Wireless Sensor Network Scheduling Schemes

by

Bo Wang

A thesis submitted to the faculty of graduate studies
Lakehead University
in partial fulfilment of the requirements for the degree of
Masters of Science in Computer Science

Department of Computer Science
Lakehead University
July 2009

Lakehead
UNIVERSITY

OFFICE OF GRADUATE STUDIES

NAME OF STUDENT:	Bo Wang
DEGREE AWARDED:	Masters of Science in Computer Science
ACADEMIC UNIT:	Department of Computer Science
TITLE OF THESIS:	Wireless Sensor Network Scheduling Schemes

This thesis has been prepared under my supervision
and the candidate has complied with the Mater's regulations.

Signature of Supervisor

Date

Contents

List of Tables

List of Figures

Abstract

Wireless Sensor Network (WSN) consists of small sensing devices with constrained power and resources. It is widely used for environment monitoring. WSN usually works in hostile environments and performs critical tasks, for example, battlefield surveillance, forest fire detection. This requires sensor network applications to conserve the limited energy and meanwhile provide reliable service. Our goal was set to address these issues on WSN.

In this thesis, two sensor scheduling algorithms are proposed for power saving on WSNs. In the first proposal a new concept of local Clique is introduced. Based on that, a new distributed node scheduling algorithm CCMS (Coverage and Connectivity Maintaining Scheduling scheme) is designed, which can allocate all nodes in the sensor network into k different groups without requiring location information, and at the same time, it can guarantee that each group will still be connected and maintain the coverage ratio as high as possible. In the second proposed algorithm, a new concept - combinatorial assignment code (CAC) is introduced. On top of that, a new distributed node scheduling algorithm CNSA (CAC based Node Scheduling Algorithm) is designed, which can allocate all nodes in the sensor network into k ($k \leq t$) different groups $\{0,1,...,k\text{-}1\}$. The second method can also elongate the lifetime of a WSN with guaranteed connectivity. The major advantage of the second scheme is that it also has fast execution speed.

Previous works have tried to improve either a sensor network's life time, or try to guarantee coverage. Not very much has been done to address both of the issues at once. Also connectivity, which is a very important character of WSN, has constantly been overlooked in the scheduling shames. Simulation results show both of our proposals prolong the network's lifetime significantly and guarantee coverage and connectivity of the network at the same time. Simulation also indicated that our scheme outperform other schemes on life time, coverage and connectivity.

Acknowledgements

First of all, I would like to thank my supervisor Dr. Ruizhong Wei who has been providing me continuous guidance and support during the completion of my thesis. I also want to thank my co-supervisor Dr. Deli Li for the constant supports and concerns of yours. Last but not least, I want to thank Dr. Lei Wang who has had very valuable input to this research. To all of you, without your help, I could not have completed the work. Thank you!

Some special thanks to my family:

Mom and Dad,

Thank you for all the unquestioning trust and supports.

Chapter 1

Introduction

With capabilities of sensing, data processing, GPS positioning and communicating, sensors have great potential in a wide variety of enterprise and military applications, such as environmental and military monitoring, earthquake and weather forecast, underground, deep water and outer space exploration. Because of the surrounding uncertainty, usually hundreds or thousands of sensor nodes need to be deployed simultaneously. Those nodes are self-organized to collect special information of a sensory field. A wireless sensor network (WSN) is a collection of sensor nodes deployed over a region of interest for sensing or monitoring certain conditions or events, and collect data for further analysis in order to achieve the goal of such deployment[1, 2, 3]. Due to their extremely small dimension, sensor nodes have very limited energy supply, and it is usually hard to recharge the battery after deployment, either because the number of sensor nodes is too large, or because the deployment area is hostile. So, one challenge in the research of sensor networks is to obtain a prolonged system lifetime by using the limited energy sources. Because of the densely distribution of sensor nodes, node scheduling plays a critical role for energy efficiency in WSNs [4, 5, 6].

The idea of sensor scheduling scheme is to allow redundant nodes to go to sleep mode, so the power of the network is conserved. And thus the lifetime of the WSN is extended. Intuitively, there is no better way to save power than turn it off. However the nodes in sleep mode are no longer part of the network. That is a major trade-off. There will be fewer nodes working in the network after the scheduling scheme is executed. So how to decide which nodes are redundant is critical for the scheduling schemes. A well designed scheduling scheme should maintain the same performance level.

Currently, there are many node scheduling methods proposed, which can be classified into the following two major categories: round-based node scheduling and group-based node scheduling. In a round-based node scheduling method, the sensor nodes will execute the scheduling algorithm during the initialization of each round. According to certain off-duty qualification check, some nodes will be exempt from working for the round, and other nodes will keep awake instead [7,8,9]. This method requires each sensor node to execute the scheduling algorithm every time at the beginning of each round. While in a group-based node scheduling method, each node only executes the scheduling algorithm once after its deployment. After the execution of the scheduling algorithm, all sensor nodes will be allocated into some different groups. Thereafter, in each of the followed time slots, each group of nodes will keep active in turn [10, 11].

In a node scheduling method, there are two major problems needed to be considered: the maintenance of coverage and the connectivity for the whole sensory field. One difficulty of the scheduling problem is that a sensor's sensing range is totally independent with its radio transmission range. So it is generally hard to combine and solve the two problems together [10]. Most researchers focus on the research of scheduling based on sensing coverage [12,13,14,15] or scheduling based on network connectivity [16,17,18,19] separately. Since it is usually costly to obtain and maintain the location information of each sensor node, the joint scheduling problem is even more difficult if the location of each sensor node is unknown [10,20,21].

