2011 International Conference on Parallel Processing Workshops

Time-Synchronized versus Self-Organized K-Coverage Configuration in WSNs

Meng-Chun Wueng*‡ Prasan Kumar Sahoo† I-Shyan Hwang*

*Department of Computer Science and Engineering
Yuan Ze University, Taiwan
E-mail: s929401@mail.yzu.edu.tw and ishwang@saturn.yzu.edu.tw
†Department of Computer Science and Information Engineering
Chang Gung University, Taiwan
E-mail: pksahooind@yahoo.com
‡Chunghwa Telecom Co., Ltd., Taiwan
E-mail: lillian_weng@cht.com.tw

***Abstract*—The K-coverage configuration is widely exploited to monitor critical applications in wireless sensor networks. A major challenge here is how to maximize the system lifetime while preserving high-quality coverage. The existing sleep-scheduling algorithms, classified into time-synchronized and self-organized approaches, either generate many redundant active sensors or incur high computation cost. In this paper, we propose KGS and DKEA algorithms to settle all essential problems of these two approaches respectively. KGS adopts an appropriate scheduling granularity to minimize the number of active sensors. DKEA efficiently determines whether a sensor should stay active by tracing only some decision areas. We further analyzed which approach maximizes the system lifetime of the K-coverage configuration. Experimental results show that, (i) KGS minimizes the average coverage degree among several popular time-synchronized algorithms; (ii) the computation cost of DKEA is only 11% of that of a well-known self-organized algorithm; and (iii) DKEA outperforms KGS in most cases.**

Keywords: K-coverage configuration, Eligibility, Fault tolerance, Wireless sensor networks.

I. Introduction

Wireless sensor networks have been widely exploited to perform cooperative tasks for critical applications, such as the detection of forest fires or toxic gas. Each sensor conducts local observations and sends the results to a specific base station, which produces a global status of the monitored area. Since an individual sensor node is unreliable due to the low-cost hardware design, a higher degree of coverage is necessary to mask faulty sensors and obtain a high confidence in detection [1], [2]. The K-coverage configuration has attracted attention, as it guarantees that each location in an area is covered by at least K active sensors. Yen et al. [3] calculated the expected area of a K-covered region. Sheu et al. [4] addressed the re-deployment issue. Liu et al. [5] performed a K-coverage configuration using directional sensor networks. The coverage and connectivity issue was investigated in [6], [7]. The authors in [8], [9] focused on fault tolerance measurements. Most of the above studies assumed that an area is already K-covered. Although many sleep-scheduling algorithms [10], [6], [11], [12], [13], [14] have been proposed, two conditions for providing full K-coverage and maximizing the system lifetime cannot be simultaneously satisfied.

The existing sleep-scheduling algorithms can be classified into two schemes, including time-synchronized and self-organized approaches. For the former approach, the monitoring period is composed of an initialization phase and a long sensing phase. The sensing phase is further divided into numerous rounds of equal duration. During the initialization phase, each sensor establishes a working schedule by using the algorithms in [11], [15], [16], and then periodically performs a duty cycle in each round. The cost of maintaining the K-coverage configuration is highly reduced. However, a prerequisite is that each sensor must perform time synchronization with its neighbors precisely. Furthermore, since the above algorithms perform the sleep scheduling based on each virtual square grid point or intersection point within the sensing range of a sensor, many sensors must remain active most of the time due to the scheduling granularity. Thus, the average coverage degree is much higher than K. To address the problems existing in the time-synchronized approach, several K-coverage eligibility algorithms [6], [10] were proposed to be executed periodically. For the self-organized approach, the monitoring operation is divided into rounds. Each round begins with an initialization phase, followed by a sensing phase. During the initialization, each sensor performs an eligibility algorithm to determine whether it is *eligible* to stay active in the sensing phase. Compared with time-synchronized approach, the self-organized approach minimizes the coverage redundancy of the K-coverage configuration, but how to provide a low cost eligibility algorithm requires further enhancement.

