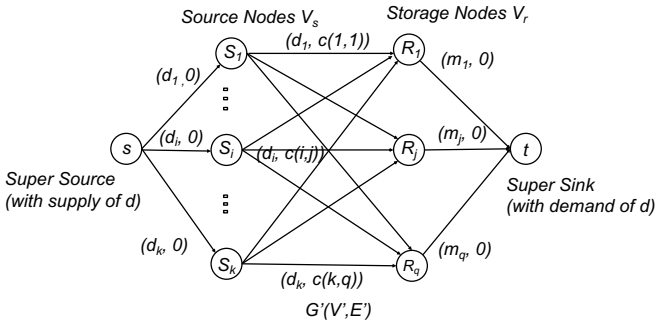


Fig. 2. DPP in BSN graph $G(V, E)$ is equivalent to MCF problem in transformed flow network $G'(V', E')$. The first number in each parenthesis is the edge's capacity, and the second is its cost.

metric to achieve this. We introduce the below definition.

Definition 2: (**Rate of Efficiency Loss (REL) of a Data Preservation NE**) Giving a DPP instance with all the players' strategy profiles A and data preservation NEs Eq, the REL of any NE $s^* \in Eq$, denoted as REL(s^*), is defined as the ratio between its preservation cost and the minimum data preservation cost; REL$(s^*) = \frac{c(s^*)}{\min_{s \in A} c(s)}$. □

Besides quantifying the performance loss of a NE, the REL is also used in Section V to validate our theoretical finding of PoAs for different data preservation algorithms.

IV. Algorithms for DPP

In this section, we present a suite of data preservation algorithms viz. *minimum cost flow* (MCF-based) [29], *Greedy* (a node-based and a cost-based), and *Random*, which serves as the benchmark. We prove they have achieved NEs (or under some conditions) and analyze their PoA and PoS.

A. MCF-Based Algorithm [29]

We first introduce MCF. Given a directed graph $G' = (V', E')$ with a super source node s and a super sink node t, each edge $(u, v) \in E'$ has a capacity $c_{u,v}$ and a cost $d_{u,v}$. Let $f(u, v)$ be the flow on edge $(u, v) \in E'$. The goal of MCF is to find a flow function f to minimize the total cost of transmitting y amount of flow from s to t, i.e. $\Sigma_{(u,v)\in E'}\big(d_{u,v} \cdot f(u, v)\big)$, subject to (i) capacity constraint: $f(u, v) \leq c_{u,v}, \forall (u, v) \in E'$ and (ii) flow conservation constraint: $\sum_{u\in V'} f(u, v) = \sum_{u\in V'} f(v, u)$, for each $v \in V' - \{s, t\}$. Our MCF-based data preservation algorithm consists of the below two steps.

Step 1: Graph Conversion. We first convert the BSN graph $G(V, E)$ to a flow network $G'(V', E')$ in Fig. 2 as follows.

First, we construct the nodes in G' as $V' = \{s\} \cup \{t\} \cup V_s \cup V_r$, where s is the super source node and t of the super sink node in the flow network. Recall that V_s and V_r are the sets of source and storage nodes, respectively.

Second, we construct the edges in G' as $E' = \{(s, S_i)\} \cup \{(S_i, R_j)\}\} \cup \{(R_j, t)\}\}$, where $S_i \in V_s$ and $R_j \in V_r$. This is a complete bipartite graph between V_s and V_r.

Third, for each edge (s, S_i), set its capacity as d_i, the number of data packets at S_i, and the cost as 0. For each edge (R_j, t), set its capacity as m_j, the storage capacity of storage node R_j, and its cost as 0. For each edge (S_i, R_j), set its capacity as d_i and cost as $c(i, j)$. Here, $c(i, j)$ is the energy cost of offloading one data packet from node S_i to R_j along their shortest path.

Finally, we set the supply at s and demand at t as d, the total number of data packets in the BSN.

Step 2: Data Preservation Computation. We then compute the data preservation by applying the MCF-Algorithm on the above flow network; the resulting flows show the data preservation solution. We adopt the implementation by Goldberg [14], a scaling push-relabel algorithm with the highest performance among all the algorithms. It has the time complexity of $O(l^2 \cdot m \cdot \log(l \cdot n))$, where l, m, and n are the number of nodes, number of edges, and maximum edge capacity of $G'(V', E')$. Next, we show that our MCF-based algorithm is an optimal DPP algorithm and achieves NE.

Theorem 1: The MCF-based algorithm gives a NE with optimal total preservation cost. Thus its PoS = PoA = 1.

Proof: Its optimality has been proved by Tang et al. [29], which shows that DPP in BSN graph $G(V, E)$ is equivalent to MCF in flow network $G'(V', E')$. Thus DPP can be solved optimally and efficiently by the MCF algorithms such as the scaling push-relabel algorithm by Goldberg [14].