If each point in a sensory field is monitored by at least t sensors, then we call the field is t-covered. Sensor nodes are usually densely deployed, i.e. we can assume that the sensory filed is m-covered [22]. If it is a scarce distribution, there is no need to use the scheduling scheme, because there is no redundant node to put in sleep. In this paper, we deal with the following challenging task: Assume that the sensor network is t-covered and connected. Without location information, how can we schedule the sensor nodes into k, ($k \leq t$) different groups $\{0,1,...,k\text{-}1\}$, such that each group will be still connected and can at the same time maintain the coverage ratio as high as possible?

In order to solve the above problem, in this thesis, first, we introduce a new structure named clique. Based on clique, we developed out new group ID assignment algorithm within each clique. This algorithm consists of two major steps. One is the native clique finding, and assign each node a group id in a clique by clique-head. Another step is to maintain connectivity, for each of the groups. This can make sure that nodes in each group are still connected after assigned group id. This algorithm's strength is thorough. However, it is relatively time consuming. Second, based on the knowledge of Combinatorics [23,24]we defined a new concept named combinatorial assignment code (CAC) firstly, which can be used to assign group IDs evenly to nodes in a local neighboring cluster, and then, based on the concept of CAC, we designed a new node scheduling scheme, which can allocate all the nodes in the whole sensor network evenly into all these k groups through a distributed way. And finally, we give a connectivity maintenance algorithm to guarantee the connectivity for each group.

For verification and testing purpose, we chose Network Simulator version 2 (shorten as NS2) as the testing platform. NS is originally designed for net work simulation, but not for wireless Sensor Network (WSN). Fortunately, NS2 includes the wireless network extension and energy model to fit our simulation purpose for WSN.

The rest of the thesis is organized as follows. First we extensively discuss relevant topics regarding sensor network's energy consumption, coverage, and connectivity in chapter two. In chapter three we analyzed three sensor scheduling scheme for case study and comparison purpose, especially the RSGC scheme. Then in chapter four and five we presented two of our proposals CCMS and CNSA with detailed algorithm and performance analysis. Then in Chapter six we summarized the work in this thesis, and provided some outlook for the future work.

Chapter 2

Research Background

In this section, we will discuss researches which are related to our study. Then we will exploit the problems that we are going to address.

Lots of previous works has been done on topics of wireless sensor network. Especially, nowadays, the processing capability of the embedded device increase dramatically, which allow the sensors to host more sophisticated applications. With the fast developing technology, the cost of the sensor is also reduced, so the large range of wireless sensor network is available. Popular researches about WSN include *energy-aware routing, sensor deployment problem, data aggregation, etc.* Among all of these topics, to guarantee the coverage, and conserve the limited battery energy of the sensor devices are of significant importance. Full-filling this purpose is also the target of our research. In this section, we listed some of the relevant works.

2.1 Wireless Sensor Deployment

Sensor deployment scheme is a popular subject in WSN studies. A well designed deployment can receive the result of better coverage and longer life time. Its drawback is that not all the environment is suitable for a designed deployment to be performed, such as the interior of an active volcano.

Chu-Fu Wang and Shu-Chien Huang [25] proposed an efficient sensor deployment scheme that effectively extends the life time of the monitoring sensor network. Meanwhile the coverage holes problem can also be solved. The proposed scheme is based on the sensor balanced deployment scheme [26] and route traffic analysis. In the balanced deployment scheme, it defines the sensor field reaches a balanced state when the number of sensors in each unit area is equal in a Mesh structure. So the target of the balanced deployment scheme is to maximum distribute the sensors in the monitored field evenly. Only one constraint is that the number of sensors in the unit area (defined by density, notion d) would be greater than or equal to $\frac{S_{unit}}{\pi \times {r_s}^2}$ (S_{unit} stands for the size of the unite area in square meters, and r_s is the sensing radius). The scheme seems to be sound if the power usage for each sensor is uniform. However, things do not work this way in practical. From the perspective of energy consumption, reaching the balanced state may not meet the goal of elongating the network lifetime since the nodes that are near the sink may intuitively consume more energy than others. To be more accurate, the nodes on the route with heavy traffic drain the battery power faster than others. Against the problem of the balanced deployment, [25] combine the balanced deployment scheme with traffic analysis to compensate the heavy-traffic area with more nodes at deployment time. Two rules apply in the scheme: first, above constraint $d \geq \frac{S_{unit}}{\pi \times {r_s}^2}$ still needs to meet for the entire sensor field. Second, when try to deploy more nodes to the heavy-weight area, still need to guarantee the remaining nodes can provide basic coverage. The scheme can work with both static and dynamic routing protocol. The scheme is depend on the distribution of the monitored events, and the traffic is weighted statistically. So the scheme assumes the distribution of the events is known, or otherwise assumes that it is a uniform distribution. The simulation focuses on the life-time comparison between the balanced deployment scheme and the proposed scheme.