In this paper, we propose two algorithms that settle the problems existing in the time-synchronized and self-organized approaches. For the former approach, we design an efficient K-group scheduling algorithm, called KGS. Unlike [11], [15], [16], the scheduling granularity adopted in KGS is based on grouped sensors. KGS minimizes not only the redundancy of the K-coverage configuration, but also the cost of maintaining connectivity among the sensors. In addition, the latency of reporting events is highly reduced. For the self-organized approach, we propose a distributed energy-efficient K-coverage eligibility algorithm (DKEA), which is capable of finding out a region with a lower coverage degree within the sensing range of each sensor. In contract to CCP [6], DKEA traces only the intersection points surrounding candidate regions rather than all intersection points. Extensive experimental results show that, (1) KGS minimizes redundant active sensors among several popular time-synchronized algorithms; (2) the computation cost of DKEA is only 11% of that of CCP, while guaranteeing the K-coverage configuration; and (3) DKEA outperforms KGS because of the characteristic of the self-organization approach.

The next section reviews related work. Sections 3 and 4 introduce proposed time-synchronized and self-organized algorithms, KGS and DKEA. Simulation results are presented in Section 5. We conclude this paper in Section 6.

II. Related Work

In [11], [15], [16], each sensor establishes its duty cycle only in the initialization phase, and then performs the scheduling in the sensing phase. Yan et al. [11], [15] partitioned the sensing range of each sensor into lots of virtual square grid points. If several sensors cover the same grid point, all will monitor it for an equal duration, in turn, to balance power consumption in each round. However, for each of the sensors, since the grid points within its sensing range may be covered by different sets of its neighbors, in order to satisfy all schedules of the grid points, the sensor must stay active most of the time. The system lifetime is highly shortened due to such scheduling granularity. To address this problem, Huang et al. [16] generates a working schedule for each sensor based on the intersection points within the sensing range, which adopted the proof presented in [6]. A sensor is K-covered if all points, intersected by the sensing ranges of the sensor's neighbors and located within its sensing range, are K-covered. For a sparse wireless sensor network, the intersection-based algorithm [16] effectively enhances the performance of the grid-based algorithm [11], [15]. However, the complexity of the intersection-based algorithm is $O(n^3)$, where n denotes the number of a sensor's neighbors. When node density is high, the number of intersection points within the sensing range of a sensor is much higher than that of the grid points contrarily.

Wang et al. [6] proved that a monitored area is K-covered if all points, which are located in the area and intersected by (1) the sensing ranges of any two deployed sensors or (2) the sensing range of a sensor with monitored boundaries, are K-covered. Hence, they proposed a K-coverage eligibility algorithm, called Coverage Configuration Protocol (CCP), which traces all the intersection points within the sensing range of each sensor. CCP accurately determines the eligibility of each sensor. However, the complexity of CCP is too high to perform long-term monitoring. To reduce the complexity of CCP, Huang et al. [10] proposed the K-perimeter-covered (KPC) algorithm, which calculates only the perimeter coverage of a sensor. The complexity of this operation is effectively reduced, but the eligibility of each sensor must be determined after all its neighbors evaluate their perimeter coverage. Consequently, KPC generates extra communication cost instead.

III. KGS Algorithm

To settle all the problems existing in [11], [15], [16], KGS schedules grouped sensors to work for an equal period, in turn, to balance the power consumption of sensors. KGS has two distinct features. First, KGS classifies the deployed sensors into groups, in which each group is 1-covered and the group members have the minimum overlap with each other. Second, KGS schedules the grouped sensors through an easy model.

