Grid-based Approach for Working Node Selection in Wireless Sensor Networks

Haining Chen, Hongyi Wu, and Nian-Feng Tzeng
Center For Advanced Computer Studies
University of Louisiana at Lafayette
P.O. Box 44330
Lafayette, LA 70504
E-mail: {hxc5633,wu,tzeng}@cacs.louisiana.edu

***Abstract*— In this paper, we propose a grid-based working node (WN) selection approach for wireless sensor networks. Due to coverage redundancy, it is highly desirable to identify a minimum subset of sensors in a wireless sensor network to serve as WNs, while the remaining sensors are deactivated to save power and reduce potential interference. The basic idea of our solution approach is to represent the coverage of the sensors by a number of sample points, i.e., the intersection points of the established grid. A simple approximation algorithm and a linear programming method are employed to select as few sensors as possible to cover all sample points. In order to reduce the computational time, clusters are formed and WN selection is performed within each cluster. The performance of the proposed WN selection schemes is quantified and the tradeoff among accuracy, communication overhead and computational time is evaluated via analyses and simulations.**

Keywords: Clustering, linear programming, node and network coverage, sensor networks, set selection, working nodes (WNs).

I. INTRODUCTION

With the advances in integrated circuits, micro-mechanism, and radio technologies, the low-power, low-cost wireless integrated sensors have become available for various monitoring, assessing, and control applications [1] [2]. Due to the limited computing power, sensing range, and transmission range of individual sensors, the sensor network is formed to detect the indicated phenomenon and to deliver the collected data to one (or several) processing/aggregating sensor(s) via possible multiple hops. For some applications (such as forest environment monitoring), a large-scale sensor network can be deployed, with thousands or more sensors densely distributed in the interested area and working cooperatively for data collection and transmission. Additionally, to support effective monitoring, the sensors usually have location information through Global Positioning System [3] or local positioning systems [4] [5].

Various issues, such as sensor hardware design, location management, medium access control (MAC) and routing, have been discussed in the literature (see [6], [7], [8], [9]). This paper focuses on the topology management of sensor networks. Given the limited sensing range, each sensor has a *coverage area*, within which it can collect useful data. The union of the coverage areas of all sensors gives rise to the *system coverage*. For applications employing the sensor network in hostile environments, accurate sensor placement is usually impossible or non-cost-effective, and thus a large number of sensors are randomly distributed in the field to ensure the desired coverage and connectivity. The random sensor distribution may result in possible coverage redundancy (i.e., overlaps between the coverage areas of adjacent nodes), and it is highly desirable to identify a minimum subset of sensors, called *working nodes (WNs)*, which together provide the same system coverage; any node outside the subset can be deactivated not only to save power for future use but also to reduce potential interference in wireless channels and to lower contention in the medium access, thereby prolonging the total life span and enhancing the performance of the sensor network.

In this paper, we propose a grid-based WN selection approach. In brief, a grid with proper density is established and the coverage area of a sensor is represented by a set of intersection points of the grid. The WN selection problem can thus be converted into a set-covering problem (whose solution is NP-hard) and be solved by a simple approximation algorithm or a linear programming method. The grid density and its effects on accuracy are investigated and evaluated via analyses and simulations. In order to scale to large sensor networks, a distributed approach is developed, where the sensors are clustered by using the cluster formation algorithm [10] with the WN selection performed within each cluster. Extensive simulations are carried out to quantify the performance of the grid-based WN selection scheme, in terms of efficiency, communication overhead, and computational time. Our results show that the proposed approach can effectively select a small subset of sensors as WNs. The number of WNs selected by the linear programming approach is about 25% less than that obtained by the approximation algorithm. There is a tradeoff between the accuracy and the overhead. Specifically, with an increase in the cluster size, fewer WNs are selected (i.e., more accurate) at the expense of increased communication overhead and computing time. Our experiments show that the rational cluster size varies with the size of the network and the sensing range of the sensors. The running time of the proposed WN selection algorithm ranges from seconds to hundreds of seconds, depending on the cluster size and the grid density.

The rest of this paper is organized as follows. Section II

discusses related work. Section III introduces the principle of grid-based WN selection algorithm and its distributed implementation. Section IV presents the simulation results. Finally, Section V concludes the paper.