Figures below illustrate the simulation of the proposed deployment scheme and the balanced scheme respectively. The number of sensor devices for deployed were made on $n = 150, 175, 200, 225, \ldots, 400$. The sensing field is a $100 \times 100\, m^2$ square coordinate grid. Then it performed 500 times of network operations for random event sensing and reporting task to obtain 500 network lifetime values and their average. The radio transmission and sensing ranges of each sensor device were set to $20m$, $15m$, $14.4m,\ and\ 10m$. Figures 2.1-1 and Figure 2.1-2 give examples of the sensor deploying scheme for using the balanced deploying algorithm and the two-phased sensor deploying algorithm when $n = 200$, respectively. Figure 2.1-3 shows the performance results for the network lifetime comparison when $r_{trans} = 20m$ and $r_{sensing} = 14.4m$. Since the events were randomly occurred, the length of the network lifetime for each simulation case varied. Although the curves in the figure are not smooth, as shown in Figure 2.1-3, the proposed two-phased deploying algorithm performs much better than the balanced deploying algorithm on network lifetime comparison. Figure 2.1-4 shows the performance results for the network lifetime comparison when $r_{trans} = 15m$ and $r_{sensing} = 10m$. This figure also gives similar results.

Problems with the above proposal are that it only works for situations where the shape of the sensing field is irregular or unknown. Furthermore, the scheme assumes that the distribution of the monitored events is either uniform, or known. Without having the above conditions meet, the deployment algorithm cannot be executed.

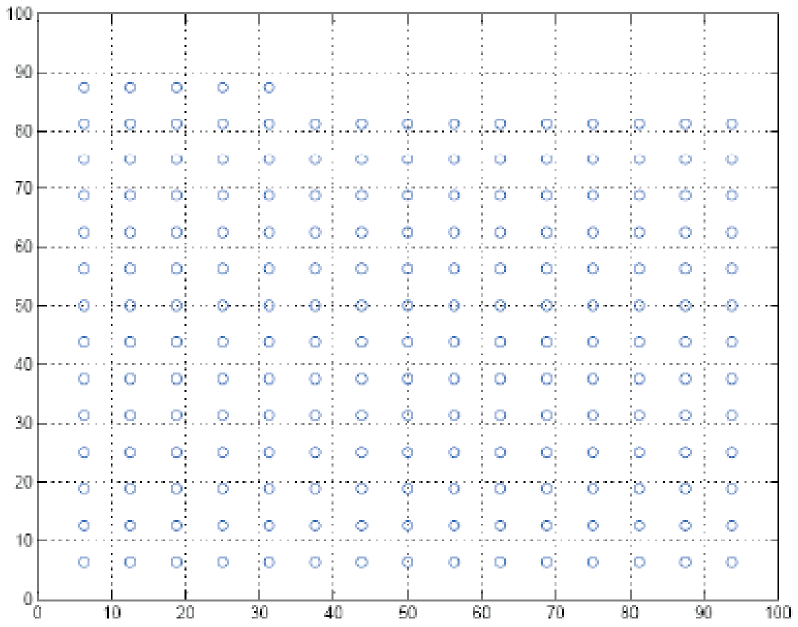


Figure 2.1-1 the balanced deploying scheme (n = 200)

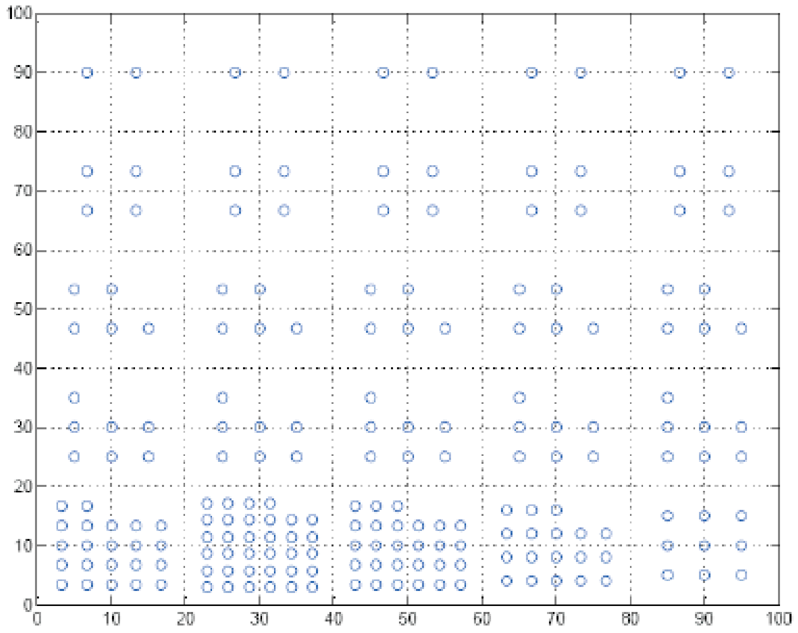


Figure 2.1-2 the two-phased deploying scheme

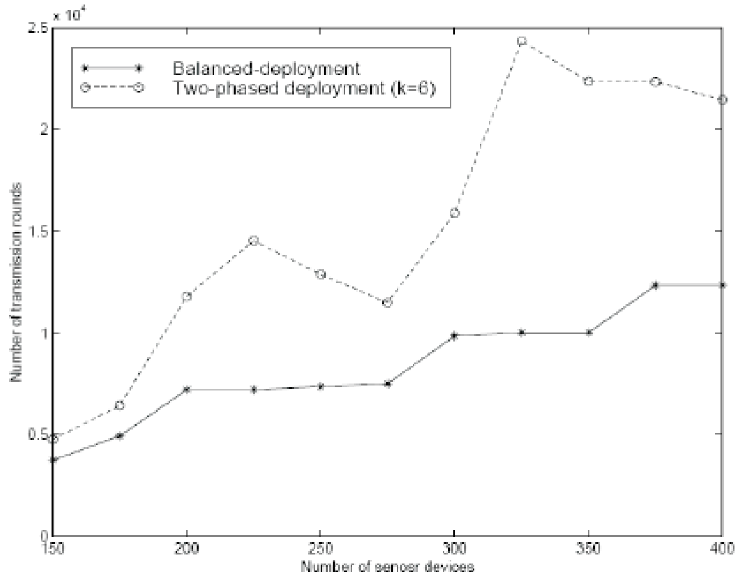


Figure 2.1-3 the network lifetime comparisons when $r_{trans} = 20m$ and $r_{sensing} = 14.4m$

