Fast-Convergent Learning-aided Control in Energy Harvesting Networks

Longbo Huang
longbohuang@tsinghua.edu
IIIS, Tsinghua University

***Abstract*— In this paper, we present a novel learning-aided energy management scheme (LEM) for multihop energy harvesting networks. Different from prior works on this problem, our algorithm explicitly incorporates information learning into system control via a step called *perturbed dual learning*. LEM does not require any statistical information of the system dynamics for implementation, and efficiently resolves the challenging energy outage problem. We show that LEM achieves the near-optimal $[O(\epsilon), O(\log(1/\epsilon)^2)]$ utility-delay tradeoff with an $O(1/\epsilon^{1-c/2})$ energy buffers ($c \in (0,1)$). More interestingly, LEM possesses a *convergence time* of $O(1/\epsilon^{1-c/2}+1/\epsilon^c)$, which is much faster than the $\Theta(1/\epsilon)$ time of pure queue-based techniques or the $\Theta(1/\epsilon^2)$ time of approaches that rely purely on learning the system statistics. This fast convergence property makes LEM more adaptive and efficient in resource allocation in dynamic environments. The design and analysis of LEM demonstrate how system control algorithms can be augmented by learning and what the benefits are. The methodology and algorithm can also be applied to similar problems, e.g., processing networks, where nodes require nonzero amount of contents to support their actions.**

I. Introduction

Recent developments in energy harvesting technologies make it possible for wireless devices to support their functions by harvesting energy from the environment. For example, by using solar panels [1] [2], by harvesting ambient radio power [3], and by converting mechanical vibration into energy [4], [5]. Due to the capability in providing long lasting energy supply, the energy harvesting technology has the potential to become a promising solution to energy problems in networks formed by self-powered devices, e.g., wireless sensor networks and mobile devices.

To realize the full benefits of energy harvesting, algorithms must be designed to efficiently incorporate it into system control. In this paper, we develop an *online learning-aided energy management* scheme for energy harvesting networks. Specifically, we consider a discrete stochastic network, where network links have time-varying qualities, and nodes are powered by finite capacity energy storage devices and can harvest energy from the environment. In each time slot, every node decides how much new workload to admit, e.g., sampled data from a field, and how much power to spend for traffic transmission (or data processing). The objective of the network is to find a joint energy management and scheduling policy, so as to maximize the aggregate traffic utility, while ensuring network stability and energy availability, i.e., the network nodes always have enough energy to support transmissions.

There have been many previous works on energy harvesting networks. Works [6] and [7] consider a leaky-bucket like structure and design joint energy prediction and power management schemes for energy harvesting sensor nodes. [8] focuses on designing energy-efficient schemes that maximize the decay exponent of the queue size. [9] develops scheduling algorithms to achieve near-optimal utility for energy harvesting networks with time-varying channels. [10] designs an energy-aware routing scheme that achieves optimality as the network size increases. [11] proposes an online energy management and scheduling algorithm for multihop energy harvesting networks. [12] considers joint compression and transmission in energy harvesting networks. [13] considers a multihop network and proposes a control scheme based on energy replenishment rate estimation.

However, we notice that the aforementioned works either focus on scenarios where complete statistical information is given beforehand, or try to design schemes that do not require such information. Therefore, they ignore the potential benefits of utilizing information of system dynamics in control, and do not provide interfaces for integrating information collecting and learning techniques [14], e.g., sensing and data mining or machine learning, into algorithm design. In this work, we try to explicitly bring information learning into the system control framework. Specifically, we develop a learning mechanism called *perturbed dual learning* and propose a learning-aided energy management scheme (LEM).

LEM is an online control algorithm and *does not require any statistical information* for implementation. Instead, it builds an empirical distribution of the system dynamics, including network condition variation and energy availability fluctuation. Then, it learns an approximate optimal Lagrange multiplier of a carefully constructed underlying optimization problem that captures system optimality, via a step called *perturbed dual learning*. Finally, LEM incorporates the learned information into the system controller by augmenting the controller with the approximate multiplier. We show that LEM is able to achieve a near-optimal $[O(\epsilon), O(\log(\frac{1}{\epsilon})^2)]$ utility-delay tradeoff for general multihop energy harvesting networks with an $O((\frac{1}{\epsilon})^{2/3}\log(\frac{1}{\epsilon})^2)$ energy storage capacity and resolves the energy outage problem. Moreover, we show that by incorporating information learning, one can significantly improve the algorithm *convergence time*, i.e., the time an algorithm takes to converge to its optimal operating point: LEM requires an $O((\frac{1}{\epsilon})^{2/3}\log(\frac{1}{\epsilon})^2)$ time for convergence, whereas existing queue-based algorithms require a $\Theta(1/\epsilon)$ time and algorithms based purely on learning the statistics

require a $\Theta(1/\epsilon^2)$ time. This fast convergence implies that learning-aided algorithms can adapt faster when the environment statistics changes, which indicates better robustness and higher efficiency in resource allocation.

Learning-aided control with dual learning was first developed in [15]. In this work, we extend the results to resolve energy outage problems in energy harvesting networks via a perturbed version of dual learning. Intuitively speaking, perturbed dual learning learns a perturbed empirical optimal Lagrange multiplier required for "no-underflow" systems, where optimal multipliers must be steered and made trackable by queues.

Our paper is mostly related to recent works [6], [13], and [11]. Specifically, both [6] and [13] try to form estimations of the harvestable energy rates and utilize the information in network control. However, they do not consider the system dynamics and do not explicitly characterize network delay performance. On the other hand, [11] focuses on achieving long term performance guarantees without learning. Moreover, these three works do not characterize the algorithm convergence speed, which is an important metric for measuring the efficiency of control algorithms in learning the optimal system operating point in dynamic environments.

We summarize the main contributions as follows:

- We propose the Learning-aided Energy Management algorithm (LEM) for multihop energy harvesting networks, and show that LEM achieves a near-optimal $[O(\epsilon), O(\log(\frac{1}{\epsilon})^2)]$ utility-delay tradeoff with an $O((\frac{1}{\epsilon})^{2/3}\log(\frac{1}{\epsilon})^2)$ energy storage capacity.
- We show that LEM possesses an $O((\frac{1}{\epsilon})^{2/3}\log(\frac{1}{\epsilon})^2)$ convergence time. This convergence time is much faster compared to the $\Theta(\frac{1}{\epsilon})$ time of existing queue-based techniques and the $\Theta(\frac{1}{\epsilon})^2$ time required for approaches that purely rely on learning the statistics.
- We analyze the performance of LEM with the *augmented drift analysis* approach, which handles the interplay between learning and control and no-underflow constraints. This analysis approach can likely find applications to other similar problems with the no-underflow constraints, e.g., processing networks [16].

The rest of the paper is organized as follows. We present the system model in Section II. We explain the algorithm design approach and present the LEM algorithm in Section III and explain the intuition. Then, we present the performance results of LEM in Section IV. Simulation results are provided in Section V. We conclude the paper in Section VI.