II. RELATED WORK

In [11], the authors have addressed a problem similar to WN selection for power-efficient organization of wireless sensor networks, where the sensors are placed at strategic locations. The coverage of a sensor is characterized by a number of *points* and the total coverage area is organized into a number of non-overlapped *fields*. A heuristic approach is proposed to solve the Set K-Cover problem (i.e., to find whether there are K disjointed covers for the sensor network). This is a centralized approach and is not scalable for large-scale sensor networks. In addition, the sensors are usually distributed randomly, instead of being placed at strategic locations.

The algorithm proposed in [12] has both centralized and distributed versions to identify a connected sensor cover, which includes a number of sensors that are fully connected and cover the entire network area. The subelements (similar to the *fields* in [11], which are the minimal regions formed by the intersection of the sensors' coverage), are assumed to be the input of the proposed algorithm. In a real system, however, such subelements are usually not available at hand and the generation of such subelements could be time-consuming (as to be discussed in Section IV-B). In addition, while the connected cover provided by [12] is useful in some scenarios, we intend to de-couple the coverage and the connectivity in this paper, because the latter depends on the routing requirement. For example, if multi-path routing is employed, a single connection maintained by [12] is not sufficient.

A recent study on the broadcast storm problem (see [13], [14], [15]) has yielded several algorithms to eliminate the excessive amount of retransmissions generated by flooding. Those algorithms enable each node to compute a subset of its neighbors for retransmissions according to local information of the node. The results, however, are local optimal with respect to the given nodes (within a range of two hops) and cannot be directly extended to the entire system. In addition, the computational method requires that all nodes have the same coverage, which is impractical for heterogeneous sensor networks at hand since such a network contains various types of sensors possibly with different transmission ranges.

III. PROPOSED APPROACH FOR WORKING NODES SELECTION

This section introduces a simple grid-based algorithm for working nodes (WNs) selection. We first discuss the selection principle and analyze its complexity and accuracy, and then present a distributed implementation of the WN selection algorithm.

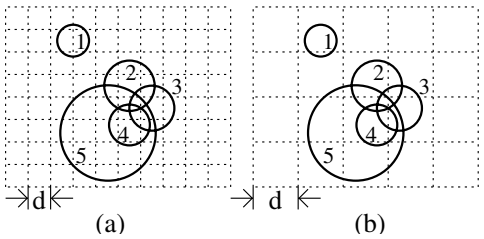

Fig. 1. Sampling with different grid distances under (a) dense grid and (b) coarse grid.

A. Principle of the grid-based algorithm

The basic idea of the grid-based approach is to establish a grid using the intersection points (referred as *sample points*) of the grid to represent the coverage area of the sensors. A number of nodes are selected as WNs to cover all sample points that span across the original network. For simplicity, we assume that the sensors have no significant movement during the period for running the algorithm.

1) Coverage Sampling: The grid can be enabled through a local positioning system to which all sensors in the network are aware. Specifically, the grid is set up in parallel with X-axis and Y-axis of the positioning system. The density of the grid (i.e., d in Fig. 1) is made available to the sensors during network initialization. Each sensor (say, sensor i) knows its own coordinates and thus can easily determine the set of sample points covered by itself, denoted as P_i, which represents the coverage area of sensor i.[1] The total coverage area of a network that includes M sensors is the union of the coverage of the M sensors, denoted by P:

$$P = \bigcup_{i=1}^{M} P_i. \tag{1}$$

On the other hand, each sample point may be covered by a number of sensors. We denote the set of sensors covering a given sample point n as Q_n. Clearly, we have

$$Q = \bigcup_{n=1}^{N} Q_n, \tag{2}$$

where Q is a set that includes all sample points in the network and N is the total number of sample points.

The grid density affects the accuracy and the time complexity of the proposed algorithm. Specifically, a coarse grid may lead to inaccurate coverage sampling. For example, node 2 and node 3 in Fig. 1(b) contain the same set of sample points although their coverage areas are different. As an extreme case, node 1 in Fig. 1(b) contains no sample points at all, and thus its coverage may be omitted completely. While increasing the grid density can improve accuracy, it also elevates the computational time of the algorithm. Our goal is to find an appropriate d that meets the accuracy requirement while at the

[1]The node coverage area is typically irregular. To simplify our illustration, however, a circle with a radius of R_i is assumed as the coverage of node i (see Fig. 1).

