Proc. IEEE International Conference on Communications, ICC09, Cape Town, South Africa, May, 2010, pp. 1-5.

The Effect of Neighbor Graph Connectivity on Coverage Redundancy in Wireless Sensor Networks

Eyuphan Bulut, Zijian Wang and Boleslaw K. Szymanski
Department of Computer Science and Center for Pervasive Computing and Networking
Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, USA
{bulute, wangz, szymansk}@cs.rpi.edu

***Abstract*—Coverage redundancy problem is one of the significant problems in wireless sensor networks. To reduce the energy consumption that arises when the high number of sensors is active, various coverage control protocols (sleep scheduling algorithms) have been proposed. In these protocols, a subset of nodes necessary to maintain sufficient sensing coverage are kept active while the others are put into a sleep mode to reduce the energy consumption. In this paper, we study the coverage redundancy problem in a sensor network where the locations of nodes and the distances between nodes are neither known nor could be easily calculated. We define a neighbor graph as the graph formed by the neighbors of a node and analyze the effect of different levels of connectivity in neighbor graphs on the coverage redundancy of sensor nodes. Moreover, we apply our results to a lightweight deployment-aware scheduling algorithm and demonstrate the improvement in the performance of the algorithm.**

I. Introduction

The advances in wireless communications and electronics have enabled the development of low-power and small-size sensor devices with limited memory and limited computing capabilities [1]. A Wireless Sensor Network (WSN) consists of a large number of these sensor nodes deployed into a region. Depending on the application of WSN (i.e. battlefield surveillance, environment monitoring), the sensor nodes can detect various phenomena including temperature, light and motion in their environment, perform simple computations and communicate with each other through radio transmission.

Sensor networks are usually deployed with high densities to have extended network reliability and lifetime. However, excessive energy consumption will occur if all the nodes operate at the same time. This will quickly cripple data acquisition by the sensor network as the increasing number of nodes will exhaust their limited energy. Therefore, in sufficiently dense networks, to avoid redundancy and increase network lifetime, a common technique called *coverage control* or *sleep scheduling* is used in which some sensor nodes are put into sleep mode and only a subset of nodes necessary to ensure coverage are kept active for sensing and communication. Coverage control algorithms control the number of active sensor nodes and enable the sleeping nodes conserve their energies for future use.

This research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Depending on different assumptions made about the sensor network features, many coverage control algorithms have been proposed with the common objective of selecting the active nodes among all sensor nodes in the network in such a way that the network field is covered by these nodes at the desired ratio while the usage of the energy by the sensor nodes is as balanced as possible. However, some of these algorithms [2], [3] assume that the nodes know their locations by either being equipped with GPS devices or by using some triangulation techniques; others [4] assume that the distances between nodes can be computed via received signal strengths while some others [5] assume the existence of some mobile nodes with controllable mobility. Satisfying each of these assumptions increases the cost of hardware deployed in the sensor network (i.e., the cost of GPS or RSSI technology) or imposes an additional communication delay and power consumption.

To prevent these extra costs and to be consistent with the simple capabilities of sensor networks, some researchers have also suggested coverage control algorithms without exploiting any location, distance or angle information about the sensor nodes. In these studies [6-11], either one-hop or two-hop neighbor counts are used to compute the coverage redundancy of sensors and to find the subset of nodes that need to be active to satisfy the required network functions (required ratio of covered area or minimum number of sensors necessary to cover each point etc.).

In this paper, we study the coverage redundancy problem under similar assumptions, as described latter. However, unlike the previous work, we utilize *neighbor graphs* to compute the expected coverage redundancy of sensor nodes. To the best of our knowledge, this is totally a novel approach. We define a *neighbor graph* as the graph formed by the neighbors of a node and analyze the effect of different configurations of neighbor graphs on the coverage redundancy of sensor nodes.

The rest of the paper is organized as follows. In Section II we describe our network model and assumptions. In Section III we give the details of expected coverage redundancy analysis. Then, in Section IV, we mention an application of our findings to a coverage control protocol. In Section V, we discuss some

issues regarding the performance of the proposed protocols. Finally, we conclude and outline the future work in Section VI.

