# Joint Energy Management and Resource Allocation in Rechargeable Sensor Networks

Ren-Shiou Liu
Department of Computer Science and Engineering
The Ohio State University
Email: rsliu@cse.ohio-state.edu

Prasun Sinha
Department of Computer Science and Engineering
The Ohio State University
Email: prasun@cse.ohio-state.edu

Can Emre Koksal
Department of Electrical and Computer Engineering
The Ohio State University
Email: koksal@ece.osu.edu

***Abstract*—Energy harvesting sensor platforms have opened up a new dimension to the design of network protocols. In order to sustain the network operation, the energy consumption rate cannot be higher than the energy harvesting rate, otherwise, sensor nodes will eventually deplete their batteries. In contrast to traditional network resource allocation problems where the resources are static, time variations in recharging rate presents a new challenge. In this paper, we first explore the performance of an efficient dual decomposition and subgradient method based algorithm, called QuickFix, for computing the data sampling rate and routes. However, fluctuations in recharging can happen at a faster time-scale than the convergence time of the traditional approach. This leads to battery outage and overflow scenarios, that are both undesirable due to missed samples and lost energy harvesting opportunities respectively. To address such dynamics, a local algorithm, called SnapIt, is designed to adapt the sampling rate with the objective of maintaining the battery at a target level. Our evaluations using the TOSSIM simulator show that QuickFix and SnapIt working in tandem can track the instantaneous optimum network utility while maintaining the battery at a target level. When compared with IFRC, a backpressure-based approach, our solution improves the total data rate by $42\%$ on the average while significantly improving the network utility.**

## I. Introduction

In various application scenarios, energy can be harvested from the surrounding environment to recharge the batteries and extend the network's lifetime. Many forms of energy such as solar, wind, water flow, thermal and vibration are being explored for driving sensor network platforms [1], [2], [3], [4]. Such sensor platforms have opened up a new dimension to the design of networking protocols. However, the strength of harvested energy is a function of various static parameters, such as the specifications and orientation of the solar panel; and time-varying parameters, such as the weather and the season. The protocol needs to adapt to such dynamics to avoid running out of battery. This is especially needed for environmental monitoring applications as they often require periodic data collection from all nodes in the network and high data throughput with fairness guarantee is desired. *Our objective in this paper is to design a distributed and adaptive solution that jointly computes the data collection rates for each node, finds the routes and schedules transmissions based on interference constraints and energy replenishment rates, such that the network utility can be maximized while maintaining perpetual operation of the network.*

This material is based upon work supported by the National Science Foundation under Grants CNS-0721434, CNS-0721817, CNS-0831919, CCF-0916664, and CCF-0635242.

Similar to the standard flow control and resource allocation (e.g. [5], [6]) approaches, we formulate the problem as a constrained optimization problem. In our formulation, we have an **energy conservation constraint** involving the replenishment rates, along with the standard flow conservation and interference constraints. The **fundamental difference** of our problem is that, our energy conservation constraint is **time-varying**, due to the time-varying nature of energy replenishment, i.e., our problem is **dynamic** and thus it has a different solution in every point in time. The standard implementations of the dual decomposition method for resource allocation involves a large number of iterations, each of which incurs a high overhead due to the necessary message exchange between neighboring nodes. In a static setting, the convergence time is less of an issue, since the solution of the problem is fixed. However, with time-varying replenishment rate, we have a time-varying optimal resource allocation in our problem.

To that end, the first major question we answer in this paper is, "how closely can we **track** that optimal point by using distributed iterations to solve the Lagrangian dual problem?" We show (in Section IV) that, in a network with arbitrary topology, the time scale of such solutions is too slow to follow the variations in the replenishment rate for the optimal point to be tracked sufficiently closely. Consequently, we focus on well-structured networks with an underlying *directed acyclic network* graph (DAG). We exploit the structure of (DAG) to develop an efficient synchronous message passing scheme, **QuickFix**, motivated by general solution structure of dynamic programs. We show that the convergence time of QuickFix is sufficiently small to track the optimal solution fairly closely.

The second major question we address is, "how does the finiteness of the total amount of energy storage in each node affect the performance of joint energy management and resource allocation?" If one accounts for the finite batteries in the original problem formulation, the solution involves complicated Markov decision processes, which does not lead to better understanding of the dynamics of the system. Instead, we propose a simple novel **adaptive localized algorithm**,

**SnapIt**, that operates above QuickFix, to slightly modify the optimal sampling rate provided by QuickFix, based on local battery states. We show that, using SnapIt along with QuickFix has two important consequences. Firstly, the fraction of time for which a node is down reduces significantly due to a complete battery discharge reduces significantly (to 0 in many instances as shown in our evaluations). Secondly, due to the short response time of SnapIt, the joint algorithm is capable of responding to the changes in replenishment rate much more quickly. Hence, the overall network utility also increases with SnapIt.

In summary, the key contributions of this paper are as follows:

- We develop a decomposition and subgradient based distributed solution, called QuickFix, that can efficiently track instantaneous optimal sampling rates and routes in the presence of a time-varying recharging rates. Prior works based on such techniques have mostly focused on static scenarios.
- For perpetual operation in the presence of rapid fluctuations in the the recharging rates, we present a new local algorithm, called SnapIt, that attempts to maintain the battery at a certain target level while stochastically maintaining a high network utility.
- Simulations using TOSSIM [7] show that the algorithms working in tandem can lead to an efficient solution for simultaneously achieving proportional fairness and perpetual operation. It is also shown to significantly outperform IFRC, a backpressure-based protocol.

## II. Related work

**Rate Control and Routing:** Routing in energy harvesting sensor network is explored in [8]. The authors proposed two heuristic protocols. The first protocol is a localized solution that allows the source node and exactly one intermediate node to deviate from the shortest path and opportunistically forward packets to a solar-powered neighboring node. On the other hand, the second protocol, with the assistance of the sink, chooses the shortest path with the minimum number of nodes that run solely on the batteries. In another work [9], an online routing algorithm which seeks to maximize the number of accepted flows is proposed. In contrast to these works, we consider joint computation of routes and rates with a different objective of fair data collection and perpetual operation.

Lexicographically maximum rate assignment and routing for perpetual data collection has been studied in [10]. It extends the framework in [11], and presents a centralized as well as a distributed algorithm. The centralized algorithm is proven to give the optimal lexicographic rate assignment. However, the distributed algorithm can only reach the optimum if the routing paths are predetermined, and the network is a tree rooted at the sink. Furthermore, the dynamics of recharging profiles is not considered. The exact recharging profile for each day must be known and it is sensitive to the initial battery levels. In contrast, our distributed solution does not require any knowledge of the future recharging rates. Furthermore, our distributed solution can work on a more generalized structure than a tree. All of these factors result in a more practical algorithm in comparison to the algorithm proposed in [10].

Other prior works on rate control in WSNs include [12], [13], [14], [15], [16]. Although some of these works [14], [15] also aim to achieve fairness in WSNs, none of them consider energy replenishment. In this paper, we re-engineered IFRC [14] to consider recharging capabilities of modern sensor platforms and use it as a benchmark.

