Tabatabaei SM, Hakami V, Dehghan M. Cognitive power management in wireless sensor networks. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(6): 1306–1317 Nov. 2015. DOI 10.1007/s11390-015-1600-8

Cognitive Power Management in Wireless Sensor Networks

Seyed Mehdi Tabatabaei, Vesal Hakami, and Mehdi Dehghan*

Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran 16846-13114, Iran

E-mail: {tabatabaei.mehdi, vhakami, dehghan}@aut.ac.ir

Received April 24, 2014; revised June 8, 2015.

Abstract Dynamic power management (DPM) in wireless sensor nodes is a well-known technique for reducing idle energy consumption. DPM controls a node's operating mode by dynamically toggling the on/off status of its units based on predictions of event occurrences. However, since each mode change induces some overhead in its own right, guaranteeing DPM's efficiency is no mean feat in environments exhibiting non-determinism and uncertainty with unknown statistics. Our solution suite in this paper, collectively referred to as cognitive power management (CPM), is a principled attempt toward enabling DPM in statistically unknown settings and gives two different analytical guarantees. Our first design is based on learning automata and guarantees better-than-pure-chance DPM in the face of non-stationary event processes. Our second solution caters for an even more general setting in which event occurrences may take on an adversarial character. In this case, we formulate the interaction of an individual mote with its environment in terms of a repeated zero-sum game in which the node relies on a no-external-regret procedure to learn its mini-max strategies in an online fashion. We conduct numerical experiments to measure the performance of our schemes in terms of network lifetime and event loss percentage.

Keywords wireless sensor network, cognitive power management, learning automata, external regret, zero-sum game

1 Introduction

Wireless sensor networks (WSNs) are comprised of a typically large number of tiny sensor motes featuring miniature units for sensing, processing and communicating events of interest within their operating environment. Despite recent advancements in microelectronics industry have significantly paved the way for the development of low cost and low power motes, energy consumption still poses a major challenge for enabling more demanding WSN application scenarios. Indeed, the WSN lifetime highly depends on the scheme used for managing the power resources on individual motes so that ideally all units on each mote remain operational for a long time. Design-time techniques for reducing power consumption are typically hardcoded into the mote's operating system or are statically embedded within its hardware components. However, the real-world environments in which WSNs are deployed hardly remain static for long durations, and characteristically exhibit a time-varying nature. Hence, the efficient management of a mote's limited energy in such environments inevitably calls for a dynamic scheme to supplement its static built-in power reduction mechanism.

Dynamic power management (DPM) has been at the forefront of research on WSNs for many years[1–5]. The basic idea is to adjust the nodes' operational mode in accordance with the intensity of the environmental or networking events. In DPM schemes, the sensor mote can operate in a number of power consumption modes. Each mode corresponds to a different activation status of the mote's constituent units (e.g., sensing unit, processing unit, transceiver unit). Decisions for transitioning from one mode to another are to be made dynamically and whenever the node becomes idle. However, the dynamic transitions between operational modes, in turn, come at the expense of some overhead (e.g., due to saving and restoring the status registers or various circuitry start-ups). In fact, switching to a lower power mode is profitable only if the next sensing event is unlikely to happen very soon; otherwise, the transition

Regular Paper

*Corresponding Author

overhead would outweigh the saving advantage of operating in low power, because the sensor mote has to immediately switch back to high power mode to handle the new event. Therefore, DPM schemes should take into account the uncertainty associated with the occurrence of sensing events within the environment[1,3,5]. The majority of the existing schemes assume that the spatiotemporal distribution of events is known as a priori for the environment under observation. Using this distribution, the sensor mote's decision problem is then reduced to an offline prediction for the next event occurrence and to determine which operational status is likely to bring about the most energy saving. Armed with this perfect statistical knowledge, it would be possible to save power by transitioning into a deeper sleep mode when it is expected that the subsequent event is not due for quite a while or on the contrary, to stand alert when the next event is likely to occur in imminent future. There have also been a few studies[6] which aim at deploying DPM in settings with unknown or imperfect statistics; however, their results are bound to restrictive assumptions on the nature of the uncertainty underlying the WSN environment. For instance, the work in [6] assumes that event occurrences are i.i.d. and that Markov property holds for the probabilistic evolution of the system parameters.

Our work in this article, on the other hand, is motivated by the need to realize DPM under more realistic assumptions reflecting the conditions we are typically faced with in a real-world setting. In many instances of real-life sensing problems, it is hard, if not impossible, to pre-characterize the event generation process. Consider, for instance, temperature auditing in some environment[7]. The workload for the sensor node strongly depends on the stochastic dynamics of the heat source affecting the monitored environment. It also depends on the sensors' locations and similar environmental conditions which are not known at design or fabrication time. In these cases, the stochastic process describing the events is initially unknown. Even worse, the workload is subject to large variations over time. For instance, the temperature data drastically changes with the time of the day or the day of the week. This argument can be generalized into many sensing applications which are subject to the varying activity of the monitored phenomena or the transitory nature of the event source (in this case, the transient heat sources, and different responses by thermal zones to thermal conditioning system, etc.).

In our DPM solution in this paper, we generally hold no prior beliefs about the environment's behavior in terms of the process it uses to control the sensing events generation: it may be as naive as a stationary stochastic process, or as complex as strategic decisions of an adversary intending to maximize the overhead inflicted upon the choices made by DPM. More specifically, we seek to deploy distribution-free online decision making mechanisms, using which the sensor motes would try to learn and proactively adapt their transitions to different power states. The merit of a distribution-free solution lies in its robustness against model variation. In fact, even if accurate statistical modeling of the environment were possible, it generally requires rigorous analysis, mathematical formulation, and validation. Moreover, the model derived for a specific setting (e.g., temperature auditing) cannot be used in a general and broader perspective.

We brand our proposed solution suite as cognitive power management (CPM) featuring two principled schemes for managing the motes' power resources in generalized environments. Our CPM schemes are essentially online randomized algorithms which adaptively update their decisions by exploiting the performance histories associated with a mote's previous transitions to the available sleep modes; however, the type of analytical guarantees we seek to fulfill is different in each scheme.

First, assuming that event occurrences follow a general non-stationary stochastic process, we model the CPM engine on each mote as an expedient linear-reward-penalty (LRP) learning automaton operating in an s-model environment[8]. The adaptation scheme used by the automaton rapidly converges and guarantees to outperform a pure chance strategy in the limit. We numerically evaluate our learning automata-based power management scheme (LAPM) through comprehensive simulations, and investigate its convergence behavior with different learning rates. In our second scheme, we envisage the most general setting and abandon any assumption regarding the stochasticity of the event generation process. Our goal in this extreme scenario is to ensure a worst-case performance guarantee; accordingly, to provide robustness against this more general type of uncertainty, the interaction of the node with its operating environment is formulated in terms of a repeated zero-sum game, i.e., a purely adversarial setting. In order to guarantee online convergence to the mini-max strategies of the formulated game, we then propose a hedge-based power management (HPM) algorithm which is built on the notion of no-external-

regret mechanism of the online-learning literature [9]. The results of the simulation experiments reveal that our CPM suite of algorithms performs well in terms of network lifetime and event loss percentage, while working without the luxury of prior statistical knowledge.