II. The System Model

We consider a general multi-hop network that operates in slotted time. The network is modeled by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \{1, 2, ..., N\}$ is the set of nodes in the network, and $\mathcal{L} = \{[n, m],\ n, m \in \mathcal{N}\}$ is the set of communication links. We use $\mathcal{N}_n^{(o)}$ to denote the set of nodes b with $[n, b] \in \mathcal{L}$ for each node n, and use $\mathcal{N}_n^{(in)}$ to denote the set of nodes a with $[a, n] \in \mathcal{L}$. We define $d_{\max} \triangleq \max_n(|\mathcal{N}_n^{(in)}|, |\mathcal{N}_n^{(o)}|)$ the maximum in-degree/out-degree that any node n can have.

A. The Traffic and Utility Model

At every time slot, the network decides how much new workload (called packets below) destined for node c to admit at node n. We call this traffic the *commodity* c data and use $R_n^{(c)}(t)$ to denote the amount of new commodity c data admitted. We assume that $0 \leq R_n^{(c)}(t) \leq R_{\max}$ for all n, c with some finite $R_{\max} > 0$ at all time.

We assume that each commodity is associated with a utility function $U_n^{(c)}(\overline{r}^{nc})$, where $\overline{r}^{nc}$ is the time average rate of the commodity c traffic admitted into node n, defined as $\overline{r}^{nc} = \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{R_n^{(c)}(\tau)\}$.[1] Each $U_n^{(c)}(r)$ function is assumed to be increasing, continuously differentiable, and concave in r with a bounded first derivative and $U_n^{(c)}(0) = 0$. We define $\beta \triangleq \max_{n,c}(U_n^{(c)})'(0)$ the maximum first derivative of all utility functions.

B. The Transmission Model

In order to deliver the admitted data to their destinations, each node needs to allocate power to the links for transmission at every time slot. To model the effect that the transmission rates typically also depend on the link conditions and that the link conditions may be time-varying, we denote $\boldsymbol{S}(t)$ the network *channel state*, i.e., the N-by-N matrix where the (n, m) component of $\boldsymbol{S}(t)$ denotes the channel condition between nodes n and m.

Denote $P_{[n,b]}(t)$ the power allocated to link $[n, b]$ at time t. At every time slot, if $\boldsymbol{S}(t) = s_i$, the power allocation vector $\boldsymbol{P}(t) = (P_{[n,b]}(t), [n, b] \in \mathcal{L})$ must be chosen from some feasible power allocation set $\mathcal{P}^{(s_i)}$. We assume that $\mathcal{P}^{(s_i)}$ is compact for all s_i, and that every power vector in $\mathcal{P}^{(s_i)}$ satisfies the constraint that for each node n, $0 \leq \sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t) \leq P_{\max}$ for some finite $P_{\max} > 0$. We also assume that for any $\boldsymbol{P} \in \mathcal{P}^{(s_i)}$, setting the entry $P_{[n,b]}$ to zero yields another power vector that is still in $\mathcal{P}^{(s_i)}$. Given channel state $\boldsymbol{S}(t)$ and power allocation vector $\boldsymbol{P}(t)$, the transmission rate over link $[n, b]$ is given by the rate-power function $\mu_{[n,b]}(t) = \mu_{[n,b]}(\boldsymbol{S}(t), \boldsymbol{P}(t))$.

For each s_i, we assume that the function $\mu_{[n,b]}(s_i, \boldsymbol{P})$ satisfies the following properties: Let $\boldsymbol{P}, \boldsymbol{P'} \in \mathcal{P}^{(s_i)}$ be such that $\boldsymbol{P'}$ is obtained by changing any single component $P_{[n,b]}$ in $\boldsymbol{P}$ to zero. Then, (i) there exists some finite constant $\kappa > 0$ that:

$$\mu_{[n,b]}(s_i, \boldsymbol{P}) \leq \mu_{[n,b]}(s_i, \boldsymbol{P'}) + \kappa P_{[n,b]}, \tag{1}$$

and (ii) for each link $[a, m] \neq [n, b]$,

$$\mu_{[a,m]}(s_i, \boldsymbol{P}) \leq \mu_{[a,m]}(s_i, \boldsymbol{P'}). \tag{2}$$

These properties can be satisfied by most rate-power functions, e.g., when the rate function is differentiable and has finite directional derivatives with respect to power [17], and when link rates do not improve with increased interference.

We assume that there exists a finite constant $\mu_{\max}$ such that $\mu_{[n,b]}(t) \leq \mu_{\max}$ for all time under any power allocation vector $\boldsymbol{P}(t)$ and any channel state $\boldsymbol{S}(t)$. We use $\mu_{[n,b]}^{(c)}(t)$ to

[1] In this paper, we assume for clarity that all limits exist with probability 1. When some limits do not exist, we can obtain similar results by replacing limit by $\liminf$ or $\limsup$, but the results are more involved.

denote the rate allocated to the commodity c data over link $[n,b]$ at time t. It can be seen that $\sum_c \mu^{(c)}_{[n,b]}(t) \leq \mu_{[n,b]}(t)$ for all $[n,b]$ and for all t.

C. The Energy Harvesting Model

Each node in the network is assumed to be powered by a *finite* capacity energy storage device, e.g., a battery or an ultra-capacitor [18]. We model such a device with an *energy queue*. We use the energy queue size at node n at time t, denoted by $E_n(t)$, to measure the amount of the energy stored at node n at time t. Each node n can observe its current energy level $E_n(t)$. In any time slot t, the power allocation vector $\boldsymbol{P}(t)$ must satisfy the following "energy-availability" constraint:[2]

$$\sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t) \leq E_n(t), \quad \forall\, n, \tag{3}$$

i.e., the consumed power must be no more than what is available.

Each node in the network is assumed to be capable of harvesting energy from the environment, for instance, using solar panels [18] or mechanical vibration [5]. To capture the fact that the amount of harvestable energy typically varies over time, we use $h_n(t)$ to denote the amount of harvestable energy by node n at time t, and denote by $\boldsymbol{h}(t) = (h_1(t), ..., h_N(t))$ the harvestable energy vector at time t, called the *energy state*. We assume that $h_n(t) \leq h_{\max}$ for all n, t for some finite $h_{\max}$. In the following, it is convenient for us to assume that each node can decide whether or not to harvest energy in each slot. Specifically, we use $e_n(t) \in [0, h_n(t)]$ to denote the amount of energy that is actually harvested at time t. We will see later that under our algorithm, $e_n(t) \neq h_n(t)$ only when the energy storage is close to full.