Next, we show that data preservation strategies computed by the MCF reach a NE for all the source nodes. Suppose a source node in the MCF solution has an incentive to unilaterally switch its data preservation strategy; it must be that the preservation cost of the new strategy is smaller than that of the MCF-based strategy. This results in a smaller total preservation cost, contradicting the optimality of the MCF. Therefore, the outcome of MCF is a NE. As it also minimizes the system data preservation cost, PoS = PoA = 1. ∎

B. Greedy Algorithms and Their PoAs

Next, we introduce two heuristic data preservation greedy algorithms viz. Node-based (i.e., Algo. 1) and Distance-based (i.e., Algo. 2). Although not optimal, they can be directly applied to the BSN graph and are more time-efficient than the MCF-based. We analyze their PoAs and PoSs as well.

Node-based Greedy Algorithm. Algo. 1 works as follows. Each source node i offloads its d_i data packets to its closest storage nodes with available spaces until all the data packets in the BSN are offloaded. Finding the shortest storage node between any pair of source and storage nodes takes $O(|E| + |V| \cdot \log|V|)$. Therefore, the time complexity of Algo. 1 is $O\big(k \cdot q \cdot \big(|E| + |V| \cdot \log(|V|)\big)\big)$.

Algorithm 1: The Node-based Greedy Algorithm.
Input: A BSN graph $G(V, E)$;
Output: Data preservation paths $f : D \to V_r$;
Notations: l_i: number of un-offloaded data packets at S_i;
h_j: number of available storage spaces at R_j;
1. **for** $(1 \leq i \leq k)$ // current data packets at S_i
2. $l_i = d_i$;

3. **for** $(1 \le j \le q)$ // current storage space at R_j
4. $h_j = m_j$;
5. **for** $(1 \le i \le k)$ // each source node S_i
6. **while** $(l_i > 0)$
7. Find the storage node in V_r closest to S_i that still has available spaces, say R_j;
8. Offload $\min(l_i, h_j)$ packets to R_j along the the preservation path between S_i and R_j;
9. $l_i = l_i - \min(l_i, h_j)$, $h_j = h_j - \min(l_i, h_j)$;
10. **end while;**
11. **end for;**
12. **RETURN** $f : D \to V_r$.

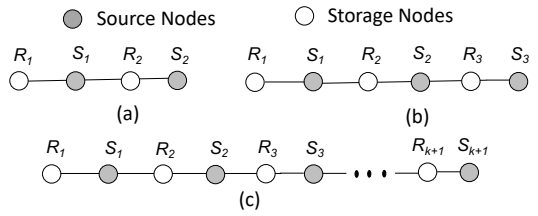


Fig. 3. Illustrating PoA $= H(d)$ for NEs resulted from Algo. 1.

Theorem 2: Algo. 1 reaches a NE. It has PoA $= H(d)$, where $H(d) \equiv 2^{d-1} + 2^{d-2} + ... + 2^0$.

Proof: According to the execution of Algo. 1, each source node minimizes its cost when it moves, and the future decisions made by other source nodes do not affect its cost. Thus no source node has incentive to unilaterally deviate from its greedy strategy, and the outcome is a NE.

We prove that PoA $= H(d)$ for Algo. 1 by induction. Since any data preservation problem can be rewritten into the scenario when each source node has one overflow data and each storage node has one capacity, we focus such a case in the proof. In what follows, we denote data nodes as $S_1, S_2, S_3, ...$ In social optimal, the corresponding destination node to which each data node offloads its data is $R_1, R_2, R_3, ...$ Denote the distance between each source node and its storage node used in social optimal as $l_1, l_2, l_3, ...$; the minimized total data preservation cost is thus $C^* = \Sigma_{i \in D} l_i$. The total preservation cost of a NE from Algo. 1 is denoted as C^N. In addition, we use xy to denote the shortest path between nodes x and y, and $|xy|$ to indicate the distance of the shortest path.

(1). Consider $d = 1$. The source node offloads along the shortest path in Algo. 1, and the outcome is socially optimal. Thus PoA $= 1 = 2^0 = H(1)$.

(2). Consider $d = 2$. Thus $C^* = l_1 + l_2$. In the worst outcome of Algo. 1, S_1 offloads data to R_2 (otherwise S_2 may offload to R_2, partially restoring social optimal), which implies that $|S_1R_2| \le l_1$. Given that R_2 is occupied, S_1 minimizes its preservation cost and it can at least offload its data to R_1. The worst case for S_2 is to go through all edges in S_2R_2, R_2S_1, S_1R_1 to offload its data to R_1. Thus the highest data preservation cost of S_2 is $l_2 + l_1 + l_1$ due to $|S_1R_2| \le l_1$. Fig 3(a) illustrates such a scenario. We have

$$\text{PoA} = \max \frac{C^N}{C^*} = \max \frac{l_1 + (l_2 + 2l_1)}{l_1 + l_2}$$
$$= \max[1 + \frac{2l_1}{l_1 + l_2}] = 3 = H(2).$$

Note that the maximization is achieved at $l_2 = 0$, i.e., when the preservation cost of S_2 at the social optimal is infinitely close to zero.

