Range-Based Sleep Scheduling (RBSS) for Wireless Sensor Networks[1]

Li-Hsing Yen[2]
Department of Computer Science and Information Engineering
National University of Kaohsiung
700, Kaohsiung University Rd., Kaohsiung, Taiwan 811, R.O.C.
lhyen@nuk.edu.tw

Yang-Min Cheng
Department of Computer Science and Information Engineering
Chung Hua University
707, Sec.2, WuFu Rd., Hsinchu, Taiwan 300, R.O.C.
cs88625@csie.chu.edu.tw

Abstract

Sleep scheduling in a wireless sensor network is the process of deciding which nodes are eligible to sleep (enter power-saving mode) after random deployment to conserve energy while retaining network coverage. Most existing approaches toward this problem require sensor's location information, which may be impractical considering costly locating overheads. This paper proposes range-based sleep scheduling (RBSS) protocol which needs sensor-to-sensor distance but no location information. RBSS attempts to approach an optimal sensor selection pattern that demands the fewest working (awake) sensors. Simulation results indicate that RBSS is comparable to its location-based counterpart in terms of coverage quality and the reduction of working sensors.

Keywords: Wireless sensor networks, Energy Efficiency, Network coverage, Network density, Sleep scheduling

1 Introduction

Rapid progress in wireless communications and micro-sensing MEMS technology has enabled the deployment of wireless sensor networks. A wireless sensor network consists of a large number of sensor nodes deployed in a region of interest. Each sensor node is capable of collecting, storing, and processing environmental information, and communicating with other sensors.

The positions of sensor nodes need not be engineered or predetermined [1] for the reason of the enormous number of sensors involved [3] or the need to deploy sensors in inaccessible terrains [1]. Due to technical limitations, each sensor node can detect only events that occur within some range from it. A piece of area in the deployment region is said to be covered if every point in this area is within the sensory range of some sensor. The area that are collectively covered by the set of all sensors is referred to as network coverage.

As sensor nodes are typically powered by batteries, power-conserving techniques are essential to prolong their operation lifetimes. In this paper, we are considering powering off redundant sensors temporarily after random deployment to conserve energy while retaining

[1]This work was supported by the National Science Council, Taiwan, under grant NSC-94-2213-E-216-001.
[2]Corresponding author. Tel: +886-7-5919518, Fax: +886-7-5919514.

sufficient network coverage. This task concerns the determination of redundant nodes as well as the arrangement of sleep time.

Existing rules for determining redundant nodes are mostly location-based [8, 7, 6, 12, 4, 9]. As these rules have sensor's location information, they can ensure 100% network coverage ratio. However, the requirement of location information may not be practical if energy-intensive GPS (Global Positioning System) device is assumed for this purpose. Other approaches determine sleep-eligible sensors based on the count of working neighbors [10], current node density [6], or expected network coverage [11]. These approaches demand no locating devices and are thus more suitable for small-size sensors. However, it is intrinsic that 100% network coverage cannot be guaranteed.

This paper proposes a range-based sleep scheduling protocol called RBSS, which needs no location information. RBSS attempts to approach an optimal sensor deployment pattern that demands the minimal number of working (awake) sensors while preserving 100% network coverage ratio. It requires the ability to estimate transmission distances between neighboring sensors, which can be fulfilled by any range measurement technique. Extended simulations were conducted for performance comparisons among the proposed protocol and its counterparts. The results indicate that our protocol performs nearly well as a location-based scheme can do in terms of coverage quality and the reduction of working sensors.

The remainder of this paper is organized as follows. The next section reviews existing sleep scheduling protocols and Section 3 details RBSS. Experimental results are presented in Section 4. The last section concludes this paper.

2 Related Work

PEAS [10] is an energy-conserving protocol that demands no location information. In PEAS, all nodes are initially sleeping. These nodes wake up asynchronously and then broadcast a probe message. Any working node receiving the message should reply. If an awakening node receives a reply to the probe message, it enters sleep mode again. Otherwise, it becomes a working node for the rest of its operation life. The performance of PEAS heavily depends on *probing range*, the transmission range of the probe message. A small probe range usually leads to high coverage ratio but also a large population of working nodes.

There are also stochastic approaches that determine sleep nodes without location information. In the scheme proposed in [6], all nodes randomly and independently alternate between working and sleep modes on a time-slot basis. Given the probability of a sensor being in working mode, the authors have analyzed the probability of a point in the deployment region being uncovered. In [11], the time periods of working and sleep modes are exponentially distributed random variables. Though the method is stochastic in nature, it is deterministic to set the means of these two distributions for a specific expected network coverage.