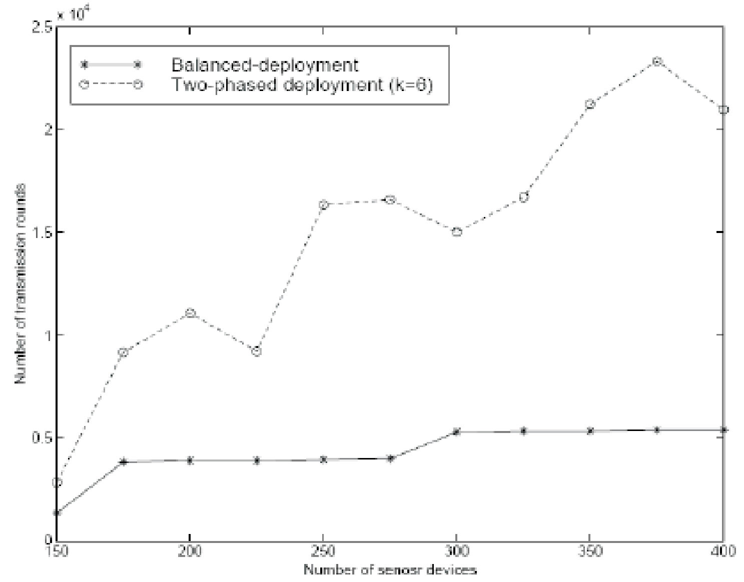


Figure 2.1-4 the network lifetime comparisons when $r_{trans} = 15m$ and $r_{sensing} = 10m$

2.2 Coverage of Wireless Sensor Network

Coverage is another fundamental characteristic of a WSN. Due to the large variety of sensors and applications, coverage is subject to a wide range of interpretations. In general, coverage is usually used to evaluate the quality of service provided by a WSN. For example, a forest fire detection sensor network. To evaluate the effectiveness of the sensor network, one may ask how well the network can observe a given area and what the chances are that a fire starting in a specific location will be detected in a given time frame. Furthermore, coverage formulations can try to find weak points in a sensor field and suggest future deployment or reconfiguration schemes for improving the overall quality of service.

The Voronoi diagram [27] has been re-invented, used, and studied in many domains. The use of Voronoi diagram can converts the problems into discrete graph problems, so that they are easier to be solved on computer. According to [27] it is believed that the Voronoi diagram is a fundamental construct defined by a discrete set of points. In 2D, the Voronoi diagram of a set of discrete sites (points) partitions the plane into a set of convex polygons such that all points inside a polygon are closest to only one site. This construction effectively produces polygons with edges that are equidistant from neighboring sites. Figure 2.2-1 shows an example of a Voronoi diagram for a set of randomly placed sites. Reference [27] presents a detailed survey of Voronoi diagrams and their applications. [28] was the first to identify the importance of computational geometry and Voronoi Diagrams in sensor network coverage.

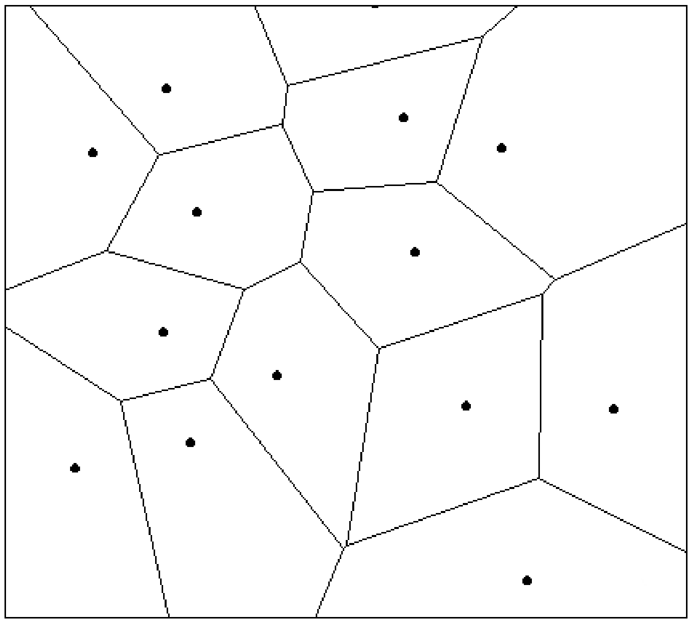


Figure 2.2-1 The Voronoi Diagram of a Set of Randomly Placed Points in a Plane.

Another structure that is directly related to Voronoi diagrams is the Delaunay triangulation [29]. The Delaunay triangulation can be obtained by connecting the sites in the Voronoi diagram whose polygons share a common edge. It has been shown that among all possible triangulations, the Delaunay triangulation maximizes the smallest angle in each triangle. Also, neighborhood information can be extracted from the Delaunay triangulation since sites that are close together are connected. In fact the Delaunay triangulation can be used to find the two closest sites by considering the shortest edge in the triangulation.