(3). Consider $d = 3$. By the same argument as in (2), the worst NE involves S_1 offloading data to R_2, driving S_2 to offload to R_3 along the longest path, which in turn triggers S_3 to offload to R_1 along the longest path. The cost minimization of each node in Algo. 1 implies that $|S_1R_2| \le l_1, |S_2R_3| \le |S_2R_2| + |R_2S_1| + |S_1R_1| \le l_2 + l_1 + l_1 = 2l_1 + l_2$. For S_3, it can at least use R_1 to offload its data, and the worst case is to go through all the edges to reach R_1. We have $S_3R_1 \le |S_3R_3| + |R_3S_2| + |S_2R_2| + |R_2S_1| + |S_1R_1| \le l_3 + (2L_1 + l_2) + l_2 + l_1 + l_1 = 4l_1 + 2l_2 + l_3$. Thus $C^N \le l_1 + (2l_1 + l_2) + (4l_1 + 2l_2 + l_3) = 7l_1 + 3l_2 + l_3$. See Fig 3(b) for such a scenario. Thus

$$\text{PoA} = \max \frac{C^N}{C^*} = \max \frac{7l_1 + 3l_2 + l_3}{l_1 + l_2 + l_3} = 7 = H(3).$$

The maximization is achieved at $l_2 = l_3 = 0$.

(4). Suppose that at $d = k \ge 4$, PoA $= H(k)$. Note that this is achieved at $l_2 = l_3 = ... = l_k = 0$, giving the total cost of preserving for $S_1, S_2, ..., S_k$ as $H(k)l_1$. We want to show that for $d = k + 1$, it holds that PoA $= H(k + 1)$. With one additional data to offload, the worst scenario of NE could be that the offloading of the original k data is with the largest possible inefficiency $H(k)$, while the last data $k + 1$ goes through the longest possible distance to offload. Thus the cost of preserving for $k+1$ is $|S_{k+1}R_{k+1}| + |R_{k+1}S_k| + ... + |R_2S_1| + |S_1R_1|$. Fig 3(c) shows a general case with $d = k+1$. Thus

$$\begin{aligned}\text{PoA} &= \max \frac{C^N}{C^*} \\ &= \max \frac{H(k)l_1 + 2^{k+1}l_1 + 2^k l_2 + ... + 2l_k + l_{k+1}}{\sum_{i=1}^{k+1} l_i} \\ &= \max \frac{H(k+1)l_1 + 2^k l_2 + 2^{k-1}l_3 + ... + l_{k+1}}{\sum_{i=1}^{k+1} l_i} \\ &= H(k+1).\end{aligned}$$

The maximization is achieved at $l_2 = l_3 = ... = l_{k+1} = 0$. ∎

Distance-based Greedy Algorithm. Algo. 2 works in iterations as well; in each iteration, it finds a source and storage node pair with the minimum distance. In particular, we first find the shortest distance between all the $k \times q$ pairs of source and storage nodes and sort them in increasing order of the distances. Next, we start from the first pair on the list and check if the source node has un-offloaded data and the storage node has available storage. If so, we will offload the minimum of these two values from this source node to this storage node; otherwise, it moves to the next pair. This continues until all

the $k \times q$ pairs are checked. Its time complexity is the same as Algo. 1, which is $O\big(k \cdot q \cdot \big(|E| + |V| \cdot \log(|V|)\big)\big)$.

Algorithm 2: The Distance-base Greedy Algorithm.
Input: A BSN graph $G(V, E)$;
Output: Data preservation paths $f : D \to V_r$;
Notations: l_i: number of un-offloaded data packets at S_i;
h_j: number of available storage spaces at R_j;
1. **for** $(1 \le i \le k)$ // current data packets at S_i
$l_i = d_i$;
2. **for** $(1 \le j \le q)$ // current storage space at R_j
$h_j = m_j$;
3. Find the shortest distance between all the (S_i, R_j) pairs;
4. Sort the pairs in the non-descending order of their distances and denote it as L;
5. **while** (L is not empty)
6. Let (S_i, R_j) be the first pair in L;
7. **if** $(l_i > 0 \wedge h_j > 0)$
8. Offload $\min(l_i, h_j)$ packets from S_i to R_j along the the data preservation path between S_i and R_j;
9. **end if;**
10. $l_i = l_i - \min(l_i, h_j)$, $h_j = h_j - \min(l_i, h_j)$;
11. **if** $(l_i == 0 \vee h_j == 0)$
12. Remove (R_i, S_j) from L;
13. **end if;**
14. **end while;**
15. **RETURN** $f : D \to V_r$.