**Dual-Decomposition Technique for Optimizing Network Performance:** The dual decomposition technique has been applied in many works to maximize the network utilization. In [5], the utility function, as ours, is defined as the summation of the $log$ of the rate achieved by each flow in the network. However, in their formulation, the route for each flow is fixed. The authors propose both a primal and a dual-based algorithm. Both algorithms tune the transmission attempt probability of each node to maximize the utility function. In contrast to [5], a general utility function is considered in [6]. It also applies the dual decomposition technique to develop a distributed algorithm. However, the algorithm not only addresses rate assignment, but also routing. Maxmin fairness of rate assignment problem in WSNs is studied in [17]. Their proposed algorithm is also based on dual decomposition. In their model, each sensor transmits with a certain probability. Thus, after Lagrangian relaxation, the dual function is not concave. To overcome this problem, they convexify their problem and apply the Gauss-Seidel method to solve the problem. Although all the above approaches allow nodes in the network self-tune certain parameters to maximize the network utilization, none of them consider energy constraints, not to mention recharging.

## III. Network Model

This section outlines the network model and the problem formulation. We consider a static sensor network represented as $G = (N, L)$, where $N$ is the set of sensor nodes including the sink node, $s$, and $L$ is the set of directed links. We assume a DAG (directed acyclic graph) rooted at the sink is constructed over the network for data collection. The problem is formulated as a convex optimization problem by exploiting the DAG structure. This formulation not only provides clear system design guidelines but also allows us to design an efficient signaling scheme. Before presenting the problem formulation, we present some of the key notations used in the paper. The notations and their corresponding semantics are also listed in Table I for reference.

For each sensor node $i \in N$, $A_i$ denotes its ancestors in the DAG, i.e., the nodes that are on some path(s) from node $i$ to the sink $s$. Conversely, $D_i$ denotes the descendants. Also, $C_i$ and $P_i$ are the children and parent nodes of node $i$ respectively. The amount of energy consumed in sensing the environment is represented by $\lambda_i^{(sn)}$. We assume that the the expected number of retransmissions over a long time is known for each link. Thus, $\lambda_{ij}^{(tx)}$ represents the average energy consumption for

TABLE I
NOTATIONS

| Symbol | Semantics |
|---|---|
| $D_i$ | The set of descendants of node $i$ |
| $A_i$ | The set of ascendants of node $i$ |
| $C_i$ | The set of children nodes of node $i$ |
| $P_i$ | The set of parent nodes of node $i$ |
| $B_i(t)$ | Battery state of node $i$ at time $t$ |
| $M_i$ | Battery capacity of node $i$ |
| $\pi_i(e)$ | Estimated average recharging rate of node $i$ in epoch $e$ |
| $\rho_i(t)$ | The instantaneous recharging rate of node $i$ at time $t$ |
| $\tau$ | Epoch length |
| $r_i$ | Sampling rate of node $i$ |
| $\lambda_i^{(sn)}$ | Energy cost for sampling at node $i$ |
| $\lambda_{ij}^{(tx)}$ | Average energy cost for TX over link $(i,j)$ |
| $\lambda_i^{(rx)}$ | Energy cost for RX at node $i$ |
| $f_{ij}$ | The amount of capacity allocated on link $(i,j)$ |
| $w_{ij}$ | The fractional amount of node $i$'s traffic that passes link $(i,j)$ |
| $z_{ij}(\mathbf{w})$ | A function of w, that represents the fractional amount of $i$'s traffic that passes $j$ |
| $\Pi$ | The feasible region of link capacity variables $f_{ij}$ |
| $W$ | The feasible region of routing variables $w_{ij}$ |
| $\mu_i$ | Lagrange multiplier for energy conservation constraint at node $i$ |
| $\upsilon_i$ | Lagrange multiplier for flow balance constraint at node $i$ |
| $\alpha$ | The constant step size used in the subgradient method |

delivering a packet over link $(i,j)$ and $\lambda_i^{(rx)}$ represents the energy cost for receiving a packet at node $i$. These parameters are the *energy costs* of the system. In particular, we consider slotted-time system and the time during a day is broken into multiple time intervals called epochs. The length of each epoch is $\tau$ slots. We use $\pi_i(e)$ to represent the average (long term) energy replenishment rate of node $i$ in epoch $e$, while $\rho_i(t)$ is used to represent the real instantaneous (short term) energy replenishment rate of node $i$ in time slot $t$. For each epoch $e$, we define $\pi_i(e) = \mathrm{E}\left[\sum_{t=(e-1)\tau+1}^{e\tau} \rho_i(t)\right]/\tau$. We assume that $\pi_i(e)$ can be estimated by each node with high accuracy. Estimating $\pi_i(e)$ is beyond the scope of this paper. Moreover, we define $w_{ij}$ to be the fractional outgoing traffic of $i$ that passes through a *parent* $j$ and $z_{ij}(w)$ to be the fractional outgoing traffic of $i$ that passes through an *ancestor* $j$. Thus, $w_{ij} = 0$, $\forall j \notin P_i$ and $z_{ij}(w) = 0$, $\forall j \notin A_i$. Note that $z_{ij}(w)$ is a function of $w$, where $w$ is the vector of $w_{ij}$ $\forall i,j$. The recursive relation between the two variables is given below.

$$z_{ij}(w) = \sum_{k \in P_i} w_{ik} z_{kj}(\mathbf{w}) \tag{1}$$

## IV. A Cross-Layer Approach to Dynamic Resource Allocation

In this section, we present an outline of the dual based cross-layer framework for distributedly computing the rates and routes in each epoch. We define the utility function $U_i(r_i)$ for node $i$ to be $\log(r_i)$, where $r_i$ is the sampling rate of node $i$. Our goal is to maximize the sum of the utility functions $\sum U_i(r_i) = \sum \log r_i$, which is strictly concave and known to achieve proportional fairness [18]. Thus, we formulate our problem as an optimization problem as follows:

$$P_e\text{:} \quad \max_{r_i, f_{ij}, w_{ij}} \quad \sum_i \log(r_i) \tag{2}$$

$$s.t. \quad \pi_i(e) \geq \lambda_i^{(sn)} r_i + \sum_{j \in D_i} z_{ji}(w) \lambda_i^{(rx)} r_j + \sum_{j \in P_i} \lambda_{ij}^{(tx)} f_{ij} \tag{3}$$

$$\sum_{j \in P_i} f_{ij} \geq r_i + \sum_{j \in D_i} z_{ji}(w) r_j$$

$$r_i \in R^+, \quad f_{ij} \in \Pi, \quad w_{ij} \in W$$

The first and the second constraints are the energy conservation and flow balance constraints, respectively. The flow balance constraint states that the sum of allocated capacity for each outgoing link should be greater than the total amount of traffic going through each node, including its own data. Besides these two constraints, the amount of capacity allocated on each link must be in the feasible capacity region $\Pi$. We assume the node exclusive interference model as in [6]. Thus, the feasible capacity region can be similarly defined as the convex hull of all the rate vectors of the matchings in $G$.

One can observe that this problem is dynamic, since the energy conservation constraint involves the time varying replenishment rate $\pi_i(e)$. Thus, the the feasible region and the optimum solution differ in each epoch. One can view this dynamic problem as a sequence of static problems. The standard method to solve each static problem involves the application of the dual decomposition and the subgradient methods. However, the implementation of these solutions in the network involves a large overhead due to message exchange between neighboring nodes. Consequently, the convergence time becomes an important issue.

