Multi-Vehicle Coordination for Wireless Energy Replenishment in Sensor Networks

Cong Wang[1], Ji Li[1], Fan Ye[2], and Yuanyuan Yang [1]

[1]*Dept. of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA*
[2]*IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA*

***Abstract*—Mobile vehicles equipped with wireless energy transmission technology can recharge sensor nodes over the air. When to recharge which nodes, and in what order, critically impact the network performance. So far only a few works have studied the recharging policy for a single mobile vehicle. In this paper, we study how to coordinate the recharging activities of multiple mobile vehicles, which provide more scalability and robustness than a single vehicle. We leverage concepts and mechanisms from NDN (*Named Data Networking*) to design energy monitoring protocols that deliver energy status information to mobile vehicles in an efficient manner. Then we study how to minimize the total traveling cost of multiple vehicles while ensuring no node failure. We derive theoretical results on the energy neutral condition and the minimum number of mobile vehicles required for perpetual network operations. We formulate the optimization problem into a Multiple Traveling Salesman Problem with Deadlines (m-TSP with Deadlines), which is NP-hard. To accommodate the dynamic nature of node energy conditions and reduce computational overhead, we present a heuristic algorithm that selects the node with the minimum weighted sum of traveling time and residual lifetime. Our scheme not only improves network scalability but also guarantees the perpetual operation of networks. Finally, we conduct extensive simulations to demonstrate the effectiveness and efficiency of our proposed design, and validate the correctness of theoretical analysis.**

***Index Terms*—Wireless sensor networks, named data networking, wireless recharging, mobile energy replenishing, perpetual operation, recharge coordination.**

I. Introduction

Wireless energy transmission technique [1], [2] is a game-changing technology for providing energy to sensor networks. It can recharge sensor nodes over the air without any wire or plug. Compared to opportunistic energy harvesting techniques [4], [5] where the energy supply is not always available, it can deliver energy to sensor nodes reliably. Mobile energy replenishing vehicles (called *SenCar*s in this paper) can move around the field and recharge nodes conveniently. The recharging policy - when which SenCar should recharge which nodes and in what order - critically impacts the efficiency and thus the lifetime of the network.

So far only a few works [8], [9] have studied the recharging policy problem, in which, basically, nodes report their energy levels periodically, and a centralized algorithm computes a specific order so that a single SenCar recharges all nodes in the next cycle. Although commendable first steps, several important practical issues are not considered, which significantly limit the applicability in a real environment.

First, these algorithms are designed for a single SenCar, which has limited recharging capacity and supports networks of limited sizes. Multiple SenCars can support larger networks, but new algorithms must be designed to coordinate the recharging activities among SenCars efficiently to best leverage their recharging capacities. Second, the timely, efficient and scalable gathering of energy status information of nodes and delivery to mobile vehicles are important and challenging issues in themselves. The previous works do not consider these problems and assume that such information is readily available. Finally, it takes nontrivial (e.g., 30-80 min) time to recharge a commercial off-the-shelf battery. Finishing one round of recharging for a network of a few hundred nodes may take several days. During this time the energy levels of nodes may have changed significantly due to unpredictable external events that can trigger extensive activities and quickly drain the battery. The recharging policy computed at the beginning of the cycle is no longer optimal. This can cause energy depletion on some nodes, leading to network disconnection or application failures.

In this paper, we propose a novel real time energy monitoring and recharging framework that optimizes the recharging policies of multiple SenCars under dynamic network conditions. Instead of letting nodes report their energy levels only after a long period, a scalable and efficient energy information aggregation protocol gathers battery levels continuously from all sensor nodes upon requests by SenCars. The SenCars receive such information and make recharging decisions based on the latest energy information. To deal with unpredictable emergencies where nodes may dramatically drain the battery in short time, the recharging of sensor nodes whose energy levels are below a critical threshold has high priority and takes precedence over those that can work for relatively longer time with their residual energy.

To ensure high scalability of the proposed framework, we apply *Named Data Networking (NDN)* [10] techniques to gather and deliver energy information to SenCars. To scale to large network sizes, we divide the network in a hierarchical fashion and the energy information is aggregated bottom-up through different levels. NDN uses names instead of locations to address data, which is a natural match for aggregated energy information that belongs to an area instead of any particular node. Thus the aggregated energy information can be addressed by the area's name. NDN also supports mobile vehicles because it has mechanisms to constantly update the routing states in intermediate nodes to follow the movements of vehicles. This is important for the SenCars to receive the energy information timely after moving.

Due to the unpredictable nature of external events, the SenCars may need to deal with multiple concurrent *emergencies* occurring at different locations. How to schedule and coordinate the SenCars to recharge these nodes within their residual lifetimes while minimizing the cost of SenCars is defined as the *Emergency Recharge Optimization with Multiple SenCars* (EROMS) problem in this paper. We first investigate the necessary conditions for perpetual operations of the network and derive the minimum number of SenCars needed to satisfy this condition. Then we find that the EROMS problem can be formulated as a Multiple Traveling Salesmen Problem with Deadlines (*m-TSP with Deadlines*) [23], which is NP-hard.

Although heuristic algorithms exist for m-TSP, they are not suitable in our context. First, most of them assume unlimited number of vehicles while the number of SenCars in our problem is limited. Second, these algorithms consider a relatively static input where the locations to be visited do not change. However, in our context, new emergencies may appear and old ones may be resolved as SenCars recharge nodes. Finally, these algorithms may produce unbalanced workloads among SenCars such that some SenCars can be idle while emergencies still exist. Therefore, we propose a new heuristic algorithm to address such deficiencies; it decides which nodes to recharge through a weighted sum of the traveling time and residual lifetime of sensor nodes. We also study the complexity of our algorithm and demonstrate its performance by extensive simulations.

We make the following contributions in this paper. First, we propose a novel real time recharging framework for wireless sensor networks, consisting of a set of scalable and efficient NDN-based energy aggregation and gathering protocols. The protocols satisfy both normal and emergency recharging needs for multiple mobile vehicles. Second, we formally analyze the conditions for the minimum number of SenCars needed for perpetual operations. Third, we discover that the emergency recharge optimization with multiple SenCars is a *m-TSP with Deadlines* problem, and further propose an efficient heuristic algorithm suitable to sensor recharging context. Finally, we conduct extensive simulations to demonstrate the effectiveness and efficiency of the framework and validate the correctness of the theoretical analysis. To the best of our knowledge, this is the first work that can coordinate the recharging activities for multiple vehicles and adapt to dynamic network conditions such as emergencies; it is also the first effort to apply NDN techniques to wireless sensor networks.

The rest of the paper is organized as follows. Section II discusses related work. Section III outlines the framework and assumptions made in the network model. Section IV describes the operations and mechanisms of our protocol followed by Section V and Section VI on deriving the minimum number of SenCars and solving the EROMS problem respectively. Finally, Section VII shows the simulation results and Section VIII concludes the paper.