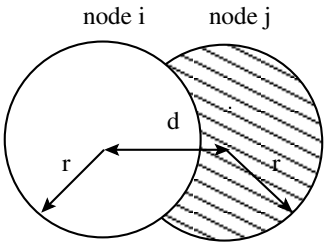

Fig. 2. Maximum undistinguishable coverage area.

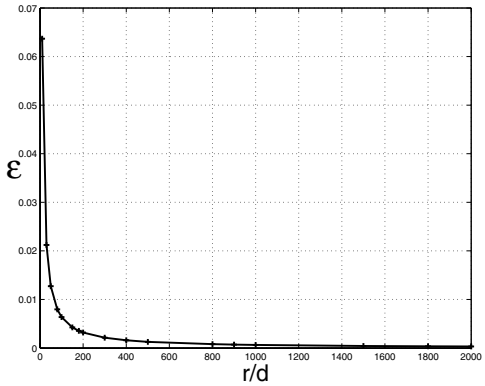

Fig. 3. ε vs. r/d.

same time keeping computational complexity low. To quantify the accuracy, we define the error of a grid to be the maximum difference between the coverage areas of two nodes i and j, with P_i = P_j (i.e., the maximum undistinguishable coverage area under such a grid). Clearly, to cover the same set of sample points, the maximum distance between node i and node j is d. Assuming R_i = R_j = r, we can derive the relative error ε by computing the ratio of the shadow area shown in Fig. 2 over the circle with radius r,

$$\varepsilon = 1 + \frac{d}{\pi r}\sqrt{1-(\frac{d}{2r})^2} - \frac{2}{\pi}\arctan\sqrt{(\frac{2r}{d})^2-1}\ , \qquad (3)$$

where $d < 2r$.[2] The relative error ε as a function of r/d is plotted in Fig. 3. As can be seen, ε decreases sharply with the increase of r/d. When $r/d \geq 800$, ε becomes less than 0.1%. An appropriate d value can be chosen according to the sensor's computing power and the accuracy requirement of applications.

Clearly, any two sample points, m and n with $Q_m = Q_n$, can be merged, creating *subelements* or *fields* similar to those mentioned in [11] and [12]. The process of generating subelements is called *preprocessing*. Without loss of generality, we consider each subelement or field to be a special sample point in the following discussions. As stated in [12], the maximum number of subelements in a 2-dimensional plane with M circles is $M(M-1)+1$. We notice that each subelement candidate must be compared with the existing subelements to determine whether it is a new subelement or not, and each determination involves at most M compare operations because every subelement is covered by at most M nodes. The above process could be repeated for at most N times because there are totally N sample points needed to be examined. As a result, the time complexity of preprocessing is $T = O((M(M-1)+1)\times M \times N)$, i.e., $T = O(N \times M^3)$.

2) Set selection: After each node is represented by a set of sample points, the WN selection problem becomes a set-covering problem, i.e., to select a minimum K WNs such that the sample points covered by the K WNs equal to P. Unfortunately, the set-covering problem is NP-hard and can be approached only heuristically by either a greedy algorithm [16] or a linear programming model [17].

In a greedy algorithm, the set with the largest number of sample points is selected, and the sample points in this set are removed from all the remaining non-empty sets. When there is a tie, the node with the smallest node ID is selected to break the tie. This procedure is repeated until all sets are empty. Without preprocessing, the time complexity of the greedy algorithm equals $T = O(N \times M \times min(N, M))$ [16]. Since N is normally much larger than M, we have $T = O(N \times M^2)$. As we can see, the time complexity of preprocessing is higher than that of the set selection algorithm without preprocessing. This suggests that the use of subelements or fields is inefficient. Therefore, no preprocessing is performed by default in the following discussion.