Denote $\boldsymbol{z}(t) = (\boldsymbol{S}(t), \boldsymbol{h}(t))$. We assume that $\boldsymbol{z}(t)$ takes values in $\mathcal{Z} = \{\boldsymbol{z}_1, ..., \boldsymbol{z}_M\}$, where $\boldsymbol{z}_m = (\boldsymbol{s}_m, \boldsymbol{h}_m)$ and is i.i.d. every time. We denote $\pi_m = \Pr\{\boldsymbol{z}(t) = \boldsymbol{z}_m\}$. We also rewrite $\mathcal{P}^{(s_i)}$ as $\mathcal{P}^m$ and $\mu_{[n,b]}(s_m, \boldsymbol{P}) = \mu_{[n,b]}(\boldsymbol{z}_m, \boldsymbol{P})$. This allows arbitrary correlations among the harvestable energy processes and channels dynamics.[3]

D. Queueing Dynamics

Let $\boldsymbol{Q}(t) = (Q_n^{(c)}(t), n, c \in \mathcal{N})$, $t = 0, 1, 2, ...$ be the data queue backlog vector in the network, where $Q_n^{(c)}(t)$ is the amount of commodity c data queued at node n. We assume the following queueing dynamics:

$$\begin{aligned} Q_n^{(c)}(t+1) \quad \leq \quad & \big[Q_n^{(c)}(t) - \sum_{b\in\mathcal{N}_n^{(o)}} \mu^{(c)}_{[n,b]}(t)\big]^+ \\ & + \sum_{a\in\mathcal{N}_n^{(in)}} \mu^{(c)}_{[a,n]}(t) + R_n^{(c)}(t), \end{aligned} \tag{4}$$

with $Q_n^{(c)}(0) = 0$ for all $n, c \in \mathcal{N}$, $Q_c^{(c)}(t) = 0\ \forall\, t$, and $[x]^+ = \max[x, 0]$. The inequality in (4) is due to the fact that some nodes may not have enough commodity c packets to fill the allocated rates. In this paper, we say that the network is *stable* if the following condition is met:

$$\overline{\boldsymbol{Q}} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n,c} \mathbb{E}\big\{Q_n^{(c)}(\tau)\big\} < \infty. \tag{5}$$

Similarly, let $\boldsymbol{E}(t) = (E_n(t), n \in \mathcal{N})$ be the vector of energy queue sizes. Due to the energy availability constraint (3), for each node n, the energy queue $E_n(t)$ evolves according to the following:

$$E_n(t+1) = E_n(t) - \sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t) + e_n(t), \tag{6}$$

with $E_n(0) = 0$ for all n. Note that with (6), we start by assuming that each energy queue has infinite capacity. We will show later that under our algorithm, a finite buffer size is sufficient for achieving the desired perfromance.

E. Utility Maximization

The goal of the network is to design a joint flow control, routing and scheduling, and energy management algorithm to maximize the system utility, defined as:

$$U_{tot}(\overline{\boldsymbol{r}}) \quad = \quad \sum_{n,c} U_n^{(c)}(\overline{r}^{nc}), \tag{7}$$

subject to network stability (5) and energy availability (3). Here $\overline{\boldsymbol{r}} = (\overline{r}^{nc}, \forall\, n, c \in \mathcal{N})$ is the vector of the average expected admitted rates. We also use $\boldsymbol{r}^*$ to denote an optimal rate vector that maximizes (7) subject to (5) and (3).

F. Discussion of the Model

This model is general and can be used to model systems that are self-powered and can harvest energy, e.g., environment monitoring wireless sensor networks, or networks formed by mobile cellular devices. The same model was also considered in [11]. There, two online algorithms were developed for achieving near-optimal utility performance. In this work, we use a very different approach, which *explicitly incorporates learning* into algorithm design and *explores the benefits* of historic system information. Moreover, while previous works mostly focus on long term average performance, we also investigate the algorithm convergence time, defined to be the time it takes for the algorithm (and the system) to learn the optimal operating point.

III. Algorithm Design via Learning

In this section, we present our algorithm and the design approach. To facilitate understanding, we first discuss the intuition behind the approach. Then, we provide detailed descriptions of the algorithm.

A. Design Approach

We first consider the following optimization problem, which can be intuitively viewed as the solution to our

[2] We measure time in unit size slots, so that our power $P_{[n,b]}(t)$ has units of energy/slot, and $P_{[n,b]}(t) \times (1\,\text{slot})$ is the resulting energy consumption in one slot. Also, the energy harvested at time t is assumed to be available for use in time $t+1$.

[3] The i.i.d. assumption is made for ease of presentation. Our results can be extended to the case when $\boldsymbol{z}(t)$ evolves according to a general finite-state Markovian.

problem.[4]

$$\max: \quad \phi = V\sum_{n,c} U_n^{(c)}(r^{nc}) \tag{8}$$

$$\text{s.t.} \quad r^{nc} + \sum_m \pi_m \sum_{a\in\mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(\boldsymbol{z}_m, \boldsymbol{P}^m) \tag{9}$$
$$\leq \sum_m \pi_m \sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(\boldsymbol{z}_m, \boldsymbol{P}^m), \forall\ (n,c)$$

$$\sum_m \pi_m \sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}^m = \sum_m \pi_m e_n^m, \forall n \tag{10}$$

$$\boldsymbol{P}^m \in \mathcal{P}_m, \forall \boldsymbol{z}_m,\ 0 \leq r^{nc} \leq R_{\max}, \forall\,(n,c)$$

$$\sum_c \mu_{[n,b]}^{(c)}(\boldsymbol{z}_m, \boldsymbol{P}^m) \leq \mu_{[n,b]}(\boldsymbol{z}_m, \boldsymbol{P}^m),\ \forall\,[n,b]$$

$$0 \leq e_n^m \leq h_n^m,\ \forall\ n, \boldsymbol{h}_j.$$

Here $V \geq 1$ is a constant and corresponds to a control parameter of our algorithm (explained later). Intuitively, problem (8) computes an optimal control policy. To see this, note that we can interpret r^{nc} as the traffic admission rate, $\boldsymbol{P}^m$ as the power allocation vector under state $\boldsymbol{z}_m$, and $\boldsymbol{e}^m$ as the energy harvesting decision. (9) represents the queue stability constraint and (10) denotes the energy consumption constraint.

In practice, one may not always have the statistics (π_m, m) a-prior. As a result, online algorithms have been proposed, e.g., ESA in [11], [13]. However, doing so *ignores the historic system information* one can accumulate over time and loses its value. In our case, we try to explicitly utilize such information and to explore its benefits. Specifically, we first try to build an empirical distribution for the system dynamics $\{\boldsymbol{z}_m\}_{m=1}^M$. Then, we solve a *perturbed empirical* version of the dual problem of (8) to obtain an empirical Lagrange multiplier (called *perturbed dual learning*). After that, we incorporate the empirical multiplier into an online system controller (Fig. 1 shows its steps).

B. Learning-aided Energy Management

Here we present our algorithm, which consists of an online controller and a learning component. We first present the algorithm and then explain the controller in Section III-D.

