Optimal Task Scheduling Policy in Energy Harvesting Wireless Sensor Networks

Vijay S. Rao, R. Venkatesha Prasad, Ignas G. M. M. Niemegeers

Embedded Software, Faculty of EEMCS
Delft University of Technology, the Netherlands
{V.Rao, R.R.VenkateshaPrasad, I.G.M.M.Niemegeers}@tudelft.nl

Abstract—**Ambient energy harvesting for Wireless Sensor Networks (WSNs) is being pitched as a promising solution for long-lasting deployments in various WSN applications. However, the sensor nodes most often do not have enough energy to handle application, network and house-keeping tasks because amount of energy harvested highly varies spatially and temporally. Moreover the ambient source cannot be assumed to be continuously available. When harvested energy is in excess, it is desirable that the nodes take up higher loads. The nodes should switch to highly energy efficient schemes when the energy is not sufficient. Hence harvesting-aware scheduling of tasks is required. The two most important challenges for harvesting-aware scheduling are (a) to determine the amount of energy to be expended in a time slot, and (b) to utilize this energy for execution of tasks maximally. To increase energy utilization for task execution, we decompose application level tasks into subtasks, some of which can be executed concurrently. In this article, we propose a dynamic optimization model, based on Markov Decision Process (MDP) that takes into account priorities and deadlines of the tasks, and stored and harvested energy to derive an optimal scheduling policy. Since the complexity of the MDP is intractable in real-time, we propose a greedy scheduling policy. We compare its performance with the optimal policy.**

I. Introduction

Wireless sensor networks (WSNs) have been deployed for a wide variety of applications. They are designed and expected to run for considerably long periods of time. A major drawback of these networks is the limited lifetime of the nodes, since they are typically powered by batteries. Frequent battery replacement is labor intensive in some cases. In many other situations, battery replacement is impractical due to physical or deployment conditions. The other associated problems of batteries, such as, increased size for increased capacity, higher leakage, etc., make them unattractive.

A promising approach, for perpetual network operations, is to harvest energy from ambient sources, such as light, radio frequency, thermal, wind, water, salinity gradients and moving objects [1]. Unfortunately, merely replacing the batteries with harvesters does not work. Ambient energy sources do not provide constant power. While the harvested energy can at times be very low, it can be in excess of the storage capacity of the nodes at times. For instance, statistics show that the difference among the available solar power in shadowy, cloudy and sunny environments can be up to three orders of magnitude [2]. The harvested energy from these ambient sources varies drastically over location and time. Therefore, nodes across the network may not have the same energy levels. Consequently, energy harvesting in these devices necessitates a redesign of algorithms, communication techniques, network protocols, and transceiver hardware to achieve *energy neutrality* while offering perpetual operations [3]. The foremost step in building towards an energy harvested wireless sensor networks (EH-WSNs) is to manage the power on a single node.

Scheduling tasks on EH-WSN node has been an active topic of research [1], [4], [5], [6]. Moser et al. [4] propose an optimal scheduling algorithm called Lazy Scheduling Algorithm (LSA). However, this algorithm requires the tasks to be preemptive. Moreover, the algorithm necessitates that the future incoming energy is predicted accurately. DEOS [5] is a dynamic scheduler for energy harvesting sensor networks. In DEOS, tasks are decomposed and combined when they can be, and concurrent execution is adopted on these tasks. Nodes look at being energy efficient while maximizing utility. While LSA considers the future energy arrivals, it does not maximally utilize the energy. On the contrary, DEOS maximizes the utility but does not consider energy arrivals. Therefore, the node may die if there is no incoming energy.

When scheduling tasks for EH-WSNs, two fundamental questions need to be answered together:

i How much energy should be expended in the current time period?
ii How to utilize this allocated energy maximally?

