Power Efficient Organization of Wireless Sensor Networks

Sasa Slijepcevic, Miodrag Potkonjak
Computer Science Department, University of California, Los Angeles
Los Angeles, CA 90095-1596

***Abstract* -- Wireless sensor networks have emerged recently as an effective way of monitoring remote or inhospitable physical environments. One of the major challenges in devising such networks lies in the constrained energy and computational resources available to sensor nodes. These constraints must be taken into account at all levels of system hierarchy.**

The deployment of sensor nodes is the first step in establishing a sensor network. Since sensor networks contain a large number of sensor nodes, the nodes must be deployed in clusters, where the location of each particular node cannot be fully guaranteed a priori. Therefore, the number of nodes that must be deployed in order to completely cover the whole monitored area is often higher than if a deterministic procedure were used. In networks with stochastically placed nodes, activating only the necessary number of sensor nodes at any particular moment can save energy. We introduce a heuristic that selects mutually exclusive sets of sensor nodes, where the members of each of those sets together completely cover the monitored area. The intervals of activity are the same for all sets, and only one of the sets is active at any time. The experimental results demonstrate that by using only a subset of sensor nodes at each moment, we achieve a significant energy savings while fully preserving coverage.

I. INTRODUCTION

Sensor networks are wireless networks comprised of a large number of miniscule devices equipped with one or more sensors, some processing circuits, and a wireless transceiver. Such devices are called sensor nodes. The dimensions of a sensor node are small enough to allow easy deployment of a large number of nodes into remote and inhospitable areas. Once deployed, sensor nodes organize a network, so that they can combine their partial observations of the environment. By combining those partial observations, a network offers to a user a global view of a monitored area.

The source of energy for a node is most often an attached battery cell. Since the size of a cell is limited, the amount of available energy is also limited. Therefore, sensor network architectures and applications, as well as deployment strategies, must be developed with low energy consumption as one of the important requirements.

We study the problem of the placement of sensor nodes into a monitored area and their organization so that the full coverage is achieved with minimal energy consumption. Specifically, we have developed a heuristic that organizes the available sensor nodes into mutually exclusive sets where the members of each of those sets completely cover the monitored area. Only one such set is active and consumes power at any moment. After a specified interval another set is activated, while the first one is deactivated. In one round all sets are used, and then the whole process repeats until the sensors are out of power. The lifetime of the whole system directly corresponds to the number of allocated sets; therefore, the goal of the algorithm is to maximize the number of the sets.

The first phase of the algorithm determines how many different sensor nodes cover the different parts of the monitored area. The second phase allocates sensor nodes into mutually independent sets. For simulation purposes, we assume that the sensing area of a sensor is a circle with the radius of the circle equal to the sensing range of a sensor. However, the heuristic does not require a circular sensing range. The heuristic is based on the assumption that for each sensor the sensing range of any shape can be either determined before the deployment as a static approximation of the sensor's capabilities or as a function of a specific location and surrounding environment after the deployment of a sensor node. Furthermore, the sensing range may depend on the observed target, e.g. a seismic sensor can detect a tank at a larger distance than it can detect a soldier on foot.

II. RELATED WORK

Research in sensor networks is motivated by the capability of sensor networks to closely connect the physical world and the existing Internet infrastructure, as described in [1][2][3]. To create such networks, the low power hardware components and customized system architectures are necessary. The issue of low power consumption in sensor networks is denoted as one of the most important requirements for sensor networks in [4][5][6]. Low power hardware components and general sensor network architecture are developed in WINS project [6][7] at UCLA. Low power sensing and processing are achieved by taking into account specifics of sensor networks, namely higher tolerance to latency and low sampling rates. Besides sensor node platforms based on radio communication, the "SmartDust" project [4] uses *motes*, small size devices with optical communication capabilities. New low power network protocols and architectures customized for sensor networks are proposed in [5][8][9].