For our algorithm, we need the dual problem of (8):

$$\min: \quad g(\boldsymbol{\upsilon}, \boldsymbol{\nu}), \quad \text{s.t. } \boldsymbol{\upsilon} \succeq \boldsymbol{0},\ \boldsymbol{\nu} \in \mathbb{R}^N, \tag{11}$$

where $\boldsymbol{\upsilon} = (\upsilon_n^{(c)}, \forall\,(n,c))$ and $\boldsymbol{\nu} = (\nu_n, \forall\, n)$ are the Lagrange multipliers, and $g(\boldsymbol{\upsilon}, \boldsymbol{\nu}) \triangleq \sum_m \pi_m g_m(\boldsymbol{\upsilon}, \boldsymbol{\nu})$ is the dual function with $g_m(\boldsymbol{\upsilon}, \boldsymbol{\nu})$ defined as:

$$g_m(\boldsymbol{\upsilon}, \boldsymbol{\nu}) = \sup\Bigg\{ V\sum_{n,c} U_n^{(c)}(r^{nc}) - \sum_n \upsilon_n^{(c)}\Big[r^{nc} + \sum_{a\in\mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(\boldsymbol{z}_m, \boldsymbol{P}^m) - \sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(\boldsymbol{z}_m, \boldsymbol{P}^m)\Big] - \sum_n \nu_n\Big[e_n^m - \sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}^m\Big] \Bigg\}. \tag{12}$$

[4]Technically speaking, one has to solve a "convexified" version of (8) to find an optimal control policy. But (8) is sufficient for our algorithm design and analysis.

Here the sup is taken over $\boldsymbol{r}$, $\boldsymbol{P}^m \in \mathcal{P}^m$, $\boldsymbol{\mu}$, and $\boldsymbol{e}^m \preceq \boldsymbol{h}^m$. In the following, we use $(\boldsymbol{\upsilon}^*, \boldsymbol{\nu}^*)$ to represent an optimal solution of $g(\boldsymbol{\upsilon}, \boldsymbol{\nu})$.

We now present the algorithm, which uses a control parameter $V \geq 1$ to tradeoff utility and delay, and specifies a learning time $T_L = V^c$ for some $c \in (0,1)$.

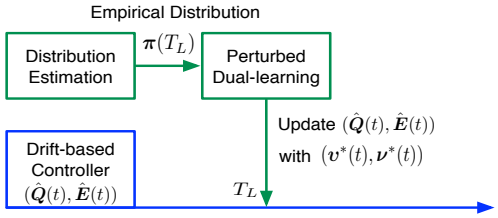

Fig. 1. There are three main components in LEM: (i) Build an empirical distribution $\boldsymbol{\pi}(t)$ for $\boldsymbol{z}(t)$. (ii) Perform perturbed dual learning and obtain the empirical optimal multiplier at time T_L. (iii) Incorporate the multiplier into the controller.

Learning-aided Energy Management (LEM): Initialize $\boldsymbol{\xi}_Q = \boldsymbol{0}$, $\boldsymbol{\xi}_E = \boldsymbol{0}$, and set $T_L = V^c$ with $c \in (0,1)$. At every time t, observe $\boldsymbol{Q}(t)$, $\boldsymbol{E}(t)$, $\boldsymbol{z}(t)$, and define the following *augmented queue vectors*:

$$\hat{\boldsymbol{Q}}(t) = \boldsymbol{Q}(t) + \boldsymbol{\xi}_Q, \quad \hat{\boldsymbol{E}}(t) = \boldsymbol{E}(t) + \boldsymbol{\xi}_E. \tag{13}$$

Then, do:

- **Energy harvesting:** If $\hat{E}_n(t) - \theta_n < 0$, harvest energy, i.e., set $e_n(t) = h_n(t)$. Else set $e_n(t) = 0$.
- **Data admission:** For each n, choose $R_n^{(c)}(t)$ by solving the following optimization problem:

$$\max:\ VU_n^{(c)}(r) - \hat{Q}_n^{(c)}(t)r,\ \ s.t.\ 0 \leq r \leq R_{\max}. \tag{14}$$

- **Power allocation:** Define the weight of commodity c data over link $[n,b]$ as:

$$W_{[n,b]}^{(c)}(t) \triangleq \big[\hat{Q}_n^{(c)}(t) - \hat{Q}_b^{(c)}(t)\big]^+. \tag{15}$$

Then, define the link weight $W_{[n,b]}(t) = \max_c W_{[n,b]}^{(c)}(t)$, and choose $\boldsymbol{P}(t) \in \mathcal{P}^{(\boldsymbol{z}(t))}$ to maximize:

$$G(\boldsymbol{P}(t)) \triangleq \sum_n \Bigg[\sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}(t) W_{[n,b]}(t) + (\hat{E}_n(t) - \theta_n) \sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t) \Bigg]. \tag{16}$$

- **Routing and scheduling:** For every node n, find any $c^* \in \text{argmax}_c W_{[n,b]}^{(c)}(t)$. If $W_{[n,b]}^{(c^*)}(t) > 0$, set:

$$\mu_{[n,b]}^{(c^*)}(t) = \mu_{[n,b]}(t),\ \mu_{[n,b]}^{(c)}(t) = 0,\ \forall\, c \neq c^*. \tag{17}$$

That is, allocate the full rate over link $[n,b]$ to any commodity that achieves the maximum positive weight over the link. Use idle-fill if needed.
- **Queue update and packet dropping:** Use Last-In-First-Out (LIFO) for packet selection. If for any node n, the resulting $\{P_{[n,b]}(t), m\}$ in (16) violates constraint (3), set $\sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t) = E_n(t)$ and drop all the packets that are supposed to be transmitted. Update $Q_n^{(c)}(t)$ and $E_n(t)$ according to (4) and (6), respectively.
- **Perturbed Dual-learning at T_L:** Let N_m be the number of times states $\boldsymbol{z}_m$ appear in $\{0, ..., T_L - 1\}$. Denote

$\pi_m(T_L) = \frac{N_m}{T_L}$ the empirical distribution of $\boldsymbol{z}_m$. Solve:

$$\min: \sum_m \pi_m(T_L) g_m(\boldsymbol{\upsilon}, \boldsymbol{\nu} - \boldsymbol{\theta}), \text{ s.t. } \boldsymbol{\upsilon} \succeq \mathbf{0}, \boldsymbol{\nu} \in \mathbb{R}^N, \quad (18)$$

and obtain the optimal multiplier $(\boldsymbol{\upsilon}^*(T_L), \boldsymbol{\nu}^*(T_L))$. Change $\boldsymbol{\xi}_Q$ and $\boldsymbol{\xi}_E$ in (13) to:

$$\boldsymbol{\xi}_Q = \boldsymbol{\upsilon}^*(T_L) - V^{1-\frac{\epsilon}{2}} \log(V)^2 \cdot \mathbf{1} \quad (19)$$

$$\boldsymbol{\xi}_E = \boldsymbol{\nu}^*(T_L) - V^{1-\frac{\epsilon}{2}} \log(V)^2 \cdot \mathbf{1}. \diamond \quad (20)$$

We will explain the controller in the next subsection. Here, we first note that the perturbed dual learning step is performed *only once* at time $t = T_L$.[5] Also, although LEM is equipped with a packet dropping option to ensure zero energy outage, dropping rarely happens, i.e., $O(V^{-\log(V)})$. Moreover, we will show that the energy availability constraint is always ensured.