II. Network Model and Assumptions

We assume that N static homogeneous sensors are deployed randomly with uniform distribution in a two dimensional field. Nodes neither know nor attempt to compute their locations, the distances to their neighbors and the angles of their neighbors with respect to their own coordinate systems. We use the same sensing and communication model as in most sensor network studies and assume that each sensor node has a circular sensing area with radius R_s and a circular communication area with radius R_t centered at the location of the sensor node. The sensing and communication are reliable, i.e. any event occurring within sensing range can be detected and the node can communicate with any other node within its communication range.

We assume the existence of a mechanism which enables nodes to know their one and two-hop neighbors. Obviously, every node can learn the nodes within two hops away by two broadcasts of hello messages. In the first one, each node will only send their ids. Then, after they gathered all the ids of their one-hop neighbors (it can simply be achieved by setting time out value for receiving the first hello messages), each node will send the second hello message which will also include the ids of its neighbors in addition to its own id. Observe that in a static sensor network it is sufficient to perform this process only once (after deployment), therefore it has negligible effect on the communication cost of the network. Moreover, all coverage control algorithms assuming no location and distance information make also similar assumptions (i.e. [7] assumes the existence of a mechanism to learn one-hop neighbors periodically and [8] assumes that nodes know their one and two-hop neighbors).

III. Expected Coverage Redundancy Analysis

In this section, we analyze the expected coverage redundancy of a sensor node in two cases: (i) when it knows only the number of its one-hop neighbors, and (ii) when it also knows its neighbor graph. We denote the set of one-hop neighbors of a node i by N_i and we define $G_i = (V_i, E_i)$ as a neighbor graph of node i where $V_i = N_i \cup \{i\}$ and E_i is the set of edges between nodes in V_i. If two neighbor graphs with the same N_i can be generated from each other by relabeling the nodes, they are isomorphic. Figure 2 shows the two possible non-isomorphic neighbor graphs when N_i=2.

For the sake of simplicity, throughout the analysis and simulations in this paper, we assume that $R_t = R_s = R$ for all sensors. However, our model can easily be adapted to the more general case when there is no relation between R_t and R_s.

A. Case when only N_i is known:

Let $P_n(i)$ denote the expected coverage redundancy (i.e., the expected ratio of sensing area covered by other nodes) of a node i by its n one-hop neighbors. If a node i knows the value of N_i, it can compute $P_{N_i}(i)$ as follows. Consider the sensor node i illustrated in Figure 1. Without loss of generality, we assume that R=1. Consider a point P inside the sensing area with distance x from node i. For P to be covered by a node j $\in N_i$, node j must be within the common area of circles centered at i and P. Let's call this area $A(x)$. Since we assume the uniform distribution of nodes in the network field, the probability that point P will be covered by a node $j \in N_i$ is $A(x)/\pi$. Clearly, $A(x) = 2\theta - \sin 2\theta$, where $\theta = \arccos x/2$. By integrating over all such points P within the sensing area of node i, $P_1(i)$ (or simply P_1) becomes:

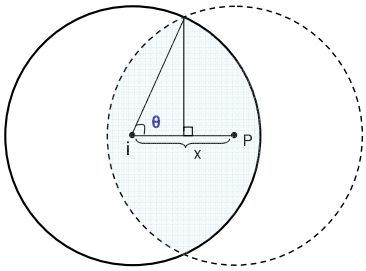

Fig. 1. The probability that a point P inside the sensing area of a node i will be covered by a one-hop neighbor of node i is equal to the ratio of the overlapping area (of circles) to the whole sensing area.

$$\begin{aligned} P_1 &= \int_0^1 2\pi x \frac{A(x)}{\pi} dx \\ &= \frac{8}{\pi} \int_{\pi/3}^{\pi/2} \cos\theta \sin\theta (2\theta - sin2\theta) d\theta \\ &= 1 - \frac{3\sqrt{3}}{4\pi} = 0.586 \end{aligned}$$

If there are n one-hop neighbors, then:

$$P_n = \int_0^1 2\pi x \left(1 - \left(1 - \frac{A(x)}{\pi}\right)^n\right) dx$$

When n = 2, 3, 4, 5, 6 the above formula gives 0.808, 0.906, 0.952, 0.974, 0.986 respectively. That is, for example if $N_i = 4$, node i can expect that 95.2% of its sensing area is covered by its one-hop neighbors.