In this paper, we address these questions, which have not been considered together. We model the incoming energy as a stochastic process [6], [7]. Since we consider future energy arrivals in the system, the problem at hand is to dynamically optimize the energy expenditure while maximizing the utility, given the variations in harvesting energy. To solve this stochastic dynamic optimization problem, we model our problem based on Markov Decision Process (MDP). We compute the optimal policy to maximize the average utility over infinite time horizon. The model has high execution complexity since all possible combinations need to be evaluated on an embedded device. Hence, we propose a low-complexity policy that can be executed on the nodes and compare its performance with the MDP's policy. Our contribution of this paper:

- we propose an MDP based model for generating optimal scheduling policies. Through this model, we can deter-

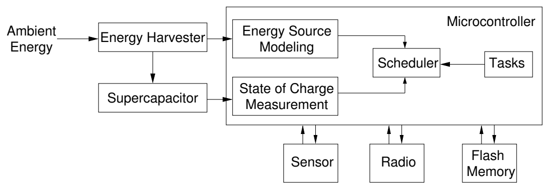


Fig. 1: Block diagram of the system

mine (a) energy to expend with the knowledge of future energy arrival and (b) maximal set of tasks to execute.

- we propose a suboptimal *Greedy Policy* since the complexity of the MDP model is high.

The remainder of the paper is organized as follows. In Section II, we present the system under consideration and the assumptions made. In Section III, we introduce Markov Decision Processes, discuss our model based on MDP for generating optimal policies and also propose a suboptimal policy. Section IV demonstrates the usability of the model and policies through results. In Section V, we describe the state of the art scheduling mechanisms in EH-WSNs. We present our conclusions in Section VI.

II. System Model

Figure 1 depicts the system model of an energy harvesting sensor node. An energy harvesting circuit scavenges energy from the ambient source and charges a supercapacitor. The supercapacitor powers a microcontroller, which controls the operation of the sensor node and other components connected to it namely, sensor, radio and flash memory. The microcontroller manages and schedules the access to the components, and hence, the energy consumption of the device.

We specifically consider a supercapacitor as the energy storage buffer due to its high energy density and theoretically unlimited charge-discharge cycles, unlike rechargeable batteries. Moreover, the remaining energy in the supercapacitor can be estimated by $E \approx CV^2/2$, where C is the capacitance of the supercapacitor and V is the voltage[1]. We discretize the energy in the supercapacitor. Different possible energy levels are denoted by $E = \{e_0, e_1, \ldots, e_{max}\}$. e_0 is the zero energy state. We define $e_{min} \geq e_1$ as the minimum amount of energy required to keep the node operational. We model the harvesting source as a stochastic process. Many works have assumed on-off processes and Poisson arrivals of energy. Markov chains based synthetic (solar and wind) data generators have been demonstrated to have good accuracy [4], [7], [9].

A *task* is a sequence of operations that are executed on a node. Tasks, for example, can be 'report measured sensor values', 'forward a packet', 'update routing table' etc. We assume the task arrival is also modeled as a stochastic process.

[1]Other types of storage elements such as rechargeable batteries may be used, however, estimating their charge in real-time is an issue. Sophisticated techniques such as [8] are being developed and can be used in the model when available.

III. MDP Model and Policies

A. Preliminaries

In Markov Decision Process (MDP) [10], a decision maker makes decisions for a probabilistic system at regular intervals of time (discrete version of the MDP). The system is represented by a set of states. The decision is to perform an *action* that maximizes his/her goal (or *reward*) with respect to some predetermined performance criterion. As a result of choosing an action in a state, two things happen: the decision maker receives a reward, and the system evolves to a possibly different state at the next decision epoch. Both the rewards and the transition probabilities depend on the current state and the choice of action, and not on states occupied and actions chosen in the past (*Markov* property). A policy is a sequence of actions. The goal of the MDP (and the decision maker) is to choose a policy that maximizes the reward.

Formally, a MDP is defined as a quintuple:

$$\{\Lambda, S, A_s, p_k(\cdot|s,a), r_k(s,a)\}, \tag{1}$$

where, $\Lambda = \{k, 2k, \ldots K\}$ are the decision epochs; S is the set of states of the model; A_s is the set of allowable actions in state $s \in S$. $r_k(s,a)$ is a real-valued function, defined for a $s \in S$ and $a \in A_s$, that denotes the reward. The system state at the next decision epoch is determined by $p_k(s,a)$, which is the state transition probability.