II. Related Work

A. Wireless Rechargeable Sensor Networks

Recently, there have been great research efforts in wireless energy transmissions from both academia and industry [3], [6]–[9]. In [6], the impact of wireless charging technology on sensor networks was investigated using devices from Powercast [3], and heuristic algorithms were developed to solve the deployment and routing problem. In [7], deployment problems were studied in a rechargeable sensor network built from industrial wireless sensing platform and commercial off-the-shelf RFID readers. In [9], the problem of periodic recharging of each sensor node using a single mobile vehicle was considered. A near-optimal solution was provided to calculate the optimal traveling path of the mobile car. It forms the shortest Hamilton cycle through all sensor nodes. In [8], the problem of joint optimization of effective energy recharging and data collection with bounded data latency was studied. A two-step approach was proposed to recharge nodes with the least residual energy while maximizing network utility.

The above works make pioneering steps in wireless rechargeable sensor networks. However, several important practical issues are not considered. First, none of them consider the coordination of multiple recharging vehicles. Second, how the energy information can be aggregated and reported to mobile vehicles is not considered. Third, the dynamic changes in energy levels that occur inevitably and unpredictably during long recharging cycles are not handled.

B. Named Data Networking (NDN)

Named Data Networking is a new network architecture proposed recently for the Internet [10]. In NDN, data are addressed by their names instead of hosting nodes' locations. The operation is based on two types of messages, *Interest* and *Data*, and the communication is initiated by the receiver. A receiver interested in certain data sends Interest messages carrying the name of the desired data. The Interest message propagates in the network following FIB (Forward Interest Base) states towards nodes hosting desired data. It also leaves a "trail" of PIT (Pending Interest Table) states in intermediate nodes. Once the Interest reaches a node hosting the desired data, Data messages can follow PIT states to traverse back to the receiver.

So far NDN research has largely focused on the Internet, with some efforts on mobile networking. Whether it can be used to satisfy the needs of wireless sensor networks is still unexplored. In this paper, we use wireless recharging as a case study to investigate its applicability in wireless sensor networks. We find that its hierarchical naming structure fits naturally with energy aggregation needs, and its inherent ability to handle mobile receivers is attractive for information delivery to recharging vehicles.

C. Coordination of Mobile Vehicles

Coordinating multiple mobile vehicles has been studied for data collection in wireless sensor networks. In [11], the problem of minimizing the total traveling cost of multiple mobile vehicles was studied. It formalizes the problem into covering salesman problems and presents a tour-planning heuristic. In [12], a set of heuristics were proposed to schedule the data collection of multiple mobile vehicles to meet sensors' dynamic buffer overflow time constraints. A sensor may be visited by one or more mobile vehicles depending on its buffer status. In [13], a set of protocols were proposed to achieve spatial coverage equivalence, vehicle mutual avoidance and load balancing. All the existing works focus on data collection where mobile vehicles only need to cover sensors in its transmission range, and a sensor may be visited by one or more vehicles during a short period. However, in wireless energy replenishment, the effective wireless recharging range is very short compared to data transmissions, and multiple mobile vehicles recharging the same sensor node incurs high cost that should be completely avoided.

III. A Novel Framework for Wireless Rechargeable Sensor Networks

In this section, we describe the components, network model and assumptions for our NDN-based wireless recharging framework. NDN has a few attractive benefits for our environment. First, by sending out new Interest packets, a mobile receiver can continuously update the routing states (i.e., PIT entries) in intermediate nodes. Data can follow the reverse paths traversed by the most recent Interest packets and reach the new location of the receiver. This solves the mobility issue of SenCars and ensures that the latest energy information can reach them in a timely manner. Second, to scale to large network sizes, we divide the network in a hierarchical fashion and energy information is gathered in aggregated forms. Thus the data is bounded to an area rather than any particular node. This makes a natural fit for NDN: the data can be addressed by the area's name. Compared to a flat topology that requires flooding messages throughout the network, such hierarchical aggregation saves considerable overhead by confining message propagation to parts of the network. Third, NDN provides network robustness when intermediate nodes fail. This is ensured by the receiver resending Interest packet when Data does not arrive. The new Interest packet explores alternative routes and bypass failures [10].

A. Network Components

The network consists of the following components.

SenCars and Service Station: SenCars query the network for energy information and recharge nodes based on the energy information collected. They can be commanded by the administrator via a service station with computing and communication capabilities.

Head nodes: A head is a sensor node delegated to aggregate energy information from its subordinate area. When requested by a SenCar or by the head of its upper level, a head queries energy information from subordinate sub-areas at the lower level, aggregates such information and sends to the requester.

Proxy: A proxy node aggregates *emergencies* from sensor nodes and reports such information to the SenCar when queried. Only top-level head nodes are proxies.

Normal Node: A sensor node not selected as a head is a normal node. It reports energy information to head nodes, or sends emergency directly to its proxy when its energy level drops below an emergency recharge threshold.

B. Name Assignments and Network Model

We assume sensors are scattered uniformly randomly. The network field is divided into several areas and each area is further divided recursively. The division of the area is based on geographical coordinates of the sensing field. Each division generates some new *sub-areas* and increases the number of *levels* in the network. This process repeats until the bottom level

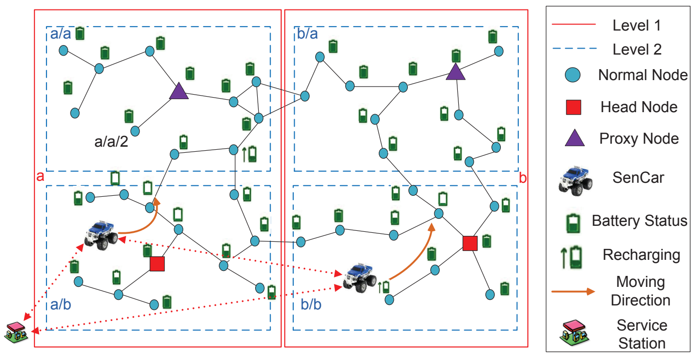

Fig. 1. An illustration of area names and network components.

subarea becomes small enough such that during the time for a SenCar to recharge such a subarea, the normal energy levels of the network do not change too much to warrant interruption of that recharge. Fig. 1 gives an example of a 2-level network with 2 areas (solid lines), each further split into 2 sub-areas (dashed lines), where each sub-area on the second level contains about 10 sensor nodes.

Based on the results of area divisions, we assign NDN data names for different subareas in a hierarchical manner. For example, Fig. 1 shows all the name assignments for different subareas (e.g., the first level areas are “a” and “b”, and the second level has “a/a”, “a/b”, . . .). Thus each subarea is identified by its unique hierarchical name. Each node has an ID including the name of its containing bottom-level subarea plus an identifier. For example, “a/a/2” is node “2” in subarea “a/a”. The hierarchical names make it easy to confine the propagation scope of a message to any subarea: nodes beyond the intended subarea (carried by the message) can simply stop further propagation.

In addition, we have the following assumptions: 1) Sensor nodes are stationary and each node knows its location. 2) Nodes have the same transmission range and messages are forwarded over multiple hops in large networks. 3) The SenCars have positioning systems and know their locations. The IDs and locations of all sensor nodes, and the subarea names are known to the SenCars (e.g., through a one-time effort at the initialization stage). 4) The SenCars are equipped with powerful antennas so that they can communicate among themselves and to the service station directly. 5) Sensors might perform different tasks thus the energy consumption is not uniform among nodes.

