A Self-Adaptive Sleep/Wake-Up Scheduling Approach for Wireless Sensor Networks

Dayong Ye and Minjie Zhang

***Abstract*—Sleep/wake-up scheduling is one of the fundamental problems in wireless sensor networks, since the energy of sensor nodes is limited and they are usually unrechargeable. The purpose of sleep/wake-up scheduling is to save the energy of each node by keeping nodes in sleep mode as long as possible (without sacrificing packet delivery efficiency) and thereby maximizing their lifetime. In this paper, a self-adaptive sleep/wake-up scheduling approach is proposed. Unlike most existing studies that use the duty cycling technique, which incurs a tradeoff between packet delivery delay and energy saving, the proposed approach, which does not us duty cycling, avoids such a tradeoff. The proposed approach, based on the reinforcement learning technique, enables each node to autonomously decide its own operation mode (sleep, listen, or transmission) in each time slot in a decentralized manner. Simulation results demonstrate the good performance of the proposed approach in various circumstances.**

***Index Terms*—Self-adaptation, sleep/wake-up scheduling, wireless sensor networks (WSNs).**

I. Introduction

DUE TO recent technological advances, the manufacturing of small, low power, low cost and highly integrated sensors has become technically and economically feasible. These sensors are generally equipped with sensing, data processing and communication components. Such sensors can be used to measure conditions in the environment surrounding them and then transform these measurements into signals. The signals can be processed further to reveal properties about objects located in the vicinity of the sensors. The sensors then send these data, usually via a radio transmitter, to a command center (also known as a "sink" or a "base station") either directly or via several relaying sensors [1]. A large number of these sensors can be networked in many applications that require unattended operation, hence producing a wireless sensor network (WSN). Currently, there are various applications of WSNs, including target tracking [2], health care [3], data collection [4], security surveillance [5], [6], and distributed computing [7], [8].

Typically, WSNs contain hundreds or thousands of sensors which have the ability to communicate with each other [9]. The energy of each sensor is limited and they are usually unrechargeable, so energy consumption of each sensor has to be minimized to prolong the life time of WSNs. Major sources of energy waste are idle listening, collision, overhearing and control overhead [10]. Among these, idle listening is a dominant factor in most sensor network applications [11]. There are several ways to prolong the life time of WSNs, e.g., efficient deployment of sensors [12], optimization of WSN coverage [13], and sleep/wake-up scheduling [14]. In this paper, we focus on sleep/wake-up scheduling. Sleep/wake-up scheduling, which aims to minimize idle listening time, is one of the fundamental research problems in WSNs [15]. Specifically, research into sleep/wake-up scheduling studies how to adjust the ratio between sleeping time and awake time of each sensor in each period. When a sensor is awake, it is in an idle listening state and it can receive and transmit packets. However, if no packets are received or transmitted during the idle listening time, the energy used during the idle listening time is wasted. Such waste should certainly be minimized by adjusting the awake time of sensors, which is the aim of sleep/wake-up scheduling. Recently, many sleep/wake-up scheduling approaches have been developed (see [14], [16], [17]). These approaches roughly fall into three categories: 1) on-demand wake-up approaches; 2) synchronous wake-up approaches; and 3) asynchronous wake-up approaches, as categorized in [18].

In on-demand wake-up approaches [19], [20], out-of-band signaling is used to wake up sleeping nodes on-demand. For example, with the help of a paging signal, a node listening on a page channel can be woken up. As page radios can operate at lower power consumption, this strategy is very energy efficient. However, it suffers from increased implementation complexity.

In synchronous wake-up approaches [16], [21], [22], sleeping nodes wake up at the same time periodically to communicate with one another. Such approaches have to synchronize neighboring nodes in order to align their awake or sleeping time. Neighboring nodes start exchanging packets only within the common active time, enabling a node to sleep for most of the time within an operational cycle without missing

Manuscript received June 1, 2016; revised October 30, 2016 and January 11, 2017; accepted February 13, 2017. This work was supported by the ARC Discovery from Australian Research Council, Australia, under Project DP140100974 and Project DP150101775. This paper was recommended by Associate Editor R. Selmic.

D. Ye is with the State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China, and also with the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, VIC 3122, Australia (e-mail: dye@swin.edu.au).

M. Zhang is with the School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: minjie@uow.edu.au).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2017.2669996

any incoming packets. Synchronous wake-up approaches can reduce idle listening time significantly, but the required synchronization introduces extra overhead and complexity. In addition, a node may need to wake up multiple times during a full sleep/wake-up period, if its neighbors are on different schedules.

In asynchronous wake-up approaches [23]–[25], each node follows its own wake-up schedule in the idle state. This requires that the wake-up intervals among neighbors are overlapped. To meet this requirement, nodes usually have to wake up more frequently than in synchronous wake-up approaches. The advantages offered by asynchronous wake-up approaches include easiness of implementation, low message overhead for communication, and assurance of network connectivity even in highly dynamic networks.

Most current studies use the technique of *duty cycling* to periodically alternate between awake and sleeping states [14], [17]. Here, duty cycle is the ratio between the wake up time length in a predefined period and the total length of that period [10]. For example, suppose a period is 1 s and a node keeps awake for 0.3 s and keeps asleep for 0.7 s in the period. Then, the duty cycle is 30% (or 0.3). The use of duty cycling incurs a tradeoff between energy saving and packet delivery delay [26]: a long wake-up time may cause energy waste, while a short wake-up time may incur packet delivery delay. However, in WSNs, both energy saving and packet delivery delay are important. Because each node in WSNs is usually equipped with an un-rechargeable battery, energy saving is crucial for prolonging the lifetime of WSNs. Because delay is unacceptable in some applications of WSNs, e.g., fire detection and tsunami alarm [27], reducing packet delivery delay is crucial for the effectiveness of WSNs. An intuitive solution to this tradeoff is to dynamically determine the length of wake-up time. The solution proposed in [28] can dynamically determine the length of wake-up time by transmitting all messages in bursts of variable length and sleeping between bursts. That solution can save energy but it may exaggerate packet delivery delay, because each node has to spend time to accumulate packets in its queue before each node transmits these packets in bursts. Another solution, proposed in [29], enables senders to predict receivers' wake-up times by using a pseudo-random wake-up scheduling approach. In the future, if senders have packets to transmit, senders can wake up shortly before the predicted wake-up time of receivers, so the energy, which senders use for idle listening, can be saved. In this case, senders do not have to make the tradeoff, because their wake-up times are totally based on receivers' wake-up times. Receivers still face the tradeoff, however, since a receiver's wake-up time relies on a pseudo-random wake-up scheduling function and different selections of parameters in this function will result in different wake-up intervals. In addition, before a sender can make a prediction about a receiver's wake-up time, the sender has to request the parameters in the receiver's wake-up scheduling function. This request incurs extra energy consumption.

In this paper, a self-adaptive sleep/wake-up scheduling approach is proposed, which takes both energy saving and packet delivery delay into account. This approach is an asynchronous one and it does not use the technique of duty cycling. Thus, the tradeoff between energy saving and packet delivery delay can be avoided. In most existing duty cycling-based sleep/wake-up scheduling approaches, the time axis is divided into periods, each of which consists of several time slots. In each period, nodes adjust their sleep and wake up time, i.e., adjusting the duty cycle, where each node keeps awake in some time slots while sleeps in other time slots. In the proposed self-adaptive sleep/wake-up scheduling approach, the time axis is directly divided into time slots. In each time slot, each node autonomously decides to sleep or wake up. Thus, in the proposed approach, there is no 'cycle' and each time slot is independent. Fig. 1 roughly displays how the proposed approach works.

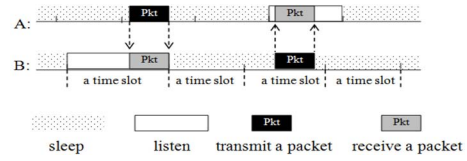


Fig. 1. Overview of the proposed approach.

In Fig. 1, *A* and *B* are two neighboring nodes whose clocks may not be synchronized. They make decisions at the beginning of each time slot autonomously and independently without exchanging information. There are two points in the figure which should be noted. First, for the receiver, if the length of a time slot is not long enough to receive a packet, the length of the time slot will be extended automatically until the packet is received successfully (see the first time slot of node *B*). Second, when a node decides to transmit a packet in the current time slot and the length of the time slot is longer than the time length required to transmit a packet, the node will also decide when in the current time slot to transmit the packet (see the third time slot of node *B*).

The proposed approach is not designed incorporating a specific packet routing protocol. This is because if the sleep/wake-up scheduling approach is designed incorporation with a specific packet routing protocol, the scheduling approach may work well only with that routing protocol but may work less efficiently with other routing protocols. For example, in [14] sleep/wake-up scheduling approach is designed incorporation with a packet routing protocol. Their scheduling approach uses staggered wake-up schedules to create unidirectional delivery paths for data propagation to significantly reduce the latency of data collection process. Their approach works very well if packets are delivered in the designated direction, but it is not efficient when packets are delivered in other directions.

The contributions of this paper are summarized as follows.

1) To the best of our knowledge, this approach is the first one which does not use the technique of duty cycling. Thus the tradeoff between energy saving and packet delivery delay, which is incurred by duty cycling, can be avoided. This approach can reduce both energy consumption and packet delivery delay.
2) This approach can also achieve higher packet delivery ratios in various circumstances compared to the benchmark approaches.