To that end, we introduce QuickFix, which, in each iteration of the subgradient method, exploits the special structure of DAG to form an efficient control message exchange scheme. This scheme is motivated by the general solution structure of a dynamic program. QuickFix is based on the hierarchical decomposition approach as the starting point. By relaxing the energy conservation and flow balance constraints in Problem $P_e$, we get the partial dual function $q(\mu, \upsilon)$ as follows:

$$q(\mu,\upsilon) = \max_{r_i, f_{ij}, w_{ij}} \quad \sum_i \log(r_i)$$

$$+ \sum_i \mu_i \left( \pi_i(e) - \lambda_i^{(sn)} r_i - \sum_{j:(i,j) \in D_i} z_{ji}(w) \lambda_i^{(rx)} r_j - \sum_{j \in P_i} \lambda_{ij}^{(tx)} f_{ij} \right)$$

$$+ \sum_i \upsilon_i \left( \sum_{j \in P_i} f_{ij} - r_i - \sum_{j:(i,j) \in D_i} z_{ji}(w) r_j \right)$$

$$s.t. \quad r_i \in R^+, \quad f_{ij} \in \Pi, \quad w_{ij} \in W$$

The dual problem is therefore:

$$\min_{\mu_i \geq 0, \upsilon_i \geq 0} q(\mu, \upsilon) \tag{4}$$

The dual problem in (4) can be decomposed hierarchically [19], [20], [21] as follows. The top level dual master problem

is responsible for solving the dual problem. Since the dual function is not differentiable, the subgradient method [21], [22] is applied to iteratively update the Lagrange Multipliers $\mu_i$ and $\upsilon_i$ at each node using (5). The notation $[.]^+$ means projection to the positive orphan of the real line $R$.

$$\begin{aligned}\mu_i^{k+1} =& \Big[\mu_i^k - \alpha_1\Big(\pi_i(e) - \lambda_i^{(sn)} r_i \\ & - \sum_{j\in D_i} z_{ji}(w)\lambda_i^{(rx)} r_j - \sum_{j\in P_i} \lambda_{ij}^{(tx)} f_{ij}\Big)\Big]^+ \\ \upsilon_i^{k+1} =& \Big[\upsilon_i^k - \alpha_2\Big(\sum_{j\in P_i} f_{ij} - r_i - \sum_{j\in D_i} z_{ji}(w) r_j\Big)\Big]^+ \end{aligned} \tag{5}$$

Each time the Lagrange Multipliers are updated, a primal decomposition is performed to decouple the coupling variables $f_{ij}$, such that when $f_{ij}$ are fixed, the problem can be further decomposed into two layers of subproblems. The upper level optimization problem can then be further decomposed into many smaller subproblems, one for each $i$, as shown in (6):

$$\begin{aligned} \max_{r_i, w_{ij}} \quad & \log(r_i) - \mu_i \left(\lambda_i^{(sn)} r_i + \sum_{j\in D_i} z_{ji}(w)\lambda_i^{(rx)} r_j\right) \\ & - \upsilon_i \left(r_i + \sum_{j\in D_i} z_{ji}(w) r_j\right) \\ s.t. \quad & r_i \in R^+, \quad w_{ij} \in W \end{aligned} \tag{6}$$

At the lower level, we have the optimization problem shown in (7), which is in charge of updating the coupling variables $f_{ij}$

$$\begin{aligned} \max_{f_{ij}} \quad & \sum_{(i,j)\in L} (\upsilon_i - \lambda_{ij}^{(tx)}\mu_i) f_{ij} \\ s.t. \quad & f \in \Pi \end{aligned} \tag{7}$$

The lower level optimization problem in (7) is equivalent to the maximum weight matching problem. Under the node exclusive interference model, it can be solved in polynomial time. However, in order to solve the maximum weight matching problem in a distributed fashion, we utilize the heuristic algorithm in [23]. While applying the algorithm, instead of the queue difference between neighboring nodes, we use the combined energy and queue state of a node ($\upsilon_i - \lambda_{ij}^{(tx)}\mu_i$) to modulate the MAC contention window size, when a node $i$ attempts to transmit a packet over the link $(i, j)$.

Note that in the upper level optimization problem, the $f_{ij}$ and $\upsilon_i$ variables are fixed, and since the objective function is strictly concave in $(r, w)$, it admits a unique maximizer as shown in (8).

$$r_i^* = \frac{1}{\lambda_i^{(sn)}\mu_i + \upsilon_i + \sum_{j\in A_i} z_{ij}(w^*)(\lambda_j^{(rx)}\mu_j + \upsilon_j)} \tag{8}$$

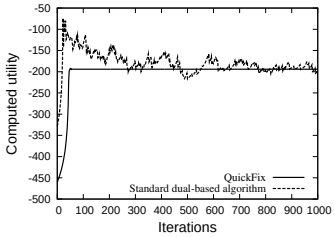



Fig. 1. The convergence time comparison between QuickFix and the standard dual-based algorithm.

We refer to $\lambda_i^{(sn)}\mu_i + \upsilon_i$ in Eq. (8) as node $i$'s local price, and $P_i = \sum_{j\in A_i} z_{ij}(w)(\lambda_j^{(rx)}\mu_j + \upsilon_j)$ as its aggregate price. Observe that if a node wants to maximize its rate, it should find the path(s) such that its aggregate price $P_i$ is minimized, i.e., it is a joint routing and rate control problem. Since our formulation utilizes the DAG structure, this allows a node to calculate its aggregate price recursively from those of its parents as stated in Proposition 1. Furthermore, Proposition 1 implies that a node should select the parent with the minimum sum of local and aggregate prices as its relay node in the DAG. This motivates the following distributed routing and rate control algorithm. Each node collects the local and aggregate prices from all its parents and selects the parent(s) with the minimum sum of the local and aggregate prices as the relay node(s) in the DAG. Then, each node uses (9) to calculate its own aggregate price and then applies (8) to determine its rate. Having determined the local rate and the outgoing link(s) to use, a node distributes its local and aggregate prices to its children, so that the children nodes can determine their routes and rates. This process continues until the leaf nodes are reached. Now, starting from the leaf nodes, each node reports its aggregate traffic to its parents, so that the parent nodes can have the needed information to update their local prices. Aggregate traffic $T_i$ of node $i$ is the total amount of traffic generated by the descendants that goes through node $i$. Similar to the computation of the aggregate price, a node can compute its aggregate traffic recursively using (10). The proof of Propositions 1 and 2 can be found in the Appendix.

**Proposition 1.** *The aggregate price $P_i$ can be recursively computed as*

$$P_i = \sum_{j\in P_i} w_{ij}\Big((\lambda_j^{(rx)}\mu_j + \upsilon_j) + P_j\Big) \tag{9}$$

**Proposition 2.** *The aggregate traffic $T_i$ of node $i$ can be recursively computed as*

$$T_i = \sum_{k\in C_i} w_{ki}\,(r_k + T_k) \tag{10}$$

Figure 1 compares the convergence time of QuickFix with the standard dual-based algorithm when a fixed $\pi_i(e)$ is given. Here, QuickFix is run for a DAG of 67 nodes. The

**Algorithm 1**: QuickFix: Distributed routing, rate control and scheduling algorithm for energy-harvesting sensor networks based on our proposed DAG formulation

**while** *true* **do**
  `/* Phase I: */`
  Each node locally adjusts the Lagrange Mulpliers $\mu_i$, $\upsilon_i$ using Eq. (5);
  The sink initiates a new iteration by broadcasting its aggregate price, which is 0, and a new iteration number. Non-sink (sensor) nodes wait until the aggregate price of all the parents have been collected. Once the aggregate price from all the parents have been collected, select the link(s) with the minimum aggregate price. If multiple links are selected, equally split flows among them.;
  Compute aggregate price using Eq. (9);
  Broadcast the aggregate price to the children. Compute the maximum rate using Eq. (8);
  `/* Phase II: */`
  Once a leaf node has determined its rate and outgoing links in Phase I, it reports its local rate to its parent nodes of the selected links. Non-leaf nodes must collect the aggregate traffic from all its children nodes on the DAG. After that, a non-leaf node computes aggregate traffic using Eq.(10) and report it to all parent nodes.;
  Inform MAC layer of newly selected link(s) and their weight(s) $\upsilon_i - \lambda_{ij}^{(sn)}\mu_i$ ;
  Start applying the newly computed rate and routing paths;
**end**

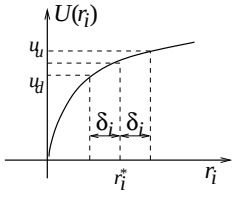



Fig. 2. With our energy management scheme, instantaneous utility alternates between $u_u$ and $u_d$

improvement in convergence rate with QuickFix relative to the standard dual-based solution is apparent. However, we leave the convergence rate analysis for future work.