Seapahn Meguerdichian, Farinaz Koushanfar, and their colleagues discussed the coverage problem in WSN in [30]. In their study, they defined the coverage problem from several points of view including deterministic, statistical, worst and best case. By combining computational geometry and graph theoretic techniques, specifically the Voronoi diagram and Delaunay triangulation, with graph search algorithms, they proposed optimal polynomial time worst and average case algorithm for coverage calculation. Two perspectives of coverage are introduced into this study to evaluate the coverage: *worst* and *best* case coverage. In worst-case coverage, attempts are made to quantify the quality of service by finding areas of lower observability from sensor nodes and detecting breach regions. In best-case coverage, finding areas of high observability from sensors and identifying the best support and guidance regions are of primary concern. Our main objective is the design of robust, efficient and scalable algorithms to be used in wireless multi-sensor integration. The use of Voronoi diagram, efficiently and without loss of optimality, transforms the continuous geometric problem into a discrete graph problem. Furthermore, it enables direct application of search techniques in the resulting graph representation. [30] used the properties of the Voronoi diagram and Delaunay triangulation to solve for best and worst case coverage. Now take a look at how the worst and best case coverage is established. The worst case coverage is decided by the search of Maximal Breach Path (P_B) between any two arbitrary points initial (I) and final (F) in a sensing field (A) with sensors set (S). P_B is defined as a path through the field A, with end-points I and F and with the property that for any point p on the path P_B, the distance from p to the closest sensor is *maximized.* Since by construction, the line segments of the Voronoi diagram maximize distance from the closest sites, the Maximal Breach Path P_B, must lie on the line segments of the Voronoi diagram corresponding to the sensors in S. If any point p on the path $P_B$deviates from Voronoi line segments, by definition, it must be closer to at least one sensor in S. Thus, the solution to the problem is obtained by finding P_B from the Voronoi diagram of S using Binary-Search and Breadth-First-Search. It must be noted that the original Voronoi diagram is unbounded. However, A is in fact a finite area Therefore, it is necessary to clip the Voronoi diagram to the boundaries of A, and a bounded Voronoi diagram it became. The Maximal Breach Path is not necessarily unique. In fact, in general, there are many paths that can qualify as the Maximal Breach Path. However, they all use edges with weights that are larger or equal to the *Breach Weight* determined in the binary search phase of the algorithm. The *Breach Weight* found by the algorithm is the minimum distance from sensors that an agent traveling on any path through the field A (from I to F) must encounter at least once. If new sensors can be deployed or existing sensors moved such that this *Breach Wright* is decreased, then the worst-case coverage is improved. Similar to the worst case coverage, best case coverage is decided by the maximal support path (P_S). Definition of P_S is similar to P_B, except that distance from p to the closest sensor is minimized. Since the Delaunay triangulation produces triangles that have minimal edge lengths among all possible triangulations, $P_S$must lie on the lines of the Delaunay triangulation of the sensors in S. The algorithm for solving for P_S is very similar to the breach algorithm with a few changes. First, Voronoi diagram is replaced by the Delaunay triangulation. Second, the weight of the edges becomes the length of themselves in the Delaunay triangulation. Third, the *Breach Weight* becomes the Support Weight. The maximal support path may also not be unique. The situation is analog of the worst case coverage. If changes made to the sensor network such that *Support Weight* is decreased, then the best-case coverage is improved.

The complexity of the above algorithms is also an important factor, since sensors are limited in resource and computing capability. The best known algorithms for the generation of the Voronoi diagram have $O(n \log n)$ complexities. The conversion to graphs and weight assignments can be accomplished in linear time and therefore do not add any significant overhead to the computation. In most cases, BFS (and DFS) have $O(m)$ complexity where m is the number of edges in the graph. Since in realistic cases we deal with sparse networks the actual complexity is $O(n)$ while it could be as high as $O(n^2)$. Binary search is accomplished in $O(log\ range)$ where range is usually limited. So, while the worst case complexity of the algorithm is $O(n^2 \log n)$, in practice the networks are sparse and the overall complexity is $O(n \log n)$, dominated by the Voronoi procedure which has large constant factor in its complexity.

The following example demonstrates the application of the coverage scheme. Figure 2.2-2 shows an instance of the coverage problem, where 30 sensors are deployed at random. The Maximal Breach Path (P_B) and the corresponding edge with *Breach Weight* depicts where the breach takes place in the field. The Maximal Support Path (P_S) and the corresponding edge with *Support Weight* are also shown.

Figure 2.2-3 shows the underlying bounded Voronoi diagram for the same problem instance. Extra edges with 0 weight are used to connect the I and F regions to the structure so that all possible paths can be considered in the search algorithm. Figure 2.2-4 shows the corresponding Delaunay triangulation. In this case, only two extra edges are introduced to connect I and F to the closest sensors in the structure.

The edges corresponding to *Breach Weight* described previously can be used as a guide for future sensor deployments. Since *Breach Weight* corresponds to the edge in the breach path where P_B is closest to the sensors, deploying additional sensors along that edge can improve overall coverage. Figure 2.2-5 shows the average improvement in breach coverage when up to 4 additional sensors are introduced in the network according to the heuristic described above. The results represent average improvements over 100 random deployments. It is interesting that even after deploying 100 sensors, breach coverage can be improved by about 10% by deploying just one more sensor.

