A Heuristic Approach for K-Coverage Extension with Energy-Efficient Sleep Scheduling in Sensor Networks

Kenji Leibnitz, Indika Suranjith Abeyweera, Naoki Wakamiya, Masayuki Murata

Osaka University, Graduate School of Information Science and Technology
1-5 Yamadaoka, Suita, Osaka, 565-0871 Japan
{leibnitz,a-indika,wakamiya,murata}@ist.osaka-u.ac.jp

ABSTRACT

In this paper we consider the problem of coverage in a wireless sensor network where nodes already have been previously deployed. Our goal is to design an efficient strategy to add further nodes for achieving a certain target K-coverage throughout the entire monitoring region. A heuristic approach based on a dynamic model resembling gas bubbles is applied to solve this problem. Once the additional coverage has been reached, each node uses a self-organizing mechanism to determine its sleep schedule, depending on the activity of its surrounding nodes.

Keywords

Sensor network deployment, K-coverage, sleep scheduling

1. INTRODUCTION

Future ambient information network infrastructures aim at providing ubiquitous services, which are custom tailored and highly adaptable to the users' requirements. A necessary prerequisite is the efficient deployment of an underlying network infrastructure of integrated sensing devices which can monitor the environment and thus adapt to changes in the demand pattern of the users. By deploying networks of sensors, information about behavior, conditions, and positions of entities in an environment are gathered and forwarded to a sink for further processing [2]. The nodes are equipped with a sensing device, radio transmitter, and are usually battery operated. Since they are designed to operate autonomously, they must be able to set up a communication network in an ad-hoc manner and be able to adapt to changes in the network topology, when individual nodes may fail due to exhausted energy resources. Conservation of energy is, thus, a key issue in the deployment of sensor networks. The energy consumption differs depending on the state of a node and mechanisms which control the duty cycle of sensor nodes in an energy-efficient manner have been a key issue in research [13].

Bionetics '08, November 25-28, 2008, Hyogo, Japan

The coverage problem in sensor networks has been studied in several papers [4, 9–11]. This problem deals with placing nodes in the monitoring region while maintaining a desired degree of coverage. Each sensor node will cover a certain sensing area and the more redundant nodes are placed, the greater the reliability of the network is, once the battery of individual nodes gets depleted. Additionally, obtaining K-coverage, i.e., each location is covered by K sensor nodes, can be utilized to schedule the duty cycles of the nodes and prolong the lifetime of the network. Such sleep scheduling should be performed in an entirely distributed manner.

However, the performance of the network depends on the placement and coverage of the nodes. Due to the limited computational capabilities of sensor nodes, more sophisticated algorithms can be applied if the sink or an omniscient operator performs the location optimization. In our case, we apply a stochastic optimization heuristic which resembles gas bubbles being attracted to certain regions of a surface and either attach themselves to this surface or break with a pop once they reach a certain age.

Once the location of additional sensor nodes has been determined, the sleep scheduling mechanism can operate in an entirely distributed manner. For this task we consider that each node makes a local decision of sleeping or waking based on the local exchange of messages. Note that the scheduling problem can be considered completely independent of the coverage planning problem, but we will here discuss both problems here in conjunction, since the deployment of the sensor nodes will greatly affect the performance of the scheduling mechanism.

In this paper we discuss the application of a dynamic and self-organizing *"gas bubble"* approach to the sensor network coverage problem. We can assume any size and shape of node coverage region as well as any desired location-dependent target coverage value. The remainder of this paper is organized as follows. In Section 2, we give the problem statement and introduce the proposed gas bubble algorithm. Following this, in Section 3, we investigate the performance of our proposal. Then, in Section 4, we describe the integration with a sleep scheduling method. Finally, this paper is concluded with an outlook on future work in Section 5.

2. THE GAS BUBBLE ALGORITHM

In this section we will present the gas bubble algorithm. Our goal is to add a certain amount of nodes to an existing sensor network to maintain that each location in the monitoring region is covered by at least K different sensor nodes.

2.1 Problem Statement

Let us consider a sensor network with a set of $\mathcal{S}$ nodes that are already deployed in a monitoring region $W \in \mathbb{R}^2$ and the cardinality of $|\mathcal{S}| = M$. The goal is to add additional N sensor nodes that all points within W have at least K-coverage, which is defined as follows.