Let $\pi = \{d_1, \ldots, d_{K-1}\}$ denote a policy where d_k denotes the decision rule for an action to be taken in epoch k. The value of the policy π is defined as

$$V^\pi = \mathbb{E}^\pi \left[\sum_{k=1}^{K-1} r_k(s_k, a) \right],$$

where $\mathbb{E}$ is the expectation operator and K is the horizon over which the decisions are to be made. The goal is to find a policy π_* such that V^π is maximized, that is,

$$V^{\pi_*} = \arg\max_{\pi} \mathbb{E}^\pi \left[\sum_{k=1}^{K-1} r_k(s_k, a) \right]. \tag{2}$$

The above formulation is a *finite horizon* MDP; an *infinite horizon* MDP can also be defined. In this case, the value of a policy π thus becomes

$$V^{\pi_*} = \arg\max_{\pi} \lim_{K \to \infty} \mathbb{E}^\pi \left[\sum_{k} r_k(s_k, a) \right]. \tag{3}$$

The optimality equation for a given start state s, also known as Bellman's equation, is given by

$$V^\pi(s) = \max_{a \in A_s} \left\{ r(s,a) + \sum_{l \in S} p(l|s,a) V(l) \right\}. \tag{4}$$

The optimal policy is π_* that has the value $V^{\pi_*}(s) \geq V^\pi(s)\ \forall \pi \in \Pi$, where Π is the set of all policies. Without loss of optimality in Eqn. 4, we only consider the set of policies that results in an average reward independent of the initial state.

Before we describe the MDP based model, we discuss briefly about tasks and utility.

B. Tasks, subtasks and utility functions

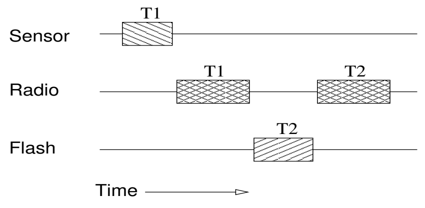
A task τ_i is characterized by $(\zeta_i, \delta_i, \rho_i, R_i, \epsilon_i)$, where $i \in \{1, \ldots, N\}$. Here, ζ_i is the time of arrival, δ_i is the deadline, ρ_i is the priority, R_i is the resources required for execution of the task. Each task τ_i consumes ϵ_i amount of energy.

Decomposition of tasks into subtasks. Typical WSN applications have tasks such as 'report sensor values', 'relay data', 'in-network processing of data before forwarding', 'update routing table in the flash', etc. It is possible to divide them into subtasks, and execute them. For example, 'report sensor values' can be split into *sense* and *transmit*. Similarly, 'relay data' can be decomposed into *receive* and *transmit*; 'in-network processing before forwarding' can be split into *receive*, *compute* and *transmit*. Thus the set of all possible subtasks is {*sense, transmit, receive, write, read, compute*}. All the tasks can be split into a combination of these in a sequence.

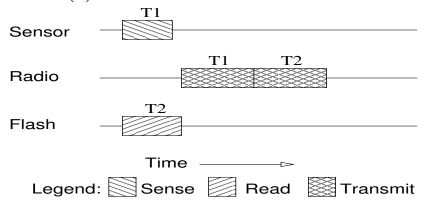
A task τ_i is decomposed into $\tau_i = \{\tau_{i1}, \tau_{i2}, \ldots, \tau_{im}\}$. Each subtask τ_{ij} can be characterized by $(\zeta_{ij}, \delta_{ij}, \rho_{ij}, R_{ij}, \epsilon_{ij})$. The subtasks derive the values for arrival time and priority from their parent task. R_{ij} points to the resource the subtask requires i.e., radio, sensor, flash memory etc. Energy required for a subtask is determined from a lookup table, which can easily be constructed from the datasheet of the component. Without the loss of generality, we assume the total amount of energy consumed by the task τ_i is $\epsilon_i = \sum_{j=1}^{m} \epsilon_{ij}$.