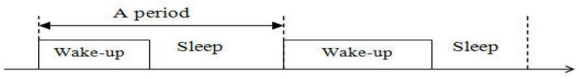


Fig. 2. Formulation of the problem.

3) Unlike recent prediction-based approaches [29], [30], where nodes have to exchange information between each other, this approach enables nodes to approximate their neighbors' situation without requesting information from these neighbors. Thus, the large amount of energy used for information exchange [14] can be saved.

The rest of this paper is organized as follows. In the next section, the proposed self-adaptive sleep/wake-up scheduling approach is described in detail. The simulation and analysis of the proposed approach is given in Section III. Finally, this paper is concluded in Section IV. Due to the page limitation, the related work, theoretical analysis, a part of simulation and future work are provided in a separate supplementary material which can be found in IEEE Digital Library.

II. Approach Design

In this section, we begin with the formulation of the sleep/wake-up scheduling problem. Then, the details of the algorithms, which are involved in the proposed approach, are provided.

A. Formulation of the Problem

As described in Section I, the research of sleep/wake-up scheduling studies how to adjust the ratio between sleeping time and awake time of each sensor in each period as shown in Fig. 2.

According to Fig. 2, formally, we have the following definitions.

Definition 1 (Sleep): A sensor cannot receive or transmit any packets when it is sleeping, i.e., in sleep state. A sensor in sleep state consumes very little energy.

Definition 2 (Wake-Up): A sensor can receive and transmit packets when it is awake, i.e., in wake-up state. A sensor in wake-up state consumes much more energy compared to sleep state.

Definition 3 (Sleep/Wake-Up Scheduling): Sensors adjust the sleeping time length and the awake time length in each period in order to save energy and meanwhile guarantee the efficient transmission of packets.

Generally, the radio transceiver in a sensor node has three modes of operations (termed actions): 1) *transmit*; 2) *listen*; and 3) *sleep* [31]. In transmit mode, the radio transceiver can transmit and receive packets. In listen mode, the transmitter circuitry is turned off, so the transceiver can only receive packets. In sleep mode, both receiver and transmitter are turned off. Typically, among these actions, the power required to transmit is the highest, the power required to listen is medium and the power required to sleep is much less compared to the other two actions. The example provided in [20] shows these power levels: 81 mW for transmission, 30 mW for listen, and 0.003 mW for sleep.

TABLE I
Meaning of Each Symbol or Term

symbol or term	meaning
a player	a node/sensor
row player and column player	two neighbouring nodes
three actions	action 1: *transmit*, action 2: *listen*, action 3: *sleep*
state of a node	Each node has four states: s_0, s_1, s_2, s_3, where s_0 means the node does not have any packet in its buffer, s_1 means the node has 1 packet in its buffer, s_2 means the node has 2 packets in its buffer, s_3 means the node has 3 packets in its buffer.
R and C	payoff matrices for row player and column player, respectively
r_{ij}	payoff obtained by row player when row player takes action i and column player takes action j
c_{ij}	payoff obtained by column player when row player takes action i and column player takes action j
$\alpha_1, \alpha_2, \alpha_3$	the probability for row player to select actions 1, 2, 3, respectively
$\beta_1, \beta_2, \beta_3$	the probability for column player to select actions 1, 2, 3, respectively
$\mathcal{U}$	Successful packet transmission reward
$\pi(s, a)$	the probability for a node to select action a in state s, where a is one of the three actions and s is one of the four states
$Q(s, a)$	the reinforcement value for a node to take action a in state s. The value is be updated using payoff and then the value is used to update $\pi(s, a)$.
$\xi, \delta, \zeta, \epsilon$	Learning rates
γ	Discount factor
η	Gradient step size
TTL	Time to live

B. Model Description

Table I describes the meaning of each symbol or term that will be used in this paper.

Interaction between two neighboring nodes is modeled as a two-player, three-action game, where two players,[1] a row player and a column player, represent two neighboring nodes and three actions mean *transmit*, *listen*, and *sleep*. The three terms, *player*, *node*, and *sensor*, are used interchangeably in this paper. Game theory is a mathematical technique which can be used to deal with multiplayer decision making problems. During the decision-making process, there may be conflict or cooperation among the multiple players. Such conflict or cooperation can be easily modeled by game theory via properly setting the payoff matrices and utility functions. In WSNs, there are conflict and cooperation among sensors during many processes, such as packet routing and sleep/wake-up scheduling. Thus, in this paper, game theory is used to deal with the sleep/wake-up scheduling problem among sensors in WSNs.

The game is defined by a pair of payoff matrices

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \text{ and } C = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

where R and C specify the payoffs for the row player and the column player, respectively. Each of the two players selects an action from the three available actions. The joint action of

[1]It is not necessary that one player must be a sender and the other player must be a receiver. Both of them can send packets to the other party simultaneously, although this will cause collision.

the players determines their payoffs according to their payoff matrices. If the row player and the column player select actions i and j, respectively, the row player receives payoff r_{ij} and the column player obtains payoff c_{ij}. The players can select actions stochastically based on a probability distribution over their available actions. Let α_1–α_3 denote the probability for the row player to choose actions 1–3, respectively, where $\alpha_1+\alpha_2+\alpha_3=1$. Let β_1–β_3 denote the probability for the column player to choose actions 1–3, respectively, where $\beta_1+\beta_2+\beta_3=1$. The row player's expected payoff is

$$P_r=\sum_{1\le i\le 3}\left(\sum_{1\le j\le 3} r_{ij}\alpha_i\beta_j\right) \tag{1}$$

and the column player's expected payoff is

$$P_c=\sum_{1\le i\le 3}\left(\sum_{1\le j\le 3} c_{ij}\alpha_i\beta_j\right). \tag{2}$$

Let actions 1–3 denote *transmit*, *listen*, and *sleep*, respectively. The values of those payoffs in the payoff matrices can be defined by the energy used by a node (which is a negative payoff). In addition, if a packet is successfully transmitted, the payoff of the transmitter/receiver is the energy, used to transmit/receive the packet, plus a positive constant, $\mathcal{U}$, say $\mathcal{U}=98$. Constant $\mathcal{U}$ is added on the energy consumption, if and only if a packet is successfully transmitted. The payoff for action *sleep* is -0.003 (the energy consumed during sleeping period) irrespective of the opponent's action, where the negative sign means that the energy is consumed. The value of the constant $\mathcal{U}$ is larger than the energy used for transmitting or receiving a packet. For example, if the row player has a packet to transmit and it selects *transmit* and the column player selects *listen*, the packet can be successfully transmitted. The payoffs for both players are positive, which can be calculated using the energy they use to transmit/receive the packet plus the constant $\mathcal{U}$. Then, the row player gets payoff $-81+98=17$ and the column player obtains payoff $-30+98=68$, where 81 and 30 are energy consumption for transmitting and receiving a packet, respectively, and the negative sign means that the energy is consumed. However, if the column player selects *sleep*, the packet cannot be successfully transmitted. Then, the row player gets payoff -81 (the energy used for transmitting a packet) and the column player gets payoff -0.003 (the energy used for sleeping). It should be noted that if a node does not have a packet to transmit, it will not select *transmit*.

In a time slot, each node is in one of several states which indicate the status of its buffer. For example, if a node's buffer can store three packets, there are four possible states for the node: s_0–s_3, which imply that the node has 0–3 packets in its buffer, respectively. The aim of each node is to find a policy π, mapping states to actions, that can maximize the node's long-run payoff. Specifically, for a node, $\pi(s,a)$ is a probability, based on which the node selects action a in current state s, and $\pi(s)$ is a vector which is a probability distribution over the available actions in current state s. Thus, policy π is a matrix. For example, a node's buffer can store three packets, so the node have four states: s_0–s_3, as described above. Also, the node has three actions: transmit, listen, and sleep, denoted as 1–3, respectively. Hence, the policy of the node is

$$\pi=\begin{pmatrix}\pi(s_0,1) & \pi(s_0,2) & \pi(s_0,3)\\ \pi(s_1,1) & \pi(s_1,2) & \pi(s_1,3)\\ \pi(s_2,1) & \pi(s_2,2) & \pi(s_2,3)\\ \pi(s_3,1) & \pi(s_3,2) & \pi(s_3,3)\end{pmatrix}.$$

Initially, as the node does not have any knowledge, each action is considered to be equally important in each state. Because for each state, s, $\sum_{1\le i\le 3}\pi(s,i)$ has to be 1, each element in π is set to be $(1/3)$. Then, through learning, the node will adjust the value of each element in π. The detail will be give in the following sections.

Here, the terms $\pi(s,a)$ and α, β denote the probability of selecting an action. $\pi(s,a)$ takes states into consideration while α and β do not do so. α and β are used only for description convenience of the model and the algorithms.

C. Algorithm

Based on the proposed model, we present a reinforcement learning algorithm, which is employed by a player to learn its optimal actions through trial-and-error interactions within a dynamic environment. The algorithm is called Q-learning (Algorithm 1). Q-learning is one of the simplest reinforcement learning algorithms. Both reinforcement learning and evolutionary computation are subfields of machine learning [32]. Reinforcement learning aims to solve sequential decision tasks through trial and error interactions with the environment [33]. In a sequential decision task, a participant interacts with a dynamic system by selecting actions that affect state transitions to optimize some reward function. Evolutionary algorithms are global search techniques derived from Darwin's theory of evolution by natural selection [32]. An evolutionary algorithm iteratively updates a population of potential solutions, which are often encoded in structures called chromosomes. Thus, the major difference between reinforcement learning and evolutionary algorithms is that reinforcement learning is used by participants to maximize their individual rewards while evolutionary algorithms are used to achieve global optimization. Moreover, reinforcement learning algorithms are mainly decentralized and participants need only local information, whereas evolutionary algorithms are primarily centralized or require global information [34]. In this paper, WSNs are distributed environments and each sensor has only local information about itself, so reinforcement learning is more suitable than evolutionary algorithms to the sleep/wake-up scheduling problem.