DEFINITION 1 (K-COVERAGE). *We say that a region W has obtained K-coverage, iff*

$$\forall_{x \in W}\, C(x) \geq K$$

where $C : \mathbb{R}^2 \rightarrow \mathbb{N}$ is the number of sensor nodes covering a point $x \in W$.

This problem is obviously NP-complete. Therefore, we suggest applying a stochastic optimization heuristic technique with a dynamic growing structure. The model is loosely based on the self-organizing approach of *Growing Neural Gas* (GNG) [3]. Growing Neural Gas is a special kind of self-organizing feature maps [6], which are dynamic neural networks that map the topologies from an input space to an output space, while preserving the topological relationship. However, in contrast to GNG, we do not require to maintain any topology, therefore, any neighborhood relations are not really taken into account, but only the best matching neuron is stimulated by an input signal. Self-organizing maps for coverage optimization have also been considered in [7,8]. While [8] considers only the application of GNG to sensor network coverage planning, [7] also deals with the problem of K-coverage. However, these papers do not discuss the problem in combination with sleep scheduling.

The key feature during the adaptation of self-organizing feature maps is that an intrinsic energy which influences the reaction of neurons to input stimuli is reduced over the lifetime of the network. This concept is for example also applied in *simulated annealing* [1] and attractor selection [5]. We will use this feature as well and let the neurons be stimulated by input signals generated randomly in the monitoring area. When the neuron's energy has reduced to fall below a threshold, the neuron will be considered as candidate for the location of a sensor node. At this point the coverage gain of the potential sensor is evaluated and if it results in an increase in coverage a new sensor will be placed at this location. On the other hand, if putting a sensor at this location would not yield any benefits in coverage, it will be simply removed.

The overall behavior of our proposed method resembles gas bubbles which are attracted towards a material with lower density. Once the bubble reaches this area, it becomes inert and merges with this material. Similarly, if it does not reach the area and exceeds its lifetime, the bubble simply breaks and is removed from the system.

We assume that with knowledge of the monitoring region and the location of the existing sensor nodes, it is possible to calculate the coverage degree with a function $C(x)$ at any location $x \in W$. In the simplest case, we can consider that the sensing coverage area is circular with a radius of r.

$$C(x) = |\{s \in \mathcal{S} : \|s - x\| < r\}| \qquad (1)$$

The operator $\|\cdot\|$ denotes the Euclidean distance.

Depending on whether a point $\xi \in W$ has already reached its target coverage, the point ξ randomly generates an input signal as a stimulus. The neuron located closest to the stimulus reacts to this input by moving towards it and storing its cumulative traveled distance, see Fig. 1. If this distance gets very large, this is an indicator that this neuron is responsible for too many stimuli and therefore requires that a new neuron is generated. In order to be accepted as sensor location, the new location must contribute to coverage, since it is of no use adding a sensor node to an already sufficiently covered area. On the other hand, when a neuron has not been moved for a certain time, it becomes unnecessary and is removed from the network.

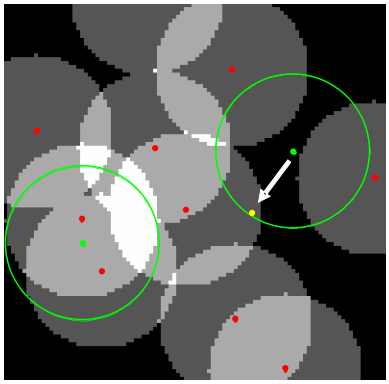


Figure 1: Attraction of neuron to input signal

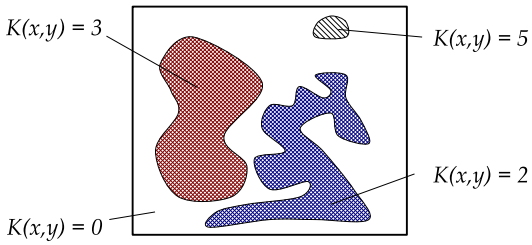


Figure 2: Inhomogeneous target coverage

A great advantage of this algorithm is that it is easily extended to take into account also inhomogeneous target coverage values. For example, consider a scenario where the degree of coverage depends on the location, see Fig. 2. Some areas may be more critical and must be observed with a higher target coverage. In this case, the algorithm is easily extended by generating the input signals ξ dependent on a location value $K(\xi)$ instead of a constant K. However, throughout this paper we will assume only a constant K.