For generating 1-covered groups, the sensor with the smallest *node_id* is selected as the main node, which selects one of its neighbors with minimal overlapping as the first reference node. The main node continues to select other reference sensors counter-clockwise. A sensor, which overlaps with the main node and the first reference node, and has the maximum sum of distance to them, is selected as the next reference node. The main node activates sensors until its perimeter is fully covered by the selected reference nodes. The first reference node then becomes the next main node to activate sensors for covering its perimeter. Note that Huang et al. [10] proved that tracing the perimeter coverage on each sensor is at very low cost, especially for 1-coverage. When the monitored area is fully covered, all selected sensors belong to the same group, which guarantees to be 1-covered. Since each of the group members is chosen depending on its distance to existing selected sensors, the overlapping coverage among those sensors is thus minimized. The grouping procedure continues until the selected sensors cannot form 1-coverage. In KGS, a sensor belongs to one group. For the sensors, which do not belong to any group, will go to sleep temporarily for fault tolerant recovery later.

As time passed away, some sensor may stop working silently. KGS adopts the same fault tolerant mechanism used in [11], [16]. Each active sensor sends a heartbeat signal periodically to its neighbors. If a sensor detects the failure of one of its neighbors, it wakes up all the neighbors of the failed sensor to re-schedule the following duty cycle. Since KGS effectively reduces the computation cost of performing the scheduling on grouped sensors, KGS provides a longer system lifetime of the K-coverage configuration than those of [11] and [16]. Detailed performance is in Section 5.

IV. DKEA Algorithm

In this section, we introduce the proposed self-organized algorithm, DKEA, which determines the eligibility of each sensor by checking only candidate regions within the sensing range of each sensor with very low cost. Before describing this algorithm, several assumptions and definitions are presented. All deployed sensors are assumed to be stationary and location-aware. Each sensor has identical sensing range, denoted by *R*. The transmission rang is set to be twice the sensing range to maintain the connectivity among sensors. Considering the signal decay of a sensor node, a point located exactly at the edge of the sensing range of a sensor may not be detected correctly. We assume that point *p* is not covered by sensor *s* when $d(s, p) = R$. The above assumptions were adopted in many studies [15], [10], [6]. The distinct feature of DKEA is that the neighbor set of each sensor is classified into two groups, called R_neighbors and R-2R_neighbors.

Definition 1: R_neighbors and R-2R_neighbors. The R_neighbors of sensor *i* is defined as $R\_neighbors(i) = \{j \mid j \in N, j \neq i, 0 < d(i,j) < R\}$ where *N* is the set of sensors located in the monitored area, and *d(i,j)* denotes the distance between sensor *i* and sensor *j*. The R-2R_neighbors of *i* is defined as $R\text{-}2R\_neighbors(i) = \{j \mid j \in N, j \neq i, R \leq d(i,j) < 2R\}$.

A. R_neighbors and R-2R_neighbors

According to our observations, there are two reasons to classify the neighbor set of a sensor as R_neighbors or R-2R_neighbors. First, the longer the distance from the neighbors to the sensor, the less coverage degree is contributed by them on its sensing range. Hence, if a sensor has only R-2R_neighbors, its coverage degree can be immediately determined as 1. This is because a sensor cannot be fully covered by its R-2R_neighbors based on our assumption. Second, the number of R_neighbors of a sensor is limited to its sensing range. We observed that the eligibilities of many sensors can be determined by tracing only the points intersected by their R_neighbors. Since the number of R_neighbors is only $1/3$ of that of R-2R_neighbors, when the number of the deployed sensors is large enough, the computation cost of many cases is greatly reduced. From the above analyses, we argue that the characteristics of the R_neighbors and R-2R_neighbors of a sensor are worthy of being addressed.