Some subtasks can be executed in parallel for e.g., sensing a physical parameter and receiving a packet can be executed concurrently. This concurrent execution makes efficient use of resources and reduces energy consumption. However, depending on the hardware, some subtasks cannot be executed simultaneously. For example, in the popular Tmote Sky mote, a radio operation cannot happen in parallel to a flash read/write operation [11]. Such mutually exclusive cases need to be taken into consideration. While decomposing of tasks has been proposed in DEOS [5], this work distinguishes itself from it in the manner of combining the tasks. Moreover, we consider the future incoming energy for task scheduling.

An example of splitting and combining subtasks is shown in Figure 2. Task 1, denoted by 'T1', is 'report sensor value' i.e., Sense and Transmit. Task 2, denoted by 'T2', is 'report a stored value' i.e., Read and Transmit. A case in point for energy saving through this method is illustrated here. In Figure 2a, the tasks are executed one after the other i.e., T1 followed by T2. When the tasks are split, the sensing subtask and the reading subtask can be executed together, followed by transmission subtasks. This is shown in Figure 2b. This saves time, allowing the microcontroller to be put in sleep state earlier, and hence is more energy efficient. Apart from decomposition and efficient utilization of the components and energy, we can also see the precedence constraints in this example. The transmission subtasks are not executed until the sensing and reading subtasks complete even though the



(a) Normal execution of tasks



(b) Execution with subtasks

Fig. 2: Example of splitting and combining tasks for execution.

radio resource is free. We represent the order of execution of the subtasks in a directed acyclic graph and the subtasks are selected accordingly.

Utility functions: Typically utility maximization frameworks have been defined in the context of fairness, but can be extended to applications such as scheduling [7]. To compare tasks and to make choices, we define utility of a task τ_i as a function of the deadline, priority and energy consumed for the task i.e., $\upsilon_i = f(\delta_i, \rho_i, \epsilon_i)$. In general, the utility function is defined as $\upsilon : \tau \rightarrow \mathbb{R}^+$ [12]. A utility function should satisfy the following two properties: (a) *completeness*, which means every task τ_i has a utility value υ_i defined by the utility function, and (b) *transitivity*, which means if $\upsilon_x \geq \upsilon_y$ and $\upsilon_y \geq \upsilon_z$, then $\upsilon_x \geq \upsilon_z$. Such functions can be defined. For example, the function $\upsilon(\cdot) = log((\cdot))$ can be used to describe the utility of a task. The function satisfies both the conditions of completeness and transitivity. We extend the utility function for subtasks rather than tasks, since we seek to construct policies for scheduling subtasks.

C. Proposed MDP based model

In a causal system, the problem of task scheduling would have been to maximize utility subject to energy availability in the supercapacitor. However in our non-causal system, we need to find both how much energy should be spent and then maximize the utility. To tackle this *dynamic optimization* problem, we model the problem with Markov Decision Process (MDP).

To this end, we define a policy π as *admissible* policy if there exists at least one procedure or sequence in which all subtasks selected according to the policy π adhere to the precedence and mutual exclusion constraints, and can be scheduled on the sensor mote. We denote the set of all admissible policies by Φ.

For the sake of simplicity, we consider a slotted system[2], where subtasks have to be scheduled at the beginning of each

[2]We assume the slots are big enough for any type of subtask to be executed.

slot. The beginning of the timeslots are the decision epochs. The state space S is defined as $S = E \times Q^{\mathcal{A}} \times Q^{\mathcal{B}} \times \cdots \times Q^{\mathcal{X}}$. $\Gamma = \{\mathcal{A}, \mathcal{B}, \ldots \mathcal{X}\}$ represents the set of type of subtasks, e.g., $\mathcal{A}$ could represent the sensing subtask. $Q^{\mathcal{A}}$ represents the queue of sensing subtasks i.e., queue of subtasks of type $\mathcal{A}$. The queue lengths are finite.

Let P_I denote the harvested energy arrival process and let P_τ denote the task arrival process. Since these processes are independent of each other, the state transition probabilities can be computed by their product.