Most existing protocols finding redundant nodes require location information. Cărbunar et al. [4] transform the problem of detecting redundant sensors to that of computing the Voronoi diagram that corresponds to the current node deployment, which requires node location information. Xing et al. [9] also exploit Voronoi diagram to ensure k-coverage, which refers to the condition that every point in the deployment region is covered by at least k sensor nodes. They have shown that k-coverage is ensured if every critical point (where two sensor's coverage areas intersect or a sensor's coverage area and a border line intersect) is covered by at least k sensors. Their protocol needs location information of every sensor as well.

A coverage-preserving sleep scheduling scheme presented in [8] demands that each sensor

advertises its location information and listens to advertisements from neighbors. After calculating its coverage and its neighbors', a node can determine if it is eligible to turn off its sensory circuitry without reducing overall network coverage. To avoid potential "coverage hole" due to simultaneous turning off, this scheme uses a back-off protocol which requires each off-duty eligible sensor to listen to other sensor's status advertisements and, if necessary, announce its own after a random back-off time period expires. The behaviors of many other schemes [7, 6, 12] are similar to [8] in that they all require the exchanges of location information and eligibility status. Among them, OGDC [12] aims to arrange a particular deployment pattern of working sensors. It has been shown [12] that, to minimize the population of working sensors while preserving network coverage, the locations of any three neighbor sensors should form an equilateral triangle with side length $\sqrt{3}r_s$, where r_s is the sensory range. Extending this argument, the optimal deployment that demands the minimal number of working sensors is to surround each working sensor S with six working neighbors (called *co-workers*) which collectively form a regular hexagon centered at S with side length $\sqrt{3}r_s$. Provided that the node density is sufficiently high, it is feasible to seek this pattern among deployed sensors.

Network connectedness is another issue that should be addressed when powering off sensors. It has been proven [12, 9] that given 100% coverage ratio, $r_t \geq 2r_s$ suffices to ensure network connectedness, where r_t is the transmission range of every sensor. Many protocols [12, 9] therefore focus on maintaining full coverage and simply set $r_t = 2r_s$ to ensure network connectedness at the same time.

Our approach assumes the availability of a ranging technology that estimates the distance between pair-wise neighbors. Several ranging techniques have been proposed for wireless sensor networks. One possible way is to establish a mathematical or empirical model that describes radio signal's path loss attenuation with distance [2]. A received signal strength indication (RSSI) can thereby be translated into a distance estimate. Another trend of ranging technologies turns signal propagation time into distance information. If the sender and the receiver of a radio signal are precisely time-synchronized, the distance in-between can be derived from the time of arrival (ToA). If two signals (one is RF and the other is acoustic signal, for example) are transmitted simultaneously, the time difference of the arrivals (TDoA) can be used for ranging [5].

3 Range-Based Sleep Scheduling

3.1 Design Rationale

The basic idea behind RBSS is that the optimal deployment can be approached without exact location information. If the transmission range of each sensor in the optimal deployment is uniformly $\sqrt{3}r_s$, S's co-workers are exactly S's neighbors that have the maximum transmission distance to S. S can first search for one such co-worker, say, A, then repeatedly look for nodes that are both the co-workers of S and an already-found co-worker. If the second co-worker found is B (C), the third co-worker will be C or D (B or E). If the third co-worker is B or C, the fourth co-worker will be D or E. In this way, all six co-workers, if exist, can be found without knowing their exact locations.

This approach, however, must face the following issues that arise in the above context.

1. **Job competitions**. When a sensor calls for a neighbor that has the maximum transmission distance to it, more than one node may be qualified to respond as the qualification rule should typically allow a reasonable number of candidates to prevent the

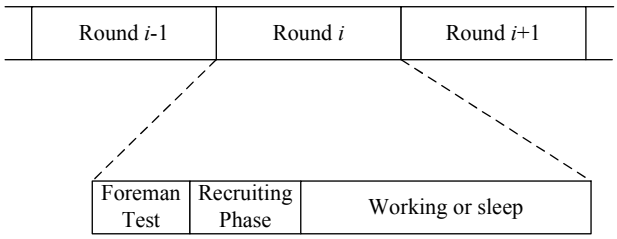

Figure 1: The time structure of a round

extreme case of no response. In that case, it is a challenge how these candidates compete distributively with each other for the job, under the constraint that they are not even aware of whom they are competing with.