In the last decade, techniques for low power design and compilation have been attracting a great deal of attention. Numerous techniques for reducing power consumption using either architecture and integrated circuit design approaches or compilation and operating system schemes have been proposed [10]. The most relevant result to our approach is that batteries have approximately twice as long life time, if they are discharged in short bursts with significant off time than in the continuous mode of operation [11]. Therefore, a mode of operation of a network where sensor nodes frequently oscillate between an active and an inactive state extends a battery lifetime.

The problem of efficient coverage of an indoor area with base stations for wireless networks resembles the problem of covering an area with sensor nodes. Optimal positions of base stations are acquired by running subsequent simulations beginning with an initial placement of stations and then using genetic algorithms [12] or simulated annealing [13] to generate new sets of positions until one solution is accepted. However, there are significant differences that do not allow straightforward implementation of the given procedure in sensor networks. The number of sensor nodes is significantly higher than the number of the stations. Also, the sensor nodes are chosen from the set of already existing locations, while in mentioned works the locations can be chosen freely.

Since we are modeling an area where a sensor detects objects as a circle, the problem can be seen as an instance of the circle covering problem discussed in [14][15]. The

solution for that problem requires solving complex equations [14] and it can be done in a reasonable time for only a small number of circles [15].

III. MOTIVATION

A. Controlled vs. Random Node Placement

There are two main reasons why a deterministic placement of sensor nodes is impractical:

- Sensor networks are often deployed in remote or inhospitable areas, which prevent individual sensor node deployment. Also, the current state-of-the-art sensor nodes are not capable of dynamic adjustment of their positions.
- The number of sensor nodes in a network is large, so the deterministic placement is associated with increased cost and latency in the deployment of the network.

Therefore, the preferred method of sensor placement is bulk dispersion of sensor nodes from an aircraft [9]. Still, it is instructive to consider the deterministic case when we can control placement of nodes since it provides the lowest bound on the required number of nodes needed to cover the area.

Fig. 1. shows the optimal regular placement of nodes, so that the whole area is covered with the minimal number of nodes. The minimal number of sensor nodes is given by the equation [16]:

$$\frac{N * r^2 \pi}{P_{AREA}} = \frac{2\pi}{\sqrt{27}},$$

where P_{AREA} is the size of the monitored area, N is the minimal number of nodes needed to cover the area, and r is the sensing range of a sensor in a sensor node. It is assumed that the sensing range of the nodes is significantly smaller than the dimensions of the monitored area.

B. Motivational Example

Here, we further clarify the problem discussed in this paper using the following example. Let us assume that the monitored area is divided into five fields. (A detailed description of a field is given in the following section.) Let F be a set of those parts: $\{F_1, F_2, F_3, F_4, F_5\}$. In a general case, each sensor covers one or more of these fields and each field is observed by at least one sensor node. In our example, we have six nodes, denoted by S_i, $1 \le i \le 6$. The sensors cover the following fields: $S_1=\{F_1, F_2, F_3, F_4\}$, $S_2=\{F_1, F_2, F_5\}$, $S_3=\{F_2, F_3, F_4, F_5\}$, $S_4=\{F_2, F_3, F_5\}$, $S_5=\{F_1, F_3, F_5\}$, and $S_6=\{F_3, F_4, F_5\}$.

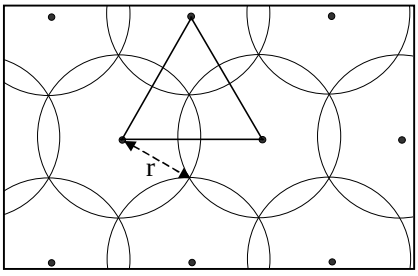

Fig. 1. Sensor nodes with a sensing range ***r*** placed in a regular structure that ensures the coverage of the whole area with a minimal number of nodes. The lines connecting three nodes whose sensing ranges intersect form an equilateral triangle. The side of the triangle is $\boldsymbol{r}\sqrt{3}$.