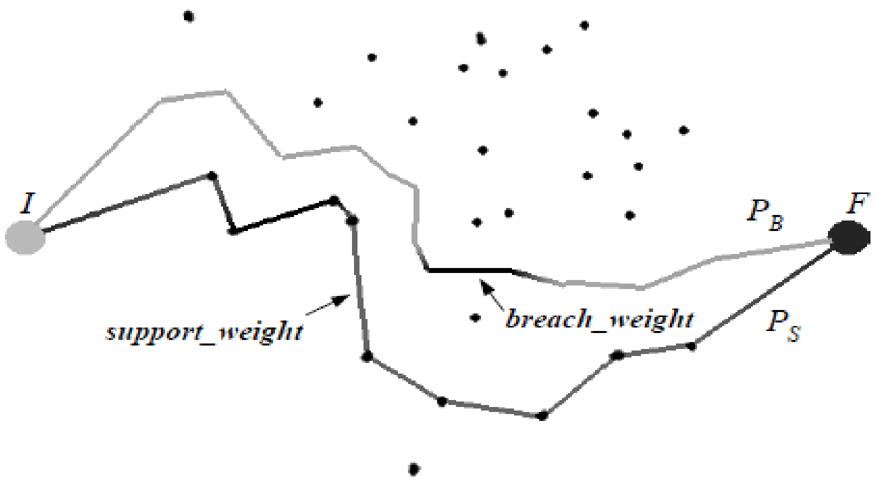


Figure 2.2-2 Sensor Field with Max Breach Path (P_B) and Max Support Path (P_S)

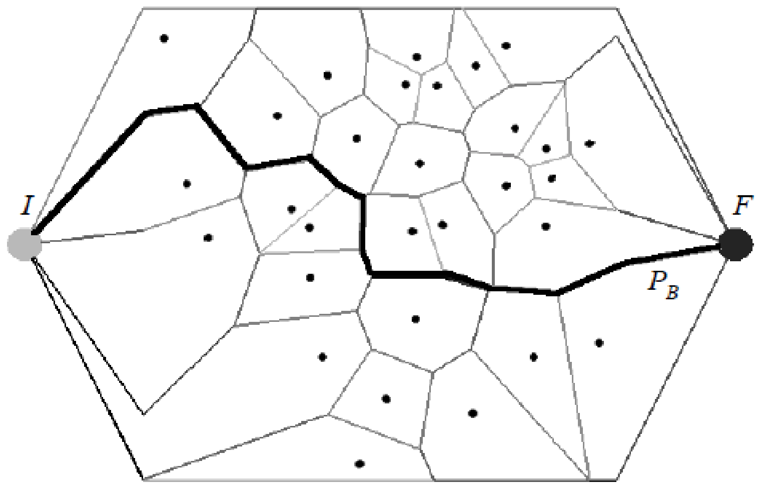


Figure 2.2-3 Sensor Field with Weighted Voronoi Diagram and Maximal Breach Path

Similarly, the *Support Weight* and the mid-point of the corresponding edge in the Delaunay triangulation can be used as a heuristic for deploying additional sensors to improve support coverage. As shown in Figure 2.2-6, on average, a 50% improvement can be achieved in support coverage by adding 1 additional sensor when 5 nodes have already been randomly deployed. After deploying 100 random sensors, on average a 10% support coverage improvement can be expected by using the heuristic to deploy one more sensor.

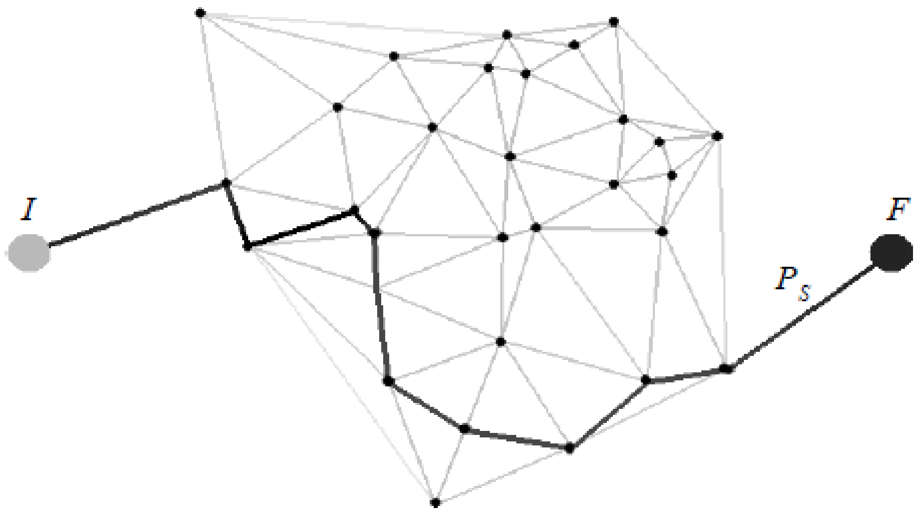


Figure 2.2-4 Sensory Field with Weighted Delaunay Triangulation and Maximal Support Path (P_S)

