Advances in Aeronautical Science and Engineering
ISSN: 1674-8190

Optimizing Energy Efficiency in Wireless Sensor NetworksG.

A. Meera
Lecturer, Department of Computer Science, Tamil Nadu College of Engineering, Coimbatore, Tamil Nadu, India

***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 trade-off. 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, wire-less sensor networks (WSNs).**

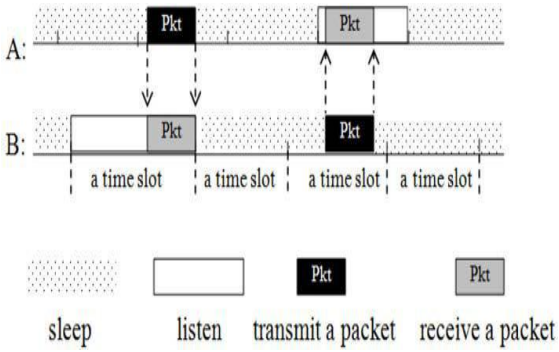
INTRODUCTION

In the last years, wireless sensor networks (WSNs) have gained increasing attention from both the research community and actual users. As sensor nodes are generally battery-powered devices, the critical aspects to face concern how to reduce the energy consumption of nodes, so that the network lifetime can be extended to reasonable times. In this paper we first break down the energy consumption for the components of a typical sensor node, and discuss the main directions to energy conservation in WSNs. Then, we present a systematic and comprehensive taxonomy of the energy conservation schemes, which are subsequently discussed in depth. Special attention

has been devoted to promising solutions which have not yet obtained a wide attention in the literature, such as techniques for energy efficient data acquisition. Finally we conclude the paper with insights for research directions about energy conservation in WSNs.

our proposed algorithm performs scheduling that is dependent on traffic loads. Nodes adapt their sleep/wake schedule based on traffic loads in response to three important factors, (a) the distance of the node from the sink node, (b) the importance of the node's

location from connectivity's perspective, and (c) if the node is in the proximity where an event occurs. Using these heuristics, the proposed scheme reduces end-to-end delay and maximizes the throughput by minimizing the congestion at nodes having heavy traffic load.



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*).

ALGORITHM

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

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

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 any-thing, while a factor of 1 means that the node considers only the most recent information . A discount factor determines the importance of future rewards. The value of a discount fac-tor is in the range [0, 1]. A factor of 0 means that the node is myopic by only considering current rewards, while a fac-tor approaching 1 means that the node strives for a long-term high reward. 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 *trans-mit*, 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.
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 divid-ing the total number of actions . The former calculation method, however, is more efficient and is more widely used than the latter one . 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 *k*th 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 meth-ods can be used here [40]. Then, at the beginning of the next time slot, the node repeats Algorithm 1

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 neighbours' situations, where such approximation does not need communication with neighbours. 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.

REFERENCES

[1]Y. Xiao et al., "Tight performance bounds of multihop fair access for MAC protocols in wireless sensor networks and underwater sensor net-works," 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 hetero-geneous wireless sensor networks," IEEE/ACM
[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.