C. Remarks on LEM

LEM only requires knowledge of the *instantaneous* state $\boldsymbol{z}(t)$ and queue states $\boldsymbol{Q}(t)$ and $\boldsymbol{E}(t)$. *It does not require any statistical information about $\boldsymbol{S}(t)$ or any knowledge of the energy state process $\boldsymbol{h}(t)$.* This is a very useful feature, as exact knowledge of the energy source may be difficult to obtain at the beginning.

There is an explicit learning step in LEM. This distinguishes it from previous algorithms for energy harvesting networks, e.g., [13], [19], [6], where sufficient statistical knowledge of the energy source is often required and no learning is considered. We will show in Theorem 2 that LEM converges in $O(V^{2/3})$ time (up to a log factor), which is much faster than the $\Theta(V)$ time for algorithms based purely on queues, or the $\Theta(V^2)$ time for algorithms based purely on learning the statistics.

The perturbation approach here is needed for guaranteeing the feasibility of dual learning and the resulting algorithm. Specifically, it "shifts" the optimal Lagrange multiplier to a positive value via $\boldsymbol{\theta}$. This step allows us to track the negative multiplier with positive queue sizes for decision making, and is critical for networks with the "no-underflow" constraint, e.g., processing networks [16].

Finally, note that due the general rate functions $\{\mu_{[nm]}(\boldsymbol{z}, \boldsymbol{P})\}$, our problem inevitably requires a centralized controller for achieving optimality. Thus, LEM requires centralized implementation. In the special case when network links do not interfere with each other and the dynamics are all independent, nodes can estimate the local distributions and pass the information to a leader node to compute $(\boldsymbol{\upsilon}^*(T_L), \boldsymbol{\nu}^*(T_L))$. Then, the leader node sends back the multiplier information to the nodes. After that, LEM can be implemented in a distributed manner.

D. Information Augmented Controller

Here we provide mathematical explanations for our controller. As we will see, the control rules are results of a drift minimization principle [20], *augmented by the information learned in perturbed dual learning*.

[5]One can also devise a version of LEM which does continuous learning.

To start, we define a *perturbed* Lyapunov function as follows:

$$L(t) \triangleq \frac{1}{2} \sum_{n,c \in \mathcal{N}} \big[Q_n^{(c)}(t)\big]^2 + \frac{1}{2} \sum_{n \in \mathcal{N}} \big[E_n(t) - \theta_n\big]^2. \quad (21)$$

Denote $\boldsymbol{Y}(t) = (\boldsymbol{Q}(t), \boldsymbol{E}(t))$ and define a one-slot conditional Lyapunov drift as follows:

$$\Delta(t) \triangleq \mathbb{E}\big\{L(t+1) - L(t) \mid \boldsymbol{Y}(t)\big\}. \quad (22)$$

We then have the following lemma from [11].

Lemma 1: Under any feasible data admission action, power allocation action that satisfies constraint (3), routing and scheduling action, and energy harvesting action that can be implemented at time t, we have:

$$\Delta(t) - V\mathbb{E}\big\{\sum_{n,c} U_n^{(c)}(R_n^{(c)}(t)) \mid \boldsymbol{Y}(t)\big\} \quad (23)$$

$$\leq B + \sum_{n \in \mathcal{N}} (E_n(t) - \theta_n)\mathbb{E}\big\{e_n(t) \mid \boldsymbol{Y}(t)\big\}$$

$$-\mathbb{E}\big\{\sum_{n,c}\big[VU_n^{(c)}(R_n^{(c)}(t)) - Q_n^{(c)}(t)R_n^{(c)}(t)\big] \mid \boldsymbol{Y}(t)\big\}$$

$$-\mathbb{E}\Big\{\sum_n \Big[\sum_c \sum_{b \in \mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t)\big[Q_n^{(c)}(t) - Q_b^{(c)}(t)\big]$$

$$+(E_n(t) - \theta_n) \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(t)\Big] \mid \boldsymbol{Y}(t)\Big\}.$$

Here $B \triangleq N^2(\frac{3}{2}d_{\max}^2\mu_{\max}^2 + R_{\max}^2) + \frac{N}{2}(P_{\max} + h_{\max})^2$, and $d_{\max}$ is defined as the maximum in-degree/out-degree of any node in the network. $\diamond$

Proof: See [11]. ■

Now add to both sides of (23) the following *drift-augmenting* term, which carries the information learned in the dual learning step, i.e., $\boldsymbol{\xi}_Q$ and $\boldsymbol{\xi}_E$ in (19) and (20) as follows:

$$\Delta_A(t) \triangleq -\mathbb{E}\Big\{\sum_n \xi_{Q,n}^{(c)}\Big[\sum_{b \in \mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t) \quad (24)$$

$$- \sum_{b \in \mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(t) - R_n^{(c)}(t)\Big] \mid \boldsymbol{Y}(t)\Big\}$$

$$-\mathbb{E}\Big\{\sum_n \xi_{E,n}\Big[\sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(t) - e_n(t)\Big] \mid \boldsymbol{Y}(t)\Big\}.$$

Doing so, one obtains the following *augmented drift*:

$$\Delta(t) + \Delta_A(t) - V\mathbb{E}\big\{\sum_{n,c} U_n^{(c)}(R_n^{(c)}(t)) \mid \boldsymbol{Y}(t)\big\} \quad (25)$$

$$\leq B + \sum_{n \in \mathcal{N}} (\hat{E}_n(t) - \theta_n)\mathbb{E}\big\{e_n(t) \mid \boldsymbol{Y}(t)\big\}$$

$$-\mathbb{E}\big\{\sum_{n,c}\big[VU_n^{(c)}(R_n^{(c)}(t)) - \hat{Q}_n^{(c)}(t)R_n^{(c)}(t)\big] \mid \boldsymbol{Y}(t)\big\}$$

$$-\mathbb{E}\Big\{\sum_n \Big[\sum_c \sum_{b \in \mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t)\big[\hat{Q}_n^{(c)}(t) - \hat{Q}_b^{(c)}(t)\big]$$

$$+(\hat{E}_n(t) - \theta_n) \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(t)\Big] \mid \boldsymbol{Y}(t)\Big\}.$$

Comparing (25) and LEM, we see that LEM is constructed to *minimize the right-hand-side (RHS) of the augmented drift*

(25). This augmenting step is important and provides a way to incorporate learning into control algorithm design.

IV. Performance analysis

Here we present the performance results for LEM. We first state the assumptions. Then, we present the theorems.

A. Assumptions

In our analysis, we make the following assumptions.