2. **Transmission collisions**. Transmission collisions may occur when multiple geographically-related sensors send their requests or responses at the same time. Collision-avoidance techniques such as setting back-off time should be used to minimize the possibility of concurrent transmissions.

3. **Ranging errors**. Signal propagation problems such as environmental interference and multi-path fading introduce estimation errors to almost all existing ranging technologies. The degree of errors is environment dependent. In harsh networking environments, the errors can be so high that make ranging techniques ineffective. The proposed method relies on distance estimate to determine which sensor is preferred when dealing with job competitions. With inaccurate distance measures, sensors that turn out to be co-workers may not be the most appropriate ones. This increases the population of co-workers, degrading the performance of the proposed approach.

The first and the second issues concern spatial and temporal race conditions among sensors, respectively. The proposed protocol uses a number of timers with sophisticated settings to deal with these issues. The third issue is important for practical implementations of the protocol. However, it is very challenging to precisely model the impact of varied terrain types and environments on ranging errors. We also note that most location-based approaches assume a perfect locating technique behind their protocols. As a fair comparison and also a demonstration of the best-case performance, ranging errors are not taken into account currently. This remains to be explored in the future.

3.2 Protocol Description

RBSS divides time into fixed-length time periods called rounds (Fig. 1). Each round begins with a Foreman Test phase, in which every sensor individually determines whether it can become a *foreman*, a node that actively recruits other co-workers. The recruitment protocol is then executed in the following phase. Foremen and co-workers are all working nodes, while others can enter sleep mode in the rest of this round.

Three control messages are used by the protocol: Co-worker Request, Co-worker Response, and Recruitment Done. Every sensor locally maintains two lists: neighbor list and worker list. Both are emptied at the beginning of each round. The neighbor list keeps the ID (identification) of each known neighbor. Whenever a control message is received or overheard from a previously unseen node, the node's ID is added into the list. The worker list of every node records IDs of foremen and co-workers currently known by it (excluding the node itself). Every Co-worker Request sent by a node is attached with the sender's worker list. A sensor that receives or overhears a Co-worker Request adds into its own

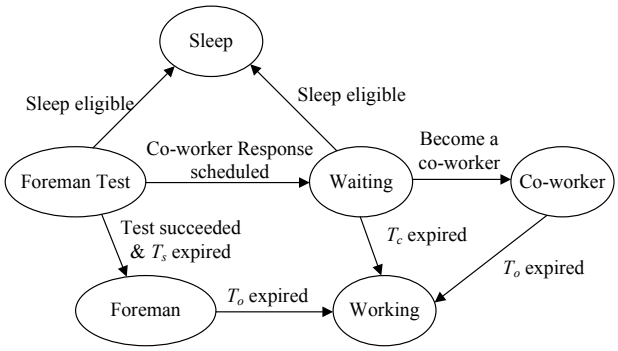

Figure 2: State transition diagram of the proposed protocol

Table 1: Parameter/Timer setting

Parameter/Timer	Meaning/Usage	Value
p_0	Initial probability of a node being a foreman	$1/n$
r_t	Transmission range of each sensor	$\sqrt{3}r_s$ m
T_s	Back-off timer for sending Co-worker Request	$[0, 0.01]$ sec.
T_r	Back-off timer for sending Co-worker Response	$[0, 1.6577]$ sec.
T_o	Timer to protect Co-worker Request	2 sec.
T_d	Additional delay added to T_r in certain cases	0.25 sec.
T_c	Maximum time for a node to stay in Waiting state	5 sec.
D_m	Minimum allowable distance between two working sensors	$r_t/2$ m

Note: An interval value means a value randomly generated within the interval.

worker list all previously unseen IDs associated with the message (including the sender's ID and the sender's worker list).

Figure 2 shows the state transition diagram of RBSS and Table 1 lists settings of some parameters and timers used by the protocol. Each node in the beginning of a round is in Foreman Test state, testing if it can become a foreman. The test is pure stochastic; a node can be a foreman with initial probability p_0, where p_0 is a variable inversely proportional to the node density of the network. This is to limit the expected density of foremen. Each node repeats the test every second until it passes or aborts the test. The instantaneous probability of a node passing the test at a given second exponentially increases with time: it is $\min\{2^{i-1}p_0, 1\}$ in the ith second. A node aborts the test if, before it passes the test, the node hears Co-worker Request or Co-worker Response from one of its neighbors. The receipt of such message implies that some nearby sensor has successfully become a foreman. The node that aborts the foreman test then executes the procedure shown in Fig. 3.