Minimize Z

Subject To

1) $Z = \sum_{k=1}^{M} S_k$
2) $S_k = \{0, 1\}, k = 1, 2, ..., M$
3) For each $Q_n \neq \phi, n \in \{1, 2, ..., N\}$, $\sum_{k \in Q_n} S_k \geq 1$

Fig. 4. A linear programming model for set covering problem.

The greedy algorithm usually yields non-optimal results, i.e., it selects more WNs than what is actually needed. In order to improve the solution quality, we resort to a linear programming model. We define a set of 0-1 variables $\{S_i |\ 1 \leq i \leq M\}$, with $S_i = 1$ indicating that node i is selected as a WN and $S_i = 0$ reflecting node i is deactivated. For every sample point n $(1 \leq n \leq N)$, if $Q_n \neq \phi$, we have the constraint

$$\sum_{k \in Q_n} S_k \geq 1, \qquad (4)$$

to guarantee sample point n is covered by at least one WN. The objective is to minimize the total number of WNs selected, i.e., to minimize $Z = \sum_{k=1}^{M} S_k$. The linear programming model is summarized in Fig. 4.

B. Distributed Implementation

The approach discussed above needs a central node to perform calculation, thus potentially resulting in a performance

[2] In order to include at least one sample point in any P_i, d should be smaller than the diameter of node i's coverage, i.e., $d < 2R_i$.

bottleneck and heavy signaling overhead. In this subsection, we introduce a distributed implementation of the proposed WN selection approach in order to reduce the signaling overhead and to disperse the computing load. Each sensor runs a distributed algorithm to form clusters. In this work, the Max-Min D-cluster formation algorithm [10] is adopted for its low time complexity ($O(D)$, where D is the maximum radius of the cluster formed by the algorithm, in terms of hops) and its scalability to large sensor networks. The readers are referred to [10] for the details of this algorithm. Once the clusters have been formed, every sensor sends to the cluster head a *report message*, which includes two types of information, namely, cluster information and coverage information. Cluster information includes the node's ID, its cluster head's ID, and the routing path from itself to the cluster head. Coverage information includes the coordinates and the sensing range of the sensor. When only one cluster head is formed (e.g., by setting D to be large enough), it turns into the centralized approach. Obviously, the distributed approach can reduce system overhead at the expense of result accuracy. This tradeoff will be studied and discussed in Section IV.

It is assumed in [10] that all the *report messages* sent from the cluster members can reach their cluster heads successfully, and we also assume this to be true, because the reliability of the intra-cluster communication is beyond the scope of this paper. After a cluster head has accumulated a full set of information from all nodes within its cluster, it runs the WN selection algorithm as discussed in Section III-A. In an alternative approach, the cluster head may pass all necessary information needed for computation to a control node which is much more powerful than an ordinary sensor in terms of computing ability. Upon being chosen, each WN is informed by a *notification message* from its cluster head with the remaining sensors being deactivated.

IV. SIMULATION AND DISCUSSION

In this section, we evaluate the proposed WN selection scheme via simulations and discuss the results in terms of the number of WNs selected, computing time, and signalling overhead. We simulated $M = 100$ sensors distributed randomly in an area of $50m \times 50m$, with the default sensing range $r = 10m$ and grid density $d = 0.1m$. All simulations run on a PC with one Pentium-4 CPU of 2GHz.

A. Number of WNs selected

In this subsection, we study the impact of such factors as different algorithms, sensor's density and node distributions, on the number of WNs selected.

1) linear programming vs. greedy algorithm: First of all, we compare the greedy algorithm with the linear programming approach. A tool called *lpsolve* is employed to solve the linear programming problem [18]. Eight experiments have been conducted with different sensor distributions. The results are shown in Table I. In experiments 1-4, the number of WNs selected by linear programming is about 25% less than that selected by the greedy algorithm. However, the linear programming approach takes much longer computing time than the greedy algorithm does. For example, under the default simulation setting, the greedy algorithm needs 2 seconds to finish WN selection while linear programming takes more than 1000 seconds. Worse yet, the linear programming approach fails to yield converged results in experiments 5-8.