IV. The NDN-based Real Time Wireless Recharging Protocol

In this section, we present the detailed design of NDN-based Real Time Wireless Recharging Protocol. We first give an overview of the protocol design in Section IV-A. Then we describe different operating phases of the protocol in Section IV-B.

A. Protocol Overview

In our protocol, the SenCars obtain the most up-to-date energy information from sensors and makes recharge decisions in real time. The energy information is aggregated on heads at different levels. To be robust, the head is usually selected as the node having the maximum energy level in its area. This selection process is done at the beginning of network startup through the propagation of *head selection* messages. The detailed will be discussed in the next subsection.

To start energy information collection, SenCars send out *energy interest* messages to poll the *heads* on the top-level. Once the heads receive such messages, they send lower level *energy interest* messages to their child-heads in respective subordinate areas. This process is repeated down the head node hierarchy, until finally the bottom-level *energy interest* messages reach the nodes in the bottom-level subareas.

Once a sensor node receives a bottom-level *energy interest*, it responds by sending out an *energy* message containing its ID and battery level. When the heads on the bottom-level receive such *energy* messages, they select sensor nodes with energy level below a normal recharge threshold (defined by the administrator), and send the names of these nodes and their energy information in an *energy* message to their parent head nodes. This is repeated up the head node hierarchy, until finally the top-level head nodes have the aggregated energy information and send it to the SenCar. When multiple SenCars query energy information simultaneously, the top-level head nodes send the aggregated energy information to the nearest SenCar. Thus SenCars recharge nearby normal nodes to reduce their travel costs. The details are explained later in this section.

To reduce transmission overhead, the head is delegated partial responsibilities to pre-select sensor nodes to be recharged. In the bottom level, this is done by the heads selecting nodes with low energy levels. In upper levels, a head selects the subordinate area which can be recharged with the most amount of energy. Thus the SenCars can replenish the network with more energy in one movement.

Such normal energy aggregation is conducted at the requests of the SenCars. For emergency nodes that have dangerously low battery levels below an emergency threshold, they send out *emergency* messages to the proxy that manages its area. The route to its proxy is built by *head selection* messages from the proxy.

After completing the recharge of any node, a SenCars sends out an *emergency interest* message to query whether any emergency has appeared. These messages are directed to proxies where lists of emergency nodes are stored. The proxies respond by sending back the emergency node IDs, estimated residual lifetimes and energy levels. SenCars receive the messages and use the emergency recharge algorithm to decide which nodes to recharge. Note that different from normal recharging, the SenCar recharging an emergency node may not be the nearest one. This is due to the urgency to avoid any battery depletion.

When a head node is low on energy, it can choose another node with high energy, and send out a *head notification* message to notify the latter to become the new head.

B. Protocol Design

We describe the detailed protocol assuming the head hierarchy has l levels.

1) Head Selection: After the areas and names have been configured, the network performs head selection from the bottom up starting at the l-th (i.e., lowest) level. Since initially sensor nodes have about the same level of energy, any of them may become a head. Each sensor node i generates a random number x. If $x > K$, where K is a pre-determined threshold, the node floods a *head selection* message in its l-th level subarea, containing the name of this subarea, $x_{max} = x$, and ID_{max} set to its own ID. Otherwise the node waits for messages from other nodes in the area.

A node receiving such a *head selection* message compares the x_{max} in the message with its local record x_{max}. If its local record is larger, the message is discarded. Otherwise, the sensor updates its local x_{max} to that in the message, sets ID_{max} to that in the message, and forwards the message to its neighbors except the node that sent this message. Finally, the node with the maximum x wins the election and is recorded by all the nodes in this subarea as the head.

New heads at the l-th level then contend to become heads of the $(l-1)$-th level. They flood new *head selection* messages in the $(l-1)$-th level subarea, carrying the area's name, their respective x values and IDs. Intermediate nodes perform similar comparisons. This will elect the heads at the $(l-1)$-th level. This process is repeated recursively until head nodes of all levels are elected.

One difference for the *head selection* messages starting from the $(l-1)$-th level and up is that messages carrying smaller x than the local copy are not discarded. Instead, they are propagated throughout the respective subarea. This builds routing states in intermediate nodes of the subarea: An intermediate node has one entry for each child-head, pointing to the neighbor from which the message from that head arrives first. Duplicate copies of the same message arriving later are discarded.

Such states are effectively FIB (Forward Interest Base) entries in NDN. Later a parent head at the $(k-1)$th-level can send *energy interest* messages to its child-heads at the k-th level using such states. To build FIB entries for the 1st level head nodes, they each flood a top-level *head selection* message throughout the whole network. Later the energy interest queries from the SenCars can use such states to reach them.

2) Normal Energy Interest Propagation: After the head hierarchy is constructed, the SenCars send *energy interest* messages to query for nodes needing recharge. The energy information is gathered on demand, and top down in the hierarchy. We will describe normal energy information collection first. Emergency information is collected similarly, but with only top-level heads involved to reduce latency.

For normal energy information, interest messages are sent by the SenCars (e.g., with data name set to "/energy/normal/a" to collect energy information from area a). Intermediate nodes use the FIB entries established by top-level *head selection* messages to forward it to corresponding top-level head nodes. To guide the return of future data from a top-level area, an intermediate node also sets up a PIT (Pending Interest Table) entry pointing to the neighbor from which the interest message towards this area is received.

To avoid duplicate selection of the same normal nodes and reduce travel costs, we want only the nearest SenCar to receive a head node's normal energy information. To this end, the *energy interest* message from each SenCar carries a hop count increased by one at each intermediate node. When multiple such messages towards the same top-level area are received, an intermediate node updates its PIT entry to record only the neighbor sending the message with the smallest hop count. . Later energy information from a head can follow such directions to reach the nearest SenCar. After an intermediate node forwards energy information from a top-level area, it deletes the corresponding PIT entry.

Upon receiving an energy interest message, a top-level head sends a new energy interest message to its child-heads, with the data name set to all subareas of its children (e.g., from the head node of area /a, "/energy/normal/a/*"). Similarly, these messages reach all child-heads following FIB entries. Intermediate nodes also set up PIT entries so that later energy information from child-heads can go back to their parent head. This process is repeated down the hierarchy, until finally heads at bottom-level flood their respective subareas with interest messages.

3) Normal Energy Report and Node Recharge: When a sensor node receives an l-th level *energy interest* message, it responds with an *energy* message including its ID and residual energy. With the help of PIT entries, the message is returned to the head of the l-th level.

The head examines if the reported residual energy is less than the normal recharge threshold. If so, the ID of the node is added to a list, and the energy that can be recharged to this node is added to a summation counter. After the head has collected these messages, it sends an aggregation message, containing the list, the summation counter and its subarea name to its parent head. A parent head compares such messages from its child-heads, selects the one with the largest summation counter (i.e., the bottom-level subarea that can be recharged of the greatest amount of energy), and forwards to its parent head. This process is repeated upwards in the hierarchy. Finally, the SenCar nearest to a top-level head receives a message for the bottom-level subarea with the largest summation counter. It moves there and recharges those nodes in the ID list one by one. Only after recharging those nodes will the SenCar sends another normal energy query.