2.2 Algorithm Formulation

We can now formulate the algorithm in this way. Initially starting with two randomly located neurons, we generate at each time step an input signal among all points in W which have not yet reached their target coverage K. Following the principle of *Hebbian Learning*, this input signal stimulates the neuron nearest to it which is moved by a certain fraction towards this stimulus. Unlike GNG or other topology-preserving mechanisms, neighboring nodes are not influenced. The amount of movement of each neuron is stored in the variable *error* and is an indication of how well the input space is represented. If it moves a lot, then more neurons are needed. and so after a fixed number of t_{add} time steps, a new neuron is added at the location of the neuron with the highest *error* value, which is then

split among the old neuron and the new neuron. In order to limit the growth of neurons, we use an age counter which is reset whenever a neuron is selected as best matching neuron. Once a certain age threshold t_{age} is reached, the bubble breaks and it is removed from the system. If a node has a low *error* value after each s_{add} steps, it has found a good location in the input space and is then pinned to this position to become the location of a new sensor node.

In order to make the system become stabilized, the error value of each neuron is decreased at each time step and its age is incremented. This procedure is repeated until the whole monitoring area reaches its target coverage.

1. Initialize $step = 0$.
2. While there are still locations left with coverage less than K:
   - 2.1 $step = step + 1$
   - 2.2 Start with two neurons n_1 and n_2 at random positions in the window W.
   - 2.3 Generate an input stimulus ξ randomly among all uncovered points in W proportional to $K - C(\xi)$.
   - 2.4 Find the nearest neuron b to ξ and update its error value and reset its age.

$$error(b) = error(b) + \|b - \xi\|$$
$$age(b) = 0$$

   - 2.5 Move b by ϵ towards ξ: $b = b + \epsilon\,(\xi - b)$
   - 2.6 If $t_{add} \bmod step == 0$, add a neuron at the location of the neuron with the highest error and split this total error among the old and new neuron.
   - 2.7 if $s_{add} \bmod step == 0$, the neuron with lowest error becomes a sensor node if there is a coverage gain.
   - 2.8 Remove all neurons n_i with $age(n_i) > t_{age}$.
   - 2.9 Decrease the error of each neuron n_j by a decay factor δ and increment its age.

$$error(n_j) = \delta\ error(n_j)$$
$$age(n_j) = age(n_j) + 1$$

3. NUMERICAL RESULTS

We conduct simulation runs to investigate the behavior of our algorithm. The parameters are $t_{add} = 200$, $s_{add} = 500$, $\epsilon = 0.2$, $t_{age} = 200$, and $\delta = 0.95$. Obviously, in order to guarantee a minimal K coverage value, we will have a lot of overlap when the radius is large since circular coverage areas are assumed. Fig. 3 shows a histogram of the 3-coverage for a square window W of size $w = 100$ and $r = 15$. It can be seen that the average coverage is much higher at about 5.5 and overlaps of up to 11 nodes can be found.

3.1 Approximation of Optimal Coverage

We can theoretically estimate the best coverage value, when we assume a regular hexagonal grid layout of the sensor nodes as this resembles the most optimal possible layout with circular coverage areas, see Fig. 4. In order to achieve a higher K, we can consider different overlaid layers of such grids, so we essentially just need to calculate the number of circles as well give an estimate for the overlap area between

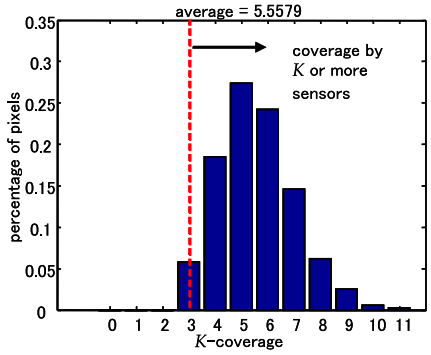


Figure 3: Histogram of K-coverage

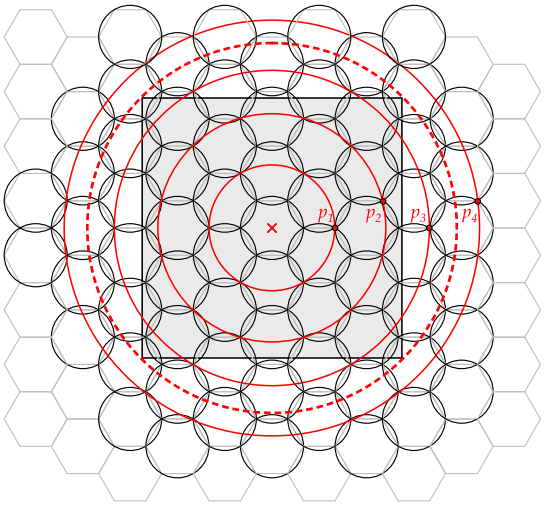


Figure 4: Hexagonal coverage layout

two circles. Since we are looking at the totally optimal situation, we will ignore any influence from the existing sensor network in this approximation.