TABLE I

NUMBER OF WNS SELECTED OUT OF $M = 100$ NODES DEPLOYED RANDOMLY IN AN AREA OF $50m \times 50m$, WITH $10m$ OF SENSING RANGE (WHERE '-' INDICATES THAT THE ALGORITHM FAILS TO CONVERGE)

Expriment	1	2	3	4	5	6	7	8
Linear Prog.	16	16	16	17	-	-	-	-
Greedy Algo.	21	20	21	22	20	19	20	21

For another set of 8 experiments, we increase the number of senors to $M = 400$, and enlarge the area to $100m \times 100m$. Linear programming fails in all eight experiments, while the greedy algorithm succeeds always, taking less than 30 seconds for each experiment. The reason why linear programming takes such a long time can be attributed to the huge number of constraint functions generated. It can be inferred from Fig. 4 that for each effective sample point n with $Q_n \neq \phi$, there is a constraint function. In our default simulation network size, there are some 27 thousand constraint functions, which sometimes make linear programming fail, as observed in Table I. When the network area is enlarged by 4 times, N also increases by 4 times, and the number of constraint functions increases to 130 thousand, almost 4 times as many as before. It is thus not surprising to experience much more failures in convergence when trying to satisfy so many constraint functions under linear programming. Clearly, while it provides a result closer to the optimal WN selection if converged, the linear programming approach is not practical for large sensor networks because of its extremely high time complexity. Thus, the experiments discussed in the following sections are solved by the greedy algorithm only.

2) Impact of node density: We study the impact of node density on WN selection with M ranging from 10 to 200. The hop D used in cluster formation is set to be 10. The results are shown in Fig. 5 and Fig. 6.

As can be observed in Fig. 5, more clusters are formed with the decrease of M because the sensors are largely separated due to their sparse distribution. When M is small, the overlap degree of sensor coverage is negligible and thus WN selection is not necessary because almost all nodes are then selected. Another interesting observation is that K (the number of WNs selected) does not increase after the node density reaches a certain value. As shown in Fig. 6, when M exceeds 30, K remains the same. This is reasonable because the whole area is almost fully covered by the deployment of 21 WNs selected out of 30 or more sensors, and thus adding more sensors will not increase the number of WNs selected.

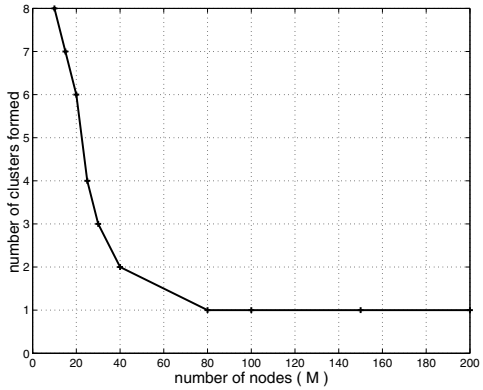

Fig. 5. Number of clusters under different node density.

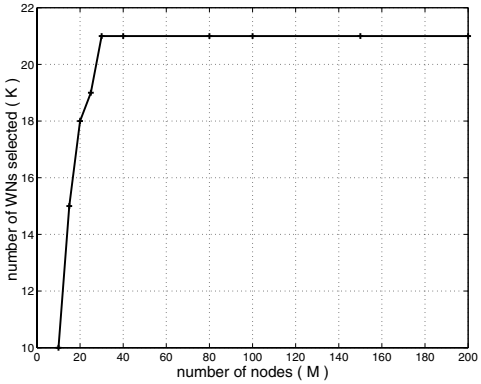

Fig. 6. Number of WNs under different node density.

3) Impact of different node distributions: In this scenario, we study different nodal distributions with only one cluster formed. For each given nodal distribution, we start with the densest grid of $d = 0.1m$, and expand d by an incremental amount of $0.1m$ repeatedly until a different set of WNs are selected (i.e., until the coarse grid introduces inaccurate results). We denote d^* to be the maximum d that yields the same set of WNs as those selected under $d = 0.1m$.

Though the results are not shown here, we have observed that different distributions result in widely different d^*, varying from $0.3m$ to $2.1m$ in this simulation. There is no single "best" grid density for a given set of network size, number of sensors and sensor coverage, because different distributions generate different errors in coverage sampling.