## V. SnapIt: A Localized Energy Management Scheme for Variable Replenishment Rate

We introduce a localized scheme called SnapIt that uses the current battery level to adapt the rate computed by QuickFix with the objective of maintaining the finite-capacity battery at a target level. This mechanism does not require any control signalling between nodes. Furthermore, we observe that by attempting to maintain the battery at a target level, the interval (epoch) of running the QuickFix algorithm can be extended, leading to reduced control overhead. Another approach with a similar motivation of utilizing energy efficiently with replenishing batteries is given in [24]. There, each node manages energy to keep its duty cycle period as smooth as possible and at the same time tries to keep the battery state close to a certain desired level. Although we do not consider duty-cycling in this work, SnapIt enables sensor nodes to achieve perpetual operation.

If we denote the total size of the battery of node $i$ as $M_i$, SnapIt uses the mid point, i.e., $M_i/2$ as the target battery state. Each node takes into account the instantaneous energy state, $B_i(t)$, of its battery and makes slight variations on the rate allocation in order to keep the drift toward $M_i/2$. These variations are small enough to guarantee a total utility close to the optimal.

The optimal rate assignment for node $i$ in epoch $e$ is $r_i^*(e) = r_i^*(\pi_i(e))$, provided by QuickFix by solving **P**. Since the solution to **P** depends on $\pi_i(e)$, and not the state of the battery, QuickFix is the battery-state oblivious static assignment $r_i^*(e)$ for all times $t$ during epoch $e$. QuickFix is inclined to choose the rates such that the energy conservation constraint (3) at a node is kept active if possible. Consequently, energy is drained at a rate, identical to the average replenishment rate possibly over multiple successive epochs. This leads to a high rate of battery discharge and hence a low network utility.

SnapIt chooses the rate (and hence the transmission power), independently at each node $i$ based on the current state of the battery as follows: For $t,\ (e-1)\tau + 1 \le t \le e\tau$,

$$r_i^{\text{SnapIt}}(t) = \begin{cases} r_i^*(e) - \delta_i, & B_i(t) \le M_i/2 \\ r_i^*(e) + \delta_i, & B_i(t) > M_i/2 \end{cases}, \qquad (11)$$

for some $\delta_i > 0$, which we will specify later on. Consequently, the transmit cost (power) reduces by $\delta_i \lambda_i^{(sn)}$ if the battery is less than half full and increases by the same amount when it is more than half full. Each node can run SnapIt based solely on the local battery state $B_i(t)$ to make rate assignments. In general, one may choose to update the rate assignments at longer intervals $\tau_S > 1$, where $\tau_S \ll \tau$. The algorithm is detailed as follows:

**Algorithm 2**: SnapIt: Localized Energy Management

**foreach** $\tau_S$ *time units* **do**
  Check out the battery state $B_i(t)$
  **if** $B_i(t) \le M_i/2$ **then**
    $r_i^{\text{SnapIt}}(t) \leftarrow r_i^*(e) - \delta_i$
  **else**
    $r_i^{\text{SnapIt}}(t) \leftarrow r_i^*(e) + \delta_i$
  **end**
**end**

As shown in Fig. 2, the instantaneous utility associated with SnapIt alternates between $u_u(e) = \log(r_i^*(e) + \delta_i)$ and $u_d(e) = \log(r_i^*(e) - \delta_i)$, depending on the battery state. Due to concavity of the log function, the average utility at epoch $e$

will be lower (by Jensen's inequality) than the optimal value $\log(r_i^*(e))$, which can only be achieved if $M_i = \infty$. The smaller the value we select for $\delta_i$, the closer the average utility of SnapIt gets to $\log(r_i^*(e))$. However, a small $\delta_i$ implies a small drift away from the complete discharge state, and hence a higher likelihood of complete discharge. The important question is how to choose $\delta_i$ such that, not only the average utility approaches to the optimal value, but also the complete discharge rate decays to 0 sufficiently fast. Next, we show that this is possible under rather weak assumptions on the instantaneous replenishment rate $\rho_i(t)$.

In particular, we assume that the asymptotic semi-invariant log moment generating function

$$\Lambda_{\rho_i}(s) = \lim_{T\to\infty} \frac{1}{T} \log \mathrm{E}\left[\exp\left(s \sum_{t=1}^{\infty} \rho_i(t)\right)\right] \qquad (12)$$

of $\rho_i(t)$ exists and is finite for all $s \in (-\infty, s_{\max})$ for some $s_{\max} > 0$. Note that this existence requires an exponential (or faster) decay for the tail of the sample pdf of $\rho_i(t)$ and it rules out the possibility of long range dependencies in the $\{\rho_i(t)\}$ process. We also assume that the estimate, $\pi_i(e)$ used for the average replenishment rate for each epoch $e$ is unbiased, i.e., $\pi_i(e) = \frac{1}{\tau}\mathrm{E}\left[\sum_{t=(e-1)\tau+1}^{e\tau} \rho_i(t)\right]$ for all $e \geq 1$.

In presenting our result, we use the following notation: $a_n = \mathrm{O}(b_n)$ if $a_n$ goes to 0 at least as fast as $b_n$, $a_n = \mathrm{o}(b_n)$ if $a_n$ goes to 0 strictly faster than $b_n$, and $a_n = \Theta(b_n)$ if $a_n$ and $b_n$ go to 0 at the same rate. Also $p_i^{\text{SnapIt}}(M_i)$ is the probability of complete battery discharge for node $i$ as a function of the size, $M_i$, of its battery, $\bar{U}_i^{\text{SnapIt}} = \mathrm{E}\left[U(r_i^{\text{SnapIt}}(t))\right]$ and $\bar{U}_i^* = \mathrm{E}\left[U(r_i^*(e))\right]$ are the time average utilities achieved by SnapIt and the optimal rate allocation with an unlimited battery size respectively.

**Theorem V.1.** *If the variance, $\sigma^2_{\tilde{\rho}_i} \triangleq var\left(\frac{1}{\tau}\sum_{t=1}^{\tau}\rho_i(t)\right)$ is bounded and the utility function is the log function, $U(\cdot) = \log(\cdot)$, then, given any $\beta \geq 1$, SnapIt achieves $p_i^{SnapIt}(M_i) = O(M_i^{-\beta})$ and $\bar{U}_i^* - \bar{U}_i^{SnapIt} = \Theta\left(\frac{\log M_i}{M_i}\right)$ with the choice of $\delta_i = \frac{\beta\sigma^2_{\tilde{\rho}_i}\log M_i}{\lambda_i^{(sn)} M_i}$.*

*Proof:* See Appendix A. ∎

This theorem shows that it is possible to have a quadratic decay for the probability of complete battery discharge, and at the same time achieve a utility that approaches the optimal value (that of an unlimited energy source) approximately[1] as $1/M_i$. To understand the strength of this SnapIt, note that there exists no scheme that achieves (even asymptotically) the optimal utility with an exponential decay for the probability of complete discharge. Very briefly, the proof has the following sketch. By choosing $\delta_i = \kappa\frac{\log M_i}{M_i}$, we show for any choice of $\kappa > 0$, the desired scaling for the utility function is achieved. Then, by choosing $\kappa = \beta\sigma^2_{\tilde{\rho}_i}/\lambda_i^{(sn)}$, we prove that we can achieve the desired quadratic decay for the probability of complete discharge.