B. Case when both N_i and G_i are known:

Observe that once the nodes know their one and two-hop neighbors (using the aforementioned mechanism), they can form their neighbor graphs using the ids of these nodes. In this section, we will show how the knowledge of G_i enables nodes to compute their expected coverage redundancy more accurately.

Let's assume that a node i has N_i one-hop neighbors. Consider the question: *how many non-isomorphic G_i's can node i have?*. Every node $j \in N_i$ has an edge to node i in G_i. When we remove these edges, this question becomes: (*how many non-isomorphic graphs can be generated with n vertices?*) studied in graph theory under graph enumeration topic. Several solutions such as Polyas theorem have been proposed to define the number of non-isomorphic graphs with

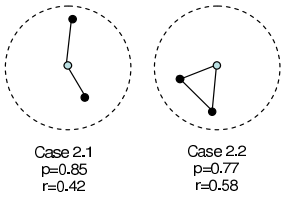

Fig. 2. Two possible non-isomorphic neighbor graphs with their p and r values when there are two one-hop neighbors.

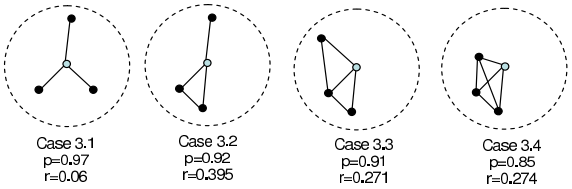

Fig. 3. Four possible non-isomorphic neighbor graphs with their p and r values when there are three one-hop neighbors.

a given vertex count. For example, when n = 3, 4, 5 the number of non-isomorphic graphs ($G(n)$) is 4, 11, 34, respectively. That is, for example when $N_i = 3$, then neighbors of node i can connect to each other in four different ways.

Next, we will show that when node i knows its G_i case among all possible $G(N_i)$ configurations, then it can compute its expected coverage redundancy more accurately. To do this, we will find the answers of the following two questions for each neighbor graph case generated with given N_i; 1) what is the expected coverage redundancy (p) of node i and 2) what is the occurrence rate (r) of the case under uniform distribution?

Here, we will use Monte-Carlo method to find p and r values for each case. The method works as follows. We create a node (i) centered at origin. Then, for each N_i value, we randomly deploy N_i nodes within the sensing range of node i. We first find the case of generated neighbor graph among all possible $G(N_i)$ cases and increase its occurrence count by 1. Then, by dividing the sensing region of node i into grids, we compute the percentage of all grids which are also covered by any of these N_i nodes. This ratio gives the coverage redundancy of node i. We add this ratio to the variable keeping the sum of all computed redundancy ratios for that specific case. When we repeat this process a large number of times (i.e. 10^6) and take the average of all redundancy ratios computed for each case we obtain the value of p for each case. Moreover, dividing the total occurrence count of each case by the total test count gives us the value of r for that specific case.

In Figures 2, 3 and 4, we show all possible neighbor graphs with their p and r values when there are two, three and four one-hop neighbors, respectively. The results show how the expected coverage redundancy of a node differs in different cases of neighbor graphs even when the number of one-hop neighbors remains the same. For example, consider two nodes i and j having four neighbors and assume that node i has G_i of case 4.2 and node j has G_j of case 4.10. From the expected redundancy analysis given in the previous section, both expect that 95.2% of their sensing areas will be covered by their one-hop neighbors. However, using the results of this section, node i expects that 99.1% of its sensing area will be covered, while node j expects that only 94.0% of its sensing area will be covered. This clearly shows how a small additional information available nearly free can improve the accuracy of expected coverage redundancy of a node.

IV. Utilization of Neighbor Graphs on Coverage Control Algorithms

In this section, we show a sample application of our results from the previous section to the design of a coverage control algorithm. For this purpose, we will modify Lightweight Deployment-Aware Scheduling Algorithm (LDAS) [7] as it only uses the number of neighbors of a node while deciding which nodes will stay active.