B. Time complexity of the grid-based approach

1) Impact of different number of nodes on computational time: In order to study the impact of a large M on the computational time of the WN selection algorithm, we vary M from 100 to 2000. Only one cluster is formed in the simulation. The results are shown in Fig. 7. The computing time required for 2000 nodes is less than 35 seconds, which is acceptable for many applications. When the number of nodes is less than 300, the time required is less than 5 seconds. Note that the computing time required for WN selection decreases accordingly when more clusters are formed.

2) Impact of different grid density on time complexity: We have also studied the time spent in WN selection under different grid density. We simulate $M = 100$ sensors distributed in an area of $2km \times 2km$. Different sensors have different sensing ranges with minimum, mean and maximum values of $10m, 100m$ and $500m$, respectively. Only one cluster is formed. The simulation is conducted with and without preprocessing, and the results are shown in Fig. 8. It is observed that the time complexity of WN selection is proportional to N (or $1/d^2$), confirming our analyses in Section III-A. In addition, it takes longer with preprocessing to merge sample points and create subelements than without it, signifying that the subelement generation is time consuming and usually not beneficial in solving the WN selection problem.

C. Signalling overhead of the grid-based approach

In this subsection, we study the signalling overhead in the distributed approach. We define the signalling overhead as the number of report messages and notification messages forwarded between cluster head and the sensors in its cluster. As can be observed in Fig. 9, when D increases, the number of cluster decreases, and the number of WNs selected also drops accordingly. The reason is that two overlapped sensor nodes may belong to different clusters when D is small, and when D is increased, they can be grouped together in one cluster such that only one of them needs to be elected as WN. Fig. 10 shows that the communication overhead grows with the increase of D. According to a given accuracy requirement, proper D can be chosen to limit the signalling overhead while achieving high accuracy at the same time. Although the results are not shown here, we also increase the sensing range of the sensors (r) from 10 to 15. We observe that with the increase of r, fewer clusters are formed, fewer number of WNs are selected, and less communications overhead is experienced.

V. CONCLUSION AND FUTURE WORK

We have proposed a grid-based approach for the working node (WN) selection problem in wireless sensor networks. The sensor's coverage area is represented by a number of sample points, and the WN selection problem is translated to a set-covering problem, whose solution is NP-hard. We have employed a greedy algorithm and a linear programming method to approximate the solutions for the problem. Our simulation results show that the two proposed approaches can effectively select a small set of sensors acting as WNs. The accuracy of the greedy algorithm is within 25% of that of the linear programming approach. Due to its high time complexity, merging sample points into subelements is not suitable for solving the WN selection problem. As a tradeoff exists between accuracy and computing complexity, proper parameters can be chosen for the proposed greedy algorithm to meet the accuracy and delay requirements. In our future research, we will rectify the proposed approach by taking node failures and mobility into consideration.

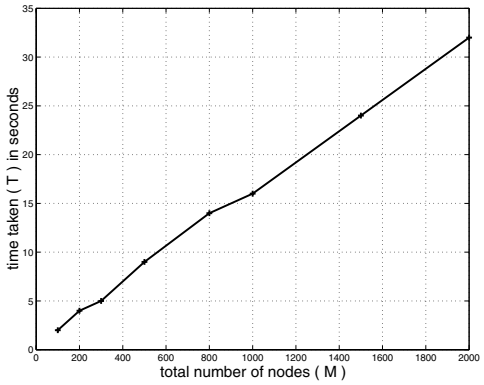

Fig. 7. Time used for WN selection under large number of nodes.

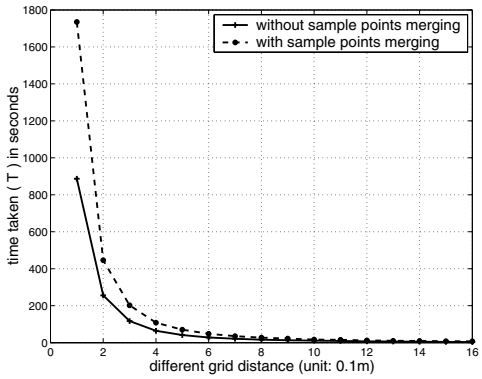

Fig. 8. Time used for WN selection under different grid density.