The reason we delegate selection partially to head nodes is twofold. First, we expect much less variation in normal energy levels. Thus the SenCar can choose one bottom-level subarea and finish recharging all listed nodes. Only after the whole subarea is recharged, we expect enough changes in normal energy distribution that warrants a new normal energy query from the SenCar. Second, this also keeps the return message sizes small and reduces overhead.

4) Emergency Energy Report and Node Recharge: Emergency energy report is slightly different due to the urgency. If a node detects that its energy level is below the emergency recharge threshold, it immediately sends an *emergency* message containing its ID and energy level to its proxy (i.e., its top-level head node). Because the head node floods a top-level *head selection* message during head election, the same FIB entries can be used to forward emergency messages to the head.

Instead of waiting for recharging a whole bottom-level subarea, a SenCar sends out *emergency interest* messages to each proxy after finishing recharging any single normal node or emergency node (e.g., with data name set to "/energy/emergency/a" to collect emergency information from the proxy of area a). To guide the return of future data from the proxy of a top-level area, an intermediate node sets up a PIT entry pointing to the neighbors from which the emergency interest messages towards this area are received. Later emergency information from a proxy can follow such directions to return to the SenCar. After an intermediate node forwards emergency information from a proxy, it deletes the corresponding PIT entry.

When an *emergency interest* message is received, the proxy returns its list of IDs, energy levels and estimated residual lifetime of emergency nodes, if there exists any. The SenCar uses the algorithm in Section VI to decide which node to recharge. It switches back to normal operation mode only when no emergency is reported. When multiple SenCars query emergency information simultaneously, they coordinate with each other and make an optimal decision to assign the emergency nodes to each of them. The procedure of emergency assignment is described in Section VI.

5) Head Hierarchy Maintenance: A head can be short on energy due to activity monitoring and message forwarding like normal nodes. When this happens, a new head is needed. Because only heads of bottom-levels contend for higher level elections, a head at any level is always the head of its bottom-level subarea. It receives the energy reports from normal nodes in its bottom-level subarea upon the normal interest query from the SenCar. So it can choose a node with the highest energy, and floods a *head notification* message to notify all nodes in the bottom-level subarea of the new head.

The new head then triggers a new head election process in its $(l-1)$-th level subarea. It propagates a new *head selection* message in its $(l-1)$th subarea, but carrying its energy level instead of the random number x. Other heads in this $(l-1)$-th level subarea do the same. Then a new $(l-1)$-th head with the maximum energy is elected. If this is the same head, the process stops. Otherwise, the new $(l-1)$-th level head triggers the same process in its upper level subarea, until finally a new top-level head is elected.

C. Summary of Protocol Design

We now summarize how we use NDN to route different messages briefly. First, FIB entries are established during the head selection process so that the *interest* message can be sent from parent head nodes to child ones. Second, the propagation of *interest* messages from the SenCar, or from parent to child-heads, establishes PIT entries for later return of *energy* messages from top-level heads or child-heads. Third, FIB entries to top-level heads (i.e., proxies) allow emergency nodes to send reports to proxies without waiting for the emergency interest queries from the SenCar, which minimizes latency.

TABLE I
TABLE OF NOTATIONS

Notation	Definition
$\mathcal{N}$	Set of sensor nodes with N elements
$\mathcal{M}$	Set of emergency nodes with M elements at the time when SenCar makes a query
$\mathcal{S}$	Set of SenCars with S elements
N	Number of sensors in the network
p	Probability for a node to consume unit energy in a time slot
n	Number of time slots
R_n	Energy replenished for a node in n time slots
E_n	Energy consumed for a node in n time slots
E_0	Initial energy of a sensor node
C	Total battery capacity
t_r	Maximum recharge time of a sensor node
t_i	Recharge time of node i
α	Weight parameter $\alpha \in [0, 1]$
$\mathcal{L}$	Set of residual lifetime of emergency nodes

V. Theoretical Analysis of Energy Neutrality and Minimum Number of SenCars

In this section, we study a couple of important theoretical questions. Given a sensor network, what is the necessary condition for it to operate perpetually and what is the minimum number of SenCars needed to satisfy this condition? For a rechargeable sensor network, the *energy neutral* condition must be satisfied, i.e., for each sensor node the energy replenished is no less than the energy consumed in any arbitrarily long time period.

We use a simple Bernoulli process to model a node's energy consumption: with probability p it consumes unit energy in a unit time slot [15]. The assumption is quite natural for most of the sensing applications. For example, in an event-based sensor network, events occur sporadically and are governed by a Poisson distribution. Once the unit time slot is small enough such that only one event can occur during a unit time slot, the Poisson distribution is equivalent to Bernoulli distribution [16]. A long time period consists of n unit time slots. Let R_n and E_n denote the energy replenished and consumed for a sensor node in n time slots, respectively, and E_0 denote the node's initial energy. Table I summarizes the general notations and their corresponding definitions in this paper.

Hence, the energy neutral condition is

$$R_n + E_0 \geq E_n \tag{1}$$

Eq. (1) states that on each sensor node, the sum of replenished energy and initial energy should be at least as large as the consumed energy. This is a *necessary condition* for perpetual operation of the network.

From Eq. (1), we can derive the minimum number of SenCars S needed. We first estimate a loose upper bound for R_n in terms of S. Intuitively, SenCars reach their maximum recharging capacity when they can "barely" keep up with the recharging needs. This is when they keep recharging node after node without any idle time in between, and each node has almost zero energy before being recharged. A SenCar can replenish at most the battery's full capacity in the full recharge time[1]. During n time slots, the total recharged energy for the whole network is the recharge rate $((C/t_r)S)$ times the time duration n (e.g., battery capacity C = 780 mAh , full recharge time t_r = 73.4 min for a Panasonic Ni-MH AAA battery [14]). The recharged energy R_n is averaged on each sensor node by dividing the number of sensor nodes N in the network. Thus, the upper bound of R_n is $\frac{nCS}{t_r N}$.

[1]We assume fully recharging batteries to avoid "memory effects" that can reduce the number of charge cycles and maximize the lifetime of a rechargeable battery.

Note that on the right hand side of Eq. (1), E_n is a random variable. We have the following lemma.

Lemma 1. *The probability for the energy neutral condition to hold is* $P_{op} = \Phi\left(\frac{R_n+E_0-np}{\sqrt{np(1-p)}}\right)$.