In order to fully cover the area square area W with side w, we assume the layout as shown in Fig. 4. First, we need to calculate the distance of the center ('x') to the series of points marked as $p_i, i \geq 1$, which give us a the radius y_i of the i-th tier of circles.

$$y_i = \begin{cases} r\ (2 + 3\,(i-1)) & \text{if } i \text{ is odd} \\ \frac{r}{2}\sqrt{(7 + 6\ (i-2))^2 + 3} & \text{if } i \text{ is even} \end{cases} \quad (2)$$

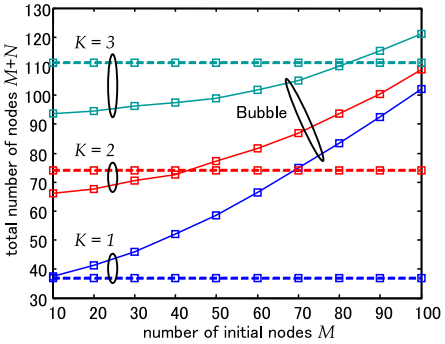
We need to find the i-th tier of circles which completely includes the square W by comparing $y_{i-1} < \frac{w}{2}\sqrt{2} \leq y_i$. For the i-th tier we have in total L_i nodes.

$$L_i = 1 + 6 \sum_{k=1}^{i} k = 1 + 3\,i\ (i+1) \quad (3)$$

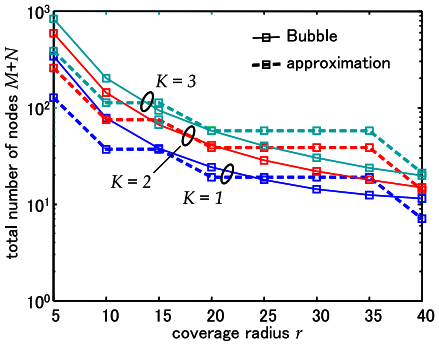
Note that this overestimates the number of nodes needed especially when i is large, since the segments enclosed by the circle and the square will become very large and contain circles which are not used for the coverage. However, since we anyway ignore the previously configured original network nodes, we can assume that the errors are compensated to get

Table 1: Radii of tiers for $w = 100$ and $r = 15$

i	1	2	3	4	5	6	7
y_i	30.0	54.1	120.0	143.1	210.0	232.9	300.0



(a) Initial number of nodes M



(b) Coverage radius

Figure 5: Comparison of total nodes $M + N$

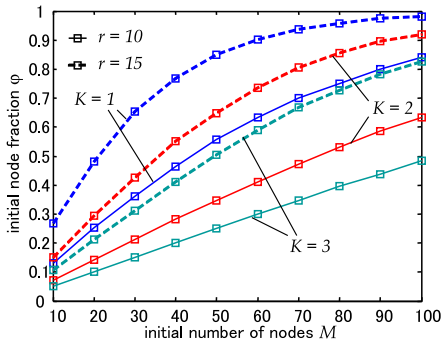
a very rough estimate for the number of nodes. Multiplication of L_i with K yields the total number of nodes.

For example, let us consider the sample configuration that we used before. Then, the values of y_i are as shown in Table 1. We can see that $y_2 < w < y_3$, so with $i = 3$ we have $L_3 = 37 \times K$ nodes in total, depending on K.

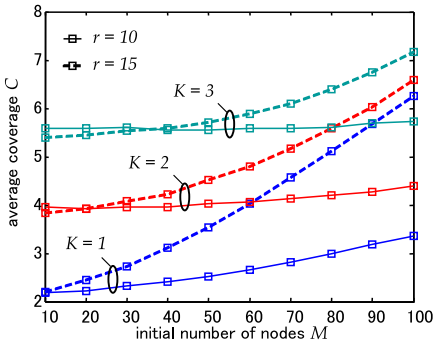
In Fig. 5(a) the values are compared between the theoretical approximation and the proposed bubble algorithm. The number of initially existing nodes is M and the number of added nodes is denoted as N. We can see that the when $K = 1$, only for $M = 10$ both methods achieve comparable results. However, unlike the optimal method where $M + N$ remains constant, the total nodes increase with the previously available sensor nodes M. However, for larger K our approach proves to be competitive requiring similar or fewer nodes than the approximation. The same observation can be made in Fig. 5(b), where we varied the coverage radius and kept $M = 10$ constant. The bubble algorithm always lies lower than the approximation methods.