The benefit of reinforcement learning is that a player does not need a teacher to learn how to solve a problem. The only signal used by the player to learn from its actions in dynamic environments is payoff (also known as reward), a number which tells the player if its last action was good or not [33]. Q-learning as the simplest reinforcement learning algorithm is model-free, which means that players using Q-learning can act optimally in Markovian domains without building overall maps of the domains [35]. During the learning process, a player takes an action in a particular state based on a probability distribution over available actions. The higher

Algorithm 1: Sleep/Wake-Up Scheduling of a Node

1 Let ξ and δ be the learning rates and γ be the discount factor;
2 For each action, initialise value function Q to 0 and policy π to $\frac{1}{n}$, where n is the number of available actions;
3 **repeat**
4 select an action a in current state s based on policy $\pi(s, a)$;
5 **if** *the selected action a is transmit* **then**
6 the node determines when to transmit the packet in the time slot; /* ref **Algorithm 2** */
7 observe payoff p and next state s', update Q-value $Q(s, a) \leftarrow (1-\xi)Q(s, a) + \xi(p + \gamma max_{a'} Q(s', a'))$;
8 **if** *the selected action a is not sleep* **then**
9 based on the updated Q-value, approximate the policy of the neighbour that interacted with the node in the current time slot;
10 based on the approximation, for each action $a \in A$, update the node's policy $\pi(s, a)$;
11 **else**
12 calculate the average payoff $\bar{P}(s) \leftarrow \sum_{a \in A} \pi(s, a) Q(s, a)$;
13 **for** *each action* $a \in A$ **do**
14 $\pi(s, a) \leftarrow \pi(s, a) + \delta(Q(s, a) - \bar{P}(s))$;
15 $\pi(s) \leftarrow Normalise(\pi(s))$; /* ref **Algorithm 3** */
16 $\xi \leftarrow \frac{k}{k+1} \cdot \xi$;
17 $s \leftarrow s'$;
18 **until** *the process is terminated*;

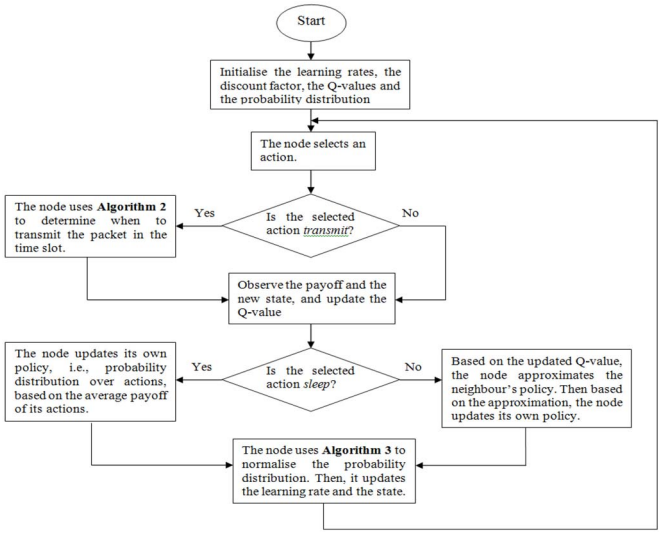


Fig. 3. Flowchart of Algorithm 1.

the probability of an action is, the more possible the action could be taken. Then, the player evaluates the consequence of the action, which the player just takes, based on the immediate reward or penalty, which it receives by taking the action, and also based on the estimate of the value of the state in which the action is taken. By trying all actions in all states repeatedly, the player learns which action is the best choice in a specific state.

Algorithm 1 describes how the proposed approach works in one time slot and is summarized as a flowchart in Fig. 3. The approach is described in the perspective of an individual node. According to Fig. 3, it can be seen that the proposed approach is composed of three steps. First, a node selects an action based on a probability distribution over the three actions: *transmit*, *listen*, or *sleep*, in the current state s. Second, the node carries out the selected action and observes the immediate payoff and the new state s'. Finally, the node adjusts the probability distribution over the three actions in state s based on the payoff and the approximation of the interacted neighbour's policy. The detail of Algorithm 1 is presented as follows.

At the beginning of each time slot, Algorithm 1 repeats from line 3, except for the first time slot where the algorithm starts at line 1. In line 1, a learning rate determines to what extent the newly acquired information will override the old information. The value of a learning rate is in the range $[0, 1]$. A factor of 0 means that the node does not learn anything, while a factor of 1 means that the node considers only the most recent information [36]. A discount factor determines the importance of future rewards. The value of a discount factor is in the range $[0, 1]$. A factor of 0 means that the node is myopic by only considering current rewards, while a factor approaching 1 means that the node strives for a long-term high reward [36]. At the beginning of each time slot, a node has to decide in which mode it will be in this time slot. The node thus selects an action based on the probability distribution over its available actions in its current state (line 4). The initial probability distribution can be set equally over available actions. For example, in this paper, there are three actions. Initially, the probability of selecting each action can be set to $(1/3)$. Later, during the learning process, the probability distribution over the actions will be updated based on the consequence of each action. If the selected action is *transmit*, the node needs to decide when to transmit the packet in the time slot (lines 5 and 6, where the detail will be described in Algorithm 2). The node then receives a payoff and reaches a new state. It updates the Q-value of the selected action in its current state based on the received payoff and the maximum Q-value in the new state (line 7). Here, Q-value, $Q(s, a)$, is a reinforcement of taking action a in state s. This information is used to reinforce the learning process. The formula in line 7 is a value iteration update. Initially, Q-value is given arbitrarily by the designer. Then, the Q-value is updated using the current Q-value, $(1-\xi)Q(s, a)$, plus the learned knowledge, $\xi(p + \gamma \max_{a'} Q(s', a'))$. The learned knowledge consists of the payoff obtained by the node after taking an action plus the estimate of optimal future value: $p + \gamma \max_{a'} Q(s', a')$. In lines 8–10, if the selected action is not *sleep*, the node will approximate the probability distribution over the neighbour's available actions. This neighbour is the one that has interacted with the node, i.e., transmitted a packet to the node or received a packet from the node, in the current time slot. Then, based on the approximation, the node updates its policy $\pi(s, a)$ for

each available action. In lines 11–14, if the selected action is *sleep*, which means that the node does not interact with another node in the current time slot, the node then updates its policy $\pi(s, a)$ for each available action based only on its average payoff. In line 12, the calculation of average payoff is based on the Q-value of an action times the probability of selecting the action. Certainly, average payoff can also be calculated using the sum of the payoff of each action dividing the total number of actions [37]. The former calculation method, however, is more efficient and is more widely used than the latter one [38], [39]. In line 13, the probability of selecting each action is updated. The update of the probability of selecting an action is derived using the current probability of selecting the action plus the difference between the Q-value of the action and the average payoff. If the Q-value of an action is larger than the average payoff, the probability of selecting the action will be increased; otherwise, the probability will be decreased. In line 15, the probability distribution $\pi(s)$ is normalized to be a valid distribution, where $\sum_{a\in A}\pi(s, a) = 1$ and each $\pi(s, a)$ is within the range $(0, 1)$. The details of the normalization will be described in Algorithm 3. Finally, in line 16, the learning rate ξ is decayed, where k means that the current time slot is the kth time slot. The learning rate is decayed to guarantee the convergence of the algorithm as shown in Theorem 3 in the supplementary material. The decay method is not unique. Actually, any progressive decay methods can be used here [40]. Then, at the beginning of the next time slot, the node repeats Algorithm 1 from line 3.

Algorithms 2 and 3 are parts of Algorithm 1, where Algorithm 2 is the expansion of line 6 in Algorithms 1 and 3 is the expansion of line 15 in Algorithm 1. However, the description of Algorithms 2 and 3 is too complex to be integrated into Algorithm 1. Thus, for the clarity purpose, Algorithms 2 and 3 are separated from Algorithm 1 to become independent algorithms.

D. Algorithm in More Detail

In Algorithm 1, there are four issues which have to be addressed.

1) If a node decides to transmit a packet in the current time slot, how does the node determine when to transmit the packet in the time slot (line 6 of Algorithm 1).
2) How does a node approximate its opponent's probability distribution without asking any information from the opponent (line 9 of Algorithm 1).
3) How does a node update its policy based on the approximation (line 10 of Algorithm 1).
4) How does an invalid probability distribution become normalized to a valid one (line 15 of Algorithm 1).

The solutions of the four issues are presented as follows.

1) Issue 1: As discussed in Section I, if the time length of a time slot is longer than the time length required to transmit a packet, a node should determine when to transmit the packet in the current time slot. The time length of a time slot and the time length for each packet transmission are known ahead of time. Suppose that the time length of a time slot is long enough for transmitting m packets.[2] A time slot is then divided into m subslots. The length of a subslot is equal to the time length required to transmit a packet. A node now needs to select a subslot from the m subslots to transmit the packet. Two intuitive solutions are that: 1) the node randomly picks up a subslot to transmit the packet and 2) from the beginning of the time slot, the node sends the packet. If the packet is not sent successfully, the node sends it again. This process continues until the packet is sent successfully or the current time slot expires. However, the two solutions may incur a large amount of energy waste, especially the second solution. To handle this issue, another learning algorithm is devised.