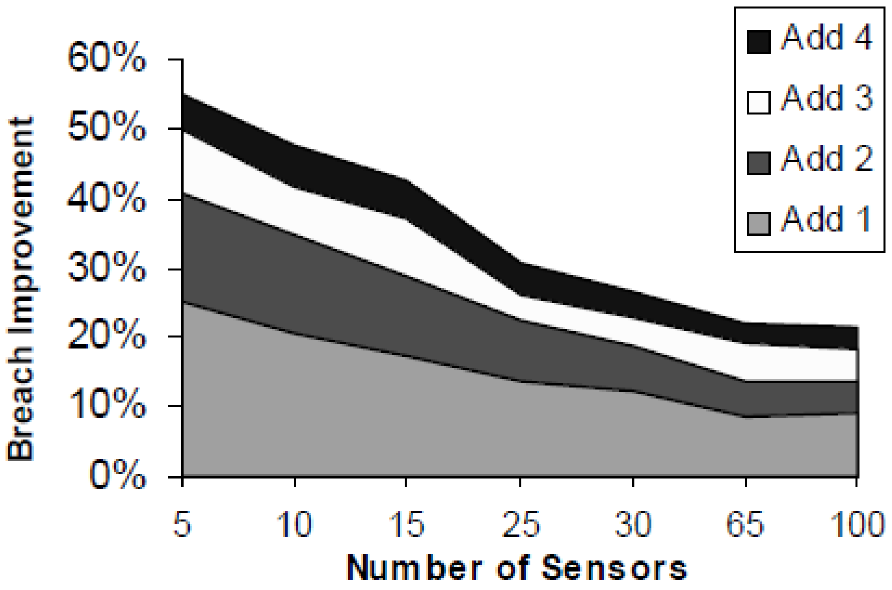


Figure 2.2-5 Average Breach Coverage Improvement by Additional Sensor Deployment

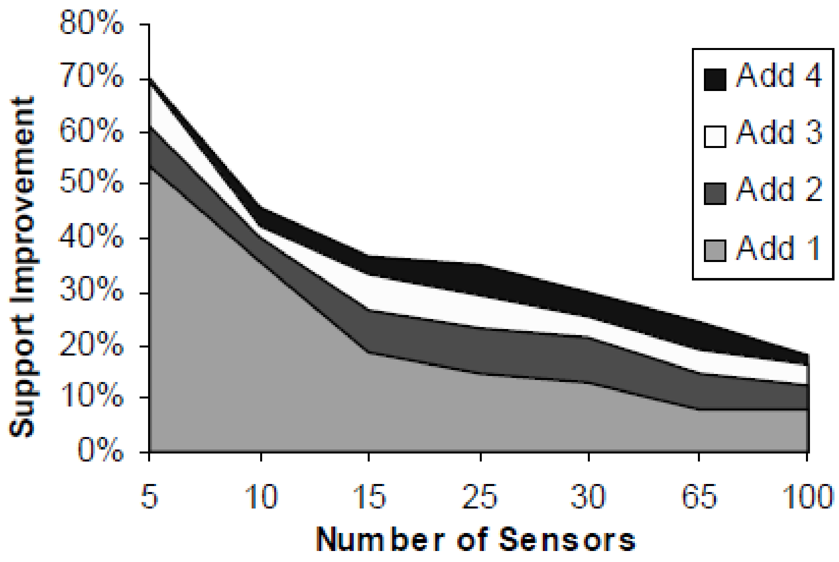


Figure 2.2-6 Average Support Coverage Improvement by Additional Sensor Deployment

2.3 Location Awareness of Wireless Sensor Network

Studies of coverage problem arises the demand for the location aware systems and services. Intuitively, GPS (Global Positioning System) is the reasonable answer. However, GPS is an unattractive solution for a wireless sensor network due to cost, power, and accuracy constraints. There have been some geo-locating algorithms without having to use GPS proposed, such as [31]. In [31] a few of the sensor nodes called beacons know their coordinates in advance, either from satellite information (GPS) or pre-deployment. The geo-location scheme then relies on signal strength information embedded in the inherent radio frequency communication capabilities of the nodes in approximating neighbor distances. Each node that can hear from a minimum of three beacon neighbors can determine its own location by trilateration and become a beacon. Iterative trilateration are then used to locate as many nodes as possible. [31] also includes heuristics to compensate for the errors in the initial beacon locations and distance information. Simulations showed that in a reasonably dense network, by having 1% or less of the nodes as initial beacons, almost all other nodes can locate themselves at the end of the location process. That is even if using GPS for the beacons' initial location, only 1% nodes need to equip GPS. The cost is reduced significantly. (Note: during the coverage discussion the nodes without location information will be ignored.)

Sometimes, a relative position between sensors is required (i.e., one node is located at which direction of the other node). The incoming signals have already been discussed in the IEEE antennas and propagation community as the Angle-Of Arrival (AOA) problem. This can be accomplished by using more than one directional antenna, as mentioned in [32].

2.4 Energy Consumption Analysis for Wireless Sensor Network

Since we discuss the energy conserving measurements, analysis for the energy consumption factors in the WSN is essential.

A sensor node's radio can be in one of the following four states: transmit, receive, idle, or sleep. The idle state is when the transceiver is neither transmitting nor receiving, and the sleep mode is when the radio is turned off. As presented in [33] an analysis of the power usage for WINS Rockwell seismic sensor indicates power consumption for the transmit state between 0.38W and 0.7W, for the receive state 0.36W, for the idle state 0.34W and for the sleep state 0.03W. The receive and idle modes may require as much energy as transmitting, while the sleep mode requires the less energy. Another observation is the communication against computation power usage ratio, which can be higher than 1000 (e.g. for Rockwell WINS [33] is from 1500 to 2700), therefore local data processing, data fusion and data compression are highly desirable.