The outline of the rest of the paper is as follows. In Section 2, we briefly review the state-of-the-art power management schemes for WSNs, and highlight their limitations. Section 3 elaborates on the system model as well as on the assumptions made for the purpose of this research. Section 4 is devoted to the presentation of our proposed algorithms for realizing cognitive power management in WSNs. Section 5 shows the experimental results. Finally, in Section 6 we conclude the paper.

2 Related Work

The lifetime of a sensor network depends highly on the specifics of the power consumption at each sensor node. A more efficient power management results in a longer network lifetime. Dynamic power management aims to reduce power consumption without adversely affecting the application's requirements. The basic idea is to shut down devices when not needed and wake them up when necessary[1]. This can be challenging because it is hard to predict when something is going to happen in the future and components do not start instantaneously but require an activation time, which depends on the components used. In the majority of the existing power management schemes, only two power modes are used in DPM (see [10] for a survey).

In the context of multi-mode DPMs, [1] is a pioneer work which provisions for dynamic transitions to five operational modes for a sensor node. More specifically, in [1], the authors proposed an OS-directed power management technique to improve the energy efficiency of the sensor nodes. Given the fact that transitions from an active to a sleep mode and vice versa have some latency and energy overhead in their own right, a threshold is computed for transitioning to each sleep mode, depending on the expected event occurrence time. When a node becomes idle, it switches to the lowest possible power mode that the probability of event occurrence during its corresponding threshold is negligible. The work in [3] comes up with more precise computation of the thresholds in [1] by also accounting for the energy expenditure due to awakening the sensor node back to the active state.

In the majority of DPM schemes[1,3,5], the probability of event occurrences in the network is bound to follow Poisson and exponential spatial and temporal distributions, respectively. There exist other studies like [11-15] which also assume a known environment but the occurrence of events is not needed to necessarily follow a Poisson or exponential distributions. For instance, [11] models the power management problem in a sensor node as an average reward Markov decision process and solves it using dynamic programming. As dynamic programming methods are model-based, they need to have access to the complete knowledge of the environment's stochastic dynamics. In the same vein, [13] presents a prediction-based dynamic energy management method to predict the mobile target's position and implements an awakening mechanism which hinges on Bayesian sampling estimations. As is well known, Bayesian-based schemes are dependent on known initial priors. Hence, the work in [13] is again restricted to statistically known environments.

In real operating environments, however, the distribution of event occurrences is unknown. Attempts have been made to implement DPM in statistically unknown environments, but their results are bound to restrictive assumptions on the nature of the uncertainty underlying the environment. For instance, in [6], a reinforcement learning-based solution is presented. It is model-free and easily applicable in both single and multi-hop scenarios. However, it is assumed in [6] that event occurrences are i.i.d. and that Markovian property holds for the probabilistic evolution of the system parameters. Moreover, given that [6] relies on R-learning for the determination of optimal policies, its convergence is not guaranteed theoretically[16].

The motivation for our work in this article is to realize dynamic power management for more realistic settings. In effect, we aim at the most general case of DPM in terms of the event generation process. Our solution is particularly suited for scenarios where zero knowledge of the event distributions and network topology is available. Event occurrences may follow an arbitrary stochastic process, be it stationary, non-stationary or even adversarial.

3 System Model and Assumptions

We assume that a typical WSN consists of nodes with sensing, processing, memory, analog/digital (AD) convertor, and transceiver units. Almost akin to the model discussed in [1], each unit is assumed to be able to operate in different power modes, e.g., the processing unit can be in active, idle, or sleep modes; likewise,

the transceiver module can be actively transmitting, idly listening, or otherwise be completely powered down (off). Thus, each sleep mode configuration for a given sensor node corresponds to a particular combination of power modes for its constituting units. Table 1 lists five different states a node can operate in, according to the directions given by its power management policy. The operational meaning associated with each state is as follows:

• *Active*: the node is busy processing an event, or has just finished processing;

• *Ready*: in this state, the processor is kept on standby so as to be "ready" in responding to a critical sensing/traffic event which is likely to occur pretty soon;

• *Monitor*: only the sensing and receiver units are left on to "monitor" ordinary sensing/traffic events which are likely to occur in the node's vicinity;

• *Observe*: in this state, the node senses its surroundings without distributing its collected data to other nodes and this provides the ability to aggregate data locally over a time period;

• *Deep Sleep*: all modules are turned off since no event is expected in the near future.

Table 1. Sensor Node Modes

Node State	Processor	Memory	Sensor A/D Convertor	Radio
s_0 (active)	Active	Active	On	Tx, Rx
s_1 (ready)	Idle	Sleep	On	Rx
s_2 (monitor)	Sleep	Sleep	On	Rx
s_3 (observe)	Sleep	Sleep	On	Off
s_4 (deep sleep)	Sleep	Sleep	Off	Off

Note: Tx stands for "transmit", and Rx for "receive".

Sleep state transitions are assumed to occur from the fully alert state s_0 directly to one of the lower power states $\{s_i\}_{i=1:4}$, and vice versa (see Fig.1). Transition decisions are made whenever the processing unit is done with its current task in s_0. While from s_0 to $\{s_i\}_{i=1:4}$, transitions are made proactively, from $\{s_i\}_{i=1:4}$ to s_0, transitions are reactive and in response to the occurrence of an event.

In the transition from the "active" mode to low power modes, the CPU performs an orderly shutdown of on-chip activity. Also, in a transition from any other state to "active" mode, the chip steps through a rather complex wake-up sequence before it can resume normal activity. As pointed out in [1], there is some overhead associated with mode switching due to: stabilizing the power supply, waiting for the phase-locked loops to lock, the clock to stabilize, and the processor context to be restored. Powering down a sensor node also gives rise to some energy overhead and extra time. Also, in general, it is the case that the deeper the sleep state, the more penalty has to be paid since it actually takes longer to enter, for instance, the "deep sleep" state than to enter the "ready" state. For example, in the three-state configuration of StrongARM SA-1100 processor[17], the time for exiting "sleep" is much longer than that for exiting "idle" (10 μs versus 160 ms). On the other hand, the power consumed by the chip in "sleep" mode (0.16 mW) is much smaller than that in "idle" mode (50 mW). In the rest of this section, we derive closed-form energy cost formula which, given a particular event occurrence time t_e, can express the total overhead incurred by a transition decision. The symbols used throughout this section are listed in Table 2.