3.2 Influence of Initial Configuration

The first series of experiments considers the original sensor network configuration and the number of nodes that were already previously deployed and which we wish to extend to K-coverage. Fig. 6(a) depicts the initial node fraction. These values shows us the proportional growth rate from the



(a) Node growth ratio



(b) Average coverage

Figure 6: Different number of initial nodes

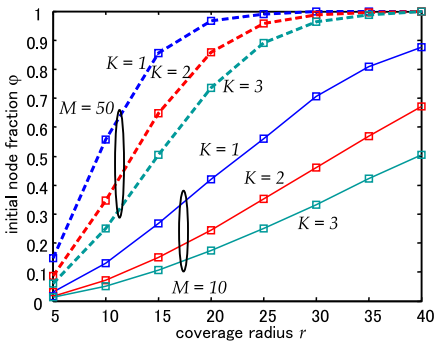
initial number of nodes and is defined as $\varphi = M/(M + N)$.

Hence, a value of $\varphi = 1$ means that there $M = M+N$ and thus no additional nodes were added. On the other hand a value of $\varphi \approx 0$ means that $N \gg M$, so a huge amount of additional nodes were required. Fig. 6(a) shows this growth fraction for $K = 1, 2, 3$ and the two coverage radii $r = 10, 15$. When $K = 3$, we have the smallest φ and the most nodes are added. This is expected, since we require that each point in the monitoring region W must be covered $K = 3$ times. Additionally, the larger the radius r, the less nodes are needed and best results are with $K = 1$ and $r = 15$.

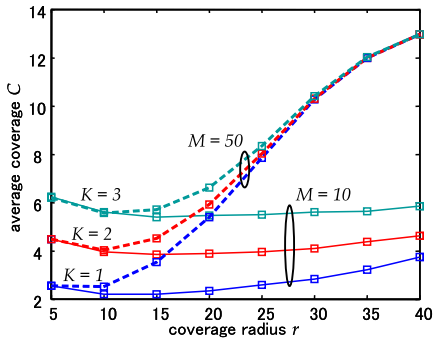
In Fig. 6(b) the average coverage value C of each point in W is shown as defined in Eqn. (1). The first thing we notice is that in order to obtain a guaranteed K-coverage, nearly the double amount of coverage is performed. The smaller the radius is, the less the results get influenced by the initial node configuration. When K is small the difference in r becomes clearer for larger M than for large K. This means that the influence of the radius is reduced when K gets larger. An interesting observation that can be made is that if $r = 10$, an optimal value of initial nodes M exists somewhere between $20 < M < 40$ which shows the least excess coverage. This effect is not visible when $r = 15$.

3.3 Influence of Node Coverage Radius

We will now take a more detailed look at the influence of the coverage radius r on the node growth ratio, cf. Fig. 7(a). When the initial number of sensor nodes is large, e.g. $M = 50$, only few nodes are added. A large coverage radius drastically reduces the number of nodes and this is enforced when



(a) Node growth ratio



(b) Average coverage

Figure 7: Influence of coverage radius

the required K gets smaller.

In Fig. 7(b), the average coverage is shown as a function over the coverage radius r. This figure shows us that a lot of the excess coverage is introduced by the already existing sensor nodes. When the algorithm has a chance to operate with only few initial nodes, e.g. $M = 10$, the average coverage remains rather constant, but it rapidly increases when M and r are large. In this case, the different target level plays no role, since so much overhead coverage is produced anyway. Again, we can clearly see that for $M = 10$ a radius r exists, so that the average coverage is minimized. This optimal radius lies between $10 < r < 15$, depending on K.

4. DYNAMIC SLEEP MANAGEMENT

The advantage of having K-coverage in a sensor network is that each location in the monitoring region is now redundantly covered by at least K sensor nodes. This improves robustness to individual node failures, higher diversity in sensing data, and the possibility of combination with a sleep scheduling scheme to prolong network lifetime. We now briefly outline how efficient coverage planning can be applied to a sleep scheduling scheme, based on the Coverage Configuration Protocol (CCP) [12] that we extend by energy-efficient timers and message filtering.