An extensive performance analysis of SnapIt is given in Section VI along with some comparisons to the static scheme that assigns a rate, fixed at the optimal value $r_i^*(e)$ during the entire epoch $e$. As we shall illustrate, in many scenarios, the dynamic scheme significantly reduces the battery discharge rate, and consequently increases the overall utility considerably.

[1]Note that the scaling laws given in the theorem are asymptotic in the battery size $M_i$ for fixed values of $\beta$.

## VI. Evaluation

In this section, we evaluate QuickFix/SnapIt and compare it with IFRC [14] using TOSSIM 2.0 [7]. The parameters used in the simulations are listed in Table II. We build the recharging profiles of the nodes using the real solar radiation measurements collected from the Baseline Measurement System at the National Renewable Energy Laboratory [25]. The data set used is Global 40-South Licor, which measures the solar resource for collectors tilted 40 degrees from horizontal and optimized for year-round performance. Unless explicitly specified, we use the profile of a sunny day (Feb. 1st 2009). The data is appropriately scaled to create a recharging profile for a solar panel with a small dimension ($37mm \times 37mm$). The battery capacity of sensor nodes is assumed to be $2100mAh$. Throughout the evaluation, we focus on the performance measures during the daytime because the energy harvesting rate is zero at night. However, based on the application's minimum sampling rate requirement, one can determine the minimum battery level that can support the minimum sampling rate at night and the SnapIt algorithm will maintain the battery at that level to ensure the network remains active during the night time. It should also be noted that although we did not consider the energy cost for signaling in our formulation, we did take that into account in the simulations. In the remaining section we compare our algorithms with the instantaneous optimum computed using MATLAB in each epoch; contrast it with a backpressure-based algorithm (IFRC); and, evaluate the sensitivity of the results with respect to the parameter $\delta$.

### A. QuickFix, SnapIt and Instantaneous Optimum

We first demonstrate the operation of SnapIt using a small 6-node network which has three levels. Node 0, at the first level, is the sink. Node 1 and node 2 are at the second level and they are the immediate children nodes of the sink. Nodes 3, 4 and 5 are at the third level. Nodes 3 and 5 have only one parent (nodes 1 and 2, respectively) and node 4 has two parents (both node 1 and node 2). In this set of simulations, we use the same recharging profile for all the sensor nodes. The performance metric include the network utility, the sum of data rates, the cumulative downtime of sensor nodes, and the cumulative battery full time. The network utility and the sum of data rates are computed based on the packet reception rates at the sink. The sum of data rates is used as one of the metrics because the utility function is in the logarithmic scale and has a small slope. The latter metric makes it easier

to visualize the difference in performance between different solutions, especially at higher data rates.

It can be seen in Figures 3(a), 3(b), 4(a) and 4(b) that the network utility and the sum rates observed at the sink are close to the optimum no matter whether SnapIt is used or not (the optimum is computed using Matlab without considering the battery levels of the sensor nodes). However, if the battery level is high (above the target level), SnapIt will exploit the excessive energy in the battery and increase the rates by $\delta$. This benefit is especially observable during 6-8 AM in Figure 4(a).

In Figure 3(c), we observe the cumulative downtime of nodes 1 and 2 when the initial battery levels of all the sensor nodes are at a very low level ($0.15\%$ of the full capacity). We only observe nodes 1 and 2 because these are the only potential bottleneck nodes in the network. Without using SnapIt, the cumulative downtime for both nodes are high. This is due to the fact that the QuickFix algorithm only runs coarsely (one iteration every $5$ minutes), and thus its computed rates can be inaccurate and even infeasible. The SnapIt algorithm mitigates this problem by reducing the rates when the battery level is below the target level. Therefore, the cumulative downtime for both nodes are zero (thus invisible in Figure 3(c)) when SnapIt is used.

In contrast to Figure 3(c), we observe the cumulative battery full time when the initial battery levels of all the sensor nodes are at a very high level ($99\%$ of the full capacity). It can be observed that, without using SnapIt, the batteries of nodes 1 and 2 spend more time in the full state. This causes nodes 1 and 2 to miss the opportunities to harvest more energy. In contrast, if SnapIt is used, both nodes 1 and 2 spend less time in full battery state, and the additional harvested energy is leveraged to increase the network utility.

TABLE II
PARAMETERS

| Parameter | $\lambda_i^{(sn)}$ | $\lambda_{ij}^{(tx)}$ | $\lambda_i^{(rx)}$ | $\alpha_1$ & $\alpha_2$ | $\delta$ |
|---|---|---|---|---|---|
| Value | $105\mu W$ | $63mW$ | $69mW$ | $0.001$ | $0.1 \times r_i$ |

### B. QuickFix/SnapIt v.s. Backpressure-based Protocol

Next, we compare our protocol with a backpressure-based protocol, IFRC [14], which aimes to achieve maxmin fairness in WSNs [16][14]. IFRC uses explicit signaling embedded in every packet to share a node's congestion state with the neighbors. Rate adaptation is done by using AIMD. Several queue thresholds are defined in IFRC. A node will reduce its rate more aggressively as a higher queue threshold is reached. Since IFRC does not consider battery state and energy replenishment, we similarly defined several thresholds for battery levels and energy harvesting rates so that all the nodes can maintain the battery at half of the full capacity. We use a tree instead of a DAG when performing the comparison as IFRC assumes a tree network. The tree is constructed using $67$ nodes based on Motelab's [26] topology. We used the recharging profiles of a sunny day (Feb. 1st 2009) as well as that of a cloudy day (Feb. 2nd 2009) for the evaluation. The initial battery level for all nodes is set to $50\%$ of the full capacity. Figure 5 clearly shows that QuickFix can achieve both higher network utility and sum of the data rates. The sum of rates is $42\%$ higher than IFRC on average. The main reason is as follows. In order to maintain the battery at $50\%$, IFRC halves the rate of a sensor and that of all its descendants if the battery drops below $50\%$. In contrast, SnapIt slightly reduces the rate by a small amount, $\delta$. It is possible to improve the performance of IFRC, but a smarter rate control algorithm is needed.

### C. Effect of different $\delta$

Larger $\delta$ can result in a higher network utility when there is extra energy in the battery. However, large $\delta$ has a negative impact on the battery levels as it can cause a node and its ancestors to consume the energy at a higher rate. We manually select three nodes and observe their battery levels over time. The three selected nodes $A$, $B$, and $C$ are 1-hop, 3-hops, 6-hops away from the sink respectively. And nodes $A$ and $B$ are on a path from node $C$ to the sink, i.e. both node $A$ and $B$ are ancestors of node $C$. Figure 6 shows that the performance of our solution is not very sensitive to the exact value of $\delta$ if $\delta$ is small. However, high values of $\delta$ ($= r_i$) should be avoided due to the consequent high fluctuations in the battery that also increases the chances of a node to run out of battery.