The procedure in Fig. 3 decides whether a node receiving a related message is eligible to sleep or should reply the request. The receiver R is sleep-eligible if it does not contribute substantial coverage to S. This condition consists of two cases: either R is very close to the sender S (specifically, the in-between distance is less than D_m) or more than two of R's neighbors are already in S's worker list. If R is not eligible to sleep, it should reply a Co-worker Response as an application for S's co-worker. A back-off timer for the reply, T_r, is used to resolve potential competitions among qualified applicants. T_r should be set to let the most appropriate node reply first. Intuitively, the most appropriate node should be the one that is most distant from S (but still a neighbor of S) and has the fewest working neighbors. To quantify the appropriateness and reflect it in T_r, the following notations are

```
/* W_S denotes S's worker list */
/* N_R denotes R's neighbor list */
/* d_{i,j} denotes the estimated distance between nodes i and j */
procedure sleep_or_reply()
    if d_{S,R} ≤ D_m then
        enter sleep mode directly
    else
        Ω = W_S ∩ N_R
        if |Ω| > 2 then
            enter sleep mode directly
        T_r = D(d_{S,R}) + Σ_{j∈Ω} D(d_{R,j})
        if |Ω| = 0 and |W_S| > 0 then
            T_r = T_r + T_d
        start T_r
        enter Waiting state
    end if
```

Figure 3: The procedure for node R to process related messages received from S

introduced:

- W_S: S's worker list.
- N_R: R's neighbor list.
- $d_{i,j}$: the estimated distance between nodes i and j.

The value of T_r is set by the following equation

$$T_r = D(d_{S,R}) + \sum_{j \in \Omega} D(d_{R,j}), \tag{1}$$

where $\Omega = W_S \cap N_R$ and function $D(x)$ is defined as

$$D(x) = \left(1 - \exp\left(\frac{x}{r_t} - 1\right)\right).$$

$D(x)$ is inversely proportional to the ratio of x (distance) to r_t (the maximum transmission range). Obviously, $D(x) \to 0$ when $x \to r_t$ and the maximum value of $D(x)$ occurs when $x \to 0$. It should be noted that the definition of $D(x)$ is not unique: it can be any function that shares the same property. An example is $D(x) = 1 - x/r_t$.

By (1), T_r increases as $|W_S \cap N_R|$ increases, but the actual value of T_r also depends on relative distances between R and associated neighbors. When $|\Omega| = 0$, we further differentiate the case $|W_S| > 0$ from that $|W_S| = 0$ by adding an additional delay T_d to T_r in the former case. This gives the latter case a priority in sending the reply.

After T_r is set, R enters Waiting state, in which the Co-worker Response is scheduled to be sent to S when T_r expires. If, before T_r expires, R overhears a Co-worker Response addressed to S from another sensor U, U must be more qualified to be S's co-worker than R for its shorter back-off delay. In that case, R cancels the scheduled response by stopping T_r and updates its neighbor list by including the sender's ID. R then stays in Waiting state because whether it should be active or sleep in this round is still pending.

```
/* Assume that R has a scheduled response to S */
On overhearing Co-worker Response from another node U to S
    stop timer T_r
    set and start timer T_c
    /* remains in Waiting state */

On receiving Co-worker Request from node V
    if V = S and R's ID is in the attached worker list then
        broadcast R's Co-worker Request
        set and start timer T_o
        enter Co-worker state
    else if V ≠ S then
        stop timer T_r
        call sleep_or_reply()
        S ← V /* change the destination of the response to V */
    end if

On receiving Recruitment Done from S
    stop T_r
    enter Sleep state

expired T_r then
    send Co-worker Response to S
    set and start timer T_c
end expired

expired T_c then
    enter Working state
end expired
```

Figure 4: Event handling procedures for node R in Waiting state

In the Waiting state, R may receive a new Co-worker Request from another sensor V, which can be an independent foreman or one of S's co-workers. In either case, R first stops T_r, which effectively cancels the pending response if T_r is still running, and calls procedure sleep_or_reply() shown in Fig. 3 to determine whether the new incoming request makes R eligible to enter sleep mode. If R is not sleep-eligible, the procedure updates and starts T_r accordingly, which schedules a response to the new request. The reason of not responding both requests is that issuing multiple responses complicates the protocol and may increase redundant co-workers. So we simply replace the old request with the new one. Intuitively, such a replacement extends R's stay in the Waiting state and thus gives R more opportunities of being sleep due to the receipt of subsequent messages. However, it also extends the protocol execution time and may increase energy consumption to some extent.