We allocate sensor nodes into covers, mutually exclusive sets of sensor nodes, where each cover $C_{i,\ i=1\ldots N,}$ completely covers the area A. Our goal is to maximize the cardinality of the set of covers $C = \{C_1,\ldots,C_N\}$. For this simple example, the optimal solution is found using the exhaustive search:

$C_1 = \{S_1, S_4\}$, $C_2 = \{S_3, S_5\}$, $C_3 = \{S_2, S_6\}$.

The heuristic we developed to solve the problem is based on the maximally constrained – minimally constraining paradigm. We first try to cover fields that are covered by a small number of sensors. In the example, the fields that are covered by the minimal number of sensors are F_1 and F_4. Only three nodes cover each of those fields. Later we try to avoid including into the same cover the sensors that cover those sparsely covered fields. The detailed description of the heuristic is given in the following section.

IV. COVERAGE OPTIMIZATION

In this section we present the SET K-COVER problem, which is a generalization of the coverage problem described in Section III. In the first subsection, we introduce the preliminaries, and provide relevant definitions. In the second subsection, we present the heuristic approach for solving the SET K-COVER problem.

A. Problem Formulation

The first step in solving the problem is to identify the parts of the area covered by different sensor nodes. A straightforward approach to the question would be to treat each point in the area (assuming some finite resolution) as a distinctive part of the area. In that case a sensor node would be characterized by a list of all points it covers. In order to reduce dimensionality of the problem, we introduce notion of the field, defined in the following way:

DEFINITION: *A field is a set of points. Two points belong to the same field iff they are covered by the same set of sensors.*

An example how points are organized in fields is given in Fig. 2. The part of the algorithm that determines fields is shown in Fig. 3. The algorithm uses the locations of the nodes as the input, and allocates the points in the area into fields. Whenever a new sensor is added to the set of sensors, the status of all points covered by that sensor is examined. The new sensor may cover some points that were not previously covered by any sensor, as well as points already belonging to some established fields. All points previously not covered belong to one new field. A part of an existing field or a whole existing field covered by the new sensor becomes a new field, whose covering set consists of the covering set of sensors for the original old field with addition of the new sensor.

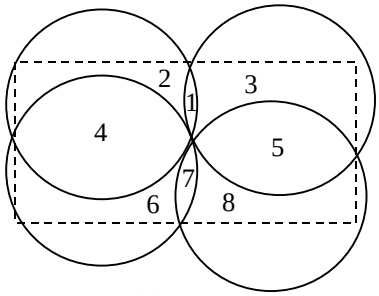

Fig. 2. After the locations of four sensor nodes are known, the area is divided in eight fields, where each field contains points covered by a same set of sensors.

```
for each sensor {
 go through all points covered by that sensor {
    if (the point was not previously covered) {
       if (a field for previously uncovered points doesn't exist)
          add new field that will cover previously uncovered points;
       add the point to the field that covers previously uncovered points;
    }
    else if (the point already belongs to some of existing fields) {
       if (no new field corresponding to old field that contains the point)
            add new field for that old field;
       add the point to the new field corresponding to the old field
          that contains the point;
    }
 }
 add all new fields to the list of the fields covered by the new sensor;
 for all new fields {
     add to the list of sensors covering that field all sensors
        covering the old field corresponding to that new field;
     add new sensor to the list of sensors covering that field;
 }
}
```

Fig. 3. The algorithm that organizes points into fields.

After the fields are established, for each sensor a list of all fields covered by that sensor is created. The set of all fields in the area is denoted as A, and set of sensors as C. We can now formally define the SET K-COVER problem:

PROBLEM: SET K-COVER

INSTANCE: Collection C of subsets of a set A, positive integer K.

QUESTION: Does C contain K disjoint covers for A, i.e. covers C_1, C_2,..., C_k, where $C_i \subseteq C$ such that every element of A belongs to at least one member of each of C_i?

We have proved that the SET K-COVER is NP-complete problem using polynomial time transformation from the minimum cover problem [17]. For the sake of brevity the proof is omitted.

B. Most Constrained – Minimally Constraining Covering Heuristic

In this subsection we present a heuristic solution for the SET K-COVER problem, first at the intuitive level and then with all the main technical details. The theoretical analyses of the heuristic indicate that the worst runtime is $O(N^2)$, where N is the number of deployed sensor nodes.