## VII. Conclusions and Future work

Achieving proportional fairness in energy harvesting sensor networks is a challenging task as the energy replenishment rate varies over time. In this paper, we showed that our proposed QuickFix algorithm can be applied to track the instantaneous optimum in such a dynamic environment and the SnapIt algorithm successfully maintains the battery at the desired target level. Our evaluations show that the two algorithms, when working together, can increase the total data rate at the sink by $42\%$ on average when compared to IFRC, while simultaneously improving the network utility. As part of the future work, we plan to generalize our solution to arbitrary graphs rather than a DAG. We will also explore dynamic adaptation of our parameter $\delta$ for faster operation, while continuing to avoid battery overflows and underflows.

## References

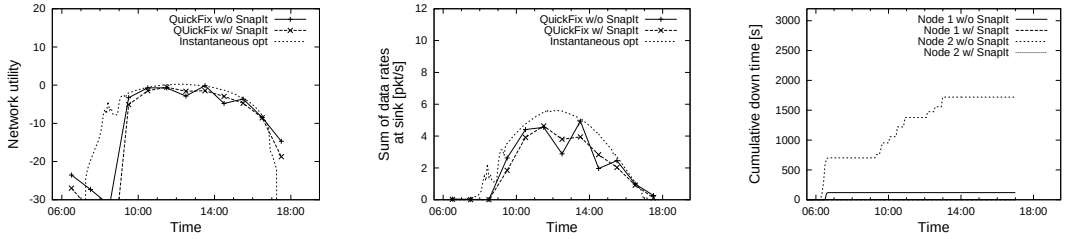

[1] C. Park and P. Chou, "AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes," in *Proc. of SECON*, Sept. 2006, pp. 168–177.

[2] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," in *Proc. of IPSN*, April 2005, pp. 463–468.

[3] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, "Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments," in *Proc. of IPSN*, April 2006, pp. 407–415.

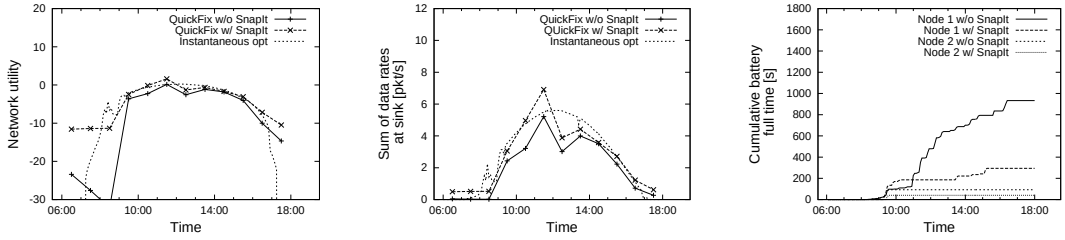
[4] K. Lin, J. Yu, J. Hsu, S. Zahedi, D. Lee, J. Friedman, A. Kansal, V. Raghunathan, and M. Srivastava, "Heliomote: Enabling Long-lived Sensor Networks Through Solar Energy Harvesting," in *Proc. of SenSys*, 2005, pp. 309–309.

[5] X. Wang and K. Kar, "Cross-layer Rate Control for End-to-end Proportional Fairness in Wireless Networks with Random Access," in *Proc. of MobiHoc*, 2005, pp. 157–168.



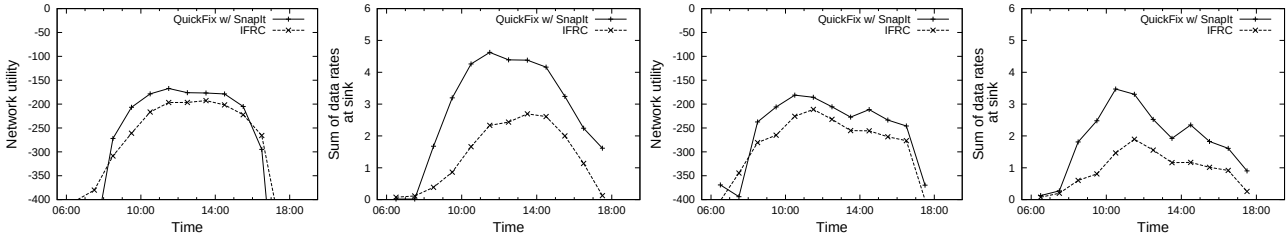

(a) Network utilityuy (b) Sum of the data rates (initial battery level = $0.15\%$) (c) Accumulative down time. With SnapIt both nodes eliminate their downtime (corresponding curves coincide with x-axis).

Fig. 3. QuickFix and SnapIt vs. Instantaneous Optimum (initial battery = $0.15\%$). Our algorithms attain similar utility and sum of rates as optimum while significantly reducing the battery downtime.



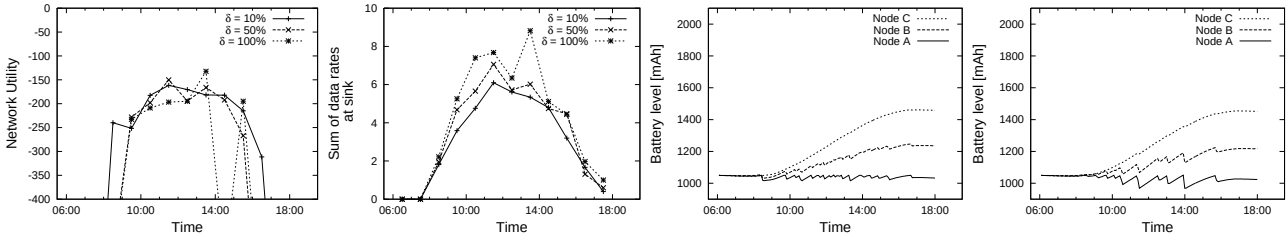

(a) Network utilityuy (b) Sum of the data rates (c) Accumulative battery full time

Fig. 4. QuickFix and SnapIt vs. Instantaneous Optimum (initial battery = $99\%$). Our algorithms attain similar utility and sum of rates as optimum while significantly reducing the time for which battery is full.




(a) Network utility (sunny day) (b) Sum of the data rates (sunny day) (c) Network utility (cloudy day) (d) Sum of the data rates (cloudy day)

Fig. 5. QuickFix vs. IFRC (67-node network, initial battery = 50%). Recharging profile of nodes are obtained by varying a base profile by an amount randomly selected in the [-5%,5%] region.




(a) Network utility using $10\%$, $50\%$ and $100\%$ of nodal sampling rate as $\delta$. (b) Sum of the data rates (c) Battery level ($\delta = 0.1 \times r_i$) (d) Battery level ($\delta = r_i$)

Fig. 6. A comparison of network utility (a), sum rates (b) and battery levels (c)(d) when $\delta = 0.1 \times r_i$, $\delta = 0.5 \times r_i$, and $\delta = r_i$ (network size is 67 nodes). High values of $\delta$ can cause rapid fluctuations in battery levels leading to increased downtime.

[6] M. C. Lijun Chen, Steven H. Low and J. C. Doyle, "Cross-Layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks," in *Proc. of Infocom*, 2006, pp. 1–13.

[7] "TinyOS-2.x," Website, http://www.tinyos.net/.

[8] T. Voigt, H. Ritter, and J. Schiller, "Utilizing Solar Power in Wireless Sensor Networks," in *Proc. of LCN*, 2003, p. 416.

[9] L. Lin, N. B. Shroff, and R. Srikant, "Asymptotically Optimal Energy-aware Routing for Multihop Wireless Networks with Renewable Energy Sources," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1021–1034, 2007.

[10] K.-W. Fan, Z.-Z. Zheng, and P. Sinha, "Steady and Fair Rate Allocation for Rechargeable Sensors in Perpetual Sensor Networks," in *Proc. of SenSys*, 2008.