If R receives Recruitment Done from S before it issues the Co-worker Response, meaning that S has recruited adequate co-workers, R aborts the scheduled response and enters sleep mode directly. If R eventually issues Co-worker Response to S, R stays in Waiting state for a maximum period of T_c seconds. If, before T_c expires, R receives a Co-worker Request from S with R's ID in the attached worker list, it means that R's application for co-worker has been approved by S. When this happens, R starts its own

On receiving Co-worker Response from node R
  if $R \notin W_S$ **then**
    $W_S \leftarrow W_S \cup \{R\}$
    stop timer T_o
    broadcast Co-worker Request with the updated W_S
    restart timer T_o
  end if

expired T_o **then**
  broadcast Recruitment Done
  enter Working state
end expired

Figure 5: Event handling procedures for node S in states Foreman and Co-worker

recruitment by broadcasting a new Co-worker Request, setting up timer T_o, and then entering Co-Worker state. R enters Working state directly if no such approval is received before T_c expires. Timer T_c simply serves for the purpose of protecting R from being trapped in this state, which may happen, for example, when R's response collide with other messages. The choice of entering Working rather than Sleep state in this case is to prevent potential coverage hole. Event handling procedures for node R in Waiting state is shown in Fig.4.

Foreman nodes will be awakening for the whole round, consuming considerable energy. Therefore, nearby sensors that are able to cover each other should not be foremen at the same time: some of them are in fact sleep eligible. To this end, every node that passes the foreman test waits T_s seconds before broadcasting Co-worker Request to announce its role as a foreman. The value of T_s is randomly chosen to alleviate potential transmission collisions among nearby sensors. If the node overhears another Co-worker Request before T_s expires, it aborts the scheduled broadcast and executes the procedure in Fig. 3 accordingly as if it did not pass the foreman test. If no Co-worker Request is heard during that interval, the node broadcasts Co-worker Request, starts timer T_o, and then enters Foreman state.

Timer T_o is started right after a node S broadcasts an associated Co-worker Request message and before S enters the Foreman or Co-worker state. T_o will be stopped when the corresponding response is received. The value of T_o is sufficiently large ($T_o > T_r$) so that no corresponding response is expected after T_o expires. Therefore, when T_o does expire, S simply broadcasts Recruitment Done and then enters Working state. If a Co-worker Response from node R is received, S adds R into its worker list, stops T_o, waits some time for additional responses (if any), and then broadcasts a new Co-worker Request with the updated worker list. This gives S another call for additional co-workers and also instructs all the newly-recruited co-workers to start their own recruitment. The detailed procedures are shown in Fig. 5.

4 Experimental Results

We conducted simulations with ns-2 network simulator[3] for performance comparisons among three representative sleep scheduling methods: PEAS, OGDC, and the proposed scheme.

[3]http://www.isi.edu/nsnam/ns/

Table 2: Simulation setup

Parameter	Setting
Network size	50×50 m^2 and 100×100 m^2
Sensor deployment	Random (uniform distribution)
MAC	IEEE 802.11 CSMA/CA
Sensor population	100 – 1000
Sensory range (r_s)	10 m
Communication range (r_t)	$2 \times r_s$ m (PEAS and OGDC) or $\sqrt{3} \times r_s$ m (RBSS)
Probing range (for PEAS)	8, 9, or 10 m
Data transmission rate	60 Kbps

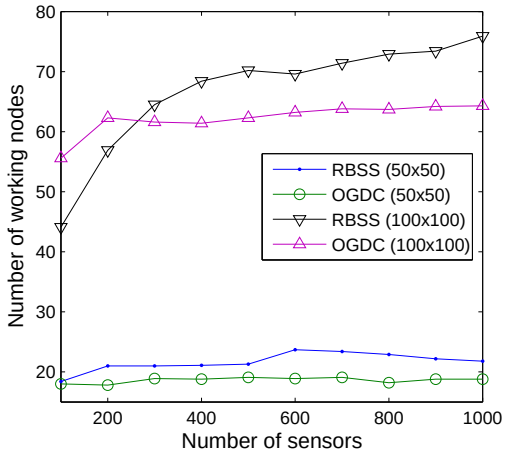

Figure 6: Number of working nodes in 50×50 and 100×100 networks

Since comparisons between OGDC and PEAS have been done already in [12] and are consistent with our results, only results of RBSS and OGDC are shown in the following. Table 2 details the simulation setting.