The maximum number of elements is $q^{\mathcal{A}}_{max}$. $d^{\mathcal{A}}_k \in \{0, 1\}$ denotes the decision taken at decision epoch k.

In our model, we use utility functions to calculate the rewards. With all elements defined for the model, the problem is to find an optimal policy π_* according to

$$U^{\pi_*} = \underset{\pi \in \Phi}{\arg\max} \lim_{K \to \infty} \mathbb{E}^{\pi} \left[\sum_{k=1}^{K} U(s_k, a) \right]. \tag{5}$$

The optimal policy corresponds to optimal scheduling policy for choosing subtasks at each epoch. The optimal policy is the solution to both the questions raised before. The explanation is as follows: for this MDP formulation, a particular action is chosen for any given state. An action here corresponds to executing a set of subtasks. The action is chosen such that the reward is maximized in the long run. That is, if the transition probability to state with higher energy level is significant (indicates more energy is expected), then MDP chooses more subtasks since the reward will be maximized. On the contrary, when chances of ending up in a state with lower energy levels (indicates lesser or no energy is expected to be harvested) is higher, then MDP chooses the 'right' set of subtasks such that the reward is maximized over time. In this way, in the optimal policy, the amount of energy to be expended and the set of tasks to be executed are chosen so that optimal reward is obtained.

The optimal policy can be obtained by *policy iteration algorithm*. For an infinite horizon MDP formulation, the optimal policy will be a stationary one [10].

It can be proven that there exists a threshold utility u_{th} such that (a) $\sum U(s_k, a) \geq u_{th}$ i.e., subtasks are selected to obtain a utility greater than or equal to the threshold, or (b) no subtasks are selected. The intuition behind this lemma is that given the current energy value and the incoming energy profile, it is rewarding to execute a set of subtasks only if their total utility is above a certain value. It is possible that some of the subtasks are not executed when in low energy state. This model looks at maximizing the utility at infinite horizon taking into consideration the incoming energy profile.In the long run, not executing lower priority tasks when low on energy will still lead to optimal utility.

The complexity of the MDP model grows exponentially with increasing number of subtasks and queue length for each subtask. The search space for optimal policy can be reduced by carefully choosing the reward function (i.e., in our case the utility function) to be a monotonic concave function [10]. We choose the following function as our utility function for a subtask j:

$$\upsilon_j(k) = \begin{cases} \frac{\rho_j}{k-\delta_j} log(1 + \epsilon_j) & \text{for } k > \delta_j \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The reward function $r(s, a)$ i.e., $U(s, a)$ can be calculated as the sum of utility functions of the selected subtasks. It can be proven that such a reward function is optimal since it is both concave and monotonic.

D. Greedy Policy

While the search space is reduced for policy iteration algorithm, the 'curse of dimensionality' [10] problem still exists. This necessitates development of low complexity policy *albeit* suboptimal.

We propose a *greedy* policy in which we maximize the utility in each decision epoch while taking into account the energy arrivals. We neglect the task arrivals since the policy is greedy. We exploit the structure of this problem in defining the policy. Notice that each task was split into subtasks based on the type of resource required. Hence, it is sufficient to look at each subtask queue, which reduces the complexity to $O(MN)$, where M is the number of subtask queues and N is the number of tasks in the node. Based on this, we propose the suboptimal greedy policy that maximizes the immediate expected reward in slot k:

$$\pi_g = \underset{b \in q^{\gamma}}{\arg\max}\, \upsilon_b(s_k) p(l|s_k, a) \quad \forall \gamma \in \Gamma \tag{7}$$

$$s.t. \ e(k+1) \geq e_{min} \tag{8}$$

The above equation says that, in a given state, for every subtask pick the one that gives maximum utility under the *condition* that outage does not occur. We update the transition probabilities for energy arrival at the end of the slot. Clearly the worst case complexity of this suboptimal policy is $O(MN)$ since we search each of the M queues based on the utility and select the desired subtasks.