The basic idea of the heuristic is to minimize the coverage of sparsely covered areas within one cover. Such areas are identified using the notion of the *critical element.*

DEFINITION: *A **Critical element** is such element from A that is a member of the minimal number of the sets in C.*

A *critical element* determines an upper bound on K, the number of covers. During the execution of the heuristic, whenever a cover is selected, all members of that cover are removed from the C. Therefore, the coverage of the elements from A changes and a new *critical element* must be determined. If there is more than one candidate for the *critical element*, any of them can be selected.

A *critical element* is determined at the beginning of each step of the heuristic. For each set from C that covers the current *critical element*, a value of the objective function is calculated. A value of the objective function for a set measures the likelihood that if the set is chosen into the current cover C_i, C_i will redundantly cover some of the sparsely covered parts of the area. A higher value of the objective function means that the likelihood is lower. Therefore, the sets with higher value of the objective function are selected into the current cover. The number of other sets from C that cover the elements from the set whose objective function is evaluated approximates that likelihood.

The pseudocode of the heuristic is shown in Fig. 4. The solution is given as a set of covers Ci, (i=1, 2,…). One cover is selected from the collection C in each pass. All sets from C belonging to the chosen cover are removed from C after each pass (line 20). Remaining members of C represent a pool for the following pass.

At the beginning of the pass *i* of the heuristic, the set of unmarked elements U contains all elements from A, and all current members of C are selected into V, the set of available sets (line 3). Members of V can be selected into the cover C_i.

At each step within the pass *i* (lines 4-20), as long as there are unmarked elements in U, one member of V is chosen into the cover Ci. A *critical element* e_{min}, which is the member of the smallest number of the available sets from V, is selected from U at line 5. For all sets V_i from V that contain e_{min}, values of the objective function *f* are calculated (lines 7-15). The objective function for a set V_i is given as

$$f(V_i) = \sum_{\substack{e \in A, e \in V_i, \\ e \in U, e \neq e_{min}}} \{M - K * (\#\text{ of } V_j \in V \mid e \in V_j)\} - \sum_{\substack{e \in A, e \in V_i, \\ e \notin U, e \neq e_{min}}} \{N - L * (\#\text{ of } V_j \in C - C_i \mid e \in V_j)\}$$

The first sum within the objective function for V_i adds the value M for each uncovered element from A that is a member of V_i. Also, for each element from V_i, the number of other available sets covering that element is taken into account. The number of available sets that an uncovered element is a member of decreases the value of the objective function for V_i. The purpose of such approach is to select a set that will cover a largest number of sparsely covered elements.

The second sum subtracts the value N from the objective function for each element that is already covered, i.e. is not an element of the set U. The second part of the second sum measures how many still available sets are covering the elements that the set V_i will redundantly cover if chosen into Ci. If some elements have to be covered by more than one element from Ci, it is better to redundantly cover elements

```
1)  Given A, C, i = 1
2)  while (each element from A covered by a set from C)
3)    Ci = ∅; U = A; V = C
4)    while (U ≠ ∅)
5)      find emin ∈ U such that emin is the member of minimal sets from V
6)      max = - MAXINT
7)      for each Vi ∈ V such that emin ∈ Vi
8)        f(Vi) = 0
9)        for each ei ∈ Vi
10)         if (ei ∈ U)
11)         then f(Vi) = f(Vi) + M - K*(# of Vj ∈ V such that ei ∈ Vj)
12)         else f(Vi) = f(Vi) - N + L*(# of Vj ∈ C-Ci such that ei ∈ Vj)
13)       end for
14)       if f(Vi) > max then Vmax = Vi; max = f(Vi);
15)       V = V – Vi
16)     end for
17)     Ci = Ci + Vmax
18)     U = U – (∀e ∈ Vmax)
19)   end while
20)   C = C – Ci i = i + 1
21) end while
```

Fig. 4. The heuristic that determines the covers.

that are covered by many other available sets in C. Such elements are unlikely to become critical elements in the following passes of the heuristic.