Theorem 3: Algo. 2 reaches a NE, which has PoA < 3.

Proof: The argument for reaching NE is similar to that for Algo. 1. As in the proof of Theorem 2, we confine to the case when each source node has one overflow data, and each storage node has one unit storage. By running Algo. 2, the order of the offloaded data is given as $1, 2, 3, ..., d$. In social optimal, the corresponding source node of each data is denoted as $R_1, R_2, R_3, .., R_d$, respectively. Denote the distance between each source node and its storage node used in social optimal as $l_1, l_2, l_3, .., l_d$. Denote the minimized total data preservation cost as $C^* = \Sigma_{i \in D} l_i$ and the total preservation cost of a NE from Algo. 2 as C^N. We use xy to denote the shortest distance between nodes x and y, and its cost is $|xy|$.

(1). Consider $d = 1$. In this case, the source node offloads along the shortest path in Algo. 2, and the outcome is socially optimal. Thus PoA $= 1$.

(2). Consider $d = 2$. Thus $C^* = l_1 + l_2$. In the worst outcome of Algo. 2, S_1 offloads data to R_2 (otherwise S_2 may offload to R_2, partially restoring social optimal), which implies that $|S_1R_2| \le l_1$ and $|S_1R_2| \le l_2$. Given that R_2 is occupied and R_1 is vacant, S_2 can at least go through the edge S_2R_2, R_2S_1, S_1R_1 to offload its data to R_1, resulting in a data preservation cost of S_2 as $|S_2R_2| + |R_2S_1| + |S_1R_1| = l_1 + l_2 + |R_2S_1|$. We have

$$\text{PoA} = \max \frac{C^N}{C^*} = \max \frac{l_1 + l_2 + 2|R_2S_1|}{l_1 + l_2} = \frac{2(l_1 + l_2)}{l_1 + l_2} = 2.$$

Note that the maximization is achieved at $|S_1R_2| = l_1 = l_2$.

(3). Consider $d = k \ge 3$. By the same argument as in (2), the worst NE involves S_1 offloading data to R_2; then S_2 offloads to R_3, etc., and S_{k-1} offloads to R_k, resulting in S_k to go through the longest distance to offload to R_1. The cost minimization of each node in Algo. 2 implies that $|S_1R_2| \le l_1, l_2; |S_2R_3| \le l_3$, ..., and $|S_{k-1}R_k| \le l_k$. For S_k, its worst data offloading cost is $|S_kR_k| + |R_kS_{k-1}| + |S_{k-1}R_{k-1}| + ... + |S_3R_3| + |R_3S_2| + |S_2R_2| + |R_2S_1| + |S_1R_1| = \sum_{i=1}^k l_i + (|R_kS_{k-1}| + ... + |S_2R_3| + |S_1R_2|)$. In this scenario, $C^N = \sum_{i=1}^k l_i + 2(|R_kS_{k-1}| + ... + |S_2R_3| + |S_1R_2|)$.

$$\begin{aligned}
\text{PoA} &= \max \frac{C^N}{C^*} \\
&= \max \frac{\sum_{i=1}^k l_i + 2(|R_kS_{k-1}| + ... + |S_2R_3| + |S_1R_2|)}{\sum_{i=1}^k l_i} \\
&= \max \frac{\sum_{i=1}^k l_i + 2|S_1R_2| + 2(l_3 + ... + l_k)}{\sum_{i=1}^k l_i} \\
&= \frac{\sum_{i=1}^k l_i + (l_1 + l_2) + 2(l_3 + ... + l_k)}{\sum_{i=1}^k l_i} \\
&= \frac{3\sum_{i=1}^k l_i - (l_1 + l_2)}{\sum_{i=1}^k l_i} = 3 - \frac{l_1 + l_2}{\sum_{i=1}^k l_i} < 3.
\end{aligned}$$

PoA is bounded away from 3 since $l_1 + l_2 > 0$. ∎

C. The PoS of Greedy Algorithms

Our analysis above shows that the greedy algorithms Algos. 1 and 2 may or may not achieve social optimal for data preservation. Below we prove that there exists a greedy algorithm that achieves NEs with PoS $=$ PoA $= 1$.

Theorem 4: There exists a greedy algorithm for DPP that reaches PoS $=$ PoA $= 1$.

Proof: We consider a general form of greedy algorithms wherein source nodes make the best choice in data preservation sequentially. To prove that its PoS $=$ PoA $= 1$, it suffices to show that the social optimal is obtained through this greedy algorithm. W.l.o.g., we confine to the situation when each source node has one unit of overflow data and each storage node has one storage space. The proof involves two steps. We omit some details due to space constraints.