IV. Results and Discussions

To evaluate the policies, we simulated the system using Markov Decision Process toolbox [13] in MATLAB. We considered four tasks 'report sensor data' (sense and transmit), 'relay data' (receive and transmit), 'respond to query' (read and transmit) and 'update value' (receive and write), and the energy values were taken from [11]. We set the task arrivals to follow uniform distribution. We considered a solar-powered sensor node with 12 mJ supercapacitor as storage. For simulation purposes, we discretize this value in multiples of 200 μJ. We do not consider leakage and retransmissions in this simulations.

We constructed an energy arrival process with the solar dataset from CONFRRM [14] for the month of July 2011 and calculated the energy harvested by considering a $1\,\mathrm{m}^2$ solar panel. We constructed the transition probabilities based on this data. Based on this information, we computed the optimal policy using the MDP toolbox.

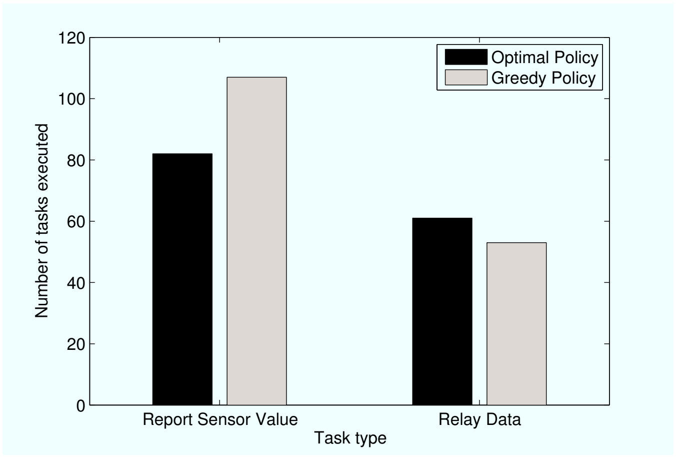
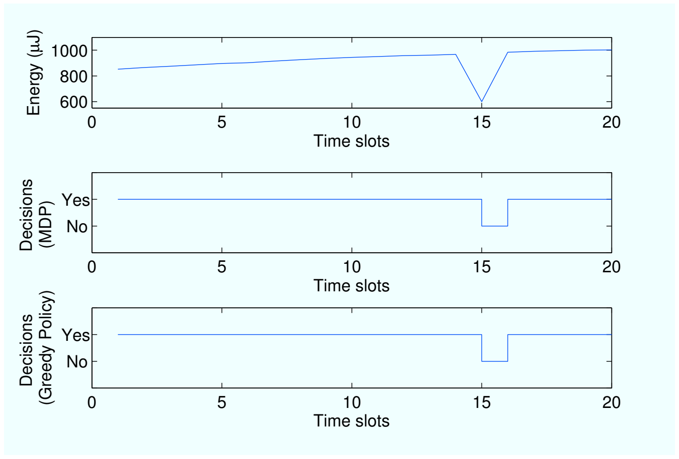


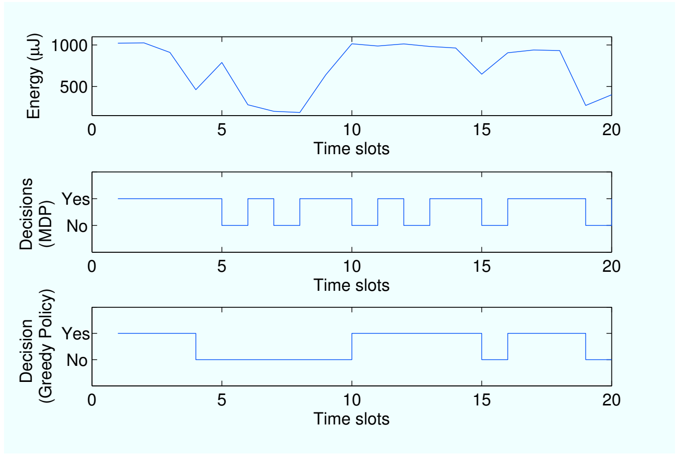
Fig. 3: Comparison of optimal and greedy policies for a random day

First, we considered a random day in July 2011 (July 28, 2011) and executed the optimal policy and the greedy policy for this sample path. We executed 'report sensor value' and 'relay data' tasks. The comparison is shown in Figure 3. The result from MDP shows that the average number of tasks selected for execution (i.e., both subtasks must be selected for the task to complete). One of the reasons for the optimal policy to have executed lesser number of tasks than the greedy policy could be the energy arrival for the considered day as compared to the energy distribution used in the MDP. From the figure, it seems that the MDP model expected lower energy arrivals and hence was conservative in executing tasks.