4.1 Overview of CCP

In CCP a sensor node follows the state transitions as illustrated in Fig. 8, depending on timers and its eligibility. Being eligible (Eligibility=TRUE) means that a sensor node must stay active and sensing to satisfy a required k-coverage,

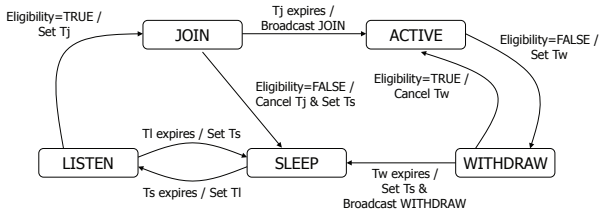


Figure 8: CCP states and transitions

with $k < K$ of that used for designing the network. In all states except for SLEEP, a sensor node maintains a neighbor list, which is updated on reception of messages before checking its eligibility. A sender of a HELLO or JOIN message is added to the list and that of a WITHDRAW message is removed from the list. An entry of the list includes an identifier, the coordinates, and the sensing range of a message sender. A detailed description of the states and their transitions can be found in [12].

In all states except for SLEEP, a sensor node evaluates its eligibility on receiving a message from neighboring nodes within the RF reception range. The algorithm proposed in [12] for evaluating the eligibility considers geometrically only intersection points of sensing ranges of neighboring sensor nodes. If all intersection points are in the sensing range of a sufficient number of sensor nodes in ACTIVE state, a node is considered ineligible.

4.2 Energy-Dependent Timer Setting

Our first extension deals with the energy-efficient setting of the timers in each state, since energy is not taken into account in the original CCP. To prolong the lifetime of the network and to keep monitoring for as long as possible, it is necessary to set the timers in accordance to the amount of residual energy. For example, by keeping a sensor node with higher residual energy longer in ACTIVE state, low-power nodes with overlapping sensing area can save battery consumption by being ineligible and entering SLEEP state.

For this purpose, we assume that a sensor node knows the amount of residual energy of its battery and HELLO, WITHDRAW, and JOIN messages include the information about the sender's residual energy at the time of message emission. When node i changes its state at time t, it sets its respective timer depending on the energy information in all messages received while it has been awake, i.e., in LISTEN, JOIN, ACTIVE, and WITHDRAW states. The ratio of own residual energy to maximum of the received energy levels is denoted as $\rho_i(t) = e_i(t)/e_{i,max}(t)$, with $e_{i,max}(t) = \max_{j \in \mathcal{N}_i} e_j(t_j)$ and $\mathcal{N}_i$ is the set of neighboring nodes from which node i has received messages, $t_j < t$ corresponds to the time that the latest message from node j was received, and $e_j(t)$ is the amount of residual energy reported by sensor node j at time t. Each timer is set as follows, where $T_x(0)$ represents the initial setting in state x.

SLEEP Timer: The sleep timer $T_s(i,t)$ is set inversely proportional to the relative amount of residual energy, so that a sensor node with the large amount of residual energy wakes up earlier than the others, i.e., $T_s(i,t) = T_s(0)/\rho_i(t)$.

WITHDRAW Timer: The withdraw timer $T_w(i,t)$ is set proportional to the relative amount of residual energy. By keeping a sensor node with the large amount of residual energy stay in the WITHDRAW state longer, the sensor

node has more chances to receive WITHDRAW messages from others and thus it is more likely to become eligible again, i.e., $T_w(i,t) = T_w(0)\,\rho_i(t)$.

JOIN Timer: The join timer $T_j(i,t)$ is set inversely proportional to the relative amount of residual energy. By making the join timer expire faster for a sensor node with the large amount of residual energy, it suppresses activation of sensor nodes with less residual energy, i.e., $T_j(i,t) = T_j(0)/\rho_i(t)$.

LISTEN Timer: The listen timer $T_l(i,t)$ is set independently of the residual energy. The duration of the LISTEN state should be long enough to receive messages of neighboring nodes. Among messages, HELLO messages are the most important for sensor node i to evaluate its eligibility, since HELLO messages imply that there are actively sensing nodes in the vicinity and it would make the sensor node ineligible. Consequently, $T_l(i,t)$ is given as, $T_l(i,t) = T_l(0) = c\,T_{hello}$, where $c \geq 1$ is a constant.