Step one. We show that the socially optimal outcome for any network involves at least one overflow data preserved via its minimum-cost path. Due to the space limit, we only give a proof sketch. Suppose this statement is not true. Then in the social optimal, the preservation cost of each source node is strictly greater than its minimum cost. We can find the data, denoted as ξ, for which its preservation cost exceeds its minimum preservation cost by the largest amount. Consider

instead switching the preservation of ξ via the minimum-cost path. If the corresponding storage is not occupied in the social optimal, contradiction follows the total preservation cost is strictly lower due to the switch. If the corresponding storage is occupied in the social optimal, switch the data originally occupied that storage to using the original social optimal storage of ξ. We can prove that because ξ is the data with the largest efficiency loss in the social optimal, such a switch will strictly lower the total data preservation cost, again a contradiction to the social optimal. We conclude that any social optimal must involve at least one data preserved via the minimum-cost path.

Step two. Let node i be a source node that uses its minimum cost path to preserve its data in the social optimal data preservation. By removing the overflow data and corresponding destination storage space (not the nodes), we get a reduced network with one fewer overflow data. By the argument in step one, for the reduced network, social optimal involves at least one source node preserving its data via its minimum-cost path. By reducing the overflow data and its destination storage, we get another reduced network, for which we can continue with the identifying and deleting procedure. Repeat this procedure until only one overflow data is in the reduced network. The whole procedure indicates a greedy algorithm that reaches the socially optimal outcome. We conclude that PoS $=$ PoA $= 1$ for the greedy algorithm. ∎

Discussions. One of the combinatorial algorithms solving MCF optimally is the *successive shortest path algorithm* [15]; thus it also solves DPP optimally. It is an iterative greedy algorithm applied to the flow network shown in Fig. 2. Each iteration finds the shortest path in the residual graph [2] of Fig. 2 until there is no path from s to t. It is optimal and achieves PoS $= 1$. However, it has a time complexity higher than our straightforward greedy algorithms Algo. 1 and 2.

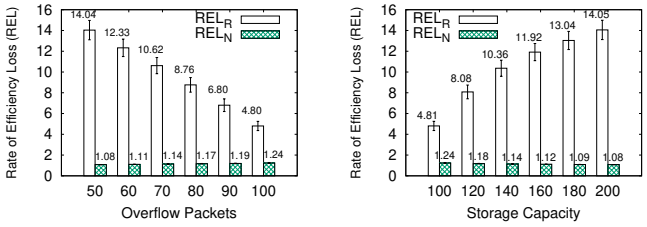
Random. We also design a Random algorithm for comparison purposes. In the Random, each source node offloads its data packets to a randomly chosen storage node with available spaces. Its time complexity is $O(k \cdot |V|)$. Different from MCF and Greedy, Random only sometimes achieves NE. Below we give the sufficient condition for Random to achieve NE.

Theorem 5: When $\sum_{j=1}^{q} m_j = d$, Random reaches NE.

Proof: In this case, the total number of data packets equals the total storage spaces available. Thus there are no free spaces in the BSN once the source nodes offload all their data packets. Since no source node can deviate by offloading its packets to other storage nodes, the outcome is a NE. ∎

V. Simulations

We write our simulator in Java on a Windows 10 machine with an Intel Processor (Intel Core i7-10750H) and 32GB of memory. We randomly place 150 sensor nodes in a $2000m \times 2000m$ sensor field. The transmission range of the sensor nodes is set as 200m, meaning an edge exists between two sensor nodes if their distance is within 200m. Among the 150 sensor nodes, 75 are randomly selected as source nodes



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

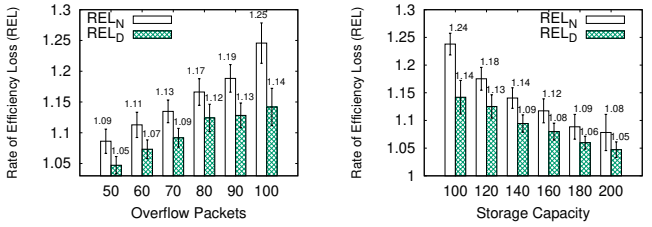
Fig. 4. Comparing Random and Greedy-N.

and the rest are storage nodes. Unless otherwise mentioned, the number of data packets at each source node $d_i = 100$, and the storage capacity of each storage node $m_j = 100$ (that is, it can store 100 data packets). Each packet is 512 Bytes.

We refer to the node-based Algo. 1 as **Greedy-N**, the distance-based Algo. 2 as **Greedy-D**, the MCF-based optimal algorithm as **MCF**, and the random algorithm as **Random**. To measure the performance degradation of different data preservation NEs, we apply each algorithm on 20 randomly generated BSN topologies and compute the average of the RELs of the resultant NEs. We denote the RELs of Greedy-N, Greedy-D, and Random as REL_N, REL_D, and REL_R, respectively. The error bars indicate 95% confidence intervals.