To demonstrate the decision making process in the policies, we did the following. From the CONFRRM dataset, we considered a good period (good sunshine with no major/abrupt changes in harvesting process) of a day (July 10 2011). Figure 4a shows the comparison of the MDP and the greedy policy. However, when a bad period (high variations in the energy harvesting process) of a day (July 17, 2011) is considered, the greedy policy fails to adapt quickly to the changes while the MDP performs well as shown in Figure 4b.



(a) Decisions in a good period



(b) Decision in a bad period

Fig. 4: Decisions made by Optimal Policy (MDP) and Greedy Policy on sample paths.

V. Related work

There is a growing body of literature on scheduling in energy harvesting sensor networks. In [15], the authors propose an utility optimal scheduling of transmissions using Lyapunov functions. Online algorithm is proposed to manage the energy and allocate power to transmissions jointly. However, the scheduler disregards the energy profile of the source, and hence, this may lead to outages in the future time slots. Moser *et al.* [4] look into the problem of real-time scheduling and proposes an optimal scheduling algorithm called Lazy Scheduling Algorithm (LSA). In this scheduler, all tasks are preemptive. The tasks are delayed in order to harvest energy as much as possible. With energy clairvoyance, they prove that LSA is the optimal scheduler. The algorithm is simple albeit the future incoming energy should be predicted accurately and the tasks should be preemptive. While some tasks are preemptive, a task such as packet transmission cannot be preempted. Moreover, only task can be executed at a time, which implies lower utility.

Audet et al. [6] consider recurring tasks on the nodes and generate equivalent virtual tasks. With considerations to the stochastic nature of the energy source, they propose schemes to adapt power levels of execution. These schemes can be used with other schedulers like LSA. However, all tasks to be executed are known beforehand, which is restrictive. Authors of [16] consider joint dynamic voltage and frequency selection and task scheduling. None of the above important works provide a dynamic scheduling framework while considering the stochastic energy profile of the source.

Markov Decision Process has been used to derive optimal transmission policies over one hop in energy harvesting sensor networks [17], [18]. In [17], the energy consumption and replenishment are modeled with a Markov chain as a birth-death process. Energy for replenishment is modeled as Poisson process. Messages to be transmitted arrive also as

Poisson process each with a value V, which also represents the immediate reward. An optimal policy is to determine a threshold vector, where a threshold at a decision epoch determines the minimum value of a message that is allowed to be transmitted in that epoch. The optimal policies are found for various energy storage models. [18] considers a solar source and models the battery states using 2D Markov chain i.e., different battery levels under three states of the solar source (*viz.* cloudy, sunny and night). The MDP model has been limited to finding effective transmission strategies while much more can be achieved with it, as will be shown in this article.

DEOS [5] proposes a dynamic scheduler for energy harvesting sensor networks. In DEOS, tasks are decomposed and combined when they can be, and concurrent execution is adopted on these tasks. Nodes look at being energy efficient while maximizing utility. An admission control procedure is also described. This approach reduces the possibilities of node deaths, however, the utility is reduced since they do not consider the incoming energy. DEOS also does not answer the question of how much energy to expend in a time period.