B. Candidate Regions with Lower Coverage Degree

In many cases, DKEA determines the eligibility of a sensor by tracing only the intersection points of their

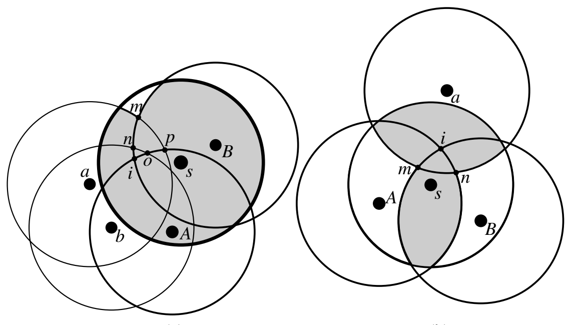

Figure 1. (a) The *candidate intersection point i* of sensor *s* is covered by *candidate R-2R_neighbors a* and *b*. (b) The lower coverage regions of *s* are surrounded by *m* and *n*.

R_neighbors. As the number of the deployed sensors increases, the coverage within the sensing range of each sensor becomes complicated. DKEA efficiently determines their eligibilities by checking the regions with lower coverage degree within their sensing ranges. Such sensors are classified into three categories. The first category is the sensors located near the edge of the monitored area. The intersection points of their R_neighbors may be out of the monitored area. The regions with a lower coverage degree are usually formed by the points intersected by one R_neighbor and one R-2R_neighbor. The second category is that a sensor has one R_neighbor and several R-2R_neighbors. Since one R_neighbor cannot cover the entire sensing range of a sensor, the lower coverage regions may be surrounded by R_neighbors and R-2R_neighbors. Hence, DKEA traces the points intersected by the R_neighbor and R-2R_neighbors. The third category is more complicated and requires further explanation. To clearly explain this case, we first introduce several terms as follows.

- *candidate intersection points*: the points intersected by any two R_neighbors and covered by the fewest R_neighbors.
- *candidate R_neighbors*: the R_neighbors, which form the *candidate intersection points*.
- *candidate R-2R_neighbors*: the R-2R_neighbors, which cover the *candidate intersection points*.
- *decision points*: the points intersected by the *candidate R_neighbors* or *candidate R-2R_neighbors*.

The third case is that a sensor has some *candidate intersection points*, which are covered by several *candidate R-2R_neighbors*, as shown in Fig. 1(a). The lower coverage regions of this sensor can be found by tracing the *decision points* and *candidate intersection points*. Taking Fig. 1(a) as an example, sensor *s* has the lower coverage region surrounded by *m*, which has the minimal coverage degree among the *decision points* (*m*, *n*, *o*, *p*), and the *candidate intersection point* (*i*). This is because the *candidate R-*

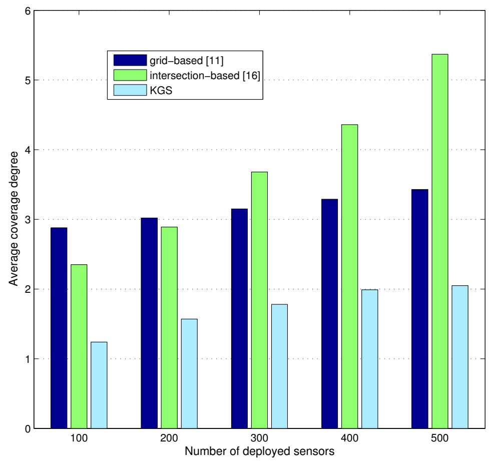

Figure 2. The average coverage degree of performing the time-synchronized approaches for K=1.

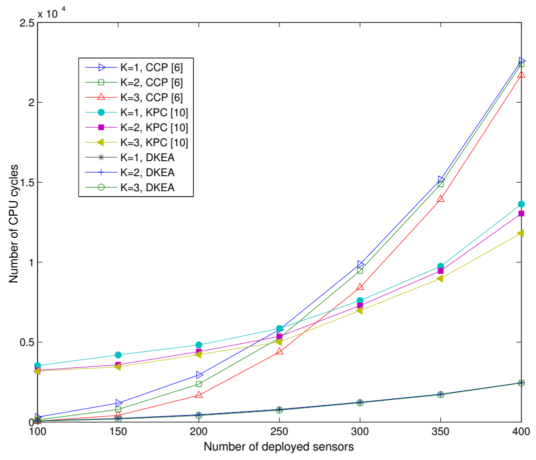

Figure 3. The computation cost of the self-organized algorithms under different K-coverage degrees.