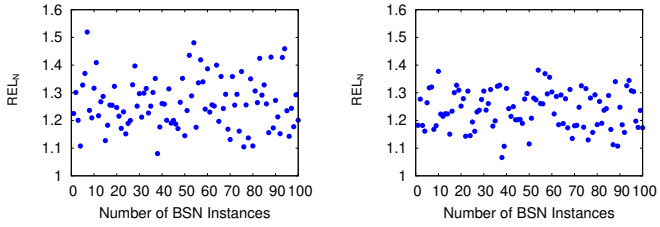
Comparing Greedy-N and Random. Fig. 4(a) compares the RELs of Greedy-N and Random by increasing d_i from 50, 60, ..., to 100 while fixing m_j as 100. After the data preservation (i.e., data offloading), the network is half-full at $d_i = 50$ and full at $d_i = 100$. First, we observe that REL_R is much larger than REL_N in all the cases. This shows that Random has a much more significant efficiency loss than Greedy-N, as it does not attempt to save the preservation cost when offloading data packets. On the other hand, the most significant REL_N is 1.24, showing the efficiency loss of the Greedy-N is at most 24% of the optimal preservation cost by the MCF. Second, it shows that when increasing d_i, the REL_N increases gradually. Although the total preservation costs of both Greedy-N and MCF increase, Greedy-N increases more than MCF does (as MCF is optimal), resulting in a larger REL_N. Third, REL_R decreases dramatically with increased d_i, showing that Random performs much better in more "crowded" scenarios. This is because when more data packets are offloaded, the randomness' negative effect of getting high preservation costs will be gradually eliminated, as each source node has fewer options to decide which storage nodes to offload data packets.

Fig. 4(b) compares the RELs of Greedy-N and Random by increasing m_j from 100, 120, ..., to 200 while fixing d_i as 100. It again shows the big difference between REL_N and REL_R. However, unlike Fig. 4(a), Fig. 4(b) shows that with the increase of the m_j, REL_R increases evidently. With the increase of the m_j, the preservation cost of the MCF decreases while that of the Random increases (due to its randomness), resulting in a larger REL_R. It also shows that with the increase of the m_j, REL_N decreases slightly, from 1.24 to 1.08. While the preservation costs of both MCF and Greedy-N decrease with the increase of the m_j, it seems



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

Fig. 5. Comparing Greedy-N and Greedy-D.



(a) $d_i = m_j = 1$. (b) $d_i = m_j = 100$.

Fig. 6. Investigating the PoAs of Greedy-N.

Greedy-N has more leeway to improve upon its preservations at smaller m_j compared to MCF.

Comparing Greedy-N and Greedy-D. As Greedy-N performs much better than Random in the RELs, we focus on Greedy-N and Greedy-D next. Fig. 5(a) compares them by varying the d_i. First, we observe that the REL_D is constantly smaller than REL_N, showing that NEs in Greedy-D have less efficiency loss for data preservation than Greedy-N does. Second, at $d_i = 50$, when there are twice as many storage spaces as the number of data packets, Greedy-N and Greedy-D yield RELs of 1.09 and 1.05, respectively. This shows that the total data preservation costs of Greedy-N and Greedy-D are at most 9% and 5% more than the optimal, indicating both are energy-efficient. Third, with the increase of d_i, both RELs increase. This shows the efficiency loss in NEs of both algorithms increases in more challenging scenarios wherein more data packets are to be preserved.

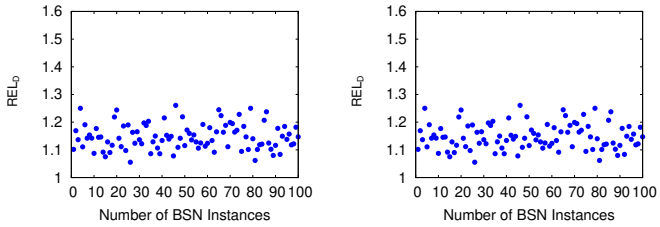
Fig. 5(b) shows that with the increase of m_j, both REL_N and REL_D decrease. This shows that more storage spaces can alleviate the NE efficiency loss. In an extreme case wherein each storage node has infinite storage capacity, Greedy-N and Greedy-D would achieve the same minimum total preservation cost as MCF; in this case, there is no efficiency loss for any NEs. In general economic terms, when there are infinite social resources, strategic interactions among selfish players will be released; as such, NEs will perform as well as the optimal resource allocation schemes.