VI. Conclusions

In this paper, we brought up the two important challenges to be handled in a harvesting-aware scheduling for an energy harvesting wireless sensor node: (a) the amount energy to be expended and (b) maximizing the utilization of the energy. To address both the issues, we proposed a model based on the popular dynamic optimization methodology i.e., Markov Decision Process. This model generates an optimal policy for scheduling. The optimality is reached subject to the condition that the energy arrival process is well-modeled. Since the complexity of the MDP model is high, we proposed a greedy policy for scheduling and compared it with the MDP model. The suboptimal greedy policy is necessary since (a) it can easily be implemented on the nodes and (b) it learns and updates the transition probabilities, thereby adapting to the immediate variations in the ambiance. The greedy policy performs sufficiently well and, with a suitable sized storage element the performance can be guaranteed.

References

[1] R. V. Prasad, S. Devasenapathy, V. S. Rao, and J. Vazifehdan, "Reincarnation in the Ambiance: Devices and Networks with Energy Harvesting," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 195–213, 2014.

[2] M. Rahimi, H. Shah, G. Sukhatme, J. Heideman, and D. Estrin, "Studying the feasibility of energy harvesting in a mobile sensor network," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 1. IEEE, pp. 19–24.

[3] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 6, no. 4, Sep. 2007.

[4] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-time scheduling for energy harvesting sensor nodes," *Real-Time Systems*, vol. 37, no. 3, pp. 233–260, Jul. 2007.

[5] T. Zhu, A. Mohaisen, and D. Towsley, "DEOS: Dynamic energy-oriented scheduling for sustainable wireless sensor networks," in *2012 Proceedings IEEE INFOCOM*. IEEE, Mar. 2012, pp. 2363–2371.

[6] D. Audet, L. C. de Oliveira, N. MacMillan, D. Marinakis, and K. Wu, "Scheduling Recurring Tasks in Energy Harvesting Sensors," in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, Apr. 2011, pp. 277–282.

[7] M. Gorlatova, A. Wallwater, and G. Zussman, "Networking Low-Power Energy Harvesting Devices: Measurements and Algorithms," *IEEE Transactions on Mobile Computing*, vol. 12, no. 9, pp. 1853–1865, Sep. 2013.

[8] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis - CODES/ISSS '10*. New York, New York, USA: ACM Press, Oct. 2010, p. 105.

[9] A. E. Susu, A. Acquaviva, D. Atienza, and G. De Micheli, "Stochastic modeling and analysis for environmentally powered wireless sensor nodes," in *International Symposium on Modeling and Optimization in Mobile, Ad hoc and Wireless Networks and Workshops (WiOPT)*, LSI, EPFL, Lausanne. IEEE, Apr. 2008, pp. 125–134.

[10] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2nd ed. John Wiley & Sons, 2005.

[11] Moteiv, "Telos B Datasheet," p. 28, 2004. [Online]. Available: http://moss.csc.ncsu.edu/~mueller/rt/rt11/readings/projects/g4/datasheet.pdf

[12] A. Mas-Colell, M. Whinston, and J. Green, *Microeconomic Theory*. New York, NY: Oxford University Press, 1995.

[13] I. Chades, G. Chapron, M.-J. Cros, F. Garcia, and R. Sabbadin, "MDP Toolbox," 2012. [Online]. Available: http://www7.inra.fr/mia/T/MDPtoolbox/MDPtoolbox.html

[14] "CONFRRM - Cooperative Networks For Renewable Resource Measurements." [Online]. Available: http://rredc.nrel.gov/solar/new_data/confrrm/

[15] L. Huang and M. J. Neely, "Utility optimal scheduling in energy harvesting networks," in *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing - MobiHoc '11*. New York, New York, USA: ACM Press, May 2011, p. 1.

[16] S. Liu, J. Lu, Q. Wu, and Q. Qiu, "Harvesting-Aware Power Management for Real-Time Systems With Renewable Energy," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 8, pp. 1473–1486, 2012.

[17] J. Lei, R. Yates, and L. Greenstein, "A generic model for optimizing single-hop transmission policy of replenishable sensors," *Wireless Communications, IEEE Transactions on*, vol. 8, no. 2, pp. 547–551, Feb. 2009.

[18] M. A. Murtaza and M. Tahir, "Optimal data transmission and battery charging policies for solar powered sensor networks using Markov decision process," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, Apr. 2013, pp. 992–997.