Assumption 1: There exists a constant $\epsilon = \Theta(1) > 0$ such that for any valid distributions $\hat{\boldsymbol{\pi}} = (\hat{\pi}_1, ..., \hat{\pi}_M)$ with $\|\hat{\boldsymbol{\pi}} - \boldsymbol{\pi}\| \leq \epsilon$, there exist a set of actions $\{R_{n,k}^{(c)}\}_{k\in\mathbb{N}_+}$, $\{\boldsymbol{P}_k^m\}_{k\in\mathbb{N}_+}^m$, $\{\boldsymbol{\mu}_k^m\}_{k\in\mathbb{N}_+}^m$, and $\{\boldsymbol{e}_k^m\}_{k\in\mathbb{N}_+}^m$, and distributions $\{\vartheta_k^m\}_{k\in\mathbb{N}_+}^m$ and $\{\varrho_k^m\}_{k\in\mathbb{N}_+}^m$ (possibly dependent on $\hat{\boldsymbol{\pi}}$), such that (i) there exists $\eta_0 = \Theta(1) > 0$ independent of $\hat{\boldsymbol{\pi}}$, so that:

$$\sum_m \pi_m \Big\{ \sum_k \vartheta_k^m [R_{n,k}^{(c)} + \sum_{a\in\mathcal{N}_n^{(in)}} \mu_{[a,n],k}^{(c)}, - \sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b],k}^{(c)}] \Big\} \leq -\eta_0, \ \forall\, n, c, \quad (26)$$

and for each n,

$$\sum_m \pi_m \sum_k \varrho_k^m e_{n,k}^m - \sum_m \pi_m \sum_k \vartheta_k^m \sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b],k}^m = 0, \quad (27)$$

and (ii) $0 < \sum_m \pi_m \sum_k \varrho_k^m e_{n,k}^m < \sum_m \pi_m h_n^m \ \forall\, n$. $\diamond$

Although Assumption 1 appears complicated, it indeed only assumes that the system has a "slackness" property, so that there exists a stationary and randomized policy that can stabilize the system, and the resulting service rates are slightly larger than the arrival rates for the queues. Assumption 1 is a necessary condition for achieving network stability and is often assumed in network optimization works with $\epsilon = 0$, e.g., [21]. Here with $\epsilon > 0$, we assume that systems with slightly different channel and harvestable energy distributions can also be stabilized with the same slack (the stabilizing policy may be different). Note that such a policy may not actually be implementable as it ignores the energy-availability constraint.

B. Performance Results

Here we present the performance results. We first define the following structural property of the system, which will be used in our analysis.

Definition 1: A system is called *polyhedral* with parameter $\rho > 0$, if the dual function $g(\boldsymbol{v}, \boldsymbol{\nu})$ satisfies:

$$g(\boldsymbol{v}^*, \boldsymbol{\nu}^*) \leq g(\boldsymbol{v}, \boldsymbol{\nu}) - \rho \|(\boldsymbol{v}^*, \boldsymbol{\nu}^*) - (\boldsymbol{v}, \boldsymbol{\nu})\|. \ \diamond \quad (28)$$

This polyhedral property often appears in practical systems, especially when the control actions are discrete (see [22] for more discussions). Moreover, (28) holds for all V values whenever it holds for $V = 1$.

Our first lemma shows that with $\theta_n = V\log(V)$, one can guarantee that at time T_L, the empirical multipliers $\boldsymbol{v}^*(t)$ and $\boldsymbol{\nu}^*(t)$ are close to their true values with high probability.

Lemma 2: For a sufficiently large V, with probability $1 - O(\frac{1}{V^{4\log(V)}})$, at time $t = T_L = V^c$ with $c \in (0,1)$, one has:

$$\|(\boldsymbol{v}^*(t), \boldsymbol{\nu}^*(t)) - (\boldsymbol{v}^*, \boldsymbol{\nu}^* + \boldsymbol{\theta})\| = \Theta(V^{1-\frac{c}{2}}\log(V)). \quad (29)$$

Here $\boldsymbol{\nu}^* + \boldsymbol{\theta} = \Theta(V\log(V)) > 0$. $\diamond$

Proof: Omitted. Please see our online report [23]. ■

Since $\boldsymbol{\nu}^* + \boldsymbol{\theta} = \Theta(V\log(V))$, we see that the relative error of $(\boldsymbol{v}^*(t), \boldsymbol{\nu}^*(t))$ is quite small, i.e., only $\Theta(V^{1-\frac{c}{2}}\log(V))$. This high accuracy (with respect to the size of $(\boldsymbol{v}^*, \boldsymbol{\nu}^* + \boldsymbol{\theta})$) contributes to achieving a good performance and fast convergence rate for LEM. Here $\boldsymbol{\nu}^* + \boldsymbol{\theta} = \Theta(V\log(V)) > 0$ is important, because without $\boldsymbol{\theta}$, we may get a non-positive $\boldsymbol{\nu}^*$ after solving (18), due to the fact that (10) is an equality constraint. In that case, it is impossible to use $\boldsymbol{E}(t)$ to track $\boldsymbol{\nu}^*$ and to base decisions on $\boldsymbol{E}(t)$.

We now state our first main theorem, which summarizes the performance of LEM.

Theorem 1: Suppose that the dual function $g(\boldsymbol{v}, \boldsymbol{\nu})$ is polyhedral with $\rho = \Theta(1) > 0$, i.e., independent of V, and has a unique optimal $(\boldsymbol{v}^*, \boldsymbol{\nu}^*)$ with $\boldsymbol{v}^* > 0$. Then, under LEM with $\theta_n = V\log(V)$ and a sufficiently large V, with probability $1 - O(\frac{1}{V^{4\log(V)}})$, we have:

(a) The average queue sizes satisfy:

$$Q_{n,\text{av}}^{(c)} \leq \frac{3}{2}V^{1-\frac{c}{2}}\log(V)^2 + O(1), \ \forall\, (n,c), \quad (30)$$

$$E_{n,\text{av}} \leq \frac{3}{2}V^{1-\frac{c}{2}}\log(V)^2 + O(1), \ \forall\, n. \quad (31)$$

In particular, in steady state, there exist $\Theta(1)$ constants D, ξ, K such that:

$$\Pr\{Q_n^{(c)}(t) \geq \frac{3}{2}V^{1-\frac{c}{2}}\log(V)^2 + D + b\} \leq \xi e^{-Kb} \quad (32)$$

$$\Pr\{E_n(t) \geq \frac{3}{2}V^{1-\frac{c}{2}}\log(V)^2 + D + b\} \leq \xi e^{-Kb}. \quad (33)$$

(b) For every data queue j with arrival rate $\lambda_j > 0$, there exist a set of packets with rate $\tilde{\lambda}_j \geq [\lambda_j - O(1/V^{\log(V)})]^+$, such that their average delay at queue j is $O(\log(V)^2)$.