Investigating the PoAs of Greedy-N and Greedy-D. Next, we validate our theoretical findings that the PoA of Greedy-N is $H(d)$, where $H(d) \equiv 2^{d-1}+2^{d-2}+...+2^0$ (Theorem 2), and the PoA of Greedy-D is less than 3 (Theorem 3). As PoA is the upper bound of the RELs defined in this paper, we generate 100 random BSN instances and find the REL of each resultant NE. Fig. 6(a) and (b) show the REL_N with $d_i = m_j = 1$ and $d_i = m_j = 100$, respectively. It shows the REL_N values fall between 1.08 and 1.52 for $d_i = m_j = 1$ while between 1.08 and 1.39 for $d_i = m_j = 100$. Our empirical results are much better than our theoretical result of PoA $< 2^{d-1}$. Fig. 7(a) and (b) study PoAs for Greedy-D. Compared to Fig. 6, the range of REL_D values is narrowed between 1.05 to 1.25. This corresponds well to our Theorem 3, which says PoA < 3 for Greedy-D. Note that Fig. 6 (a) and (b) give the same set of REL_D values. This is because we randomly generate 100 BSN instances for this set of simulations and then apply both sets of parameters (i.e., $d_i = m_j = 1$ and $d_i = m_j = 100$) on the same instance. As a result, the total preservation costs for both Greedy-D and MCF are increased 100 times in Fig. 6 (b) of those in (a), resulting in the same REL_D values.

Comparing Total Preservation Costs. Finally, we look closely at the total preservation costs of all the NEs achieved by the proposed algorithms, as shown in Fig. 8. We have two general observations. First, the costs of all three algorithms increase with the increase of d_i and decrease with the rise of m_j. Second, MCF is the most energy-efficient way to achieve NEs, followed by Greedy-D, while Greedy-N is the least energy-efficient. We also observe that the performance differences among these three algorithms diminish when the data preservation gets less challenging (i.e., when d_i is small and m_j is large). This also has an economic interpretation: When strategic competition among selfish players lessens, NE outcome tends to be closer to the social optimal of resource allocations. In other words, the effects of selfishness in NEs upon performance degradation decrease.

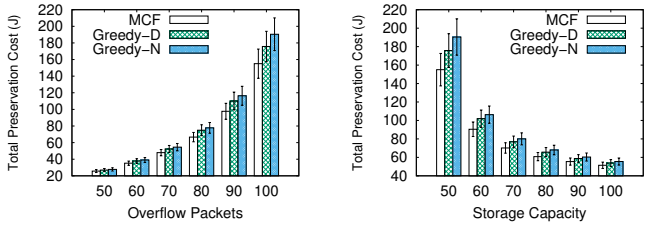
VI. Conclusion and Future Work

We studied the Nash Equilibrium (NE) for data preservation in emerging base station-less sensor networks (BSNs). We design a suite of data preservation algorithms that achieve NEs and define our own metric (besides commonly used PoA and PoS) to quantify the performance efficiency of any NEs achieved by our algorithms. We show that our minimum cost flow-based algorithm achieves not only NEs but also minimum preservation cost, thus suffering no efficiency loss common for selfish players. We also prove that the PoA is no more than 3 for one of our greedy algorithms, and a greedy algorithm



(a) $m_j = 1$. (b) $m_j = 100$.

Fig. 7. Investigating the PoAs of Greedy-D.



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

Fig. 8. Total preservation costs of different algorithms.

always exists that gives a PoS $= 1$. Finally, we conduct extensive simulations to investigate and compare the efficiency loss of various NEs and validate our theoretical results. Studying the performance of NEs is essential to our data preservation problem and critical to any multi-agent reinforcement learning systems (e.g., autonomous driving) wherein NE is an indispensable solution concept. For future works, first, we will attempt to reduce the efficiency loss of NEs through a marginal cost pricing mechanism called Shapley pricing [31]. Second, we will consider that data packets have different values. In this case, when the value of a data packet is less than its preservation cost, the selfish source node would choose not to offload it. How to incentivize them to participate in data preservation while achieving performance-optimal NEs remains a challenging problem.

ACKNOWLEDGMENT

It was supported by NSF Grant CNS-2131309 and Google exploreCSR program. We thank Chris Gonzalez for plotting.

REFERENCES

[1] M. Y. Aalsalem, W. Z. Khan, W. Gharibi, M. K. Khan, and Q. Arshad. Wireless sensor networks in oil and gas industry: Recent advances, taxonomy, requirements, and open challenges. *Journal of Network and Computer Applications*, 113:87–97, 2018.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

[3] T. AlSkaif, M. G. Zapata, and B. Bellalta. Game theory for energy efficiency in wireless sensor networks: Latest trends. *Journal of Network and Computer Applications*, 54:33 – 61, 2015.

[4] A. Attiah, M. Amjad, M. Chatterjee, and C. Zou. An evolutionary routing game for energy balance in wireless sensor networks. *Comput. Netw.*, 138:31–43, 2018.

[5] Z. Ballard, C. Brown, and A.M. et al. Madni. Machine learning and computation-enabled intelligent sensor design. *Nature Machine Intelligence*, (3):556–565, 2021.