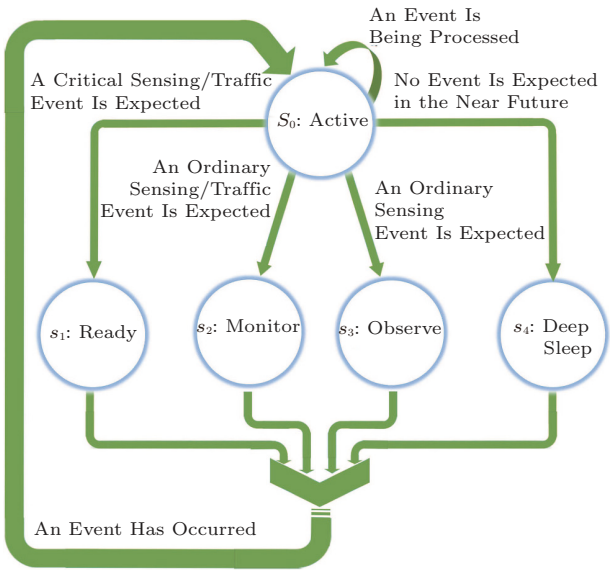

Fig.1. Sensor node state diagram.

Table 2. Symbols and Definition

Symbol	Definition
t_e	Time until next event occurs
$\tau_{\text{down},i}$	Transition time from "active" to state i
$\tau_{\text{up},i}$	Transition time from state i to "active"
P_{active}	Power consumption in state s_0
$P_{\text{sleep},i}$	Power consumption in modes $s_1 \sim s_4$

The net energy consumed by a transition from s_0 to $\{s_i\}_{i=1:4}$, depends on whether the next sensing event occurs in the midst of the transition, i.e., while it is in progress, or otherwise afterwards. Fig.2(a) depicts the case when $t_e > \tau_{\text{down},i}$. The hatched area in the figure

is the total power consumed when the transition decision has been made until the moment the node reverts from s_i to the fully operational mode s_0. We denote the total energy consumption in this case by PwC_1, and it can be calculated by the following equation:

$$PwC_1 = \frac{[P_{\text{active}} + P_{\text{sleep},i}]}{2} \times (\tau_{\text{down},i} + \tau_{\text{up},i}) + P_{\text{sleep},i} \times (t_{\text{e}} - \tau_{\text{down},i}).$$

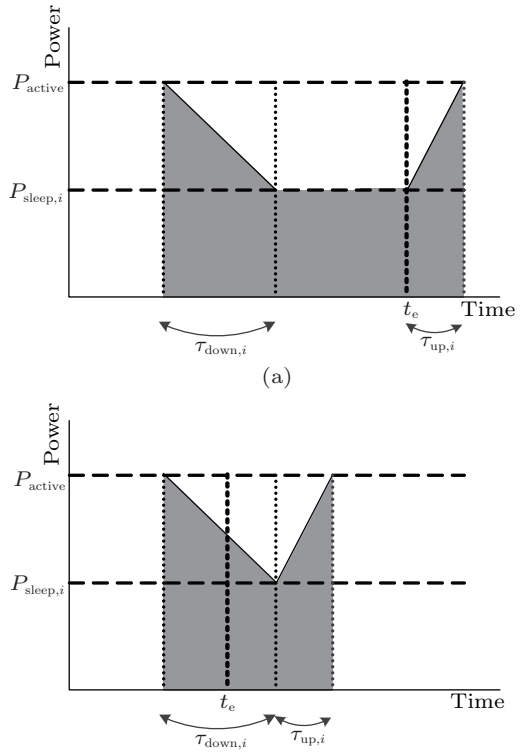

Fig.2. Transition latency and power consumption. (a) $t_{\text{e}} > \tau_{\text{down},i}$. (b) $t_{\text{e}} < \tau_{\text{down},i}$.

In case $t_{\text{e}} < \tau_{\text{down},i}$, however, since no productive work can be done in the transition period, it is assumed that the node first completes its current transition to $\{s_i\}_{i=1:4}$ and then immediately starts ascending back to s_0 (see Fig.2(b)). We denote the total energy consumption in this case by PwC_2 which is given by (1):

$$PwC_2 = \frac{[P_{\text{active}} + P_{\text{sleep},i}]}{2} \times (\tau_{\text{down},i} + \tau_{\text{up},i}). \quad (1)$$

Finally, (2) describes the energy consumption in the above two cases in a compact form:

$$PwC(s_i, t_{\text{e}}) = \mathbb{I}(t_{\text{e}} > \tau_{\text{down},i}) \times PwC_1(s_i, t_{\text{e}}) + \mathbb{I}(t_{\text{e}} < \tau_{\text{down},i}) \times PwC_2(s_i, t_{\text{e}}), \quad (2)$$

where $\mathbb{I}(.)$ is an indicator function which is equal to 1 if its argument is true and 0 otherwise.

4 Cognitive Power Management

In this section, we present the details of our CPM suite of protocols for proactive power adaptation in an unknown non-deterministic sensing environment. Our general approach is to deploy online adaptive procedures for learning to behave efficiently through repeated interactions with the environment. A reasonable choice for a learning model is the one that works within the confines of bounded rationality so that it does not impose too much computational burden on the extremely resource-constrained sensor nodes. Depending on the nature of non-determinism, we envision two sensing scenarios and propose solutions specific to each scenario. In Subsection 4.1, we present our LAPM algorithm for managing power mode transitions in non-stationary sensing environments. In Subsection 4.2, we introduce HPM for robust power management in the more general case of adversarial environments.

4.1 Power Management in Non-Stationary Settings

In case the event generation process in the WSN environment exhibits a general non-stationary behavior with unknown time-varying distributions, we resort to the learning automata (LA) toolbox for enabling our notion of cognitive power management. A learning automaton is a simple learning unit which, through repeated interactions with its environment, would ultimately find out how to adaptively respond to the changing conditions affecting its performance measure.

The type of analytical guarantee given by a learning automaton is commonly referred to as expediency in LA parlance[18]. More specifically, an expedient LA is the one that outperforms a pure-chance automaton, i.e., a simple automaton which always goes for uniform randomization over its action choices. To tackle the non-stationarity of the event generation process, we will use the well-known linear reward-penalty (L_{RP}) adaptation scheme as the LA's updating rule, which is a simple undemanding reinforcement procedure with a constant learning rate.

Our rationale for choosing L_{RP} lies in its capability for actively tracking the expedient strategy in the face of time-varying system parameters (t_{e} in our context). Also, given a node's current remaining battery level, we would be able to normalize the per-stage energy cost PwC^n incurred by the LA's current action

choice (see (2)) to obtain a continuous-valued environmental response within the interval [0, 1]. The overall setting would then comply with a standard $\mathrm{L_{RP}}$ LA interacting with an s-model environment[8].