Algorithm 2: Node Determines When to Transmit a Packet in a Time Slot

1 Let ζ and ϵ be the learning rates;
2 For each sub-slot in the current time slot, initialise Q-value to 0 and the probability for selecting each sub-slot is initialised to $x_i = \frac{1}{m}$, where $1 \le i \le m$ and m is the number of sub-slots;
3 select a sub-slot in current time slot based on the probability distribution over the sub-slots $\mathbf{x} = \langle x_1, \ldots, x_m\rangle$;
4 observe payoff p and update Q-value for each sub-slot, $Q_i \leftarrow Q_i + x_i \cdot \zeta \cdot (p - \sum_{1\le i\le m} x_i Q_i)$;
5 update x_i for each sub-slot,
$$x_i = \begin{cases} (1-\epsilon) + (\epsilon/m), \text{ if } Q_i \text{ is the highest} \\ \epsilon/m, \text{ otherwise} \end{cases}$$
6 $\mathbf{x} \leftarrow Normalise(\mathbf{x})$;

In Algorithm 2, if a node decides to transmit a packet in a time slot, it selects a subslot based on the probability distribution $\mathbf{x}$ over the subslots (line 3).[3] The node then observes the payoff p obtained for selecting the subslot and update Q-value for each subslot based on payoff p and current probability distribution $\mathbf{x}$ (line 4). Probability distribution $\mathbf{x}$ is adjusted based on the updated Q-values of subslots (line 5) and is normalized to be a valid distribution (line 6). The probability distribution update method (line 5) is ϵ-greedy exploration [41]. The ϵ-greedy exploration defines a semiuniform probability distribution. The current best subslot is selected with probability $1 - \epsilon + (\epsilon/m)$ and one of the rest subslots is selected with probability (ϵ/m), where m is the number of subslots. The purpose of exploration is to harmonize the tradeoff between exploitation and exploration such that the node can reinforce the evaluation of the subslots it already knows to be good but also explore new subslots. The importance of obtaining a Q-learning algorithm with ϵ-greedy exploration is also justified through a large number of applications [42]–[44].

[2] In this paper, it is assumed that each packet is transmitted using the same time length. Relaxing this assumption is one of our future studies.

[3] Each time (except the first time) Algorithm 2 is called, only lines 3–6 are executed. As lines 1 and 2 are used for initialization, they are executed only at the first time the algorithm is called. Also, Algorithm 2 is called and executed only when the node selects action *transmit* in Algorithm 1. Every time Algorithm 2 is called, it is executed only once in the current time slot.

It should be noted that using the above-mentioned intuitive solutions will incur energy waste, as these solutions do not have any intelligence. The proposed algorithm, i.e., Algorithm 2, however, enables nodes to intelligently select proper subslots for efficient packet transmission. Thus, compared to the intuitive solutions, the energy wasted on failed packet transmission can be saved by using Algorithm 2. Algorithm 2 needs time for convergence. However, during the convergence process, the performance of Algorithm 2 increases gradually, whereas the performance of the intuitive solutions is not improved as time progresses, because they do not have any learning or adaptive ability. This means that Algorithm 2 outperforms the intuitive approaches since the start of running. Then, the performance gap becomes bigger and bigger as time progresses.

2) Issue 2: Taking the row player for example (recall Section II-B), the expected payoffs of the row player plays actions 1–3 are

$$P_r^{(1)} = \lim_{\alpha_1 \to 1} \sum_{1 \le i \le 3} \left(\sum_{1 \le j \le 3} r_{ij} \alpha_i \beta_j \right) = \sum_{1 \le j \le 3} r_{1j} \beta_j \tag{3}$$

$$P_r^{(2)} = \lim_{\alpha_2 \to 1} \sum_{1 \le i \le 3} \left(\sum_{1 \le j \le 3} r_{ij} \alpha_i \beta_j \right) = \sum_{1 \le j \le 3} r_{2j} \beta_j \tag{4}$$

$$P_r^{(3)} = \lim_{\alpha_3 \to 1} \sum_{1 \le i \le 3} \left(\sum_{1 \le j \le 3} r_{ij} \alpha_i \beta_j \right) = \sum_{1 \le j \le 3} r_{3j} \beta_j \tag{5}$$

respectively. It can be proven that if an action is executed in each state an infinite number of times on an infinite run and the learning rate ξ is decayed appropriately, the Q-value of that action (calculated using the equation in line 5 of Algorithm 1) will converge with probability 1 to Q^*, where Q^* is the expected payoff of a player playing that action. The detailed proof will be presented in the supplementary material. According to (3)–(5), it can been found that the row player's expected payoff for executing an action depend on the column player's probabilities for executing each action. Based on this conclusion, the row player can use the current Q-value of an action to approximate the expected payoff of that action

$$Q(s, 1) = P_r^{(1)} = \sum_{1 \le j \le 3} r_{1j} \beta_j \tag{6}$$

$$Q(s, 2) = P_r^{(2)} = \sum_{1 \le j \le 3} r_{2j} \beta_j \tag{7}$$

$$Q(s, 3) = P_r^{(3)} = \sum_{1 \le j \le 3} r_{3j} \beta_j. \tag{8}$$

Using (6)–(8), the row player can calculate β_1–β_3 which are the column player's probabilities to take actions 1–3, respectively. After the calculation, the row player adjusts its own probabilities for executing each action, which will be described later. Certainly, it is infeasible to execute the same action infinite number of times. Thus, the expected payoff can only be approximated but cannot be precisely predicted. However, as the learning progresses, the Q-learning algorithm will gradually converge to the optimal Q-value, Q^* and thus the precision of the approximation will increase. Then, the column player's probability distribution can be computed more and more precisely by the row player. Thus, it can be seen that by using the approximation, the row player can predict the column player's probability distribution over the actions without communication with the column player.

3) Issue 3: Using a gradient ascent algorithm [45], [46], a player can increase its expected payoff by moving its strategy in the direction of the current gradient with some step size. The gradient is computed as the partial derivative of the player's expected payoff with respect to its probability for selecting each action. We take the row player for example and set $\alpha_3 = 1 - \alpha_1 - \alpha_2$. Then, we have

$$\frac{\partial P_r}{\partial \alpha_1} = \sum_{1 \le j \le 3} r_{1j} \beta_j - \sum_{1 \le j \le 3} r_{3j} \beta_j \tag{9}$$

and

$$\frac{\partial P_r}{\partial \alpha_2} = \sum_{1 \le j \le 3} r_{2j} \beta_j - \sum_{1 \le j \le 3} r_{3j} \beta_j. \tag{10}$$

If in the kth time slot, $\alpha_1^{(k)}$–$\alpha_3^{(k)}$ are the row player's probabilities for selecting actions 1–3, respectively, then the new probabilities for the next time slot are

$$\alpha_1^{(k+1)} = \alpha_1^{(k)} + \eta \frac{\partial P_r}{\partial \alpha_1^{(k)}} \tag{11}$$

$$\alpha_2^{(k+1)} = \alpha_2^{(k)} + \eta \frac{\partial P_r}{\partial \alpha_2^{(k)}} \tag{12}$$

and

$$\alpha_3^{(k+1)} = 1 - \alpha_1^{(k+1)} - \alpha_2^{(k+1)} \tag{13}$$

where η is the gradient step size. Based on mathematical knowledge [47], it can be known that if η is sufficiently small, the procedure will converge.

4) Issue 4: To normalize an invalid probability distribution to be a valid one, proportion-based mapping is used. The invalid probability distribution contains the learned knowledge and in order to preserve the learned knowledge, the normalized probability distribution should be as "close" as possible to the un-normalized one. Proportion-based mapping can adjust each invalid probability into the range $(0, 1)$ and meanwhile, can keep the relative magnitude of the probabilities. The function *Normalise*() is presented in pseudocode form in Algorithm 3.

From the above description, it can be seen that, compared to existing sleep/wake-up scheduling approaches, where nodes' sleep/wake-up patterns are almost predefined, this approach enables nodes to autonomously and dynamically decide whether to sleep using learning techniques. Thus, theoretically, nodes in this approach are smarter than nodes in most existing approaches.

III. Simulation and Analysis

In this section, the simulation results and the corresponding analysis are presented. The simulation was implemented using JAVA programming language and was run on Windows 7 Professional SP1 system with Intel Core i5 3.4-GHz CPU and

Algorithm 3: Normalize()

Suppose that in state s, there are m available actions, i.e., $a_1, a_2, \ldots, a_m$;
Let $d = min_{1 \le k \le m} \pi(s, a_k)$, mapping center $c_0 = 0.5$ and mapping lower bound $\Delta = 0.001$;
if $d < \Delta$ **then**
    $\rho \leftarrow \frac{c_0 - \Delta}{c_0 - d}$;
    for $k = 1$ *to* m **do**
        $\pi(s, a_k) \leftarrow c_0 - \rho \cdot (c_0 - \pi(s, a_k))$;
for $k = 1$ *to* m **do**
    $r \leftarrow \sum_{1 \le k \le m} \pi(s, a_k)$;
    $\pi(s, a_k) \leftarrow \frac{\pi(s, a_k)}{r}$;
return $\pi(s)$;

8GB RAM. "Node" is programmed as a class and each node is an object of this class. To simulate an area, a 2-D array is defined, which represents the coordinates of an area. The length of the row of the array represents the length of the area and the length of the column of the array represents the width of the area. For the connection of two nodes, if two nodes are in the communication range of each other, the two nodes are connected. For time implementation, a JAVA function is used to read the system time. To measure the time length of delivering a packet, system time is read at both beginning of and end of the delivery. Then, the time length of delivering a packet can be obtained using the system time read at the beginning of the delivery minus the system time read at the end of the delivery.