(c) Let $\overline{\boldsymbol{r}} = (\overline{r}^{nc}, \forall\,(n,c))$ be the time average admitted rate vector achieved by LEM defined in Section II-A. We have:

$$U_{tot}(\overline{\boldsymbol{r}}) \quad \geq \quad U_{tot}(\boldsymbol{r}^*) - O(\frac{1}{V}). \quad (34)$$

Here $\boldsymbol{r}^*$ is an optimal solution of our problem. Moreover, no dropping takes places before time T_L and the average packet dropping rate is $O(1/V^{\log(V)})$. $\diamond$

Proof: Omitted. Please see our online report [23]. ■

By taking $\epsilon = 1/V$, we see from Part (a) and Part (b) that LEM achieves an $[O(\epsilon), O(\log(\frac{1}{\epsilon})^2)]$ utility-delay tradeoff. We also see from Part (a) that LEM can use an energy buffer of size $O((\frac{1}{\epsilon})^{1-\frac{c}{2}}\log(\frac{1}{\epsilon})^2)$, which is much smaller than the $\Theta(1/\epsilon)$ size under previous algorithms.

Our second main result concerns the *convergence time* of LEM. The convergence time of an algorithm characterizes how fast it (or equivalently, the system) enters its steady state. A faster convergence speed implies faster learning and more efficient resource allocation. The formal definition of the convergence time is as follows [15]:

Definition 2: Let $\zeta > 0$ be a given constant. The convergence time of the control algorithm is defined as:

$$T_\zeta \triangleq \inf\{t : \|(\hat{\boldsymbol{Q}}(t), \hat{\boldsymbol{E}}(t)) - (\boldsymbol{v}^*, \boldsymbol{\nu}^* + \boldsymbol{\theta})\| < \zeta\}. \diamond \quad (35)$$

Here the intuition is that once $(\hat{\boldsymbol{Q}}(t), \hat{\boldsymbol{E}}(t))$ gets close to $(\boldsymbol{v}^*, \boldsymbol{\nu}^* + \boldsymbol{\theta})$, LEM will start making near-optimal decisions.

***Theorem** 2:* Suppose the conditions in Theorem 1 hold. Under LEM, there exists an $\Theta(1)$ constant D such that, with probability $1 - O(\frac{1}{V^{4\log(V)}})$,

$$\mathbb{E}\{T_D\} = O(V^c + V^{1-\frac{c}{2}}\log(V)^2). \quad (36)$$

In particular, when $c = \frac{2}{3}$, $\mathbb{E}\{T_D\} = O(V^{\frac{2}{3}}\log(V)^2)$. $\diamond$

Proof: Omitted. Please see our online report [23]. ■

We remark here that if one only uses pure queue-based policies to track the optimal multipliers, e.g., ESA in [11] (can be viewed as linear learning since each queue can change by an $\Theta(1)$ amount at each time), the convergence time is necessarily $\Theta(V)$, since the optimal multiplier is $\Theta(V)$ [22]. If instead one tries to compute the optimal solution only by learning the distribution, it requires $\Theta(V^2)$ time to ensure that the distribution is within $O(1/V)$ accuracy. Dual learning can be viewed as combining the benefits of the two methods, i.e., the fast start of statistical learning and the smooth learning of queue-based policies. Hence, it is able to achieve a superior convergence speed compared to both methods.

V. Simulation

This section provides simulation results for LEM. We consider the network shown in Fig. 2, which is an example of a data collecting sensor network. In this network, traffic are admitted from nodes 1, 2 and 3, and are relayed to node 4. Since we only have one commodity, we omit the superscript.

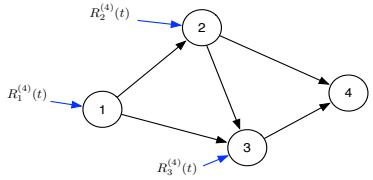

Fig. 2. A data collection network.

The channel state of each communication link is i.i.d. every time slot and can be either "G=Good" or "B=Bad." The probabilities of being in the good state for the links are given by $\boldsymbol{p}^s = (p^s_{12}, p^s_{13}, p^s_{23}, p^s_{24}, p^s_{34}) = (0.5, 0.2, 0.3, 0.5, 0.7)$. For each node, the harvestable energy $h_n(t)$ takes values 2 or 0. The probabilities of having a 2-unit energy arrival at the nodes are $\boldsymbol{p}^h = (p^h_1, p^h_2, p^h_3) = (0.6, 0.3, 0.5)$. We have a total of 32 channel states and 8 energy states.

At every time t, a node can either allocate one unit power for transmission or do not transmit. When the channel is good, one unit power can support a transmission of two packets. Otherwise it can only support one. We assume $R_{\max} = 2$ and at each time $R_n(t) \in \{0, 1, 2\}$. The utility functions are given by: $U_1(r) = 3\log(1 + r)$ and $U_2(r) = 2\log(1 + r)$, and $U_3(r) = \log(1 + r)$. We also assume that the links do not interfere with each other.

We simulate LEM with $V \in \{30, 40, 50, 80, 100, 150\}$ and $c = 2/3$. We choose to begin with $V = 30$ so that dropping does not happen. Each simulation is run for 10^6 slots. In the simulation, in order to combact the effect of V not being large enough, we slightly increase the learning time from V^c to $V^c\log(V)$ (same performance can be proven). We also reduce $V^{1-c/2}\log(V)^2$ in (19) and (20) to $V^{1-c/2}\log(V)$, the results are not affected. For benchmark comparison, we also implement the ESA algorithm in [11].[6]

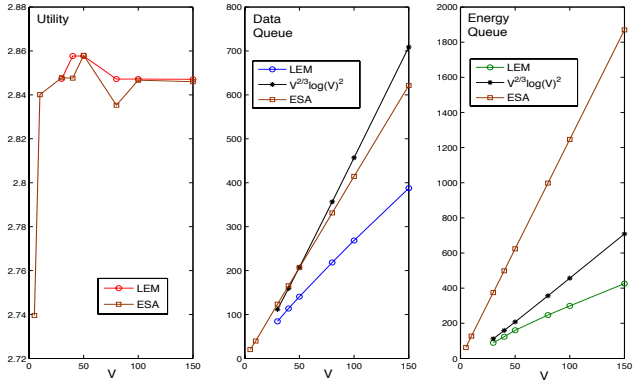

Fig. 3. Utility and queue performance of LEM.

Fig. 3 shows the utility and queue performance of LEM. We see that LEM achieves good utility performance. We also see from the middle and the right plots that both the average data queues and the average energy queues under LEM are of size $O(V^{1-c/2}\log(V)^2)$, whereas it is $\Theta(V)$ under ESA. This implies that one can implement LEM with much smaller energy buffers compared to ESA.

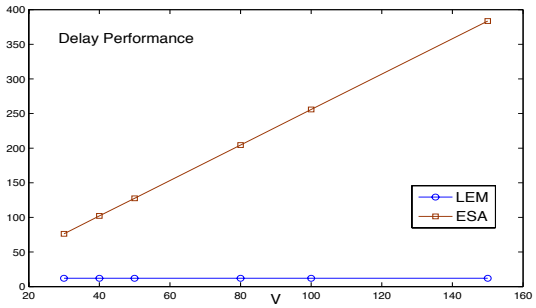

Fig. 4. Delay scaling under LEM.