2R_neighbors do not fully cover the lower coverage region. A new lower coverage region will be formed by the *candidate R_neighbors* and *candidate R-2R_neighbors* Oppositely, if the *candidate R-2R_neighbors* cover the region, since the coverage contributed by R-2R_neighbors in the sensing range of a sensor is limited, the lower coverage region is surrounded by *candidate R_neighbors* and *candidate R-2R_neighbors*, as m and n in Fig. 1(b).

V. Performance Evaluation

We evaluated the performance of KGS and DKEA through comparisons with several popular time-synchronized and self-organized algorithms on NS-2 [17]. We further analyzed which approach maximizes the system lifetime of the K-coverage configuration. In the simulation, all sensors with identical sensing range (5m) were deployed uniformly in a 50m×50m square space. Each result reported herein was the average of performing the algorithms on 100 network topologies.

A. Time-Synchronized Approach

Two popular time-synchronized algorithms [11], [16] were implemented for comparison with KGS. All three algorithms performed a lightweight protocol [18] to provide a global time synchronization among the sensors. In the time-synchronized approach, each sensor built its working schedule only in the initialization phase, and then performed the duty cycle in the following long sensing phase. The system lifetime of the K-coverage configuration is greatly affected by how many sensors stay active in each round of the sensing phase. Fig. 2 shows the average coverage degree [1] in a round after performing the three algorithms, in which the grid-based algorithm [11] divided the sensing range of each sensor into 1m×1m virtual grid points. Although the intersection-based algorithm [16] effectively reduced redundant active sensors in a sparse sensor network, it did not solve the problem existing in the grid-based algorithm with high node density. This is because the number of intersection points needed to be traced by each sensor is much higher than that of the virtual grid points. Note that the number of points that should be traced in [11], [16] affects not only the average coverage degree but also the computation cost of performing a K-coverage configuration. Unlike the above two algorithms, KGS performed sleep scheduling on several groups, in which each group is 1-covered with minimum overlap among the group members. Since the scheduling granularity adopted by KGS was based on the sensing range of a sensor, rather than the intersection points or grid points inside it, the average coverage degree in KGS was much lower than that in [11], [16].

B. Self-Organized Approach

Different from the time-synchronized approach, since the K-coverage eligibility algorithm in the self-organized approach is executed periodically in each round, the system lifetime of the K-coverage configuration mainly depends on the performance of the eligibility algorithm. To evaluate the performance of DKEA, we implemented two well-known eligibility algorithms (CCP [6] and KPC [10]) for comparison with it. All algorithms, executed on each sensor, terminate if the coverage degree of a sensor is less than or equal to K during processing. Fig. 3 shows the performance of the three K-coverage eligibility algorithms executed on Avrora [19]. Note that the calculated CPU

[1]The average coverage degree means how many active sensors cover each location in the monitored area.

Table I
THE NUMBER OF ACTIVE SENSORS (AS) AND AVERAGE COVERAGE DEGREE (ACD) OF KGS AND DKEA IN A ROUND.

	300 sensors		400 sensors		500 sensors		600 sensors	
	AS	ACD	AS	ACD	AS	ACD	AS	ACD
K=1, KGS	75.27	1.68	75.87	1.99	76.11	2.05	76.67	2.16
K=1, DKEA	74.33	1.43	74.52	1.47	74.53	1.47	74.76	1.48
K=2, KGS	131.89	2.48	132.12	2.49	132.77	2.51	133.15	2.54
K=2, DKEA	126.27	2.31	126.37	2.33	126.79	2.34	126.98	2.37
K=3, KGS	184.17	3.79	185.47	3.91	185.72	4.02	185.81	4.13
K=3, DKEA	172.67	3.42	173.31	3.51	174.07	3.67	174.18	3.88
K=4, KGS	236.47	4.84	236.17	5.19	237.21	5.23	237.65	5.48
K=4, DKEA	215.38	4.79	216.38	4.95	217.18	5.08	217.53	5.11
K=5, KGS	266.96	5.91	267.18	6.24	268.71	6.35	269.17	6.47
K=5, DKEA	257.22	5.52	258.12	5.58	258.42	5.79	259.55	6.07