To evaluate the proposed self-adaptive approach (recorded as SA-Mech.), we build a platform using JAVA to test it in comparison with four other approaches: 1) two-radio-MAC (TR-MAC) [20]; 2) DW-MAC [48]; 3) EM-MAC [29]; and 4) AS-MAC [26]. TR-MAC is an on-demand approach. DW-MAC is a synchronous approach. EM-MAC and AS-MAC are asynchronous approaches. Through comparing with these approaches, the performance of the proposed approach can be objectively demonstrated. The reason for selecting these approaches is that TR-MAC and DW-MAC are the most efficient on-demand approaches and synchronous approach, respectively.[4] Both EM-MAC and AS-MAC are the latest asynchronous approaches,[5] but, to the best of our knowledge, their performance has not been compared yet. Thus, we select both of them in our simulation. These approaches are described in detail as follows.

1) *TR-MAC:* In TR-MAC, two radios are used, where one is for waking up neighbors and the other is for sending packets. Unlike traditional on-demand approaches, in TR-MAC, when a node has a packet to transmit, it does not wake up its entire neighborhood but selectively wakes up several neighbors which have previously engaged in communication through rate estimation.
2) *DW-MAC:* DW-MAC is a synchronized duty cycle MAC protocol, where each cycle is divided into three periods: a) sync; b) data; and c) sleep. DW-MAC has to synchronize the clocks in sensor nodes periodically during the sync period. DW-MAC then sets up a one-to-one proportional mapping between a data period and the following sleep period. In a data period, the sender will send a scheduling frame to the receiver. Based on the time interval after the beginning of the data period and the duration of the scheduling frame transmission, both sender and receiver will set up their wake-up time interval during the following Sleep period to transmit/receive the packet.
3) *EM-MAC:* In EM-MAC, each node uses a pseudo-random number generator: $X_{n+1} = (aX_n + c) \bmod m$ to compute its wake-up times, where $m > 0$ is the modulus, a is the multiplier, c is the increment, X_n is the current seed and the generated X_{n+1} becomes the next seed. In this simulation, $m = 65536$, each node's a, c and X_n are independently chosen following the principles suggested by Knuth [50]. By requesting the parameters, m, a, c, and X_n, from a receiver, a sender can predict the receivers future wake-up times and prepare to send data at those times. EM-MAC does not need synchronization but it requires nodes to exchange information before nodes can make predictions.
4) *AS-MAC:* In AS-MAC, nodes wake up periodically (but asynchronously) to receive packets. Nodes intending to transmit packets wake up at the scheduled wake-up time of the intended target nodes. Neighboring nodes have to communicate periodically to exchange information about wake-up schedules to avoid long preambles at the beginning of transmission.

In addition, we also compare SA-Mech. with its synchronized version, SA-Mech.-Syn. In SA-Mech.-Syn, it is assumed that a sink node periodically broadcasts a special packet to the entire network to synchronize the nodes' clocks.[6] The aim of introducing SA-Mech.-Syn for comparison is to test how the introduction of synchronization will affect the performance of SA-Mech.

A. Simulation Setup

The simulation is operated in two types of networks: 1) grid networks and 2) random networks. For each type of networks, there are four different scales. This setting is to evaluate the performance of these approaches in different scales and different types of networks. The scale of grid networks fluctuates from 49 nodes to 169 nodes, where 49 nodes are structured as a 7×7 grid network, 81 nodes are structured as a 9×9 grid network, 121 nodes are structured as a 11×11 grid network,

[4] Actually, DW-MAC is not the latest synchronous approach. Instead, Guo *et al.* [14] approach is the latest. However, Guo *et al.*'s [14] approach heavily depends on predetermined transmission paths, so it is not very efficient once a path is changed.

[5] Actually, Hsu *et al.* [49] approach was published later than both EM-MAC and AS-MAC. However, as Hsu *et al.* [49] approach is jointly designed with a routing protocol, it may not work well with other routing protocols.

[6] Actually, clocks can only be synchronized using broadcasting only if there are not intermediate nodes for relaying. Otherwise, too much delay will incur. However, as the focus of this paper is not on clock synchronization, the synchronization process is simplified.

and 169 nodes are structured as a 13×13 grid network. In the four grid networks, each node is 200 meters from its neighbors and there are 5 sinks which are located at the four corners and the center of the network. The four scales of random networks are 50 nodes, 80 nodes, 120 nodes, and 170 nodes which are randomly located in a 1000×1000 m area. In the four random networks, there are five sinks which are also randomly located in the area. The communication radius of each node is set to 200 m.

Each node generates a packet at the beginning of each time slot based on a predefined probability: the packet generation probability. As the state of a node is determined by the number of packets in its buffer, the packet generation probability directly affects the state of each node. Then, the action selection of each node will be indirectly affected. The expiry time of a packet is based on exponential distribution. The average size of a packet is 100 bytes, and the actual size of a packet is based on normal distribution with variance equal to 10. In this simulation, four packet generation probabilities are used: 0.2, 0.4, 0.6, and 0.8. This setting is to evaluate the performance of these approaches in a network with different number of transmitted packets. For packet routing, we use a basic routing approach, gossiping [51]. Gossiping is a slightly enhanced version of flooding where the receiving node sends the packet to a randomly selected neighbour, which picks another random neighbour to forward the packet to and so on, until the destination or the maximum hop is reached. It should be noted that when the destination and some other nodes are all in the signal range of the source, based on the routing protocol, the source still relays a packet to one of neighbors and this process continues until the destination or the maximum hop is reached. The routing process is not optimized in the simulation, as this paper focuses on sleep/wake-up scheduling only. This routing protocol is not energy-efficient but it is easy to implement. Because all of the sleep/wake-up scheduling approaches use the same routing protocol in the simulation, the comparison among them is still fair.

Performance is measured by three quantitative metrics: 1) average packet delivery latency; 2) packet delivery ratio; and 3) average energy consumption. The minimum time needed by nodes to transmit or receive a packet is about 2 ms [14], e.g., using the radio chip *Chipcon CC2420*. The three metrics are described as follows.

1) Packet delivery latency is measured by the average time taken by each delivered packet to be transmitted from the source to the destination. Note that those packets, which do not reach the destination successfully, have also been taken into account. Their delivery latency is the time interval, during which they exist in the network.
2) Packet delivery ratio is measured by using the percentage of packets that are successfully delivered from the source to the destination. Each packet comes with a parameter, time-to-live (TTL), which is a positive integer. Once a packet is transmitted from a sender to a receiver (no matter whether successfully or unsuccessfully), the TTL of this packet subtracts 1. If the TTL of this packet becomes 0 and it has not reached the destination, the delivery of this packet is a failure.
3) Average energy consumption is calculated by using the total energy consumption to divide the number of nodes in the network during a simulation run.

TABLE II
PARAMETERS SETTING

Parameters	Values	Explanations
	$81mW$	Energy used for transmission
	$30mW$	Energy used for listen
	$0.003mW$	Energy used for sleep
$\mathcal{U}$	98	Successful packet transmission reward
$\xi, \delta, \zeta, \epsilon$	$0.8, 0.4, 0.2, 0.2$	Learning rates
γ	0.65	Discount factor
η	0.0001	Gradient step size
TTL	8, 15	Time to live

In this simulation, we set up the evaluation cases ourselves. The approaches used for comparison in the simulation are from four different references which use different evaluation cases. Thus, there are no common evaluation cases among these references. Because all the evaluated approaches are tested in the same cases, the comparison is still fair and the results are convincing.

The values and meanings of the parameters used in the simulation are listed in Table II, where the value of the TTL is set to 8 in grid networks and is set to 15 in random networks. The values of the parameters were experimentally chosen to provide the best performance of the proposed approach.[7] The values for energy consumption are from [20].

For the compared approaches, the duty cycle is set to 5%. A full sleep/wake-up interval is set to 1 s. As our approach does not use the duty cycle, the time length of a time slot in our approach is set to 8 ms. As described in Section II-C, learning rates are used to update the learned knowledge. Because the compared approaches do not use reinforcement learning, they do not need update and thus do not need learning rates. In each simulation case, each of these approaches has 200 repeats and each repeat consists of 5000 s. The detailed simulation result data are given in the supplementary material.

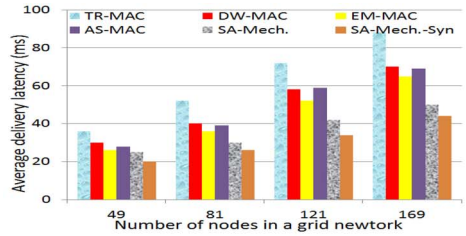
B. Simulation Results in Grid Networks

The topology of a grid network is like a chessboard as described in the previous section. Grid networks have three rules.