Hence, we formalize our LA-based power management (LAPM) solution by defining an $\mathrm{sL_{RP}}$ LA in terms of the quadruple (A, B, P, U):

- $A = \{s_i\}_{i=0}^{4}$: the set of sleep modes;
- $B = \{b_1, b_2, \ldots\}$: the set of environmental responses (normalized per-stage energy costs);
- $P = \{p(s_i)\}_{i=0}^{4}$: a probability distribution over action choices (the set of sleep modes);
- U : the $\mathrm{sL_{RP}}$ update rule:

$$p^{n+1}(s) = \begin{cases} p^n(s) + \alpha(1 - b^n(s))(1 - p^n(s)) - \\ \alpha b^n(s)\, p^n(s), \quad \text{if} \quad s = s^n, & (3) \\ p^n(s) - \alpha(1 - b^n(s))(p^n(s)) + \\ \alpha b^n(s)\left(\dfrac{1}{1 - |A|} - p^n(s)\right) - \\ \alpha(1 - b^n(s))\, p^n(s), \quad \text{if} \quad s \neq s^n, & (4) \end{cases}$$

where $0 < \alpha < 1$ is a constant learning rate. Our LA-based routine for power management is summarized in Algorithm 1. Table 3 lists the symbols used in the pseudo-code along with their definitions.

Algorithm 1. LA-Based Power Management (LAPM)

Initialization: $p^0(s) = 1/|A|, \forall s \in A; \lambda_i = 0$;
$eventCount = 0; t = 0; n = 1$;
begin
  case (event) **do**
    idle:
01: Choose $s^n = s$ with probability: $p^n(s)$;
02: **if** $(s = s_4 \& eventCount \neq 0)$
03: $t_{s_4} = -\dfrac{1}{\hat{\lambda}_i^n} \ln(p^n(s_4))$;
04: *set_timer*(t_{s_4});
05: **end if**
06: Switch to sleep mode s;
    sensing_event_occured:
07: Wake up;
08: $eventCount++$;
09: $e^n = e^{(n-1)} - PwC(s)$;
10: $b^n(s) = \frac{PwC(s)}{e^n}$;
11: **for** $(\forall \hat{s} \in A)$
12: **if** $(\hat{s} = s)$
13: Calculate $p^{n+1}(\hat{s})$ as in (3);
14: **else**
15: Calculate $p^{n+1}(\hat{s})$ as in (4);
16: **end if**
17: **end for**
18: $t = t + \mathbb{I}(t_e > \tau_{s,a}) \times (t_e + \tau_{s,u}) + \mathbb{I}(t_e < \tau_{s,d}) \times (\tau_{s,d} + \tau_{s,u})$;
19: $\hat{\lambda}_i^{n+1} = \gamma \hat{\lambda}_i^n + (1 - \gamma) \frac{\text{number of events}}{t}$;
20: $n = n + 1$; /* update the time index. */
    *timer_*t_{s_4}*_expired:*
21: Wake up;
end

Table 3. Symbols and Definitions for LAPM

Symbol	Definition
$b^n(s)$	Normalized power consumption of node i for selecting sleep mode s in the n-th iteration
$p^n(s)$	Probability of switching to mode s in the n-th iteration
s^n	Selected sleep mode in the n-th iteration
N	Time horizon
t_{s_4}	Time duration node i has spent in mode s_4
λ_i	Average event intensity estimated by node i
p_{s_4}	Probability of entering sleep mode s_4
t	Total time elapsed
e^n	Remaining energy in the n-th iteration

In the n-th iteration of LAPM algorithm, the idle sensor node switches to the sleep mode $s^n = s$ with probability $p^n(s)$. Immediately after the occurrence of the next event, t_e can be plugged into (2) in order to calculate the energy cost PwC^n for current choice $s^n = s$. Given the node's current battery level e^n, PwC^n values would be normalized into [0, 1] which can be considered as the environment's response b^n to LA. The switching probabilities would then be updated in proportion to the feedback obtained from the previous choices and according to the rule given in (3) and (4).

There is a subtlety with the case when the action drawn from p^n turns out to be s_4. In this case, the mote switches off all its units. Allowing for the mote to get totally powered down is only a viable option for uncritical monitoring tasks with non-zero tolerance of event loss. In such scenarios, one can set a timer to automatically turn on the mote once some pre-specified time period elapses. We set the wake-up timer for the deepest sleep mode using (5) below:

$$t_{s_4} = -\frac{1}{\hat{\lambda}_i^n} \ln(p^n(s_4)), \quad (5)$$

where $p^n(s_4)$ is the current probability with which a node i goes to the deepest sleep mode, and $\hat{\lambda}_i^n$ is the current estimate for the local intensity of the sensing events in node i. As a new event occurs, $\hat{\lambda}_i^n$ can be updated using a basic linear recursive filter mechanism with weighting factor γ:

$$\hat{\lambda}_i^{n+1} = \gamma \hat{\lambda}_i^n + (1 - \gamma) \times \frac{\text{number of events}}{\text{total elapsed time}}.$$

4.2 Power Management in Adversarial Settings

In this subsection, we generalize our CPM solution to also account for scenarios where no particular assumption can be made whatsoever on the uncertainty associated with the specifics of event occurrences. Within this general setting, one can take on

a distribution-free robust optimization perspective and describe the interaction of a mote with its environment as a repeated two-player zero-sum game. More specifically, we intend to come up with a decision making scheme which is capable of performing optimally in a DPM setting in which the event occurrence times may take on an adversarial (worst-case) character. Accordingly, the analytical guarantee we seek to fulfill would be to converge, in an online fashion, to the mini-max strategy of the mote in the formulated game, i.e., a strategy which is guaranteed to minimize the cumulative worst-case energy consumption of the mote in the long run. To this end, we propose an online adaptive DPM scheme by resorting to the theoretical results in adversarial learning literature. In particular, we build on the well-known Hedge algorithm for regret minimization[9] which has been shown to converge to a (possibly randomized) strategy corresponding to that of the mini-max equilibria of a repeated zero-sum game. Our power management scheme, referred to as Hedge-based power management (HPM), is particularly suited for zero-knowledge scenarios we deal with when sensor motes are deployed in an unknown hostile environment. In what follows, we first present our game-theoretic formulation of the CPM problem in Subsection 4.2.1 and then go on to describe the details of our HPM algorithm in Subsection 4.2.2.

4.2.1 Repeated Zero-Sum Game Formulation

In the absence of any prior statistical knowledge, we take on a robust optimization approach to the CPM problem and formulate the interaction of a mote with its environment in terms of a repeated zero-sum game.

Players. On one side of the game resides a sensor mote as a decision making entity armed with an online learning procedure and on the other side lies its adversary, i.e., the unknown stochastic process governing the sensing event generations within the environment.