[11] S. Chen and Y. Xia, "Lexicographic Maxmin Fairness for Data Collection in Wireless Sensor Networks," *IEEE Trans. on Mobile Computing*, vol. 6, no. 7, pp. 762–776, 2007, senior Member-Yuguang Fang.

[12] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks," in *Proc. of SenSys*, 2003, pp. 266–279.

[13] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating Congestion in Wireless Sensor Networks," in *Proc. of SenSys*, 2004, pp. 134–147.

[14] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis, "Interference-aware Fair Rate Control in Wireless Sensor Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 63–74, 2006.

[15] A. Sridharan and B. Krishnamachari, "Explicit and Precise Rate Control in Wireless Sensor Networks," in *Proc. of SenSys*, 2009.

[16] A. Sridhara, S. Moeller, B. Krishnamachari, and M. Hsieh, "Implementing Backpressure-based Rate Control in Wireless Networks," in *Proc. of ITA*, 2009.

[17] A. Karnik, A. Kumar, and V. Borkar, "Distributed Self-tuning of Sensor Networks," *Wireless Network*, vol. 12, no. 5, pp. 531–544, 2006.

[18] F. Kelly, "Charging and Rate Control for Elastic Traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.

[19] D. P. Palomar and M. Chiang, "A Tutorial on Decomposition Methods for Network Utility Maximization," *IEEE Journal on Selected Areas in Comm.*, vol. 24, no. 8, pp. 1439–1451, 2006.

[20] ——, "Alternative Decompositions for Distributed Maximization of Network Utility: Framework and Applications," in *Proc. of Infocom*, 2006, pp. 1–13.

[21] D. P. Bertsekas, *Nonlinear Programming.* Athena Scientific, 1995.

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[23] U. Akyol, M. Andrews, P. Gupta, J. Hobby, I. Saniee, and A. Stolyar, "Joint Scheduling and Congestion Control in Mobile Ad-hoc Networks," in *Proc. of Infocom*, 2008.

[24] C. Vigorito, D. Ganesan, and A. Barto, "Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks," *Proc. of SECON*, pp. 21–30, June 2007.

[25] "National Renewable Energy Laboratory," Website, http://www.nrel.gov.

[26] "MOTELAB Testbed," Website, http://motelab.eecs.harvard.edu/.

[27] R. Gallager, *Discrete Stochastic Processes*, 1st ed. Kluwer Academic Publishers, 1996.


## APPENDIX

Recall that rate allocation with SnapIt is described as follows: for $t,\ (e-1)\tau_Q+1 \le t \le e\tau_Q$,

$$r_i^{\text{SnapIt}}(t) = \begin{cases} r_i^*(e)-\delta_i, & B_i(t) \le M_i/2 \\ r_i^*(e)+\delta_i, & B_i(t) > M_i/2 \end{cases},$$

for a certain $\delta_i > 0$ and $r_i^*(e)$ is the optimal sampling rate for epoch $e$.

Let us define the processes $\tilde{\rho}_i(e) \triangleq \frac{1}{\tau_Q}\sum_{t=(e-1)\tau_Q+1}^{e\tau_Q}[\rho_i(t)-\pi_i(e)]$ and $\tilde{D}_i^-(e) \triangleq \tilde{\rho}_i(e)+\delta_i\lambda_i^{(sn)}$. Note that $D_i^-(e)$ represents the average drift of the battery of node $i$ in epoch $e$ in which $B_i(t) < M_i/2$ for all $t$ and the energy conservation constraint (3) is active. This implies that $r_i^*(e) = \lambda_i^{(sn)}\pi_i(e)$, i.e., the battery is drained at the rate it is being replenished in epoch $e$. We can write the semi-invariant asymptotic log moment generating function of process $\{\tilde{\rho}_i(e)\}$ as:

$$\Lambda_{\tilde{\rho}_i}(s) = \lim_{\tau_A\to\infty}\frac{1}{\tau_A}\log \mathrm{E}\left[\exp\left(s\sum_{e=1}^{\tau_A}\tilde{\rho}_i(e)\right)\right].$$

Likewise for process $\{D_i^-(e)\}$, the same function can e written as $\Lambda_{D_i^-}(s) = \Lambda_{\tilde{\rho}_i}(s) + s\delta_i\lambda_i^{(sn)}$.

Let $s_i^o$ be the unique negative root of $\Lambda_{D_i^-}(s)$. Note that, as $M_i \to \infty$ (i.e., $\delta_i \to 0$), $s_i^o$ goes to 0. To prove the theorem, we first prove the following lemma.

**Lemma A.1.** *The variance of* $\tilde{\rho}_i(t),\ \sigma_{\tilde{\rho}_i}^2 = \Lambda_{\tilde{\rho}_i}''(0)$ *satisfies*

$$\left.\frac{ds_i^o}{d\delta_i}\right|_{\delta_i=0} = \frac{-2\lambda_i^{(sn)}}{\sigma_{\tilde{\rho}_i}^2}. \tag{13}$$

*Proof:* Let $\Lambda_{D_i^-}^{(n)}(0) = \left.\frac{d^n\Lambda_{D_i^-}(s)}{ds^n}\right|_{s=0}$. The expansion of $\Lambda_{D_i^-}(s)$ around $s=0$ leads to

$$\begin{aligned} 0 = \Lambda_{D_i^-}(s_i^o) &= \sum_{n=1}^{\infty}\Lambda_{D_i^-}^{(n)}(0)\frac{(s_i^o)^n}{n!} \\ &= \delta_i\lambda_i^{(sn)}s_i^o + \sum_{n=1}^{\infty}\Lambda_{\tilde{\rho}_i}^{(n)}(0)\frac{(s_i^o)^n}{n!}. \end{aligned} \tag{14}$$

Since we assume that the replenishment rate estimator $\pi_i(e)$ is unbiased, $\mathrm{E}[\tilde{\rho}_i(e)] = 0$ for all $e \ge 1$ and (14) reduces to:

$$\sum_{n=2}^{\infty}\Lambda_{\tilde{\rho}_i}^{(n)}(0)\frac{(s_i^o)^{n-1}}{n!} = -\delta_i\lambda_i^{(sn)}. \tag{15}$$

Differentiating both sides with respect to $\delta_i$, (15) becomes

$$\frac{ds_i^o}{d\delta_i}\sum_{n=2}^{\infty}\Lambda_{\tilde{\rho}_i}^{(n)}(0)\frac{(n-1)(s_i^o)^{n-2}}{n!} = -\lambda_i^{(sn)}.$$

For $\delta_i = 0,\ s_i^o = 0$ and the above equality becomes

$$\left.\frac{ds_i^o}{d\delta_i}\right|_{\delta_i=0} = -\frac{2\lambda_i^{(sn)}}{\sigma_{\tilde{\rho}_i}^2} \quad\Rightarrow\quad s_i^o = -\frac{2\lambda_i^{(sn)}}{\sigma_{\tilde{\rho}_i}^2}\delta_i + \mathrm{o}(\delta_i).$$

■

**Proof of Theorem V.1:** Here we prove that our scheme satisfies the scaling properties given in Theorem V.1. By choosing $\delta_i = \kappa\frac{\log M_i}{M_i}$, we first show that the asymptotic scaling of $p_i^{\text{SnapIt}}(M_i)$ with $M_i$ is of the form $p_i^{\text{SnapIt}}(M_i) = \mathrm{O}(M_i^{-\beta})$. For node $i$, let us consider the worst case scenario, in which rate $r_i^{\text{SnapIt}}(t)$ is such that the energy is consumed at the rate it is being replenished, i.e., energy conservation constraint (3) is active at all times. The probability of complete discharge in this scenario constitutes an upper bound for the actual probability of complete discharge. The drift $D_i^-(e)$ for this scenario is treated in the above lemma. Applying Wald's identity [27] for $D_i^-(e)$, we can write:

$$p_i^{\text{SnapIt}}(M_i) = \mathrm{O}\left(\exp\left(s_i^o\frac{M_i}{2}\right)\right).$$

By Lemma A.1, we have

$$s_i^o = -\frac{2\lambda_i^{(sn)}}{\sigma^2_{\tilde{\rho}_i}}\delta_i + \mathrm{o}(\delta_i) = -\frac{2\lambda_i^{(sn)}}{\sigma^2_{\tilde{\rho}_i}}\frac{\kappa \log M_i}{M_i} + \mathrm{o}\left(\frac{\log M_i}{M_i}\right).$$

Choosing $\kappa = \beta\sigma^2_{\tilde{\rho}_i}/\lambda_i^{(sn)}$, we prove the desired result $p_i^{\text{SnapIt}}(M_i) = \mathrm{O}(M_i^{-\beta})$.