cycles of KPC here include not only computation cost but also communication cost. This is because KPC cannot accurately determine the eligibility of each sensor by tracing only its perimeter coverage. Each sensor has to check the perimeter coverage of all its neighbors. On the contrary, both CCP and DKEA accurately determine the eligibility of each sensor from collected neighbor information without generating extra communication cost. Unlike CCP, DKEA determines the eligibility of each sensor by tracing only the intersection points surrounding the regions with lower coverage degree rather than all intersection points within the sensing range. Hence, DKEA is scalable in network size. When we deployed 400 sensors for K=1, the computation cost of DKEA in the best case is only 11% of that of CCP.

C. *KGS vs. DKEA*

KGS and DKEA effectively enhance the performance of the time-synchronized and self-organized approaches. We further analyzed which one maximizes the system lifetime of the K-coverage configuration. How many sensors should stay active in each round was evaluated first. Although KGS reduces the average coverage degree with a group-based scheduling granularity, since DKEA accurately determines the eligibility of each sensor, Table I shows that DKEA reduces more redundant active sensors than that of KGS. Unlike DKEA, KGS groups K 1-covered sensors, in which some sensors may be able to sleep without sacrificing coverage degree. However, extra power consumption will be produced.

For measuring the system lifetime of KGS and DKEA, we calculated the execution rounds from the beginning of the deployment until half of the monitored area is under K-coverage. The energy consumption of each sensor was calculated based on the energy model used in [20]. Compared to DKEA, KGS reduces lots of cost of maintaining the configuration, because each sensor in KGS just follows the duty cycle generated in the initialization phase without performing the eligibility algorithm periodically. However, the time-synchronized approach like KGS needs to pay extra cost for addressing fault tolerance issue. If a sensor fails, all its neighbors need to re-schedule their duty cycles. As time passed away, more and more sensors stop working that cause many sensors to build the sleep scheduling again and again. Hence, the percentage of K-coverage in KGS drops off quickly. Although DKEA also needs to handle fault tolerance issue, it determines the eligibility of each sensor according to the remaining energy, the probability of producing faulty sensors is reduced. Furthermore, since DKEA performs the eligibility algorithm at very low cost, even though each sensor must perform the eligibility algorithm periodically, DKEA preserves a longer system lifetime than that of KGS.

VI. CONCLUSION

In this paper, we investigate several sleep-scheduling algorithms of the K-coverage configuration, which are classified into time-synchronized and self-organized approaches. The contributions of this paper are threefold. First, KGS demonstrates that adopting an appropriate scheduling granularity in the time-synchronized approach effectively minimizes redundant active sensors. Second, DKEA shows that the eligibility of each sensor can be accurately determined with the computation cost as low as only 11% of that of the well-known self-organized algorithm by tracing only some critical intersection points within the sensing range of a sensor. Third, extensive simulation results verify that DKEA outperforms KGS in most cases because of the characteristics of DKEA in the self-organized approach.

REFERENCES

[1] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, and L. Gu, "Energy-Efficient Surveillance System Using Wireless Sensor Networks," in *Proc. of the ACM MobiSys*, 2004, pp. 270–283.

[2] T. Clouqueur, K. K. Saluja, and P. Ramanathan, "Fault Tolerance in Collaborative Sensor Networks for Target Detection," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 320–333, 2004.