Strategies. The action choices for a node are denoted by the set $\mathcal{S} = \{s_i\}_{i=0:4}$ of sleep modes. The adversary's action, on the other hand, is to determine the occurrence time for the next sensing event. We denote the set of all possible adversarial outcomes by $\mathcal{T}$ which is assumed to be a compact set. At stage n of the CPM game, the occurrence time $t_{\mathrm{e}}^n \in \mathcal{T}$ corresponding to the next event may follow an arbitrary stochastic process, be it stationary, non-stationary or even adversarial. We denote a randomized strategy of the sensor node by $p \in \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ is the set of all probability distributions over the set $\mathcal{S}$ of actions. Likewise, $q \in \Delta(\mathcal{T})$ denotes the strategy used by the adversary which is assumed to be a probability density function over $\mathcal{T}$.

Loss Functions. A sensor node's loss can be defined in terms of a vector-valued function $\ell^n = \left[\frac{PwC(s_i,t_{\mathrm{e}}^n)}{e_i^{n-1}}\right]_{i=0:4}$, whose coordinates are of the form $\ell_i^n : \mathcal{S} \times \mathcal{T} \to [0,1]$ $(i = 0,\ldots,4)$. Each function ℓ_i^n expresses the energy loss due to transitioning into sleep mode $s_i \in \mathcal{S}$ when the next event occurs t_{e}^n time units past the start of the transition. The denominator e_i^{n-1} in this formula is the available mote's energy right before the n-th transition, which ensures the loss values to fall within $[0,1]$. Therefore, the loss experienced by a node depends both on its choice of a sleep mode and on the adversary's strategy for determining the occurrence time of the next sensing event. Given the zero-sum nature of the game, the adversary's loss at time step n is exactly $-\ell^n$.

Strategic Objective. A randomized strategy p of a sensor node is a mini-max strategy in the CPM game if the following condition holds

$$p = \arg\min_{p}\max_{q}\sum_{s\in\mathcal{S}} p(s)\int_{t_{\mathrm{e}}\in\mathcal{T}} q(t_{\mathrm{e}})\times\ell(s,t_{\mathrm{e}}). \quad (6)$$

With (6) satisfied, the sensor node adopting the strategy p would play optimally against an adversary which maximizes its loss. As a side note, it is worth mentioning that a mini-max strategy profile in two player zero-sum games coincides with Nash equilibrium (NE); hence, for every alternative $p' \in \Delta(\mathcal{S})$ and $q' \in \Delta(\mathcal{T})$, the mini-max strategy profile (p,q) satisfying (6) meets the following NE condition as well,

$$\begin{aligned}&\sum_{s\in\mathcal{S}} p(s)\int_{t_{\mathrm{e}}\in\mathcal{T}} q'(t_{\mathrm{e}})\times\ell(s,t_{\mathrm{e}})\\ \leqslant\ &\sum_{s\in\mathcal{S}} p(s)\int_{t_{\mathrm{e}}\in\mathcal{T}} q(t_{\mathrm{e}})\times\ell(s,t_{\mathrm{e}})\\ \leqslant\ &\sum_{s\in\mathcal{S}} p'(s)\int_{t_{\mathrm{e}}\in\mathcal{T}} q(t_{\mathrm{e}})\times\ell(s,t_{\mathrm{e}}).\end{aligned}$$

4.2.2 Hedge-Based Power Management

In the n-th iteration of the CPM game, the play proceeds as follows: the idle node chooses to switch into a sleep mode $s^n \in \mathcal{S}$, and at the same time, the adversary determines the occurrence time for the next event t_{e}^n. Upon the actual occurrence of the next event, the mote begins to revert back to the fully operational mode, and can also plug the now-determined value of t_{e}^n into (2) in order to calculate the vector of its instantaneous loss ℓ^n. In effect, given that all that is required for calculating loss values is the occurrence time t_{e}^n of the next event, the loss function can be evaluated both for the actual choice s^n made at time step n as well as for all

its alternative choices $\mathcal{S}\backslash\{s^n\}$. Given that no a priori knowledge is available regarding the decision rule used by the adversary in the sensing environment, the sensor node must rely on an online learning procedure which, through repeated interactions, ultimately leads to the emergence of its mini-max strategy in the CPM game. It is a well-known theoretical result in the adversarial learning literature that the adaptive schemes minimizing a specific measure of the repeated zero-sum game, the so-called"external regret" criterion, are capable of almost sure convergence to the mini-max strategies of the play. The "external regret" criterion measures the difference between the average loss actually experienced by an online learning algorithm and the average loss that would have been incurred by the player's best fixed choice in hindsight. Accordingly, a decision rule is "no-regret" if, given the history of the play, its average loss in the limit is no more than the average loss that would have been incurred by each of its choices had they been chosen constantly across the entire history of the game. More formally, let $\mathcal{H}^n \triangleq \mathcal{S}^{n-1} \times \mathcal{T}^{n-1}$ be the set of all play histories up until time n. In the n-th decision period, the sensor node chooses the sleep mode $s^n \in \mathcal{S}$ according to the (mixed) strategy $p^n : \mathcal{H}^n \to \Delta(\mathcal{S})$; likewise, the adversary comes up with a $t_{\mathrm{e}} \in \mathcal{T}$ following an unknown rule $q^n : \mathcal{H}^n \to \Delta(\mathcal{T})$. An online learning algorithm has "no-external regret" if the strategy sequence $\hat{p}^n$ prescribed by the algorithm satisfies the following stochastic convergence condition:

$$\lim_{N\to\infty} \sup \frac{1}{N}\Bigg(\mathbb{E}\left[\sum_{n=1}^{N} l\left(\hat{p}^n, t_{\mathrm{e}}^n\right)\right] - \min_{s\in\mathcal{S}} \sum_{n=1}^{N} l\left(s^n, t_{\mathrm{e}}^n\right)\Bigg) \leqslant 0.$$

Our CPM solution for the adversarial sensing environment scenario is founded on the following key theoretical result (see [9]).

Theorem 1. *The average loss actually experienced by a "no-external regret" algorithm in a repeated zero-sum game is upper bounded by the mini-max value of the game, i.e.,*

$$\lim_{N\to\infty} \sup \frac{1}{N} \sum_{n=1}^{N} l\left(\hat{s}^n, t_{\mathrm{e}}^n\right) \leqslant \min_{p} \max_{q} \sum_{s\in\mathcal{S}} p(s) \int_{t_{\mathrm{e}}\in\mathcal{T}} q(t_{\mathrm{e}}) \times \ell(s, t_{\mathrm{e}}) \,\mathrm{w.p.\,1},$$

where $\hat{s}^n \sim \hat{p}^n$ (i.e., $\hat{s}^n$ is drawn from the probability distribution $\hat{p}^n$).