The incentive of such objective function is to (1) favor sets that cover a high number of uncovered elements by adding the value M for each uncovered element, (2) favor sets that cover more sparsely covered elements by subtracting number of available sets for a particular element, where for sparsely covered elements a subtracted value will be low, (3) favor sets that do not cover the area redundantly by subtracting N for each already covered element, and (4) favor sets that redundantly cover the elements that do not belong to sparsely covered areas, by adding number of sets that can cover that element in the following covers.

The set that contains the element e_{min} and has a maximum value of the objective function is chosen into the cover Ci (line 20) after the evaluation of the objective function. All elements from the chosen set are deleted from the set of the unmarked elements U (line 21). All sets covering the *critical element* are marked as unavailable for that cover (line 18). Therefore, all sets containing any of the elements that were *critical elements* in previous steps of generating the current cover are unavailable in following steps during one pass of the heuristic.

Let us prove now that the given heuristic generates a cover if at the beginning of a pass for each of the elements from A there is a set from V covering that element. A selected *critical element* at any step of the heuristics has *k* sets that cover that element, $k>0$. All other still uncovered elements of A are covered by *k* or more sets. After a set is chosen, all *k* sets containing the critical element are marked as unavailable. If there is another element that was covered by the same number of sets as the *critical element*, that another element must have had at least one set that covers that element, but not the *critical element*. All other elements covered by more than *k* sets must have at least one set available after *k* sets are marked unavailable. The one that have the minimal number of available sets and is not covered by chosen set is chosen as the next *critical element*. Therefore, after each step for each element is true that a) that element is already covered, or b) there is an available set in C that covers that element. Consequently, if at the beginning of the pass for each element there is a set that covers that element, the heuristic can create a cover C_i.

When the set of the unmarked elements U is empty, one pass of the heuristic is finished, and the cover C_i covers all elements from A. All members of the cover C_i are deleted from C (line 23). The heuristic exits if at the beginning of a pass there is an element from A that is not covered by any of the members of C.

V. Experimental Results

The performance of the heuristic in solving the SET K-COVER problem is compared to the performance of an implementation of simulated annealing [18]. In this section we describe that implementation, the simulation environment, and the simulation results.

A. Simulated Annealing - Parameters

Since we are not aware of any other algorithm or a heuristic that solves the SET K-COVER problem, we have developed a simulated annealing based heuristic to get an approximate measurement of efficiency of the most constrained – minimally constraining heuristic. An execution *j* of the simulated annealing heuristic allocates the sets from C into n_j subcollections where n_j is constant for the j^{th} execution. The number of subcollections that completely cover the set A measures the quality of the solution of the simulated annealing based heuristic.

An initial state of the system is generated by a random distribution of the sets from C into n_j subcollections C_1, C_2, ... C_{nj}. One move of simulated annealing is implemented by choosing randomly one of the n_j subcollections and by choosing a set from that subcollection. The chosen set is moved to another randomly selected subcollection. The cost function of a state is given as:

$$CF = a \cdot \sum_{i=1}^{kj} 1 + b \cdot \sum_{i=k+1}^{nj} 1/m_i$$

where the subcollections $i=1,..,k_j$ are those that cover all elements from A. For the subcollections $i=k_j+1,..,n_j$ that do not cover all elements from A, the number of uncovered elements from A m_i is calculated. If a new solution has a higher value of the cost function than the current solution, the new solution is accepted. If the value of the cost function for the new solution is lower than the value for the current solution, the new solution is accepted as a current solution with a probability that decreases with the temperature of the system. According to the simulated annealing strategy, the temperature of the system is highest at the beginning and then gradually decreases. When the system cools down, the final solution is accepted as the best solution for that execution of simulated annealing. The motivation for such cost function is that those subcollections that do not cover the set A should add to the value of the cost function depending on the number of elements of A that are not covered by that subcollection.