4.1 Working Nodes and Coverage Ratios

We first measured the number of working nodes. We assumed that all sensors are initially awake and counted the number of working sensors after running a sleep scheduling protocol. Fig. 6 shows the obtained results. All values are averaged over ten experiments.

As can be seen from the figure, OGDC in general yielded fewer working sensors than RBSS. The amount of sensors raised by RBSS ranges from 2% to 25% (average 16%) in 50×50 networks and from -21% to 18% (average 7%) in 100×100 networks. The performance gain of OGDC is due to its knowledge of sensor locations. Both OGDC's and RBSS's results also possess an interesting property: the number of working sensors does not proportionally increase with network size. When we enlarged the network size from 50×50 to 100×100, which was a 400% increase in area, the average increases in sensor populations were only 334% (OGDC) and 307% (RBSS). This can be explained as sensors located in the boundary of the network are unlikely to sleep due to the lack of coverage support from their limited neighbors, and the ratio of boundary nodes becomes small when we enlarge the network size.

To ease the calculation of network coverage, we divided the deployment area into 1×1 grids, where a gird is said to be covered if the center of the grid is covered by some sensor. Coverage ratio is defined to be the ratio of the number of covered grids to the whole. When the network was partitioned, only the largest connected component (the one that covers

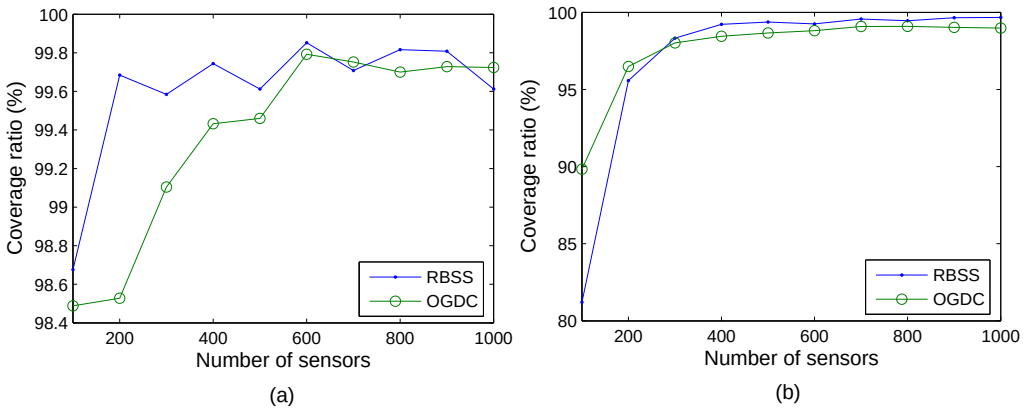

Figure 7: Coverage ratios in a (a) 50×50 and (b) 100×100 network

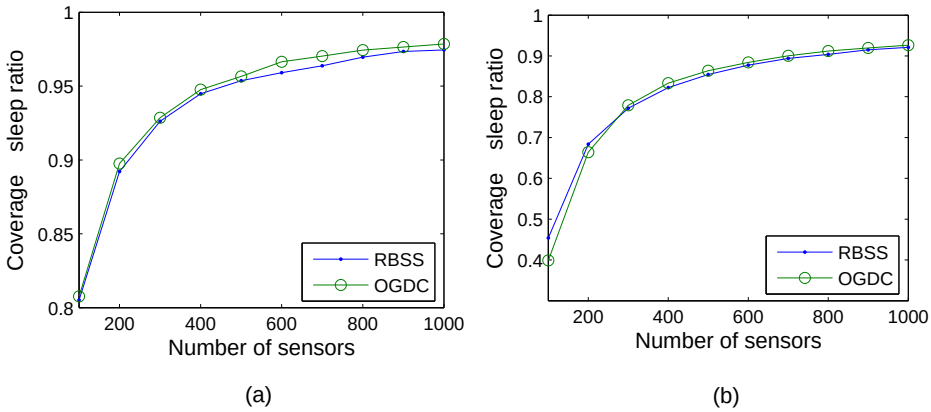

Figure 8: Sleep $\times$ coverage ratio in a (a) 50×50 and (b) 100×100 network

the largest area) was considered in the coverage ratio calculation. Therefore, even though network connectedness was not explicitly gauged, it was reflected by the degree of network coverage. Fig. 7 shows the results averaged over ten experiments.