The underlying mechanics in "no-external regret" algorithms is essentially to choose an action in each iteration with a probability proportional to the cumulative loss incurred by that action over the previous plays. In this article, we draw on Hedge algorithm[9], a simple yet one of the most celebrated "no-external regret" procedures, to cognitively manage the node's power consumption in an adversarial setting. At the heart of Hedge lies the following polynomial-weight updating rule which dynamically adjusts the weights on the actions at each stage to account for the received feedback:

$$w_i^{n+1} = w_i^n \left(1 - \eta \ell_i^n\right), \quad i = 0, \ldots, 4,$$

where w_i^n is the weight assigned to each action $s_i \in \mathcal{S}$ at stage n of the CPM game and η is a constant parameter typically determined by the number of actions $|\mathcal{S}|$ and the intended time horizon N. According to [9], one appropriate choice for η is $\eta = \min\left\{\sqrt{\frac{\log|\mathcal{S}|}{N}}, \frac{1}{2}\right\}$. By letting $p_i^n = \frac{w_i^n}{W^n}$ such that $W^n = \sum_{j=0}^{4} w_j^n$, the above update rule sets $W^{n+1} = W^n - \sum_{j=0}^{4} \eta w_j^n \ell_j^n$, where $\frac{\sum_{j=0}^{4} w_j^n \ell_j^n}{W^n}$ can be interpreted as the expected loss to be incurred by Hedge at time n. Theorem 2 expresses the theoretical guarantee given by a Hedge-based online learning procedure.

Theorem 2. *Within a finite time horizon N and for any $[0,1]$-valued loss sequence, the difference between the cumulative loss $\sum_{n=1}^{N} \ell_H^n$ actually incurred by HPM with $\eta = \min\left\{\sqrt{\frac{\log|\mathcal{S}|}{N}}, \frac{1}{2}\right\}$ and that of the best fixed choice in hindsight is upper bounded by $2\sqrt{N \log|\mathcal{S}|}$, i.e., the regret grows sub-linearly by the number of iterations.*

Our Hedge-based routine for power management is summarized in Algorithm 2. Table 4 lists the symbols used in the pseudo-code along with their definitions. As a final note, it is worth mentioning that the special case $s^n = s_4$ is handled exactly in the same way as our LAPM algorithm in Subsection 4.1.

Table 4. Symbols and Definitions for HPM

Symbol	Definition
ℓ_i^n	Normalized power consumption of node i
w_s^n	Weight of sleep mode s in the n-th iteration
p_s^n	Probability of sleep mode s
t_{s_4}	Time duration of node i in mode s_4
λ_i	Average arrival rate for node i
p_{s_4}	Probability of node i for entering s_4
t	Total time elapsed
e^n	Remaining energy in the n-th iteration
η	Constant factor for the HPM algorithm

Algorithm 2. Hedge-Based Power Management (HPM)

Initialization: $\eta = \min\left\{\sqrt{\frac{\ln|\mathcal{S}_i|}{T}}, 1/2\right\}; w_i^1 = 1; p_i^1 = \frac{1}{|\mathcal{S}_i|}$;
$\lambda_i = 0; eventCount = 0; t = 0; n = 1;$
begin
  case (event) **do**
    idle:
01: Choose $s^n = s$ with probability: $p_s^n = w_s^n / \sum_s w_s^n$;
02: **if** ($s = s_4$ & $eventCount \neq 0$)
03: $t_{s_4} = -\frac{1}{\hat{\lambda}_i^n} \ln(p^n(s_4))$;
04: *Set_timer* (t_{s_4});
05: **end if**
06: Switch to sleep mode s;
    sensing_event_occured:
07: Wake up;
08: $eventCount++$;
09: $e^n = e^{n-1} - PwC(s)$;
10: **for** $(\forall \hat{s} \in s_i)$
11: $l_i^n = (PwC(\hat{s}))/e^n$;
12: $w_{\hat{s}}^{(n+1)} = w_{\hat{s}}^n(1 - \eta l_{\hat{s}}^n)$;
13: **end for**
14: $t = t + \mathbb{I}(t_e > \tau_{s,a}) \times (t_e + \tau_{s,u}) + \mathbb{I}(t_e < \tau_{s,d}) \times (\tau_{s,d} + \tau_{s,u})$;
15: $\hat{\lambda}_i^{n+1} = \gamma\hat{\lambda}_i^n + (1-\gamma) \times \frac{\text{number of events}}{t}$;
16: $n = n + 1$; /* update the time index. */
    timer_t_{s_4}_expired:
17: Wake up;
end

5 Performance Evaluation

In this section, we do extensive simulations for evaluating the performance of our CPM suite of solutions. We organize our discussion of the results into two subsections: node-level experiments, and network-wide experiments.

5.1 Node-Level Experiments

In our simulated setting, the sensing events occur according to a Markov-modulated Poisson process (MMPP). MMPP is basically a Poisson process whose intensity parameter varies with time according to a Markov chain. Throughout the experiments, the MMPP that governs the generation of the sensing events switches between five Poisson processes whose rates are given in Table 5. The sensor nodes in our experimental setting are assumed to feature five operational modes. The transition time to and from each mode together with their associated power levels is listed in Table 5. Our setup is in line with the standard configuration envisaged in [1], which is in turn based on the data-sheet specifications for the commonly adopted sensing, communication, and processing platforms. For instance, the energy dissipation model for the processer unit is based on StrongARM family of microprocessors. The same settings have also been adopted in similar studies on DPM, e.g., in [19].

Table 5. Simulation Settings

Sensor Modes	p_{s_i}(mW)	τ_d (ms)	τ_u (ms)
s_0	1 040		
s_1	400	5	5
s_2	270	15	15
s_3	200	20	20
s_4	10	50	50

Note: MMPP: $\lambda_1 = 5$, $\lambda_2 = 15$, $\lambda_3 = 45$, $\lambda_4 = 140$, $\lambda_5 = 200$.

We evaluate our algorithms in terms of: 1) instantaneous and cumulative energy expenditures, and 2) event loss. Based on these two criteria, our HPM and LAPM algorithms will be compared against their theoretical benchmarks (i.e., the best fixed choice in case of HPM and pure-chance automaton for LAPM).

In the first experiment, the instantaneous values for HPM's energy consumption are reported for the first 50 seconds of the simulation. The results are demonstrated in Fig.3. As it turns out, the realized occurrence time for the sensing events in this experiment is so that the best fixed choice for the sensor node, when looked upon in retrospect, happens to be constantly operating in sleep mode s_2. As can be seen, at almost the 30th second, HPM begins to closely track s_2's performance, and even subsequently excels it, a behavior which corroborates the theoretical discussion in Subsection 4.2.2.

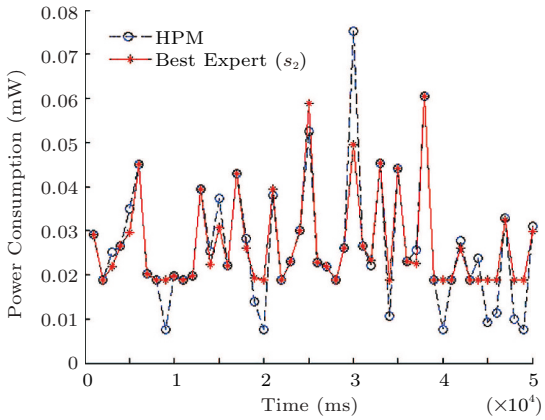

Fig.3. HPM vs theoretical benchmark: normalized power consumption.