Proof: Let $X_1, X_2, \ldots, X_n$ be independent and identically distributed Bernoulli random variables for energy consumption in each time slot with probability p. $E_n = \sum_{i=1}^{n} X_i = n\overline{X}$. When n is sampled over a long time period, by the Central Limit Theorem, we know $\overline{X} \sim (p, \frac{p(1-p)}{n})$. Thus, E_n is also normal distributed with $\mu(n) = np$ and variance $\sigma^2(n) = np(1-p)$ ($E_n \sim \mathcal{N}(np, np(1-p))$) [17]. Hence,

$$P_{op} = Pr\{R_n + E_0 > E_n\} = \Phi\left(\frac{R_n + E_0 - \mu(n)}{\sqrt{\sigma^2(n)}}\right)$$

and the energy neutral condition holds with P_{op}. ■

Proposition 1. *The minimum number of SenCars required to achieve perpetual operation is*

$$S = \left\lceil \frac{t_r N(2.33\sqrt{np(1-p)} + np - E_0)}{Cn} \right\rceil$$

Proof: Since $\Phi^{-1}(1) \to \infty$, we consider the sensor network achieves perpetual operation when $P_{op} \geq 0.99$. From $\Phi^{-1}(0.99) \leq \frac{\frac{nCS}{t_r N}+E_0-np}{\sqrt{np(1-p)}}$, we obtain the minimum number of SenCar S. ■

The derivation from Proposition 1 can help network administrator plan the network. Once the experimental parameters and the application specifics from the sensors have been determined (e.g., network size N, recharge time t_r, initial energy E_0, working probability p, operation duration n and battery capacity C), we can easily obtain the minimum number of SenCars needed. As will be seen later, we also validate the correctness of the derivation in simulations.

VI. Emergency Recharge Optimization with Multiple SenCars

In this section, we study the Emergency Recharge Optimization with Multiple SenCars problem (EROMS). Our objective is to minimize the total traveling cost of the SenCars while guaranteeing recharge before sensors' battery depletion. We formalize this problem into a Multiple Traveling Salesmen Problem with Deadlines. We show our problem is NP-hard and propose a heuristic algorithm suitable for dynamic real-time recharging.

A. Problem Formulation

Given a set of SenCars $\mathcal{S}$ and a set of emergency nodes $\mathcal{M}$, we formalize the problem as follows. Consider a graph $G = (V, E)$, where $V_0^{(k)}$ is the starting position of SenCar k, and V_i ($i \in \mathcal{M}$) is the location of emergency sensor i to be visited. E is the set of edges. Each edge E_{ij} has a latency cost $c_{ij} = t_i + t_{ij}$, where t_i is the time to recharge node i from its current energy level to full capacity, and t_{ij} is the traveling time from node i to node j. For SenCar k, $c_{0j}^{(k)}$ represents its cost from its initial position 0 to node j. For each sensor node i, the residual lifetime is L_i. A_i specifies the arrival time for a SenCar at sensor node i.

We introduce decision variables x_{ij} for edge E_{ij}. The decision variable is 1 if an edge is visited, otherwise it is 0. Additionally, $x_{0j}^{(k)}$ is 1 if SenCar k moves from its initial position to node j. u_i is the position of vertex i in the path. We virtually make the SenCars return to $V_0^{(k)}$ after recharging all the selected nodes by setting $c_{i0}^{(k)} = 0, i \in \mathcal{M}$, thus the EROMS problem can be formulated as the Multiple Start Traveling Salesman Problem with Deadlines in which multiple traveling salesmen start from different locations to visit a set of cities within their deadlines.

$$\mathbf{P1}: \quad \min \left\{ \sum_{i=1}^{M} \sum_{j=1}^{M} c_{ij} x_{ij} + \sum_{k=1}^{S} \sum_{j=1}^{M} c_{0j}^{(k)} x_{0j}^{(k)} \right\} \tag{2}$$

Subject to

$$\sum_{j=1}^{M} x_{0j}^{(k)} = \sum_{i=1}^{M} x_{i0}^{(k)} = 1, \forall k = 1, 2, \ldots, S, \tag{3}$$

$$\sum_{i=1}^{M} x_{ik} = \sum_{j=1}^{M} x_{kj} = 1; \forall k = 2, \ldots, M, \tag{4}$$

$$A_i \le L_i; \forall i = 1, 2, \ldots, M, \tag{5}$$

$$x_{ij} \in \{0, 1\}; \forall i, j = 1, 2, \ldots, M, \tag{6}$$

$$2 \le u_i \le M; \forall i = 2, 3, \ldots, M, \tag{7}$$

$$u_i - u_j + (M - S)x_{ij} \le M - S - 1; \quad \forall i, j = 2, 3, \ldots, M, i \ne j. \tag{8}$$

Constraint (3) guarantees that the recharge path starts at 0 and finishes at 0. Constraint (4) ensures the connectivity of the path and that every vertex is visited at most once. Constraint (5) guarantees the arrival time of the SenCar is within sensor's residual lifetime. Constraint (6) imposes x_{ij} to be 0-1 valued. Constraints (7) and (8) eliminate the subtour in the planned route. The subtour elimination constraints are formulated according to [18], [19].

We now show that EROMS is NP-hard. If we remove Constraint (5) and set all the SenCars to start from one position, the problem becomes finding the shortest tour of visiting every sensor exactly once by multiple SenCars with sensors having infinite lifetime, which is known to be another NP-hard problem, Multiple Traveling Salesman Problem (m-TSP). Thus EROMS is NP-hard.

B. Minimum Weighted Sum Heuristic Algorithm and Complexity Analysis

In this subsection, we propose a heuristic algorithm for the EROMS problem that jointly considers the residual lifetime and traveling time. In general, m-TSP is closely related to the Vehicle Routing Problem (VRP) in which a fleet of vehicles start from the same depot and visit client locations except that in m-TSP, salesmen are allowed to start from different locations. The m-TSP with Deadlines can be considered as a special case of the *Multiple Traveling Salesman Problem with Time Windows* (m-TSPTW)[2]. This problem is similar to Vehicle Routing Problem with Time Window (VRPTW) which has been studied in the literature and a handful of optimal and approximation algorithms are available [20]–[24].

The approaches to VRPTW are usually divided into two phases. A construction of a feasible tour is sought in the first phase and the tour is interactively improved in the second phase. In [20], a local search algorithm was proposed to reduce the computation of checking feasibility constraint of TSPTW. In [21], the minimum number of vehicles to meet the time window requirements was studied by utilizing precedence graphs. However, since checking the tour feasibility is as hard as the original problem [20], these approaches are still computationally expensive.

Several approximation algorithms have been proposed for the VRPTW problem in [22], [23]. However, these algorithms are not suitable for the recharging problem context. First, they assume the number of vehicles is unlimited but the number of SenCars is bounded. Second, existing algorithms deal with a static problem input. However, in EROMS, new emergencies may appear at any time, and residual node lifetimes also vary due to ongoing sensing activities. Maintaining an optimal schedule would become prohibitively expensive. Finally, existing algorithms may generate unbalanced workloads among SenCars, resulting in idling SenCars while emergencies still exist.

We present a heuristic algorithm that schedules recharge assignments among SenCars. Two important metrics impact the recharging order between node i and node j: the traveling time between node i to node j, and their residual lifetime L_i and L_j. If node j has a small L_j such that it would be dead if a SenCar recharges node i first, node j should be visited first.