Next we show that for any choice of $\kappa$, our scheme achieves an average utility $\bar{U}_i^{\text{SnapIt}}$ such that $\bar{U}_i^* - \bar{U}_i^{\text{SnapIt}} = \Theta\left(\frac{\log M_i}{M_i}\right)$. Let us focus on a single epoch $e$ initially. First, note that, instantaneous utility $U(r_i^{\text{SnapIt}}(t)) = 0$ if $B_i(t) = 0$ (i.e., with probability $\mathrm{O}(M_i^{-\beta})$), since $r_i^{\text{SnapIt}}(t) = 0$. Otherwise, as illustrated in Fig. 2, the utility alternates between $u_u \triangleq u_u(e)$ and $u_d \triangleq u_d(e)$ during epoch $e$. With a first order Taylor series expansion of the utility function $U(r_i^{\text{SnapIt}}(t)) = \log(r_i^{\text{SnapIt}}(t))$ around $r_i^* \triangleq r_i^*(e)$, we have

$$u_u = \log(r_i^*) + \frac{\delta_i}{r_i^*} + \mathrm{o}(\delta_i)$$

and

$$u_d = \log(r_i^*) - \frac{\delta_i}{r_i^*} + \mathrm{o}(\delta_i).$$

Let $\gamma_i^- \triangleq \gamma_i^-(e)$ be the fraction of time that battery state $B_i(t) < M_i/2$. Then,

$$\begin{aligned}\bar{U}_i^{\text{SnapIt}} &= \mathrm{E}\left[\gamma_i^- u_d + (1-\gamma_i^-)u_u\right]\left(1 - p_i^{\text{SnapIt}}(M_i)\right) \\ &= \log(r_i^*) + \mathrm{E}\left[1 - 2\gamma_i^-\right]\frac{1}{r_i^*}\left(\frac{\kappa \log M_i}{M_i}\right) + \mathrm{o}\left(\frac{\log M_i}{M_i}\right), \qquad (16)\end{aligned}$$

where (16) follows since $\kappa$ is chosen such that $\beta \geq 1$. Thus, since $r_i^* > 0$ for all $i$ and $\bar{U}_i^* = \mathrm{E}\left[\log(r_i^*(e))\right]$, we can write $\bar{U}_i^* - \bar{U}_i^{\text{SnapIt}} = \Theta\left(\frac{\log M_i}{M_i}\right)$.

### A. Proof of Proposition 1

*Proof:*

$$\begin{aligned}
P_i &= \sum_{j\in A_i}(\lambda_r\mu_j + \upsilon_j)z_{ij}(w) \\
&= \sum_{k\in P_i}(\lambda_r\mu_k + \upsilon_k)z_{ik}(w) + \sum_{\ell\in A_i\backslash P_i}(\lambda_r\mu_\ell + \upsilon_\ell)z_{i\ell}(w) && (17)\\
&= \sum_{k\in P_i}(\lambda_r\mu_k + \upsilon_k)\Big(w_{ik} + \sum_{m\in P_i\backslash\{k\}} w_{im}z_{mk}(w)\Big) \\
&+ \sum_{\ell\in A_i\backslash P_i}(\lambda_r\mu_\ell + \upsilon_\ell)z_{i\ell}(w) && (18)\\
&= \sum_{k\in P_i} w_{ik}(\lambda_r\mu_k + \upsilon_k) \\
&+ \sum_{k\in P_i}(\lambda_r\mu_k + \upsilon_k)\Big(\sum_{m\in P_i\backslash\{k\}} w_{im}z_{mk}(w)\Big) \\
&+ \sum_{\ell\in A_i\backslash P_i}(\lambda_r\mu_\ell + \upsilon_\ell)z_{i\ell}(w) && (19)\\
&= \sum_{k\in P_i} w_{ik}(\lambda_r\mu_k + \upsilon_k) + \\
&+ \sum_{k\in P_i} w_{ik}\Big(\sum_{m\in P_i\backslash\{k\}}(\lambda_r\mu_m + \upsilon_m)z_{km}(w)\Big) \\
&+ \sum_{\ell\in A_i\backslash P_i}(\lambda_r\mu_\ell + \upsilon_\ell)z_{i\ell}(w) && (20)\\
&= \sum_{k\in P_i} w_{ik}(\lambda_r\mu_k + \upsilon_k) \\
&+ \sum_{k\in P_i} w_{ik}\Big(\sum_{m\in P_i\backslash\{k\}}(\lambda_r\mu_m + \upsilon_m)z_{km}(w)\Big) \\
&+ \sum_{\ell\in A_i\backslash P_i}(\lambda_r\mu_\ell + \upsilon_\ell)\Big(\sum_{k\in P_i} w_{ik}z_{k\ell}(w)\Big) && (21)\\
&= \sum_{k\in P_i} w_{ik}(\lambda_r\mu_k + \upsilon_k) \\
&+ \sum_{k\in P_i} w_{ik}\Big(\sum_{m\in P_i\backslash\{k\}}(\lambda_r\mu_m + \upsilon_m)z_{km}(w)\Big) \\
&+ \sum_{k\in P_i} w_{ik}\Big(\sum_{\ell\in A_i\backslash P_i}(\lambda_r\mu_\ell + \upsilon_\ell)z_{k\ell}(w)\Big) && (22)\\
&= \sum_{k\in P_i} w_{ik}\Big((\lambda_r\mu_k + \upsilon_k) \\
&+ \sum_{m\in P_i\backslash\{k\}}(\lambda_r\mu_m + \upsilon_m)z_{km}(w) \\
&+ \sum_{\ell\in A_i\backslash P_i}(\lambda_r\mu_\ell + \upsilon_\ell)z_{k\ell}(w)\Big) && (23)\\
&= \sum_{k\in P_i} w_{ik}\Big((\lambda_r\mu_k + \upsilon_k) + P_k\Big) && (24)\\
& && (25)
\end{aligned}$$

(17) is true because $A_i = P_i \cup A_i\backslash P_i$. (18) follows the definition of $z_{ik}$. (19) simply separates the inner sum from the outer sum. (20) exchanges the sum and applied change of variables. (21) expands $z_{i\ell}$ by using its definition. (23) groups the second and the third sum. Finally, (24) follows because $A_k \subseteq P_i\backslash\{k\} \cup A_i\backslash P_i$ and if a node $m \in P_i\backslash\{k\} \cup A_i\backslash P_i$ and $m \notin A_k$, $z_{km} = 0$. Proposition 2 can be similarly proved. ■