[3] L.-H. Yen, C.-W. Yu, and Y.-M. Cheng, "Expected K-coverage in Wireless Sensor Networks," *Ad Hoc Networks*, vol. 5, no. 4, pp. 636–650, 2006.

[4] J.-P. Sheu, G.-Y. Chang, and Y.-T. Chen, "A Novel Approach for k-coverage Rate Evaluation and Re-deployment in Wireless Sensor Networks," in *Proc. of IEEE Globocom*, 2008, pp. 341–345.

[5] L. Liu, H. Ma, and X. Zhang, "On Directional K-Coverage Analysis of Randomly Deployed Camera Sensor Networks," in *Proc. of IEEE ICC*, 2008, pp. 2707–2711.

[6] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated Coverage and Connectivity Configuration for Energy Conservation in Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 36–72, Aug. 2005.

[7] F. Dai and J. Wu, “On constructing k-connected k-dominating set in wireless ad hoc and sensor networks,” *Journal of Parallel and Distributed Computing*, vol. 66, no. 7, pp. 947–958, 2006.

[8] A. Sen, B. Shen, L. Zhao, and B. Hao, “Fault-Tolerance in Sensor Networks: a New Evaluation Metric,” in *Proc. of IEEE INFOCOM*, 2006, pp. 1–12.

[9] H. M. Ammari and S. K. Das, “Fault Tolerance Measures for Large-Scale Wireless Sensor Networks,” *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 1, 2009.

[10] C.-F. Huang and Y.-C. Tseng, “The Coverage Problem in a Wireless Sensor Network,” *Mobile Networks and Applications*, vol. 10, no. 4, pp. 519–528, 2005.

[11] T. Yan, T. He, and J. A. Stankovic, “Differentiated Surveillance for Sensor Networks,” in *Proc. of the ACM SenSys*, 2003, pp. 51–62.

[12] H. M. Ammari and S. K. Das, “On the Design of K-covered Wireless Sensor Networks: Self- versus Triggered Sensor Scheduling,” in *Proc. of IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks and Workshops*, 2009, pp. 1–9.

[13] H. Kim, E. J. Kim, and K. H. Yum, “ROAL: A Randomly Ordered Activation and Layering Protocol for Ensuring K-coverage in Wireless Sensor Networks,” in *Proc. of IEEE ICWMC*, 2007, pp. 4–9.

[14] Y. Li and S. Gao, “Designing K-Coverage Schedules in Wireless Sensor Networks,” *Journal of Combinatorial Optimization*, vol. 15, no. 2, pp. 127–146, 2008.

[15] T. Yan, Y. Gu, T. He, and J. A. Stankovic, “Design and Optimization of Distributed Sensing Coverage in Wireless Sensor Networks,” *ACM Transactions on Embedded Computing Systems*, vol. 7, no. 3, 2008.

[16] C.-F. Huang, L.-C. Lo, Y.-C. Tseng, and W.-T. Chen, “Decentralized Energy-Conserving and Coverage-Preserving Protocols for Wireless Sensor Networks,” *ACM Transactions on Sensor Networks*, vol. 2, no. 2, pp. 182–187, 2006.

[17] UCB/LBNL/VINT, “NS-2 Network Simulation, http://www.isi.edu/nsnam/ns/.”

[18] J. van Greunen and J. Rabaey, “Lightweight Time Synchronization for Sensor Networks,” in *Proc. of ACM WSNA*, 2003, pp. 11–19.

[19] B. L. Titzer, D. K. Lee, and J. Palsberg, “Avrora: Scalable Sensor Network Simulation with Precise Timing,” in *Proc. of the ACM IPSN*, 2005, pp. 477–482.

[20] L. Yu, L. Yuan, G. Qu, and A. Ephremides, “Energy-Driven Detection Scheme with Guaranteed Accuracy,” in *Proc. of ACM/IEEE IPSN*, 2006, pp. 284–291.