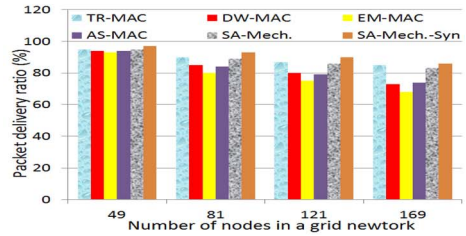
1) Each of the four nodes that are located at the four corners has two neighbors.
2) Each node on the edge has three neighbors.
3) All the other nodes have four neighbors.

Moreover, in this simulation, five sinks are located at the four corners and the center of the network, respectively. Therefore, the grid networks are highly regular. In this situation, the proposed approach, SA-Mech., can work well, because SA-Mech. is based on reinforcement learning, and something with high regularity can be easily learned [33]. In the grid networks, all the packets are transmitted to the four corners and the center, where sinks are located. This regularity can be easily learned

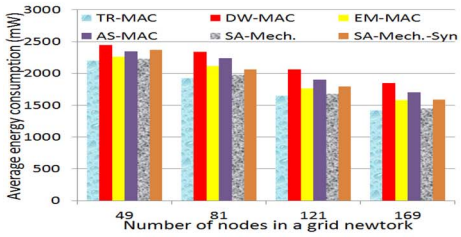
[7]The "best performance" means that the presented results are the best results that we ever attempted. It does not mean that the presented results are the best in all the combinations of parameter values. As the number of combinations of parameter values is infinite, it is infeasible to attempt all the combinations of parameter values.



(a)



(b)



(c)

Fig. 4. Performance of the approaches in different scales of grid networks. (a) Average delivery latency (ms). (b) Packet delivery ratio (%). (c) Average energy consumption (mW).

by nodes under SA-Mech. as time progresses, and based on the learned knowledge, nodes can precisely approximate their neighbors' situations. The detailed analysis of the simulation results is given in the following sections.

1) Performance of the Approaches in Different Scales of Grid Networks: Fig. 4 demonstrates the performance of these approaches in different scales of grid networks. The packet generation probability is fixed at 0.2. In Fig. 4(a), with the increase of network scale, the average delivery latency in all of these approaches rises. This is because when the network scale increases, based on the routing approach used in this simulation, on average, each packet has to be transmitted with more steps to its destination. Thus, the average delivery latency will undoubtedly increase. Specifically, it can be found that TR-MAC achieves the highest latency (36 ms in the 49-node network, 52 ms in the 81-node network, 72 ms in the 121-node network, and 88 ms in the 169-node network). This is because in TR-MAC, whenever a sender intends to transmit a packet, it has to wake-up the receiver and to wait for the response from the receiver. This process will continue until the packet reaches the destination or the TTL of the packet reaches 0, so each packet will suffer a large latency during the transmission process. DW-MAC and AS-MAC achieve nearly the same latency while the latency in EM-MAC is slightly, about 5%, lower than that in DW-MAC and AS-MAC. This can be explained by the fact that both DW-MAC and AS-MAC require nodes to periodically exchange information to schedule future wake-up intervals but EM-MAC does not need such a step, so the time used for periodical information exchange can be saved. Although in EM-MAC, nodes have to request their neighbors' information for future wake-up prediction, that request is on-demand and only once. The proposed approach, SA-Mech., achieves the lowest latency (25 ms in the 49-node network, 30 ms in the 81-node network, 42 ms in the 121-node network, and 50 ms in the 169-node network), because: 1) SA-Mech. does not require a periodical information exchange and unlike EM-MAC, SA-Mech. does not require nodes to request their neighbors' information for prediction, so the corresponding time is saved and 2) as EM-MAC is an asynchronous approach, two neighboring nodes' clocks are not synchronized. When a node makes prediction regarding its neighbour's wake-up interval, it uses its own clock, so the prediction may not be precise enough. In SA-Mech., nodes can dynamically approximate their neighbors' behavior based on their own and their neighbors' current situation, so SA-Mech. achieves a lower delivery latency, average 10%, than EM-MAC does.

In Fig. 4(b), with the increase of network scale, the packet delivery ratios in all of these approaches decrease. This is mainly because of the routing approach. In different scales of networks, each packet is given the same TTL. When the network scale increases, the TTL value is no longer large enough to guarantee successful transmission of a packet to its destination. Therefore, more and more packets are unable to be transmitted to their destinations successfully. In this scenario, TR-MAC obtains the best result (95% in the 49-node network, 90% in the 81-node network, 87% in the 121-node network, and 85% in the 169-node network). In TR-MAC, the transmission is on-demand and the transmission happens after the sender communicates with the receiver, so packets are very likely to be transmitted successfully. SA-Mech. achieves the second best result (95% in the 49-node network, 89% in the 81-node network, 86% in the 121-node network, and 83% in the 169-node network). In SA-Mech., to save energy, nodes are not allowed to communicate to exchange information. Nodes make decisions based only on approximation which may contain errors, so SA-Mech. is somewhat, about 2%, less efficient than TR-MAC in this scenario, however, SA-Mech. is still better than the three other approaches. Unlike TR-MAC and SA-Mech., where senders immediately transmit packets to intended receivers after communication or approximation, in both DW-MAC and AS-MAC, there is a periodic schedule for future wake-up times. Because in both DW-MAC and AS-MAC, packet transmission does not happen immediately after scheduling, collision is possible in the future. For example, several senders schedule with one receiver for future wake-up times and these wake-up times may overlap, which will result in packet transmission failure, because: 1) in DW-MAC, the calculation of future wake-up time is based on a one-to-one proportional mapping between a data period and the following Sleep period, so there is a probability that two senders have the same or partially overlapping wake-up times with one receiver and 2) in AS-MAC, there is not such strict one-to-one mapping but AS-MAC is an asynchronous approach, so the scheduled wake-up times may still overlap

due to the asynchrony of nodes' clocks. In EM-MAC, packet transmission does not happen immediately after scheduling and there is no synchronization in EM-MAC, so it suffers the drawbacks of both DW-MAC and AS-MAC. Hence, EM-MAC achieves the worst result in this scenario 93% in the 49-node network, 80% in the 81-node network, 75% in the 121-node network, and 68% in the 169-node network.

In Fig. 4(c), with the increase of network scale, the average energy consumption reduces gradually. Actually, with the increase of network scale, the total energy consumption increases, as packets may have to be transmitted with more steps to the destination. However, because the packet generation probability and TTL are fixed, the total energy consumption cannot increase much with the increase of network scale. Therefore, the increase speed of total energy consumption is slower than the increase speed of the number of nodes. Thus, the average energy consumption decreases with the increase of network scale. Recall that average energy consumption is computed by using total energy consumption to divide the number of nodes. TR-MAC and SA-Mech. are very energy efficient (2230 mW in the 49-node network, 1978 mW in the 81-node network, 1682 mW in the 121-node network, and 1453 mW in the 169-node network), as both of them do not require nodes to communicate between each other to obtain information.[8] The energy consumption in EM-MAC is less than that in both DW-MAC and AS-MAC (about 10% less than DW-MAC and 5% less than AS-MAC), because DW-MAC and AS-MAC require an energy-consuming periodical synchronization or scheduling period. This periodical process consumes much energy. SA-Mech. is more energy efficient than EM-MAC, since, as described above, EM-MAC suffers the drawback of asynchrony, where a sender's prediction, based on its own clock, about an intended receiver's wake-up interval may not be precise enough.

In Fig. 4, it can be seen that SA-Mech. uses less energy than SA-Mech.-Syn, but SA-Mech.-Syn can achieve higher packet delivery ratio and less packet delivery latency. This is because of the introduction of synchronization. In order to realize synchronization, a sink has to periodically broadcast to synchronize nodes' clocks and this synchronization process inevitably costs some energy. However, synchronization can make the behavior of nodes more predictable compared with asynchronized nodes. In SA-Mech., each node adapts its behavior based on the prediction and approximation of its neighbors' behavior and the introduction of synchronization can make nodes' behavior more predictable. Therefore, SA-Mech.-Syn can achieve higher packet delivery ratio and less packet delivery latency than SA-Mech. This phenomenon also exists in other circumstances (Figs. 5 and 6) due to the same reason.

2) Performance of the Approaches Under Different Packet Generation Probabilities in Grid Networks: Fig. 5 demonstrates the performance of these approaches under different packet generation probabilities. The number of nodes in the grid network is fixed at 121 (11 × 11). In Fig. 5(a), with the increase of packet generation probability, the average packet delivery latency in each of the approaches rises. This is because when more packets exist in the network, collision is more likely to happen, as two or more nodes transmitting might be within each other's signal range. Such collision will increase packet delivery latency. An interesting phenomenon is that the packet delivery latency in DW-MAC increases more sharply as the packet generation probability rises compared with the other approaches. This is because in DW-MAC, a sender has to schedule packet transmission with a receiver during a data period. Since the data period is short, when several senders want to transmit packets to one receiver, the data period is not long enough for all the senders' scheduling. Thus, some of the senders have to wait until the next cycle, which will prolong the packet delivery latency.