A. Overview of LDAS

In LDAS [7], each sensor node maintains the number of its working one-hop neighbors by periodical beacons. It also occasionally sends out tickets to its neighbors and then checks whether it has received enough tickets to be qualified to go to sleep mode. If it did, it enters a *ready-to-off* mode, otherwise it stays active until it collects enough tickets. In *ready-to-off* mode, the node first backs off for a randomly selected time. Then, if it has enough neighbors to satisfy required quality of surveillance (QoS measured by the percentage of the sensing area that needs to be covered), it goes to sleep mode and stays in that mode for a limited time. Otherwise, it continue waiting in *ready-to-off* mode until the necessary number of nodes becomes active in its neighborhood.

In LDAS, the number of tickets to be distributed depends on the number of neighbors required to achieve the given QoS. Wu et al. present in [7] a formula for the lower bound of the percentage of the redundant area with a given number of neighbors. Using this formula, each node first finds the minimum neighbor count (c) which can provide required QoS and then sends $n - c$ tickets, each to a randomly selected working neighbor among n neighbors. As a result, the nodes in dense areas send more tickets out increasing their chance to get into sleep mode compared to nodes in low density areas. The nodes also need c active neighbors to go to sleep mode after their back off in *ready-to-off* mode expires.

B. Proposed Algorithms

We provide three different algorithms adopting the basic working principles of LDAS but updating or extending it to show the effect of neighbor graphs on the performance of the algorithm. These three protocols with corresponding additions are as follows:

1) U-LDAS: In LDAS, the number of neighbors required to achieve the given QoS is set conservatively using the lower bound of coverage redundancy with the given one-hop neighbor count. Here, we modify this algorithm by enabling nodes to decide the required number of neighbors using the results from section III-A. For example, if required $QoS = 0.91$, U-LDAS requires the existence of four neighbors while LDAS requires five neighbors.

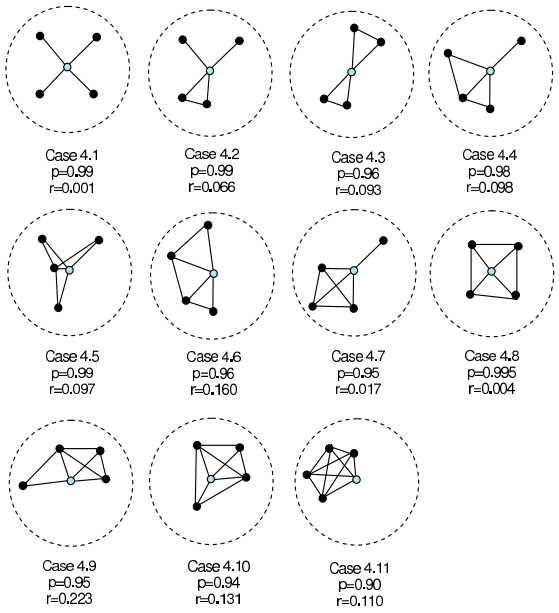

Fig. 4. Eleven possible non-isomorphic neighbor graphs with their p and r values when there are four one-hop neighbors.

2) E1-LDAS: The results in the section III-B emphasize the significant effect of neighbor graphs on computing the expected ratios of redundantly covered areas of nodes. For example, when there are three neighbors, the total occurrence rate of cases which provide a higher p value than the average value of all cases (expected coverage redundancy obtained using only N_i) is 72.6%. This indicates that we can improve the performance of coverage redundancy check algorithm up to 72.6% (the improvement depends on the required QoS). In E1-LDAS, we slightly extend U-LDAS algorithm by enabling nodes to decide the required number of active neighbors at the last step using their neighbor graphs. That is, when the back off time expires for a node in *ready-to-off* mode, instead of checking whether it has enough active neighbors to go to sleep mode using only the number of its one-hop neighbors, it performs this check according to p value of its current neighbor graph.

3) E2-LDAS: In all previous protocols including LDAS, the nodes collecting more tickets have higher chance to get into sleep mode. Here, we change it and give more chance to nodes having fewer ticket to go to sleep mode. To achieve this, we update the ticket distribution scheme as follows. After a node i forms its G_i, it first generates all subgraphs of G_i using only $s \leq N_i$ of its neighbors. Then for each subgraph of neighbor graph, it checks whether it can provide the required QoS (using the results obtained in Section III-B). If this is the case, then it increases the number of tickets that node i will give to each of the neighbors in this specific subgraph by one. After all subgraphs are processed, then node i sends the number of tickets it will give to each of its neighbors. Here, note that the voting mechanism of E2-LDAS works selectively rather than in random manner as it is done in the previous algorithms. Once all the nodes gather their tickets, the nodes assigns a back off time directly proportional to their ticket count and when it expires, they go to sleep mode if their current active neighbor count provides enough expected redundant coverage on its sensing area (bigger than required QoS). This self-selecting voting mechanism of E2-LDAS in the style of [12] forces nodes having critical connections with their neighbors (in terms of coverage redundancy) to gather more tickets so that they select longer back off times and lower their chance of getting into sleep mode.