We use a weighted sum w_{ij} of traveling time from the current node i to next node j and the residual lifetime of node j,

$$w_{ij} = \alpha t_{ij} + (1 - \alpha) L_j. \tag{9}$$

w_{ij} is used to decide which node j to recharge next. A sensor node with a smaller weighted value should be visited at a higher priority. When $\alpha = 1$, the algorithm reduces to nearest node selection that the SenCars always recharge the closest node first regardless of battery deadlines; when $\alpha = 0$, it picks the node with the earliest battery deadline first regardless of the traveling distance.

When a SenCar performs calculation, it communicates via a long range radio with other SenCars to know their positions

[2]m-TSP with Deadlines is m-TSPTW having all the release time at 0.

TABLE II
MINIMUM WEIGHTED SUM ALGORITHM

```
Input: weight parameter $\alpha \in [0,1]$ in stepsize $1/(A-1)$,
position of SenCar at node $k$, emergency set $\mathcal{M}$, traveling
time from $i$ to $j$, $t_{ij}$, residual lifetime $L_i, \forall i, j \in \mathcal{M}$,
node list $\Omega_i$ at service station, $i \in \mathcal{N}$.
Output: result weight parameter $\alpha$ and schedule sequence $Q$.
Initialize minDist $= \infty$
For $\alpha = 0, \ldots, 1$
 While $\mathcal{M} \neq \emptyset$
  Compute weight $w_{kj} \leftarrow \alpha t_{kj} + (1-\alpha) L_j$.
   Communicate service station If
   $\Omega_i = 1$, Set $w_{ki} = \infty$.
   End if
   Find $j \leftarrow \arg\min_j w_{kj}$.
   $Q_t \leftarrow Q_t + j$, $M \leftarrow M - j$.
   update $\forall i \in M$, $L_i \leftarrow L_i - t_{kj} - t_j$.
   If $L_i \leq 0$
   Declare infeasible and break (Inform service station).
   End if
   Move to position $j$, $k \leftarrow j$, recharge and update lifetime $L_j$
  End while
  If feasible
  Compute total cost dist($Q_t$).
   If dist($Q_t$) $<$ minDist,
   minDist $\leftarrow$ dist($Q_t$), $Q \leftarrow Q_t$.
   End if
  End if
End for
```

for computing the weighted sum. To avoid conflicts where multiple SenCars choose the same node for recharge, we utilize the service station to store and update the availability of each node. The procedure is similar to memory access in operating systems [25]. The service station maintains a 0-1 valued node list Ω. Once a sensor is chosen, its value is set to 1 (locked). Otherwise, it is 0. The value should be changed back from 1 to 0 when a SenCar finishes recharging that node. A SenCar can simply communicate with the service station, exclude nodes already selected by other SenCars, and notify the service station of the status of nodes it chooses. Table II shows the pseudo-code of the entire algorithm.

We now analyze the complexity of the heuristic algorithm. Note that the node selection operations are executed on each SenCar, which takes $\mathcal{O}(M + \log M)$ time. For each SenCar, it performs M/S rounds of node selections and the total number of tests on α is A. Thus, the total computational complexity of our heuristic algorithm is $\mathcal{O}(\frac{AM}{S}(M + \log M))$.

VII. PERFORMANCE EVALUATIONS

In this section, we use simulation to evaluate the effectiveness and efficiency of our framework. We have developed a discrete event-driven simulator using POSIX thread programming in C language. We examine two network sizes of 500 and 1000 sensor nodes, uniformly randomly distributed over a $200 \times 200 m^2$ and $282 \times 282 m^2$ square field, respectively. The field size is chosen so that the two cases have the same node density. The network consists of 3-level hierarchy with 4^l number of subareas at the l-th level. The energy consumption on each sensor is a Bernoulli random variable with probability p to consume unit energy (37.5 mJ). If a sensor node works continuously at this rate, the battery can last for 5 days. The relationship between recharged energy and recharge time follows that of Panasonic Ni-MH AAA battery [14]. To understand the impact of the number of SenCars on network performance, we show marginal cases where the number of SenCars is not sufficient while adding one more SenCar would guarantee perpetual operations. These cases are $S = 2, 3$ for $N = 500$ and $S = 4, 5$ for $N = 1000$. We will show these cases in the following and validate the correctness of Proposition 1. All the parameter settings in the simulation are listed in Table III.

TABLE III
PARAMETER SETTINGS

Parameter	Value
Field Length	200×200, $282 \times 282 m^2$
Number of Nodes N	500, 1000
Number of SenCars S	1, 2, 3, 4, 5
Number of Levels	3
Areas on l-th level	4^l
Battery Capacity	780 mAh
Transmission Range	18 m
Unit Energy Consumption r_c	37.5 mJ
Energy Consumption Probability p	0.5
SenCar Speed	1 m/s
Maximum Recharge Time	73.4 mins
Normal Recharge Threshold	50%
Emergency Recharge Threshold	10%
Simulation Time	6 months

A. Evaluation of Weighted-sum Algorithm

In this subsection, we evaluate the effectiveness of the weighted-sum algorithm in finding the shortest path and achieving no node failure. We examine cases when 4 SenCars are employed. We assume the locations of emergencies are randomly distributed in the field of $282 \times 282 m^2$, and the residual energy uniformly distributed from zero to the emergency threshold. The corresponding residual lifetime is calculated by dividing the residual energy by pr_c, the expected energy consumption in unit time.

Table IV shows the total distance of SenCars when the number of concurrent emergencies M increases from 72 to 96 in a step of 8. Note that when the number reaches 96, the set of 4 SenCar is not sufficient to resolve all the emergencies without complete battery depletion. For $M = 88$, $\alpha = 0.8, 1$ are not feasible and for $M = 72, 88$, $\alpha = 1$ is not feasible either. We notice that in the case when $\alpha = 1$, some nodes that suffer from energy shortage may not get recharged in a higher priority thereby rendering the result infeasible to avoid battery depletion. As we can see from this example, the choice of α is critical, when α approaches 1, the total distance is decreased at the risk of becoming infeasible. Thus, we need to search for α in our algorithm. In real applications, the value of α is subject to change and determined by real-time statistical data and parameters.

B. Performance Evaluations

In this subsection, we evaluate the energy evolution of the network, the number of emergency and nonfunctional (i.e., energy depleted) nodes of the network, and the maintenance cost of the framework.

TABLE IV
TOTAL TRAVELING DISTANCE OF SENCARS, D

M	D ($\alpha = 0$)	D ($\alpha = 0.2$)	D ($\alpha = 0.4$)
72	7524.1	7473.3	7740.2
80	7652.4	7578.9	7706.6
88	8662.6	8128.3	7251.6
96	Infeasible	Infeasible	Infeasible
M	D ($\alpha = 0.6$)	D ($\alpha = 0.8$)	D ($\alpha = 1$)
72	6843.5	**6390.6**	Infeasible
80	7271.8	**6941.0**	Infeasible
88	**6998.3**	Infeasible	Infeasible
96	Infeasible	Infeasible	Infeasible

1) Energy Evolution: First, we show the energy evolution in the network of 500, 1000 nodes served by different numbers of SenCars. In Fig. 2, the amount of energy consumed and replenished in every one-hour time slot is plotted as functions of simulation time. In Fig. 2(a) and (c), we can see that the consumed energy "steps down" to a lower level around 400 hours and then enters equilibrium. This is because a portion of sensor nodes deplete their energy and do not get recharged. In these two scenarios the energy neutral condition has been violated, simply because the number of SenCars is not enough. Fig. 2(b) and (d) show the energy evolution when the numbers of SenCars is increased by 1, both of which satisfy the energy neutral condition at the equilibrium and there is no such "step-down" effect in energy consumption.