In 50×50 networks, RBSS outperformed OGDC by -0.11% to 1.17% (average 0.24%). The outperformance of RBSS can also be observed in Fig. 7(b) when the number of sensors is larger than 300. When only 100 sensors were deployed, the node density was so low that RBSS failed to find adequate working sensors (only 44.1 working sensors in average, refer to Fig. 6) to maintain over 90% coverage.

The above results reveal that a sleep scheduling scheme may trade the ratio of sleep sensors for coverage ratio. We therefore propose sleep ratio multiplying coverage ratio as an overall performance index. This index emphasizes the balance between sleep and coverage ratios, as favoring sleep or coverage ratio alone usually does not lead to a high index value.

Figure 8 shows the results for this index. Clearly, RBSS is comparable to OGDC in terms of the proposed performance index.

4.2 Time Domain Comparison

The above comparisons focus on space domain, meaning that all values were measured by running a sleep scheduling protocol right after sensors were deployed. These values actually may change over time, as some sensors may die for energy exhaustion. In light of this, we also made performance comparisons in time domain.

We applied an energy model similar to that used by PEAS [10]. The power consumptions

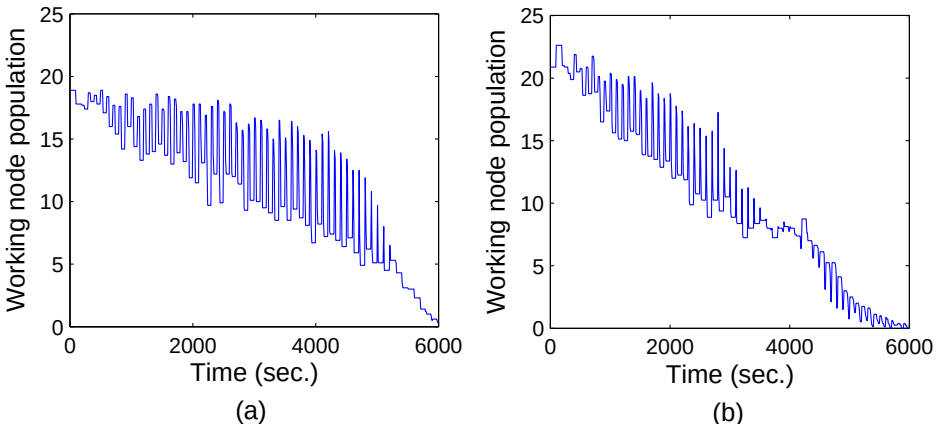

Figure 9: Number of working nodes versus time in a 50×50 network with (a) OGDC and (b) RBSS

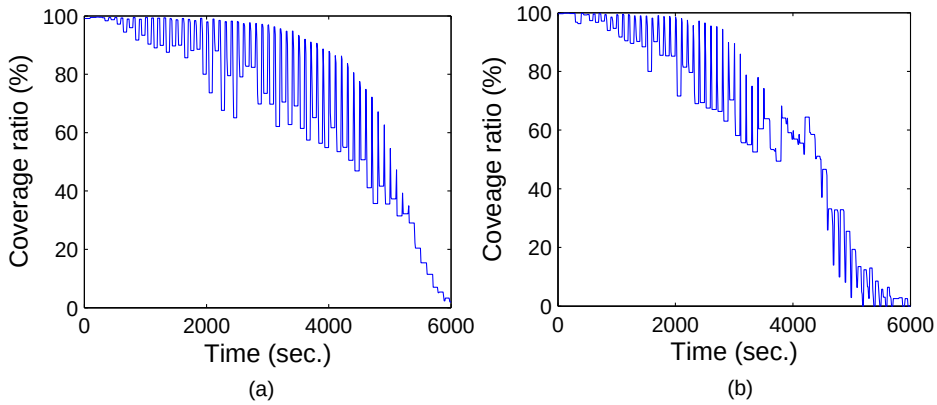

Figure 10: Coverage ratio versus time in a 50×50 network with (a) OGDC and (b) RBSS

in reception, idle, and sleep modes are 4 mW, 4 mW, and 0.01 mW, respectively. The power consumption in transmission mode is 20 mW if $r_t = 20$ m and 16 mW if $r_t = 10 \times \sqrt{3}$ m. For OGDC, the energy consumed in node locating was ignored in our energy model. Total 300 sensors were deployed, each had initial energy of 1 J. We assumed that all sensors are time synchronized, waking up and making powering-off decisions every 100 seconds.