[6] S. Basagni, L. Boloni, P. Gjanci, C. Petrioli, C. A. Phillips, and D. Turgut. Maximizing the value of sensed information in underwater wireless sensor networks via an autonomous underwater vehicle. In *Proc. of INFOCOM*, 2014.

[7] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[8] D. E. Charilas and A. D. Panagopoulos. A survey on game theory applications in wireless networks. *Comput. Netw.*, 54(18):3421–3430, December 2010.

[9] Y. Chen, J. Ly, and B. Tang. Voluntary data preservation mechanism in base station-less sensor networks. In *12th EAI International Conference on Game Theory for Networks (GameNets 2022)*.

[10] Z. Chen, Y. Qiu, J. Liu, and L. Xu. Incentive mechanism for selfish nodes in wireless sensor networks based on evolutionary game. *Computers Mathematics with Applications*, 62(9):3378–3388, 2011.

[11] M. Elhoseny, K. Shankar, and M. Abdel-Basset. *Artificial Intelligence Techniques in IoT Sensor Networks*. Chapman and Hall/CRC, 2020.

[12] L. Terray et al. From sensor to cloud: An iot network of radon outdoor probes to monitor active volcanoes. *Sensors*, 20(10), 2020.

[13] R. Ghaffarivardavagh, S. S. Afzal, O. Rodriguez, and F. Adib. Ultra-wideband underwater backscatter via piezoelectric metamaterials. In *Proc. of the ACM SIGCOMM*, 2020.

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

[15] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

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

[17] S. Hsu, Y. Yu, and B. Tang. dre^2: Achieving data resilience in wireless sensor networks: A quadratic programming approach. In *Proc. of IEEE MASS*, 2020.

[18] H. Huang, A. V. Savkin, M. Ding, and C. Huang. Mobile robots in wireless sensor networks: A survey on tasks. *Computer Networks*, 148:1–19, 2019.

[19] J. Jang and F. Adib. Underwater backscatter networking. In *Proc. of the ACM SIGCOMM*, 2019.

[20] R. Kannan and S. Iyengar. Game-theoretic models for reliable path-length and energy-constrained routing with data aggregation in wireless sensor networks. *IEEE J. Sel. Areas Commun.*, 22:1141–1150, 2004.

[21] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2016.

[22] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

[23] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[24] D. Niyato, E. Hossain, M. M. Rashid, and V. K. Bhargava. Wireless sensor networks with energy harvesting technologies: a game-theoretic approach to optimal energy management. *IEEE Wireless Communications*, 14(4):90–96, 2007.

[25] F. Pavlidou and G. Koltsidas. Game theory for routing modeling in communication networks – a survey. *Journal of Communications and Networks*, 10(3):268–286, Sep. 2008.

[26] H.-Y. Shi, W.-L. Wang, N.-M. Kwok, and S.-Y. Chen. Game theory for wireless sensor networks: A survey. *Sensors (Basel, Switzerland)*, 12:9055–97, 12 2012.

[27] M. S. Stankovic, K. H. Johansson, and D. M. Stipanovic. Distributed seeking of nash equilibria with applications to mobile sensor networks. *IEEE Transactions on Automatic Control*, 57(4):904–919, 2012.

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

[29] B. Tang, N. Jaggi, H. Wu, and R. Kurkal. Energy-efficient data redistribution in sensor networks. *ACM Trans. Sen. Netw.*, 9(2):11:1–11:28, April 2013.

[30] B. Tang, H. Ngo, Y. Ma, and B. Alhakami. dao^2: Overcoming overall storage overflow in intermittently connected sensor networks. *Accepted at IEEE/ACM Transactions on Networking*, 2023.

[31] Y. Tauman. The aumann-shapley prices: A survey. *A. Roth (Ed.), The Shapley Value: Essays in Honor of Lloyd S. Shapley. Cambridge: Cambridge University Press.*, pages 279–304, 1988.

[32] A. C. Voulkidis, M. P. Anastasopoulos, and P. G. Cottis. Energy efficiency in wireless sensor networks: A game-theoretic approach based on coalition formation. *ACM Trans. Sen. Netw.*, 9(4), July 2013.

[33] A. Wichmann, T. Korkmaz, and A. S. Tosun. Robot control strategies for task allocation with connectivity constraints in wireless sensor and robot networks. *IEEE Transactions on Mobile Computing*, 17(6):1429–1441, 2018.

[34] L. D. Xu, W. He, and S. Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014.

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

[36] Y. Yu, S. Hsu, A. Chen, Y. Chen, and B. Tang. Truthful and efficient data preservation in base station-less sensor networks. *Accepted for publication in ACM Transactions on Sensor Networks*, 2023.

[37] Y. Zhao, S. S. Afzal, W. Akbar, O. Rodriguez, F. Mo, D. Boyle, F. Adib, and H. Haddadi. Towards battery-free machine learning and inference in underwater environments. In *ACM HotMobile'22*.