C. Simulation Results

To evaluate the performance of proposed schemes, we performed a set of simulations. We randomly deployed N nodes in a 150 m x 150 m square region. We assumed all sensor nodes are identical and they have the same sensing and transmission range of 10 meters. For different N and required QoS values, we ran each proposed algorithm on ten different networks (created with different seeds) and computed the average active number of nodes that each algorithm achieve. Our initial results here only show the results of running each algorithm on the initially deployed network where all the nodes are active and have the same energy levels.

Figure 5 and Figure 6 show the number of active nodes obtained for two different required QoS (0.96 and 0.91) after running each algorithm on the initial network with different number of nodes deployed. We have selected these specific QoS ratios to show the difference of active node counts generated in U-LDAS and E1-LDAS algorithms more clearly. This is because the selected QoS values are slightly higher than the boundary values deciding the number of required number of neighbors in terms of expected redundant coverage. For example, when required $QoS = 0.91$, the number of one-hop neighbors required to achieve that QoS is four. However, as it is seen in Figure 3, three cases of 3-node neighborhood graph provide expected redundant coverage equal to or higher than 91%.

In both graphs we observe that U-LDAS generates fewer active nodes than LDAS while the required QoS is successfully achieved (all algorithms provide higher QoS than required, for brevity, we did not show the exact QoS of each algorithm). This shows the benefits of using expected coverage analysis to decide the required number of one-hop neighbors. Moreover, it is clear from the two graphs of results that the active node count generated by LDAS is always the maximum and the active node count generated by E2-LDAS algorithm is always the minimum among all four algorithms. The number of active nodes generated in E2-LDAS is sometimes half of the active nodes generated in LDAS. Comparing the performances of U-LDAS and E1-LDAS, we notice that when the required QoS is equal to 0.91 the difference in their performance is bigger than when it is 0.96. This is in agreement with the observation that we made in the description of E1-LDAS algorithm. When required $QoS = 0.91$, U-LDAS requires four neighbors but in 72.6% of all neighbor graphs with three neighbors E1-LDAS will find them sufficient for this QoS. On the other hand when required $QoS = 0.96$, U-LDAS requires five neighbors

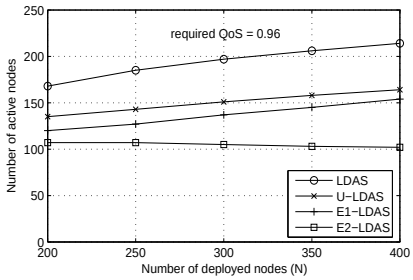

Fig. 5. Number of active nodes determined by each algorithm with different number of nodes deployed when required QoS=0.96.

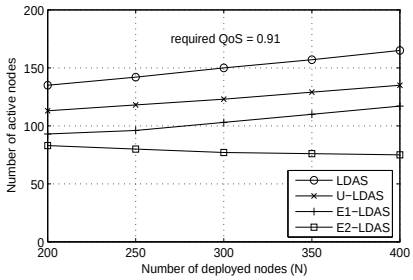

Fig. 6. Number of active nodes determined by each algorithm with different number of nodes deployed when required QoS=0.91.

but in 6% of all neighbor graphs with three neighbors E1-LDAS will require just three neighbors and in 37.6% of all neighbor graphs with four neighbors it will find four neighbors sufficient. Hence, E1-LDAS has higher chance of improvement over U-LDAS for $QoS = 0.91$ than for $QoS = 0.96$.

V. Discussions

A. Handling Connectivity

In this paper, we only focused on the coverage of sensor networks. However, the connectivity of nodes is required in most of the sensor network applications. If $R_t \geq 2R_s$ then coverage implies connectivity [2]. However, our algorithms can also be modified for different R_t/R_s ratios.