Figure 9 shows how the number of working nodes changed in every ten seconds. The observed periodic fluctuations deserve an explanation. The population of working nodes raised every 100 seconds due to scheduled executions of the protocol. However, working sensors rapidly exhausted their energy, as a working sensor in idle mode dissipates at least 0.4 J per 100 seconds. So the working sensor population dropped even before the next scheduled execution. After nearly 3000 seconds of executions, both methods cannot find out sufficient number of working sensors to maintain coverage. Fig. 10 shows the change of coverage ratio over time. It was observed that the superiority of RBSS over OGDC in terms of coverage (Fig. 7) disappears. The reason is that RBSS hires more working nodes than OGDC initially, resulting in fewer available sensors later.

5 Conclusions

We have reviewed existing sleep scheduling protocols and presented RBSS, a range-based approach. Extended simulations have been conducted for performance comparisons between

RBSS and OGDC, a state-of-the-art location-based counterpart. The results indicate that RBSS performs nearly the same as OGDC when considering both the reduction of working nodes and coverage ratio. Time-domain simulation results show that the proposed protocol consumed a little more energy than OGDC did. But this was obtained when the cost of location incurred by OGDC is not taken into account.

In the future, we shall refine the design to further reduce working sensors and the number of control messages. Timer values and other parameters will be fine tuned to shorten protocol execution time, as more energy can be saved if nodes can enter sleep more earlier. The effects of ranging errors shall also be considered.

Acknowledgement

We would like to thank the authors of [12] for kindly providing us the ns-2 source codes of OGDC.

References

[1] Ian F. Akyìldìz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Commun. Magazine*, 40(8):102–114, Aug. 2002.

[2] P. Bahl and V.N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proc. IEEE INFOCOM 2000*, pp. 775–784, Mar. 2000.

[3] Nirupama Bulusu, Deborah Estrin, Lewis Girod, and John Heidemann. Scalable coordination for wireless sensor networks: Self-configuring localization systems. In *Proc. 6th IEEE Int'l Symp. on Commun. Theory and Application*, Ambleside, U.K., July 2001.

[4] Bogdan Cărbunar, Ananth Grama, Jan Vitek, and Octavian Cărbunar. Coverage preserving redundancy elimination in sensor networks. In *1st IEEE Int'l Conf. on Sensor and Ad Hoc Communications and Networks*, pp. 377–386, Oct. 2004.

[5] Lewis Girod and Deborah Estrin. Robust range estimation using acoustic and multimodal sensing. In *Proc. 2001 IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pp. 1312–1320, Maui, USA, Oct.-Nov. 2001.

[6] Chih-Fan Hsin and Mingyan Liu. Network coverage using low duty-cycled sensors: Random & coordinated sleep algorithms. In *Int'l Symp. on Information Processing in Sensor Networks*, pp. 433–442, Apr. 2004.

[7] Jun Lu and Tatsuya Suda. Coverage-aware self-scheduling in sensor networks. In *Proc. IEEE 18th Annual Workshop on Computer Communications*, pp. 117–123, Oct. 2003.

[8] Di Tian and Nicolas D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *First ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 32–41, 2002.

[9] Guoliang Xing, Xiaorui Wang, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Trans. on Sensor Networks*, 1(1):36–72, Aug. 2005.

[10] Fan Ye, Gary Zhong, Jesse Cheng, Songwu Lu, and Lixia Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proc. 23rd Int'l Conf. on Distributed Computing Systems*, pp. 28–37, May 2003.

[11] Li-Hsing Yen, Chang Wu Yu, and Yang-Min Cheng. Expected k-coverage in wireless sensor networks. *Ad Hoc Networks*, 5(4):636–650, Sept. 2006.

[12] Honghai Zhang and Jennifer C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Wireless Ad Hoc and Sensor Networks: An International Journal*, 1(1-2):89–123, Jan. 2005.