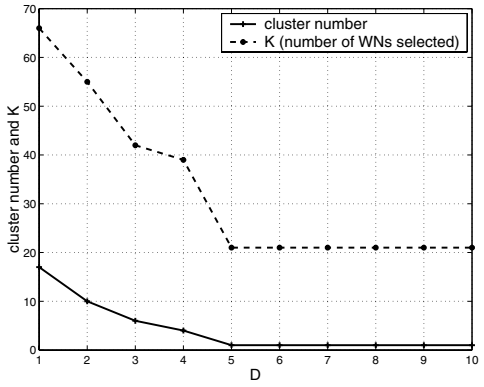

Fig. 9. Impact of D on the cluster size and number of WNs selected.

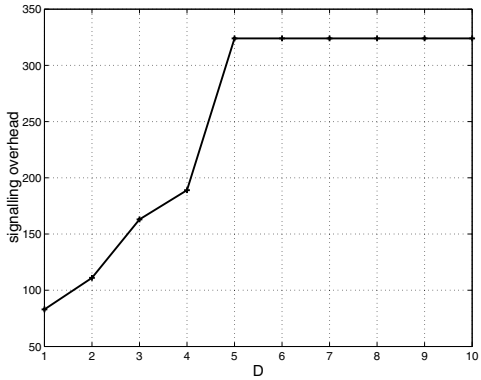

Fig. 10. Impact of D on communication overhead.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," in *IEEE Communications Magazine*, August 2002, vol. 40, pp. 102–116.

[2] Q. Fang, F. Zhao, and L. Guibas, "Lightweight sensing and communication protocols for target enumeration and aggregation," in *Proc. of ACM MobiHoc 2003*, June 2003, pp. 165–176.

[3] W. Su, S. J. Lee, and M. Gerla, "Mobility prediction in wireless networks," in *Proc. of 21st Century Military Communications Conference (MILCOM 2000)*, October 2000, vol. 1, pp. 491–495.

[4] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less lowcost outdoor localization for very small devices," in *IEEE Personal Communications*, October 2000, vol. 7, pp. 28 –34.

[5] D. Niculescu and B. Nath, "Ad hoc positioning system(APS)," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM 2001)*, November 2001, vol. 5, pp. 25–29.

[6] M. Haenggi, "Mobile sensor-actuator networks: Opportunities and challenges," in *Proc. of the 7th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA 2002)*, July 2002, pp. 283 –290.

[7] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *Proc. of ACM MobiHoc 2003*, June 2003, pp. 201–212.

[8] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proc. of IEEE Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, June 2002, vol. 3, pp. 1567 –1576.

[9] M. A. Youssef, M. F. Younis, and K. A. Arisha, "A constrained shortest-path energy-aware routing algorithm for wireless sensor networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC 2002)*, March 2002, vol. 2, pp. 794 –799.

[10] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *Proc. of 19th Joint Conference of the IEEE Computer and Communication Societies (INFOCOM 2000)*, 2000, vol. 1, pp. 32–41.

[11] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless networks," in *Proc. of IEEE International Conference on Communications*, June 2001, vol. 2, pp. 472–476.

[12] H. Gupta, S. R. Das, and Q. Gu, "Connected sensor cover: Self-organization of sensor networks for efficient query execution," in *Proc. of ACM MobiHoc 2003*, June 2003.

[13] S. Ni et al, "The broadcast storm problem in a mobile ad hoc network," in *Proc. of 5th ACM/IEEE Int'l Conf. on Mobile Computing and Networking*, August 1999, pp. 151–162.

[14] W. Peng and X. Lu, "On the reduction of broadcast redundancy in mobile ad hoc network," in *Proc. of ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing*, August 2000, pp. 129–130.

[15] M. Sun and T. Lai, "Computing optimal local cover set for broadcast in ad hoc network," in *Proc. of IEEE Int'l Conf. on Communications*, May 2002.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 2nd edition, 2001.

[17] CMU, *linear programming model to solve set-covering problem*, http://mat.gsia.cmu.edu/orclass/integer/node8.html.

[18] netlib.org, *linear programming solving tool lpsolve*, http://www.netlib.org/ampl/solvers/lpsolve/.