2) Number of Emergencies: Fig. 3 and Fig. 4 compare the percentage of nodes in emergency and nonfunctional (i.e., energy at zero) status for networks of 500 and 1000 nodes with different numbers of SenCars. First, we can see that there are surges in the numbers of emergency and nonfunctional nodes during the first 200 hours. This is due to the fact that the SenCars only responds to requests when the node energy is below the normal recharge threshold. When such requests swarm into the job queues on the SenCars at the beginning of 200 hours, we can see that the SenCars' capacity has been temporarily exceeded. As the energy of sensors is restored, the numbers of emergency and nonfunctional nodes decrease sharply.

To illustrate the consequences of insufficient number of SenCars, we vary the number of SenCars S over a range including the minimum number needed for energy neutrality. Fig. 3 and Fig. 4 show the number of emergency and nonfunctional nodes over time. For cases $N = 500, S = 2$ and $N = 1000, S = 4$ when the number of SenCars is insufficient for energy neutral, we can see that about 30% nodes are in constant emergency and 20% nodes are in nonfunctional status after the network achieves equilibrium. For $N = 500, S = 3$, there are occasional nonfunctional nodes but they were soon recharged by the SenCars. For a majority of the time, the number of nonfunctional nodes stays at zero. For $N = 1000, S = 5$, the number of nonfunctional nodes stays at zero at equilibrium with only a small number of emergencies.

Recall from Proposition 1 that the minimum number of SenCars for $N = 500$ and $N = 1000$ can be calculated as $S = \lceil 2.41 \rceil = 3$ and $S = \lceil 4.84 \rceil = 5$ for the given parameter settings in Table III. These numbers match well with our simulation results that $S = 3, 5$ are the minimum num-

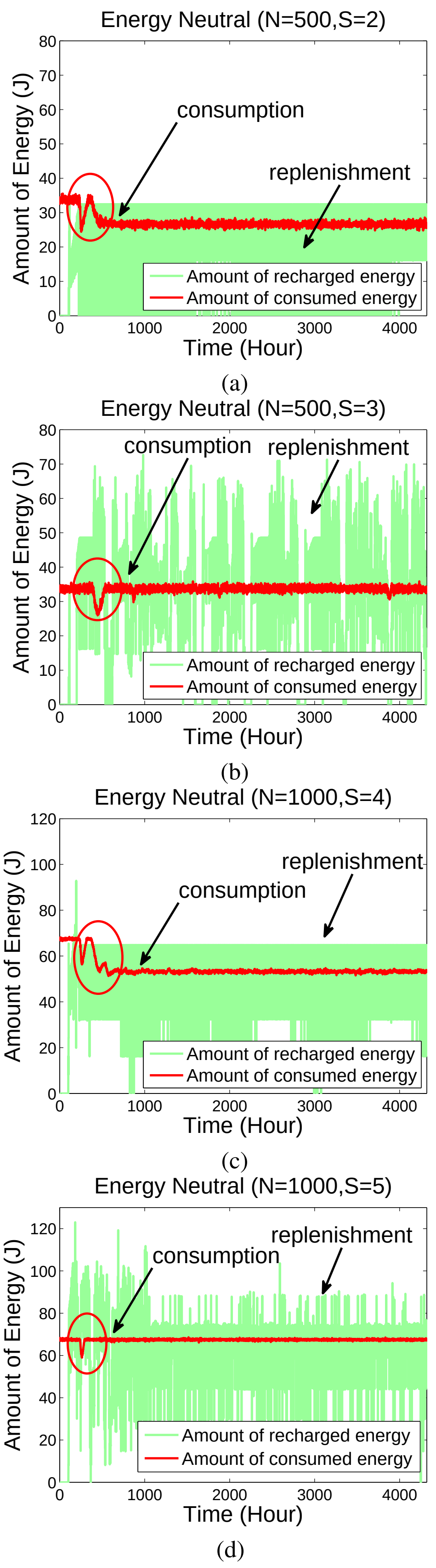

Fig. 2. Evolution of energy consumption vs. energy replenishment in 6 months time. (a) Number of nodes $N = 500$, number of SenCars $S = 2$. (b) Number of nodes $N = 500$, number of SenCars $S = 3$. (c) Number of nodes $N = 1000$, number of SenCars $S = 4$. (d) Number of nodes $N = 1000$, number of SenCars $S = 5$.

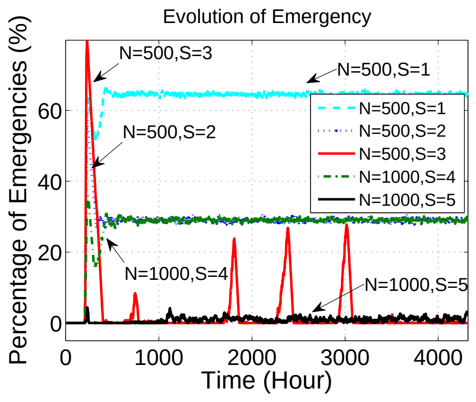

Fig. 3. Number of emergency nodes.

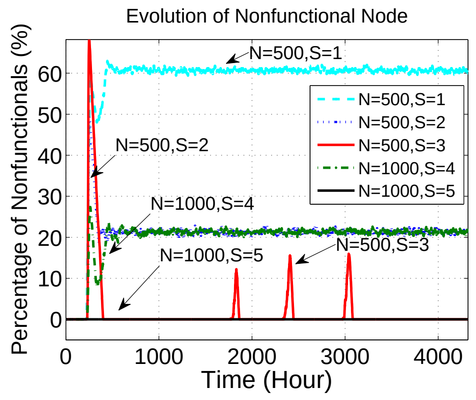

Fig. 4. Number of nonfunctional nodes.

ber of SenCars to achieve perpetual operation at equilibrium, respectively. By utilizing our theoretical analysis, the network administrator can make reasonable estimations for the minimum number of SenCars needed when planning a network.

C. Evaluation of Protocol Overhead

We evaluate the overhead introduced by our protocol, including all types of messages sent by sensor nodes or the SenCar to recharge nodes. Fig. 5 shows the average overhead per node in a 6 month period. After the networks enter equilibrium, the overhead on each sensor node is from 8 to 48 bits per second, which is negligible compared to radio transmission rates in sensor nodes (e.g., 20 - 900 kbps).