In Fig. 5(b), with the increase of packet generation probability, the packet delivery ratio keeps almost steady in TR-MAC (about 87%), DW-MAC (about 80%), AS-MAC (about 79%), and SA-Mech. (about 85%), but it goes down in EM-MAC (from 75% to 60%). This is due to the fact that the asynchrony of EM-MAC will incur senders' imprecise prediction of intended receivers' wake-up interval. Such imprecise prediction may lead to collision and thus increases the number of packet delivery failures. As the number of packets in the network increases, collision becomes more

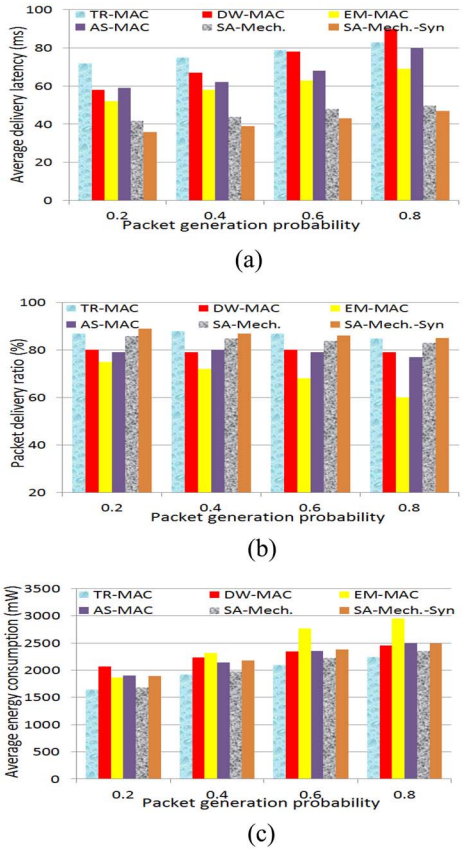


Fig. 5. Performance of the approaches under different packet generation probabilities in grid networks. (a) Average delivery latency (ms). (b) Packet delivery ratio (%). (c) Average energy consumption (mW).

[8]Recall that in TR-MAC, for each node, there is a separate wake-up radio for communication. The energy consumption of a wake-up radio is much lower than that of a normal sensor node.

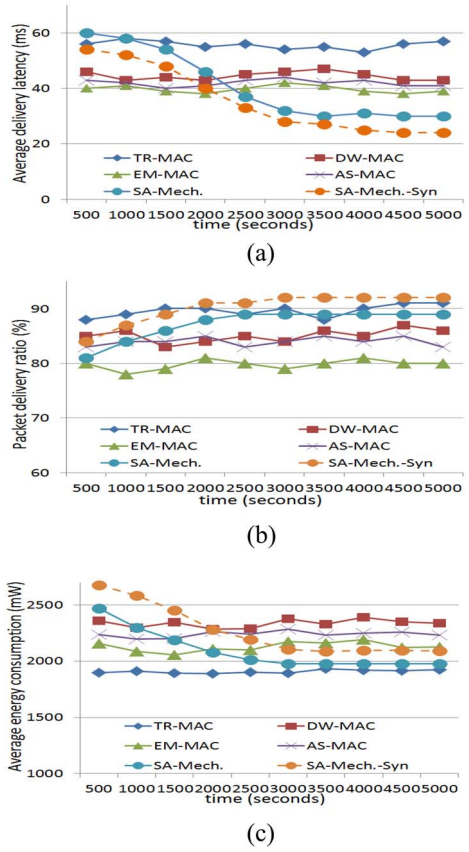


Fig. 6. Performance variance of the approaches as time progresses in grid networks. (a) Average delivery latency (ms). (b) Packet delivery ratio (%). (c) Average energy consumption (mW).

likely and the number of packet delivery failures rises correspondingly. Also, as shown in Fig. 5(c), because of such a large number of packet delivery failures, the corresponding energy is consumed substantially. Therefore, in EM-MAC, the energy consumption rises significantly (from 1898 to 2952 mW) as the packet generation probability increases, whereas in other approaches, the energy consumption just rises steadily due to the increasing number of transmitted packets in the network.

3) Performance Variance of the Approaches as Time Progresses in Grid Networks: Fig. 6 demonstrates the performance variance of these approaches as time progresses. The number of nodes is fixed at 81 and the packet generation probability is fixed at 0.4. It can be seen that as time progresses, only the performance of SA-Mech. and SA-Mech.-Syn increases gradually, whereas the performance of other approaches keeps almost steady. This is because in both SA-Mech. and SA-Mech.-Syn, nodes have a learning process. Thus, as time progresses, nodes obtain more and more knowledge and work better and better. In other approaches, nodes do not have such a learning process, so their performance is relatively stable in comparison with SA-Mech. and SA-Mech.-Syn. It has to be noted that the convergence speed of SA-Mech. and SA-Mech.-Syn depends on the value of learning rate, ξ. If the value of ξ is large, e.g., 0.9, SA-Mech., and SA-Mech.-Syn can quickly converge to a relatively stable state in the early stages but they may heavily oscillate in the late stages. On the contrary, if the value of ξ is too small, e.g., 0.1, SA-Mech., and SA-Mech.-Syn will converge very slowly but once they converge to a state, this state will be a very stable state. Therefore, in the simulation, we initially set ξ a large value and let it automatically decay in each round. There are two reasons for setting ξ in this way. First, SA-Mech. and SA-Mech.-Syn can quickly move toward a relatively stable state in the early stages and then slowly converge to an optimal state in the late stages. Thus, the oscillation can be avoided and the convergence speed is relatively quick. The second reason is that setting ξ in this way can meet the requirements of Theorem 3 (presented in the supplementary material) and then, the convergence of SA-Mech. and SA-Mech.-Syn to an optimal result can be guaranteed.

The energy cost of learning, e.g., how much energy is used during a learning process, is not evaluated in this paper, because the dominant energy cost in WSNs is idle listening [11]. The time cost of learning has been empirically evaluated. For a single process, i.e., the time period during which a node makes a decision about sleeping or awaking, the proposed approach needs less time than the other compared approaches. This can be explained by the fact that the proposed approach does not need communication among neighboring nodes, while the all of the other approaches need. Communication is usually more time consuming than computation. Thus, because of the prediction of neighboring nodes' behavior, the proposed approach needs more computation than the other approaches but, under the proposed approach, packets need less delivery time. Because the evaluation of average delivery latency has already included the evaluation of time cost, the figure about the time cost for a single process of these approaches is not presented in this paper.

In summary, the performance of the proposed approach, SA-Mech., is better than the other approaches in most circumstances under the current simulation setup. Although in some circumstances, the performance of TR-MAC is better than that of SA-Mech, the difference is quite slight, about 5%, considering that TR-MAC has to use separate communication radios. Both SA-Mech. and its synchronized version SA-Mech.-Syn. perform well under the current simulation setup. SA-Mech.-Syn. achieves 8% higher packet delivery ratio and 3% less packet delivery latency than SA-Mech. but needs 10% extra energy. Thus, which one is better depends on specific requirements. Moreover, we have also evaluated a learning-based approach, RL-Mech. [52], in the simulation. Like our SA-Mech., RL-Mech. is also based on reinforcement learning. Thus, SA-Mech. and RL-Mech. have the same trend in various situations. However, RL-Mech. is still based on duty cycle technique and nodes have to keep awake for D consecutive time slots, where D is the duty cycle fixed by users. In our SA-Mech., each time slot is independent and nodes do not have to keep awake for a consecutive series of time slots. Thus, the average energy consumption under SA-Mech. is less than that under RL-Mech. for 30%. Also, as RL-Mech. is asynchronous and nodes cannot predict their neighbors' duty cycle, a node's awake time may not overlap with another node's awake time. Therefore, the average delivery latency under RL-Mech. is longer than that under SA-Mech. for 20%,

and the packet delivery ratio under RL-Mech. is less than that under SA-Mech. for 15%. In addition, through the simulation, it has been found that the performance of these approaches depends, to some extent, on the routing approach. Such dependency may limit the applicability of these approaches in real situations. Therefore, such dependency should be avoided. We leave this as one of our future studies.

IV. Conclusion

This paper introduced a self-adaptive sleep/wake-up scheduling approach. This approach does not use the technique of duty cycling. Instead, it divides the time axis into a number of time slots and lets each node autonomously decide to sleep, listen or transmit in a time slot. Each node makes a decision based on its current situation and an approximation of its neighbors' situations, where such approximation does not need communication with neighbors. Through these techniques, the performance of the proposed approach outperforms other related approaches. Most existing approaches are based on the duty cycling technique and these researchers have taken much effort to improve the performance of their approaches. Thus, duty cycling is a mature and efficient technique for sleep/wake-up scheduling. This paper is the first one which does not use the duty cycling technique. Instead, it proposes an alternative approach which is based on game theory and the reinforcement learning technique. The performance improvement of the proposed approach, compared with existing approaches, may not be big, but the proposed approach provides a new way to study sleep/wake-up scheduling in WSNs. This paper primarily focuses on theoretical study, so there are some assumptions. These assumptions are set to simplify the discussion of our approach. Without these assumptions, the discussion of our approach will become extremely complex, which is harmful for the readability of this paper. The problem itself addressed in this paper, however, is not simplified by these assumptions. Thus, the problem is still general under these assumptions.

Acknowledgment

The authors would like to thank A/Prof. Z. Ye for his help on theoretical analysis. They would also like to thank Dr. M. S. Cincotta and Dr. I. C. Piper's help in the final language editing of this paper.

References

[1] Y. Xiao *et al.*, "Tight performance bounds of multihop fair access for MAC protocols in wireless sensor networks and underwater sensor networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 10, pp. 1538–1554, Oct. 2012.