B. Balancing Energy Consumption

To balance the energy consumption among nodes in the network, we can use the remaining energy levels of nodes. For example, in E2-LDAS, if nodes first multiply their ticket counts by their remaining energy levels and assign a back off time proportional to this new value, they attempt to sleep earlier to save energy even though they have more tickets.

C. Complexity of Algorithms

The complexity of U-LDAS and E1-LDAS algorithms are almost the same as the original LDAS algorithm (except negligible cost of knowing two-hop neighbors and forming neighbor graphs). For E2-LDAS, at first glance it seems that computation of ticket counts to be distributed to each neighbor will increase the execution cost. However, a node can remember its previous computations, so it can often avoid this additional cost. On the other hand, in dense networks, where nodes have many neighbors, the cost of ticket computation can be high. This cost can also be reduced by dividing the nodes into distinct sets as in [9], such that each set is kept active at different times and the nodes in each set cover the network area sufficiently. Then, E2-LDAS can run on each of these smaller sets (where nodes have fewer neighbors) independently.

VI. Conclusion and Future Work

In this paper, we studied the coverage redundancy problem in wireless sensor networks in which nodes neither know their locations nor the distances to their neighbors. Specifically, we looked at the effects of neighbor graph connectivity on the expected redundant coverage by sensor nodes and we demonstrated that using neighbor graphs provides more accurate information than using only neighbor counts. In simulations, we also showed that utilization of neighbor graphs can improve the performance of coverage control protocols. In future work, we will extend our simulations and we will search for improvements throughout the lifetime of network. We also plan to compare proposed algorithms with other algorithms published in the relevant literature.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, *Wireless Sensor Networks: A Survey*, Computer Networks, 2002.

[2] H. Zhang, J. C. Hou, *Maintaining sensing coverage and connectivity in large sensor networks*, Journal of Ad Hoc and Sensor Wireless Networks 1 (12) pp. 89124, 2005.

[3] E. Bulut, I. Korpeoglu, *DSSP: A Dynamic Sleep Scheduling Protocol for Prolonging the Lifetime of Wireess Sensor Networks*, Third IEEE International Workshop on Heterogeneous Wireless Networks, in conjunction with AINA, Volume 2, pp. 725 - 730, 2007.

[4] L. H. Yen, Y. M. Cheng, *Range-based sleep scheduling (RBSS) for wireless sensor networks*, Wireless Personal Commun., vol.48, no.3, pp.411-423, 2009.

[5] T. P. Lambrou, C. G. Panayiotou, S. Felici, B. Beferull, *Exploiting Mobility for Efficient Coverage in Sparse Wireless Sensor Networks*, Wireless Personal Commun., April, 2009.

[6] S.Kumar, T.H.Lai, J.Balogh, *On k-coverage in a mostly sleeping sensor network*, in Proceedings of the 10th annual international conference on Mobile computing and networking, pp. 144-158, 2004.

[7] K. Wu, Y. Gao, F. Li, and Y. Xiao, *Lightweight Deployment-Aware Scheduling for Wireless Sensor Networks*, ACM/Springer Mobile Networks and Applications Journal, Vol. 10, pp. 837-852, 2005.

[8] R. Choudhury, R. Kravets, *Location-Independent Coverage in Wireless Sensor Networks*, Technical Report, UIUC, http://people.ee.duke.edu/ romit/publications.html, 2004.

[9] C. Liu, K. Wu, Y. Xiao, B. Sun, *Random Coverage with Guaranteed Connectivity: Joint Scheduling for Wireless Sensor Networks*, IEEE Trans. Parallel Distrib. Syst. 17(6): 562-575, 2006.

[10] F. Ye, G. Zhong, S. Lu, L. Zhang, *PEAS: A robust energy conserving protocol for long-lived sensor networks*, in Proceedings of the 23rd Intl Conference on Distributed Computing Systems, pp. 28-37, 2003.

[11] B. Wang, C. Fu, and H. B. Lim, *Layered diffusion-based coverage control in wireless sensor networks*, Computer Networks, Elsevier, Vol 53, Issue 7, pp. 1114-1124, May 2009.

[12] B. K. Szymanski and G. Chen, *Computing with Time: From Neural Networks to Sensor Networks*, The Computer Journal, Vol. 51, pp. 511-522, 2008.