Fig. 4 also shows the average packet delay under LEM, computed using the packets that exit the network when the simulation ends (This accounts for more than 99.9% of the packets that enter the network). It can be seen that the average packet delay under LEM grows very slowly, i.e., $O(\log(V)^2)$, and stays around 12 slots. On the other hand, the delay under ESA grows linearly in V, and ranges from 75 to 380 slots (ESA requires a $V \geq 30$ to achieve a similar utility performance as LEM). Thus, with the same utility performance, LEM achieves a 6 to 30-fold delay saving.

Fig. 5 shows the convergence property of LEM. To show the convergence, we shorten the time to 10^4 slots and change the system statistics in slot 5000 to $\boldsymbol{p}^s = (0.3, 0.2, 0.2, 0.5, 0.7)$ and $\boldsymbol{p}^h = (0.1, 0.6, 0.2)$. We see that LEM converges much faster compared to ESA (We only show $\boldsymbol{E}(t)$ here. The data queues have similar performance). Indeed, in the first time, the energy queue sizes converge to the optimal values

[6]Other algorithms in the literature are designed for different settings and do not directly apply to our problem.

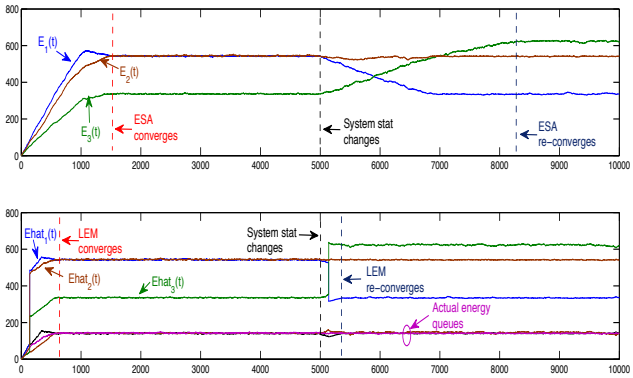

Fig. 5. Convergence comparison between LEM and ESA for $V = 150$

(the corresponding optimal Lagrange multiplier values) at around 650 slots under LEM, whereas ESA converges at around 1500 slots (an 850-slot saving!). Then, after we apply the change, LEM re-converge after about 450 slots, whereas ESA takes about 3300 slots to re-adapt to the system ($7\times$ faster, save about 2900 slots!). Moreover, we also note that the actual energy queue sizes are barely affected, except for a small change after time 5000. This clear demonstrates the effectiveness of using dual learning in accelerating the convergence of the algorithm.

VI. Conclusion

In this paper, we develop a learning-aided energy management algorithm (LEM) for general multihop energy harvesting networks. LEM explicitly utilizes historic system information and learns an empirical optimal Lagrange multiplier via perturbed dual learning. Then, it incorporates the multiplier into a drift-based system controller via drift-augmenting. We show that LEM is able to achieve a near-optimal utility-delay tradeoff with a finite energy storage capacity. Moreover, LEM possesses a provable faster convergence speed compared to existing techniques that based purely on queue-based control or based purely on learning the statistics.

VII. Acknowledgement

This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61361136003, 61303195, Tsinghua Initiative Research Grant, Microsoft Research Asia Collaborative Research Award, and the China Youth 1000-talent Grant.

References

[1] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. B. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. *Proc. of IEEE IPSN*, April 2005.

[2] Solar powered smart benches. *Energy Harvesting Journal*, July 22, 2014.

[3] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman. Challenge: Ultra-low-power energy-harvesting active networked tags (EnHANTs). *Proceedings of MobiCom*, Sept. 2009.

[4] S. Meninger, J. O. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. H. Lang. Vibration-to-eletric energy conversion. *IEEE Trans. on VLSI, Vol. 9, No.1*, Feb. 2001.

[5] Wireless train sensor harvests vibrational energy. *Energy Harvesting Journal*, July 22, 2014.

[6] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Trans. on Embedded Computing Systems, Vol.6, Issue 4*, Sept. 2007.

[7] D. Ganesan C. M. Vigorito and A.G. Barto. Adaptive duty cycling for energy harvesting systems. *Proceedings of IEEE SECON*, 2007.

[8] R. Srivastava and C. E. Koksal. Basic tradeoffs for energy management in rechargeable sensor networks. *ArXiv Techreport arXiv: 1009.0569v1*, Sept. 2010.

[9] M. Gatzianas, L. Georgiadis, and L. Tassiulas. Control of wireless networks with rechargeable batteries. *IEEE Trans. on Wireless Communications, Vol. 9, No. 2*, Feb. 2010.

[10] L. Lin, N. B. Shroff, and R. Srikant. Asymptotically optimal power-aware routing for multihop wireless networks with renewable energy sources. *Proceedings of INFOCOM*, 2005.

[11] L. Huang and M. J. Neely. Utility optimal scheduling in energy harvesting networks. *IEEE/ACM Transactions on Networking, Vol. 21, Issue 4, pp. 1117-1130*, August 2013.

[12] O. Simeone C. Tapparello and M. Rossi. Dynamic compression-transmission for energy-harvesting multihop networks with correlated sources. *IEEE/ACM Trans. on Networking*, 2014.

[13] S. Chen, P. Sinha, N. B. Shroff, and C. Joo. A simple asymptotically optimal joint energy allocation and routing scheme in rechargeable sensor networks. *IEEE/ACM Trans. on Networking*, to appear.

[14] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] L. Huang, X. Liu, and X. Hao. The power of online learning in stochastic network optimization. *Proceedings of ACM Sigmetrics*, 2014.

[16] L. Jiang and J. Walrand. Stable and utility-maximizing scheduling for stochastic processing networks. *Allerton Conference on Communication, Control, and Computing*, 2009.

[17] M. J. Neely. Energy optimal control for time-varying wireless networks. *IEEE Transactions on Information Theory 52(7): 2915-2934*, July 2006.

[18] V. Sharma, U. Mukherji, V. Joseph, and S. Gupta. Optimal energy management policies for energy harvesting sensor nodes. *IEEE Trans. on Wireless Communication, Vol.9, Issue 4.*, April 2010.

[19] L. Huang. Optimal sleep-wake scheduling for energy harvesting smart mobile devices. *Proceedings of WiOpt*, April 2013.

[20] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking Vol. 1, no. 1, pp. 1-144, 2006.

[21] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE/ACM Trans. Netw.*, 15(6):1333–1344, 2007.

[22] L. Huang and M. J. Neely. Delay reduction via Lagrange multipliers in stochastic network optimization. *IEEE Trans. on Automatic Control*, 56(4):842–857, April 2011.

[23] L. Huang. Fast-convergent learning-aided control in energy harvesting networks. *ArXiv Tech Report 1503.05665v1*, 2015.

[24] R. Durrett. *Probability: Theory and Examples*. Duxbury Press, 3rd edition, 2004.

[25] L. Huang and M. J. Neely. Max-weight achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under Markov dynamics. *arXiv:1008.0200v1*, 2010.

[26] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari. LIFO-backpressure achieves near optimal utility-delay tradeoff. *IEEE/ACM Transactions on Networking*, 21(3):831–844, June 2013.