In Fig.4, HPM's cumulative energy consumption for a 10-second simulation period is contrasted against the corresponding values obtained from constantly being in either one of modes $s_0 \sim s_4$ over the course of the same period. As can be seen, HPM's cumulative consumption is very close to that of s_3 (i.e., the best fixed choice in hindsight).

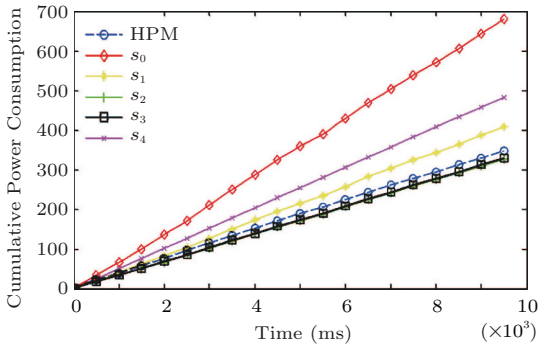

Fig.4. HPM vs theoretical benchmark: cumulative power consumption.

The next experiment is conducted to verify if the theoretical expediency guarantee holds for our LAPM algorithm. More specifically, we compare the LAPM's cumulative energy consumption with that resultant from a pure-chance automaton. As shown in Fig.5, over the course of a 50-second simulation period, LAPM outperforms pure-chance automaton and hence the empirical results corroborate with the theoretical guarantee of expediency.

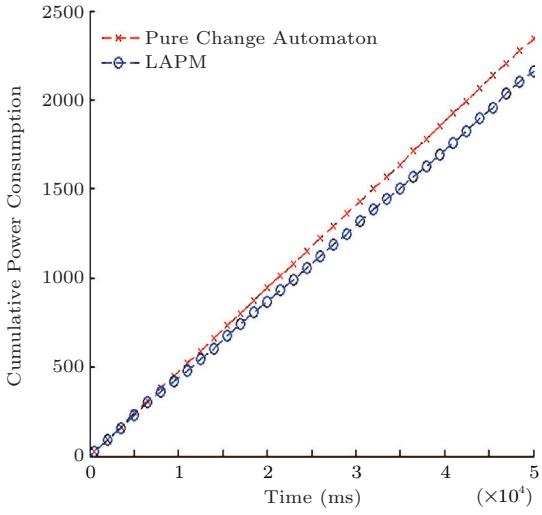

Fig.5. LAPM vs theoretical benchmark: cumulative power consumption.

Figs.6 and 7 show how LAPM's transient and asymptotic behavior can be influenced by the values of its learning rate parameter. Since LAPM makes use of a constant learning rate, smaller values for this parameter result in slower convergence, that is, power consumption is relatively higher in early iterations due to suboptimal choices made before the algorithm actually converges. However, once LAPM converges under a smaller learning rate, its asymptotic performance would excel its counterpart with a larger learning rate[20].

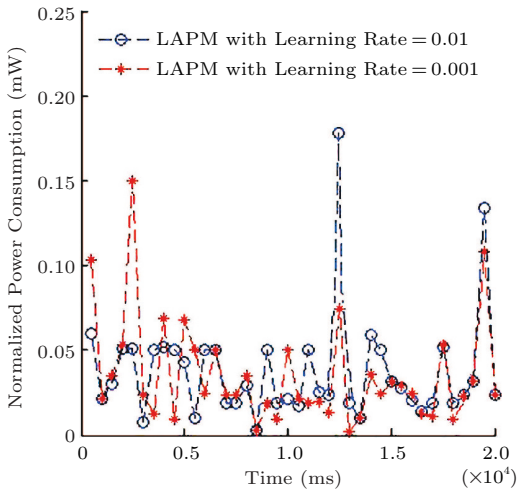

Fig.6. LAPM's power consumption under two learning rates.

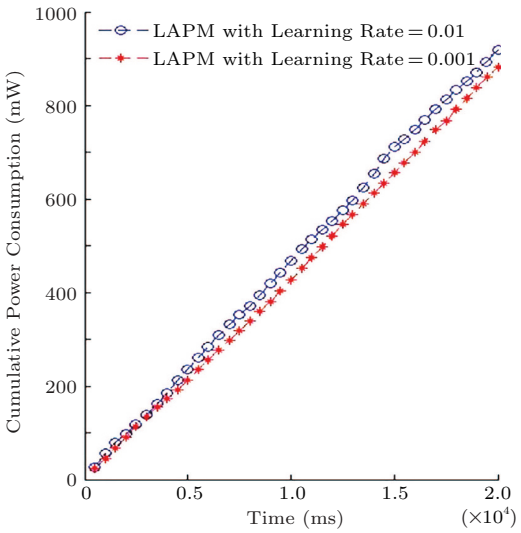

Fig.7. LAPM's cumulative power consumption with two learning rates.

Our last node-level experiment is dedicated to the comparison of HPM and LAPM in terms of the percentage of event losses. As already discussed in Subsection 4.1, an event loss takes place only when a sensor mode is in its deepest sleep mode and the next event occurs before the wake-up timer fires. The results reported in Table 6 are obtained from running these three algorithms under identical simulation settings. As can be seen, HPM's conservative mini-max strategy turns out to outperform its other two counterparts.

Table 6. Average Percentage of Missed Events

Algorithm	Average Percentage (%)
HPM	9
LAPM	12

5.2 Network-Wide Experiments

We have simulated a connected network of 100 homologous motes distributed uniformly across a square area of 10 m×10 m. The sensing range for each mote is assumed to be 1 m. Also, the nodes' initial energy is taken to be 1 J. In the simulated setting for this subsection, the spatial distribution of the sensing events is again assumed to be non-stationary, following an MMPP with intensity rates as listed in Table 5. Depending on the whereabouts of a given event, it may be processed by one or more sensor nodes.

In the first experiment, the percentage of the nodes alive in a network running HPM is compared against the constant choice of s_2 as the best fixed choice in hindsight. As demonstrated in Fig.8, HPM closely tracks s_2 and in both cases, the first node dies out in the 1 000th iteration. The next experiment is conducted to demonstrate how HPM's performance differs from the best fixed choice with respect to the network-wide spatial distribution of power consumption. As can be seen in Fig.9, the discrepancy between HPM and its theoretical benchmark is negligible.