After an execution of the simulated annealing algorithm, the number of subcollections n_{j+1} that is attempted in the next execution depends on:

- k_j, the number of the subcollections from the j^{th} execution that cover the set A,
- k_{max}, the maximum number of the subcollections achieved in all previous executions,
- *upperBound*, the number of sets from the collection C containing the element covered by minimum number of sets, e_{min},

in the following way:

$$k_j = n_j \Rightarrow n_{j+1} = \min(upperBound, 2 \cdot n_j)$$

$$k_j < n_j \Rightarrow n_{j+1} = \max(\frac{k_j + n_j}{2}, k_{max} + 1)$$

If the number of the subcollections covering A is equal to n_j, the next execution attempts to achieve higher number of the subcollections that cover A, while in the case where k_j is smaller than n_j, the attempted number of subcollections for the next execution of simulated annealing is decreased, but never below an already achieved best result k_{max}. The simulation stops if after *limit* number of attempts k_{max}, the achieved number of subcollections covering A, does not change.

B. Simulation Environment and Results

The simulations were performed on Ultra5 Sun Workstation running SunOS 5.7 with 128M of RAM memory and 256M of swap space and UltraSparc-IIi processor on

333MHhz. The simulation code is written in Java. The size of the simulator code is around 1500 lines.

Table I shows the results of the first set of simulations. The monitored area is a 500x500 units square. The sensor nodes coordinates are randomly generated using the pseudouniform distribution. The column SA denotes the average number of covers that simulated annealing achieves, while the column H denotes the average number of covers that the heuristic described in this paper achieves. The column |C| shows the number of sensors, while |A| shows the number of fields.

The most constrained – minimally constraining heuristic achieves significantly better results than the implementation of the simulated annealing, although the simulated annealing is running for a significantly longer time. Another observation is that the heuristic approaches the upper bound in all cases. The explanation for such results is that the fields in the central part of the monitored area are covered by significantly more sensors than the fields near the edge of the area. The heuristic easily generates a maximum number of covers having an abundance of the sensors covering non-critical fields to combine with the sensors covering critical fields. However, given distribution results in a low number of covers and a high number of unused sensors.

For the second set of simulations, whose results are shown in Table II, the conditions are changed in order to increase the number of covers. An additional space 70 units wide is added around the edges of the square, so the sensors are distributed over a 570x570 units area. However, only the 500x500 area is taken into account when the covers are selected. Another improvement implemented in the second set of simulations is that sensors are distributed in several stages. The density of the sensor nodes in different parts of the area is measured after a certain number of nodes are deployed. Additional sensor nodes are deployed in areas where the density is below a certain limit. From the comparison of the Tables I and II can be seen that for the same number of sensors, the number of sets that cover whole area increases when the given improvements in the deployment of the nodes are utilized.

VI. CONCLUSION

Wireless sensor networks enable efficient monitoring of physical environments. The main operating constraint is available energy. In order to maximize efficient use of batteries used by randomly placed sensors, we propose an organization of a sensor network, where the nodes are divided in mutually exclusive sets. One such set is active at any time.

We developed the most-constrained least-constraining heuristic and demonstrated her effectiveness on variety of simulated scenarios. From the simulation results can be concluded that the initial distribution of the sensor nodes in the monitored area determines the possible utilization of the available sensor nodes. Our future work will explore the possible strategies of sensor networks deployment that will ensure the best coverage of the monitored area.

TABLE I
RESULTS OF SIMULATIONS FOR 500x500 AREA

|C|	|A|	Radius	Number of covers		Runtime		upper bound
			SA	H	SA	H	
100	5403	200	6.8	9.7	5h:30	1m:30	9.7
120	8135	200	7	11.5	11h:15	5m	11.5
150	11454	200	10.5	18.5	14h:08	13m:30	18.5
180	13646	150	2.9	6.6	23h	7m	6.6
300	13510	80	2.6	4.3	27h:14	3m:36	4.3
400	24932	80	3.2	4.5	48h	10m	4.7