From Fig. 5, we can observe that all the four scenarios have a large amount of message transmission when simulations start up. Such bursts are caused by simultaneous head selection processes in all the sub-areas, during which a lot of messages are broadcast. Energy information query also contributes to the bursts which also leads to message broadcast. As time elapses, however, the energy levels of the nodes drop and emergency occurs in the network. So the top-level heads, while receiving *energy interest* messages, respond with emergency information instead of querying lower levels for energy information. Because emergency information is directly reported to top-level heads without propagating through the hierarchy, the amount of messages transmitted in the network decreases.

For $N = 500, S = 2$ and $N = 1000, S = 4$, there are a large number of emergency nodes after the networks

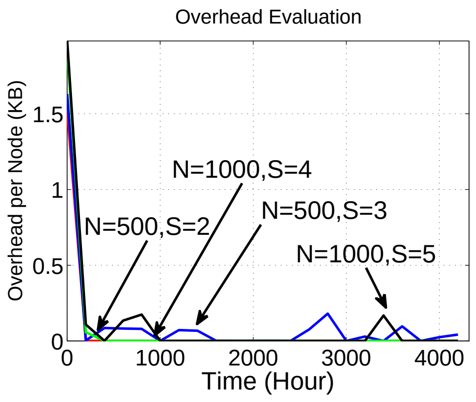

Fig. 5. Average overhead for each sensor node per second.

enter equilibrium (Fig. 3). Thus the SenCars always query for emergency information. In our protocol, *emergency interest* messages sent by the SenCars are relayed directly by proxies so the overhead of message forwarding is quite small.

As the number of SenCars increases, the number of emergencies decreases dramatically and a majority of the time the SenCars query for normal energy information and perform normal recharge. In response to *energy interest*, the heads poll energy information in a top-down method which finally results in the broadcast of *energy interest* message in subareas at the bottom-level. Such broadcast, as well as the transmission of energy information sent by each node, causes the increase of the number of messages transmitted in the networks, which is observed as the spikes in the curves.

D. Discussions

The simulation results have demonstrated the effectiveness of our framework for handling both emergency and normal recharge requests. We also validate our theoretical derivations on the minimum number of SenCars for perpetual operation. The results also show that coordinating multiple SenCars to perform the recharge assignments improves the network scalability and immunity to burst of emergencies and the communication overhead is negligible compared to the data rate in sensor networks.

VIII. CONCLUSIONS

In this paper, we study the coordination of multiple mobile vehicles to recharge wireless sensor networks. We develop a comprehensive set of communication protocols based on NDN concepts and mechanisms to enable effective and efficient energy information gathering and delivery. The protocols adapt to unpredictable network conditions and satisfy the needs for both normal and emergency recharging. We formally analyze the probability for the energy neutral condition, which is required for perpetual operations. We derive the minimum number of SenCars needed to achieve this condition. We then model the Emergency Recharge Optimization problem with multiple SenCars into the category of m-TSP with Deadline problem and provide a fast, efficient heuristic algorithm suitable for dynamic network conditions. The extensive simulation results demonstrate the efficiency and effectiveness of the proposed algorithm and the framework, and validate the correctness of theoretical analysis.

References

[1] A. Kurs, A. Karalis, R. Moffatt, J. D. Joannopoulos, P. Fisher and M. Soljacic, “Wireless power transfer via strongly coupled magnetic resonances,” *Science*, vol. 317, pp. 83, 2007.

[2] A. Karalis, J. D. Joannopoulos and M. Soljacic, “Efficient wireless non-radiative mid-range energy transfer,” *Annals of Physics*, vol. 323, no. 1, pp. 34-48, Jan. 2008.

[3] PowerCast Corp, “http://www.powercastco.com.”

[4] M. Rahimi, H. Shah, G. Sukhatme, J. Heideman and D. Estrin, “Studying the feasibility of energy harvesting in a mobile sensor network,” *IEEE International Conference on Robotics and Automation*, 2003.

[5] A. Kansal, J. Hsu, M. Srivastava and V. Raqhunathan, “Harvesting aware power management for sensor networks,” *43rd ACM/IEEE Design Automation Conference*, 2006.

[6] B. Tong, Z. Li, G. Wang and W. Zhang, “How wireless power charging technology affects sensor network deployment and routing,” *IEEE ICDCS*, 2010.

[7] S. He, J. Chen, F. Jiang, D. Yau , G. Xing and Y. Sun, “Energy provisioning in wireless rechargeable sensor networks,” *IEEE INFOCOM*, 2011.

[8] M. Zhao, J. Li and Y. Yang, “Joint mobile energy replenishment and data gathering in wireless rechargeable sensor networks,” *IEEE ITC*, 2011.

[9] Y. Shi, L. Xie, T. Hou and H. Sherali, “On renewable sensor networks with wireless energy transfer,” *IEEE INFOCOM*, 2011.

[10] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs and R. Braynard, “Networking named content,” *ACM CoNEXT*, 2009.

[11] M. Ma and Y. Yang, “Data gathering in wireless sensor networks with mobile collectors,”*IEEE IPDPS*, 2008.

[12] A. Somasundara, A. Ramamoorthy and M. Srivastava, “Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines,” *IEEE RTSS*, 2004.

[13] I. Chatzigiannakis, A. Kinalis, S. Nikoletseas and J. Rolim, “Fast and energy efficient sensor data collection by multiple mobile sinks,” *ACM MobiWac*, 2007.

[14] Panasonic Ni-MH battery handbook, “http://www2.renovaar.ee/userfiles/Panasonic_Ni-MH_Handbook.pdf”.

[15] Y. Wang, M. C. Vuran and S. Goddard, “Stochastic analysis of energy consumption wireless sensor networks,” *Proc. of IEEE Secon*, 2010.

[16] R. Nelson, *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling*, Springer, 1995.

[17] S. Ross, *A First Course in Probability*, 8th Ed, Prentice Hall, 2009.

[18] C. Miller, A. Tucker and R. Zemlin,“Integer programming formulations and traveling salesman problems,” *Journal of the ACM*, pp. 326-329, 1960.

[19] B. Gavish, “A note on the formulation of the m-salesman traveling salesman problem,” Management Science, 1976.

[20] M.W.P. Savelsbergh, “Local search in routing problems with time windows,” *Annals of Operation Research*, pp. 285-305, 1985.

[21] Snezana Mitrovic-Minic, Ramesh Krishnamurti, “The multiple TSP with time windows: vehicle bounds based on precedence graphs,” *Operations Research Letters*, pp. 111 - 120, 2006.

[22] M. Desrochers, J.K. Lenstra, M.W.P. Savelsbergh and F. Soumis, “Vehicle routing with time windows: optimization and approximation,” Elsevier Science Publisher, 1988.

[23] N. Bansal, A. Blum, S. Chawla and A. Meyerson, “Approximation algorithms for Deadline-TSP and Vehicle Routing with Time Windows,” *ACM STOC*, 2004.

[24] M. Gendreau, A. Hertz, G. Laporte and M. Stan, “A generalized insertion heuristic for the traveling salesman problem with time windows,” *Journal Operations Research*, vol. 46 no. 3, 1998, pp. 330-335.

[25] A. Tanenbaum, *Modern Operating Systems*, pp. 125, 3rd Ed, Prentice Hall, 2007.