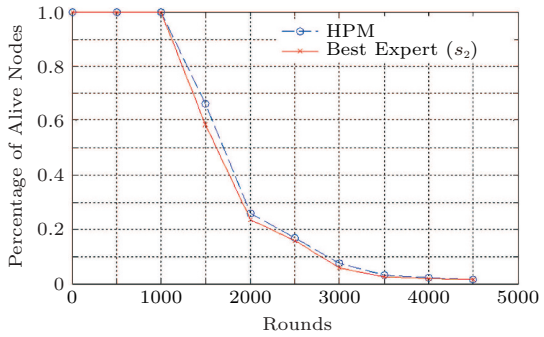

Fig.8. HPM vs theoretical benchmark: percentage of alive nodes.

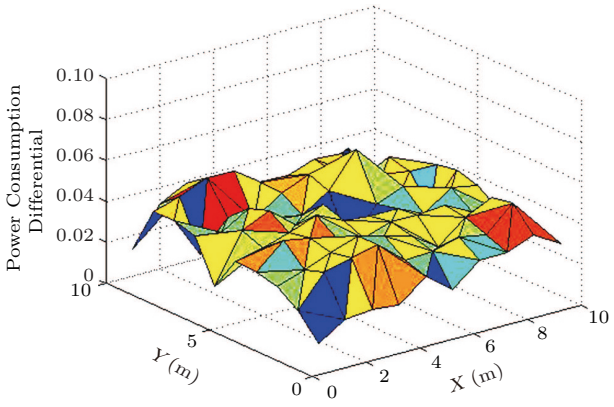

Fig.9. Discrepancy between HPM and it theoretical benchmark in terms of power consumption.

6 Conclusions

In this paper, we proposed a novel power management solution for wireless sensor networks which is particularly suited for operation in unknown environments with zero design-time knowledge. In particular, we came up with two cognitive schemes which are built on the theoretical results in stochastic and online learning literature. Our learning automata-based solution is suited for non-stationary sensing environments, while our no-external-regret scheme is specialized for adversarial settings. Unlike prior art, our solutions can guarantee reasonable performance in terms of power consumption and event loss ratio even when no knowledge is available regarding the spatiotemporal distribution of event occurrences. Individual node-level and network-wide experiments corroborated the theoretical results. The minimal computational requirement of our proposed schemes is also a merit when it comes to the deployment in real life settings.

References

[1] Sinha A, Chandrakasan A. Dynamic power management in wireless sensor networks. *IEEE Design & Test of Computers*, 2001, 18(2): 62-74.

[2] Lou R C, Tu L C, Chen O. An efficient dynamic power management policy on sensor network. In *Proc. the 19th Int. Conf. Advanced Information Networking and Applications*, March 2005, pp.341-344.

[3] Lin C, Xiong N, Park J H, Kim T. Dynamic power management in new architecture of wireless sensor networks. *International Journal of Communication Systems*, 2009, 22(6): 671-693.

[4] Kianpisheh S, Charkari N M. Dynamic power management for sensor node in WSN using average reward MDP. In *Proc. the 4th WASA*, Aug. 2009, pp.53-61.

[5] Lou R C, Chen O. Mobile sensor node deployment and asynchronous power management for wireless sensor networks. *IEEE Transactions on Industrial Electronics*, 2012, 59(5): 2377-2385.

[6] Kianpisheh S, Charkari N M. A new approach for power management in sensor node based on reinforcement learning. In *Proc. International Symposium on Computer Networks and Distributed Systems* (*CNDS*), Feb. 2011, pp.158-163.

[7] Fang L, Dobson S. In-network sensor data modelling methods for fault detection. In *Evolving Ambient Intelligence*, O'Grady M J, Vahdat-Nejad H, Wolf K *et al.* (eds.), Springer International Publishing, 2013, pp.176-189.

[8] Narendra K S, Thathachar M A L. Learning automata a survey. *IEEE Transactions on Systems, Man and Cybernetics*, 1974, 4(4): 323-334.

[9] Blum A, Mansour Y. Learning, regret minimization, and equilibria. In *Algorithmic Game Theory*, Nisan N, Roughgarden T, Tardos E, Vazirani V (eds.), Cambridge University Press, 2007.

[10] Wang L, Xiao Y. A survey of energy-efficient scheduling mechanisms in sensor networks. *Mobile Networks and Applications*, 2006, 11(5): 723-740.

[11] Kianpisheh S, Charkari N M. A power control mechanism for sensor node based on dynamic programming. In *Proc. the 2nd International Conference on Communication Software and Networks*, Feb. 2010, pp.114-118.

[12] Fallahi A, Hossain E. A dynamic programming approach for QoS-aware power management in wireless video sensor networks. *IEEE Transactions on Vehicular Technology*, 2009, 58(2): 843-854.

[13] Wang X, Ma J, Wang S, Bi D. Prediction-based dynamic energy management in wireless sensor networks. *Sensors*, 2007, 7(3): 251-266.

[14] Durand J B, Girard S, Civiza V *et al.* Optimization of power consumption and device availability based on point process modelling of the request sequence. *Journal of the Royal Statistical Society: Series C* (*Applied Statistics*), 2013, 62(2): 151-165.

[15] Jung H, Pedram M. Supervised learning based power management for multicore processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2010, 29(9): 1395-1408.

[16] Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge, Massachusetts, London, England: MIT Press, 1998.

[17] Benini L, Bogliolo A, De Micheli G. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, 2000, 8(3): 299-316.

[18] Vrancx P. Decentralised reinforcement learning in Markov games [Ph.D. Dissertation]. Department of Computer Science, Vrije Universiteit Brussel, 2010.

[19] Wang X, Ma J, Wang S. Collaborative deployment optimization and dynamic power management in wireless sensor networks. In *Proc. the 5th Grid and Cooperative Computing*, October 2006, pp.121-128.

[20] Meybodi M R, Beigy H. A note on learning automata based schemes for adaptation of BP parameters. *Neurocomputing*, 2002, 48(1/2/3/4): 957-974.

Seyed Mehdi Tabatabaei obtained his M.S. degree in information technology from Amirkabir University of Technology (Tehran Polytechnic), Tehran, in 2014. He is a member of Wireless Networks Laboratory at Amirkabir University of Technology, Tehran. His current research mainly focuses on energy saving in wireless networks.

Vesal Hakami received his B.S. degree in computer engineering (software) and his M.S. and Ph.D. degrees in information technology (computer science) from Amirkabir University of Technology, Tehran, in 2004, 2008 and 2015, respectively. His current research mainly focuses on cognitive control of wireless networks using stochastic control theory, and distributed strategic learning.

Mehdi Dehghan received his B.S. degree in computer engineering from Iran University of Science and Technology (IUST), Tehran, in 1992, and M.S. and Ph.D. degrees from Amirkabir University of Technology (AUT), Tehran, in 1995 and 2001, respectively. He is an associate professor of computer engineering and information technology at Amirkabir University of Technology (AUT). Prior to joining AUT in 2004, he was a research scientist at Iran Telecommunication Research Center (ITRC) working in the area of quality-of-service provisioning and network management. His research interests are in wireless networks, pattern recognition, fault-tolerant computing, and distributed systems.