[2] S. Zhu, C. Chen, W. Li, B. Yang, and X. Guan, "Distributed optimal consensus filter for target tracking in heterogeneous sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1963–1976, Dec. 2013.

[3] G. Acampora, D. J. Cook, P. Rashidi, and A. V. Vasilakos, "A survey on ambient intelligence in healthcare," *Proc. IEEE*, vol. 101, no. 12, pp. 2470–2494, Dec. 2013.

[4] Y. Yao, Q. Cao, and A. V. Vasilakos, "EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 810–823, Jun. 2015, doi: 10.1109/TNET.2014.2306592.

[5] S. H. Semnani and O. A. Basir, "Semi-flocking algorithm for motion control of mobile sensors in large-scale surveillance systems," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 129–137, Jan. 2015.

[6] B. Fu, Y. Xiao, X. Liang, and C. L. P. Chen, "Bio-inspired group modeling and analysis for intruder detection in mobile sensor/robotic networks," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 103–115, Jan. 2015.

[7] Y. Zhao, Y. Liu, Z. Duan, and G. Wen, "Distributed average computation for multiple time-varying signals with output measurements," *Int. J. Robust Nonlin. Control*, vol. 26, no. 13, pp. 2899–2915, 2016.

[8] Y. Zhao, Z. Duan, G. Wen, and G. Chen, "Distributed finite-time tracking of multiple non-identical second-order nonlinear systems with settling time estimation," *Automatica*, vol. 64, pp. 86–93, Feb. 2016.

[9] M. Li, Z. Li, and A. V. Vasilakos, "A survey on topology control in wireless sensor networks: Taxonomy, comparative study, and open issues," *Proc. IEEE*, vol. 101, no. 12, pp. 2538–2557, Dec. 2013.

[10] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. IEEE INFOCOM*, New York, NY, USA, Jun. 2002, pp. 1567–1576.

[11] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," in *Proc. ACM SenSys*, Boulder, CO, USA, USA, Nov. 2006, pp. 321–334.

[12] X. Liu, "A deployment strategy for multiple types of requirements in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2364–2376, Oct. 2015.

[13] C.-P. Chen *et al.*, "A hybrid memetic framework for coverage optimization in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2309–2322, Oct. 2015.

[14] P. Guo, T. Jiang, Q. Zhang, and K. Zhang, "Sleep scheduling for critical event monitoring in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 345–352, Feb. 2012.

[15] G. Wei, Y. Ling, B. Guo, B. Xiao, and A. V. Vasilakos, "Prediction-based data aggregation in wireless sensor networks: Combining grey model and Kalman filter," *Comput. Commun.*, vol. 34, no. 6, pp. 793–802, 2011.

[16] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[17] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *Proc. ACM SenSys*, Raleigh, NC, USA, Nov. 2008, pp. 1–14.

[18] S. Lai, B. Ravindran, and H. Cho, "Heterogenous quorum-based wake-up scheduling in wireless sensor networks," *IEEE Trans. Comput.*, vol. 59, no. 11, pp. 1562–1575, Nov. 2010.

[19] L. Gu and J. A. Stankovic, "Radio-triggered wake-up capability for sensor networks," in *Proc. 10th IEEE Real Time Embedded Technol. Appl. Symp.*, Toronto, ON, Canada, 2004, pp. 27–37.

[20] M. J. Miller and N. H. Vaidya, "A MAC protocol to reduce sensor network energy consumption using a wakeup radio," *IEEE Trans. Mobile Comput.*, vol. 4, no. 3, pp. 228–242, May/Jun. 2005.

[21] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw. (IPSN)*, Boise, ID, USA, 2005, pp. 20–27.

[22] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proc. 7th ACM MobiHoc*, Florence, Italy, 2006, pp. 322–333.

[23] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in *Proc. ACM MobiHoc*, Annapolis, MD, USA, 2003, pp. 35–45.

[24] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. ACM SenSys*, Baltimore, MD, USA, 2004, pp. 95–107.

[25] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *Proc. 9th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, San Francisco, CA, USA, 2009, pp. 37–48.

[26] B. Jang, J. B. Lim, and M. L. Sichitiu, "An asynchronous scheduled MAC protocol for wireless sensor networks," *Comput. Netw.*, vol. 57, no. 1, pp. 85–98, 2013.

[27] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "Minimizing delay and maximizing lifetime for wireless sensor networks with anycast," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 515–528, Apr. 2010.

[28] T. V. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. ACM SenSys*, Los Angeles, CA, USA, 2003, pp. 171–180.

[29] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, "PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks," in *Proc. IEEE INFOCOM*, Shanghai, China, 2011, pp. 1305–1313.

[30] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, "EM-MAC: A dynamic multichannel energy-efficient MAC protocol for wireless sensor networks," in *Proc. ACM MobiHoc*, Paris, France, 2011, pp. 1–11.
[31] D. Niyato, E. Hossain, M. M. Rashid, and V. K. Bhargava, "Wireless sensor networks with energy harvesting technologies: A game-theoretic approach to optimal energy management," *IEEE Wireless Commun.*, vol. 14, no. 4, pp. 90–96, Aug. 2007.
[32] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette, "Evolutionary algorithms for reinforcement learning," *J. Artif. Intell. Res.*, vol. 11, no. 1, pp. 241–276, 1999.
[33] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, 1996.
[34] Y. Gong *et al.*, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Appl. Soft Comput.*, vol. 34, pp. 286–300, Sep. 2015.
[35] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.
[36] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[37] S. Abdallah and V. Lesser, "Learning the task allocation game," in *Proc. AAMAS*, Hakodate, Japan, 2006, pp. 850–857.
[38] S. Abdallah and V. Lesser, "Multiagent reinforcement learning and self-organization in a network of agents," in *Proc. AAMAS*, Honolulu, HI, USA, May 2007, pp. 172–179.
[39] C. Zhang, V. Lesser, and P. Shenoy, "A multi-agent learning approach to online distributed resource allocation," in *Proc. IJCAI*, Pasadena, CA, USA, Jul. 2009, pp. 361–366.
[40] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artif. Intell.*, vol. 136, no. 2, pp. 215–250, 2002.
[41] E. R. Gomes and R. Kowalczyk, "Dynamic analysis of multiagent Q-learning with e-greedy exploration," in *Proc. ICML*, Montreal, QC, Canada, 2009, pp. 369–376.
[42] A. Galstyan, K. Czajkowski, and K. Lerman, "Resource allocation in the grid using reinforcement learning," in *Proc. AAMAS*, New York, NY, USA, Aug. 2004, pp. 1314–1315.
[43] E. R. Gomes and R. Kowalczyk, "Learning the IPA market with individual and social rewards," in *Proc. Int. Conf. Intell. Agent Technol. (IAT)*, Fremont, CA, USA, 2007, pp. 328–334.
[44] N. P. Ziogos, A. C. Tellidou, V. P. Gountis, and A. G. Bakirtzis, "A reinforcement learning algorithm for market participants in FTR auctions," in *Proc. 7th IEEE Power Tech. Conf.*, Lausanne, Switzerland, 2007, pp. 943–948.
[45] S. P. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 287–308, 2000.
[46] S. P. Singh, M. J. Kearns, and Y. Mansour, "Nash convergence of gradient dynamics in general-sum games," in *Proc. UAI*, Stanford, CA, USA, 2000, pp. 541–548.
[47] D. Koller and N. Friedman, *Probabilistic Graphical Models*. Cambridge, MA, USA: MIT Press, 2009.
[48] Y. Sun, D. Shu, O. Gurewitz, and D. B. Johnson, "DW-MAC: A low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks," in *Proc. ACM MobiHoc*, Hong Kong, 2008, pp. 53–62.
[49] C.-C. Hsu, M.-S. Kuo, S.-C. Wang, and C.-F. Chou, "Joint design of asynchronous sleep-wake scheduling and opportunistic routing in wireless sensor networks," *IEEE Trans. Comput.*, vol. 63, no. 7, pp. 1840–1846, Jul. 2014.
[50] D. E. Knuth, "The linear congruential method," in *The Art of Computer Programming*, vol. 2, 3rd ed. Reading, MA, USA: Addison-Wesley, 1997.
[51] S. M. Hedetniemi, S. T. Hedetniemiand, and A. L. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks*, vol. 18, no. 4, pp. 319–349, 1988.
[52] M. Mihaylov, Y.-A. L. Borgne, K. Tuyls, and A. Nowe, "Decentralised reinforcement learning for energy-efficient scheduling in wireless sensor networks," *Int. J. Commun. Netw. Distrib. Syst.*, vol. 9, nos. 3–4, pp. 207–224, 2012.

Dayong Ye received the M.Sc. and Ph.D. degrees from the University of Wollongong, Wollongong, NSW, Australia, in 2009 and 2013, respectively.

He is currently a Research Fellow of Computer Science with the Swinburne University of Technology, Melbourne, VIC, Australia. His current research interests include service-oriented computing, self-organization, and multiagent systems.

Minjie Zhang received the B.Sc. degree from Fudan University, Shanghai, China, in 1982, and the Ph.D. degree in computer science from the University of New England, Armidale, NSW, Australia, in 1996.

She is currently an Associate Professor of Computer Science with the University of Wollongong, Wollongong, NSW, Australia. Her current research interests include multiagent systems and agent-based simulation and modeling in complex domains.