4.3 State Transitions with Message Filtering

The state transition of CCP causes that sensor nodes are divided into two groups, nodes which remain in ACTIVE state and nodes which continuously switch between SLEEP state and ACTIVE state. Therefore, we additionally introduce message filtering and modify the CCP state transition rules. Here, we explain how a node acts in each awake state when it receives any message.

ACTIVE State: Sensor node i neglects HELLO messages from neighbors which have lower energy than $e_i(t)$. Then nodes which have higher residual energy can easier enter ACTIVE state. Furthermore, nodes with low energy compare their level to neighbors entering SLEEP state. Any JOIN messages, sent by nodes switching from JOIN to ACTIVE state, are neglected. However, WITHDRAW messages should be accepted regardless of the energy level.

WITHDRAW State: As this is a transient state before going to SLEEP state, sensor node i which has lower residual energy goes to SLEEP state as soon as possible, neglecting HELLO and JOIN messages from neighbors with lower than $e_i(t)$ in the same way as in ACTIVE state.

JOIN State: Messages in this transient state are handled the same way as in ACTIVE state.

LISTEN State: This state is for deciding whether the node should go to ACTIVE state after coming from SLEEP state according to its neighbors' messages. Thus, node i neglects messages which are lower than $e_i(t)$. Then sensor nodes, which have higher than $e_i(t)$ are moved to ACTIVE state and others to SLEEP. In this state WITHDRAW messages are neglected without considering the residual energy.

5. CONCLUSION AND OUTLOOK

In this paper we addressed two important problems found in sensor network deployment and operation. At first, we applied a dynamic heuristic approach for optimizing an already existing sensor network to reach K-coverage. The algorithm resembles the behavior of gas bubbles and stochastically finds a solution to the K-coverage problem. Then, after the "gas bubble" algorithm finds suitable locations of additional sensors, a dynamic sleep scheduling method can be applied to prolong the lifetime of the network. The sleep scheduling mechanism works self-adaptively with only local information. In the future we wish to further investigate the effects of energy-efficient timer settings by experimental studies and mathematical analysis. When the timer parameters are selected in an appropriate and energy-efficient way, it will be possible to ensure that the sensor network operation becomes robust and reliable in order to provide ubiquitous service in an ambient network infrastructure.

Acknowledgments

The authors would like to thank Yoshiaki Taniguchi for his help and the anonymous reviewer for his comments. This research was supported in part by "Global COE (Centers of Excellence) Program" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

6. REFERENCES

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. Wiley, 1989.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Commun. Mag.*, 40(4):102–114, 2002.

[3] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems*, volume 7, pages 625–632, 1995.

[4] C.-F. Huang and Y.-C. Tseng. The coverage problem in a wireless sensor network. In *2nd international conference on Wireless sensor networks and applications (WSNA'03)*, San Diego, CA, 2003.

[5] A. Kashiwagi, I. Urabe, K. Kaneko, and T. Yomo. Adaptive response of a gene network to environmental changes by fitness-induced attractor selection. *PLoS ONE*, 1(1):e49, 2006.

[6] T. Kohonen. *Self-Organizing Maps*, volume 30. Springer Series in Information Sciences, 2001.

[7] C. Koutsougeras, Y. Liu, and R. Zheng. Event-driven sensor deployment using self-organizing maps. *International Journal of Sensor Networks*, 3(3):142–151, 2008.

[8] A. Kulakov and D. Davcev. Distributed algorithm for a mobile wireless sensor network for optimal coverage of non-stationary signals. In *1st Workshop on Spatial Stochastic Modeling of Wireless Networks (SpaSWiN'05)*, Riva del Garda, Italy, 2005.

[9] J. Lu, L. Bao, and T. Suda. Coverage-aware sensor engagement in dense sensor networks. In *Embedded and Ubiquitous Computing: International Conference (EUC'05)*, pages 639–650, Nagasaki, Japan, 2005.

[10] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *IEEE Infocom*, Anchorage, AK, 2001.

[11] M. T. Thai, F. Wang, and D.-Z. Du. Coverage problems in wireless sensor networks: designs and analysis. *International Journal of Sensor Networks*, 3(3):191–200, 2005.

[12] 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 Trans. on Sensor Networks*, 1(1):36–72, 2005.

[13] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE Infocom*, New York, NY, 2002.