TABLE II
RESULTS OF SIMULATIONS FOR EXTENDED DEPLOYMENT AREA

|C|	|A|	Radius	Number of covers		Runtime		upper bound
			SA	H	SA	H	
100	3888	200	8.2	15.1	1h:08	1m:09	15.1
120	5916	200	8	16	8h:12	12m	16
150	7965	200	10.7	19.2	8h:42	13m:50	19.2
180	8106	150	8	15	12h:30	8m:34	16
300	11470	80	2.5	5.5	13h:45	5m	6.2
400	18230	80	4.2	8.1	52h	9m	9.1

REFERENCES

[1] D. Estrin, R. Govindan, J. Heidemann, "Embedding the Internet: Introduction," Communications of the ACM, vol.43, no.5, pp.38-41, May 2000.

[2] D. Tennenhouse, "Embedding the Internet: Proactive computing," Communications of the ACM, vol.43, no.5, pp.43-50, May 2000.

[3] G.J. Pottie, W.J. Kaiser, "Embedding the Internet: Wireless integrated network sensors," Communications of the ACM, vol.43, no.5, pp.51-58, May 2000.

[4] J. M. Kahn, R. H. Katz, K. S. J. Pister, "Next century challenges: Mobile networking for 'Smart Dust'," In Proceedings of International Conference on Mobile Computing and Networking (MobiCom 99), Seattle, WA, USA, Aug 1999.

[5] W. Rabiner Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," Proceedings of the 33rd International Conference on System Sciences (HICSS ' 00), January 2000.

[6] G. Asada, M. Dong, T.S. Lin, F. Newberg, G. Pottie, W.J. Kaiser, H.O. Marcy, "Wireless integrated network sensors: Low power systems on a chip," Proceedings of the 24th IEEE European Solid State Circuits Conference, 1998.

[7] T.-H. Lin, H. Sanchez, H.O. Marcy, W.J. Kaiser, "Wireless integrated network sensors for tactical information systems," Proceedings of the 1998 Government Microcircuit Applications Conference.

[8] J. Elson, D. Estrin, "An address-free architecture for dynamic sensor networks," Technical Report 00-724, CS Dept, USC, January 2000.

[9] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, "Next century challenges: Scalable coordination in sensor networks," In Proceedings of International Conference on Mobile Computing and Networks (MobiCom 99), Seattle, WA, USA, Aug 1999.

[10] "Low power design methodologies," edited by Jan M. Rabaey and Massoud Pedram, Boston, Kluwer Academic Publishers, 1996.

[11] L. Benini, G. Castelli, A. Macii, E. Macii, et al., "A discrete-time battery model for high-level power estimation," Design, Automation and Test in Europe Conference, pp.35-39, 2000.

[12] M. Adickes, R. Billo, B. Norman, S. Banerjee, B. Nnaji, J. Rajgopal, "Optimization of indoor wireless communication network layouts," Technical Report No. TR 99-5, Department of Industrial Engineering, University of Pittsburgh.

[13] D. Stamatelos, A. Ephremides, "Spectral efficiency and optimal base placement for indoor wireless networks," IEEE Journal on Selected Areas in Communications, pp. 651-661, May 1996.

[14] J. B. M. Melissen, P. C. Schuur, "Improved coverings of a square with six and eight equal circles," Electronic Journal of Combinatorics, vol.3, no.1, 1996.

[15] K. J. Nurmela, P. R. J. Östergård, "Covering a square with up to 30 equal circles," Research Report A62, Laboratory for Theoretical Computer Science, Helsinki University of Technology, 2000.

[16] R. Williams, "The geometrical foundation of natural structure: A source book of design," Dover Pub. Inc., New York, pp. 51-52, 1979.

[17] M. R. Garey, D. S. Johnson, "Computers and intractability: A guide to the theory of NP-Completeness," Freeman, New York, 1979.

[18] S. Kirkpatrick, C. Gelatt, M. D. Vecchi, "Optimisation by simulated annealing," Science, Vol. 220, No. 4598, pp. 671-680, May 1983.