For applications such as coverage calculations, the energy of computations per node is also a component of the energy metric. It is important to note that technology scaling will gradually reduce the processing costs, with the transmission cost remaining constant. Using compression techniques, one can reduce the number of transmitted bits, thus reducing the cost of transmission at the expense of more computation. This communication-computation trade-off is the core idea behind low energy sensor networks. From this discussion it is apparent that a good algorithm designed for wireless sensor networks will require minimal amount of communication. This is in sharp contrast with classical distributed systems [34] where the goal generally is maximizing the speed of execution. This renders the classical distributed algorithm irrelevant for developing wireless sensor networks algorithms.

Generally speaking, there are several sources of power consumption in sensor networks, and correspondent methods of reducing power consumption [35]:

- ***Idle listening*** is the major power consumption source for many networks. For most transceivers, the receive mode power consumption is on the same order of magnitude as the transmission power [36, 37, 38], and most MAC protocols put the transceiver in receive mode whenever it does not transmit, whether there is the need to receive a message or not.

- ***Retransmissions resulting from collisions*** can be quite significant if the network load is high and the collisions frequent.
- ***Control packet overhead*** (e.g., RTS, CTS. ACK) can be significant for sensor networks which, typically, have small packets.
- ***Unnecessary high transmitting power*** does not only results in higher power consumption, but may also increase the. Interference at other nodes in the network.
- ***Sub-optimal utilization of the available resources***; for example, routes that utilize the nodes with the largest (remaining) batteries should be preferred.

2.5 Power-saving Strategies for Wireless Sensor network

Based on the analysis in the previous section, we will discuss some energy-saving strategies for wireless sensor network in this part.

Minimizing energy consumption and maximizing the system lifetime has been a major design goal for wireless sensor networks. In the last few years, researchers are actively exploring advanced power conservation approaches for wireless sensor networks. On the one hand, device manufacturers have been striving for low power consumption in their products. In [39,40], low power transceiver architectures and low power signal processing systems are discussed separately. In [“a coverage-preserving node scheduling scheme for large WSN” 8], an energy-scavenging technique, which enables self-powered nodes using energy extracted from the environment, is presented. In [“a coverage-preserving node scheduling scheme for large WSN” 9], a low power data converter, signal processing, RF communication circuits are integrated into one chip. On the other hand, protocol designers are seeking an energy efficient communication architecture, which involves all levels from the physical layer to the application layer [“a coverage-preserving node scheduling scheme for large WSN” 4]. For instance, Directed Diffusion [41] and LEACH [42] are two typical data communication protocols proposed for wireless sensor networks. In directed diffusion, routes (called gradients) that link sources of interesting data to sinks are formed when interest is disseminated throughout the network. When the source has data of interest, it sends the data along the gradient paths back to the sinks. Energy is saved by reinforcement-based adaptation to the empirically best path, caching and in-network data aggregation. LEACH is a clustering-based protocol that utilizes a randomized rotation of a local cluster-head to evenly distribute the energy load among sensors in the network. It also uses localized coordination to enable scalability and robustness for dynamic networks and incorporates data fusion into the routing protocols to achieve energy conservation. [43] and [44] both propose energy efficient centralized mechanisms by dividing the sensor nodes into disjoint sets, such that every set can individually perform the coverage tasks. These sets are then activated successively, and while the current sensor set is active, all other nodes are in the sleep mode. The goal of this approach is to determine a maximum number of disjoint sets, as this has a direct impact on conserving sensor energy resources as well as on prolonging the network lifetime.

Power saving techniques can generally be classified in the following categories [45]:

1) power control by adjusting the transmission range of wireless nodes
2) energy efficient routing, data gathering
3) schedule the wireless node to states between active and sleep mode
4) reduce the amount of data transmitted and avoid useless activity

In this thesis, we concentrate on the study of the third method (Note that the above methods can be used on a wireless sensor network crossingly). We try to design an optimal sensor scheduling mechanism. There is no better way to conserve energy than to put the nodes to sleep (since using low power components only goes so far). However, a node that is sleeping is no longer part of the network, and thus cannot keep on monitoring the field and delivering the sensor data. Therefore, coverage and connectivity are both factors here. In next chapter, we will discuss recent studies on sensor scheduling schemes.

Chapter 3

Reviews of Sensor Scheduling Schemes for WSN

The idea of Sensor Scheduling Scheme is to allow maximum redundant sensors to go to sleep mode, meanwhile try to maintain the same level of coverage and connectivity. With the energy being saved in this way, the life time of the wireless sensor network is also extended.

There have been many proposals on sensor scheduling schemes for WSN. In this chapter, we will review some of these proposals and look closely at one of the schemes which we later will use for performance comparison with our own proposal.

3.1 Introduction to Sensor Scheduling Scheme

Scheduling Scheme makes it possible to put a portion of the sensors in the sensor network to sleep when they are not needed. There is no better way than putting the nodes to sleep (since using low power components only goes so far), in terms of energy-saving. However, a node that is sleeping is no longer part of the network, and thus can no longer monitor the field and deliver the sensor data. This poses a major challenge to the scheduling schemes that to have fewer sensors working without having to degrade the system performance and reliability.

In general, there are two classes of scheduling strategies in the field. One class is based on certain exempt rule, with which the system uses to decide whether a sensor is qualified to go to sleep model at a certain time. With this type of scheduling method, the scheduling scheme needs to run periodically, because the exempt qualification of sensors is dependent on time. The other class depends on some grouping policies which will assign sensors into different sets. After the sets are decided, only one set of the sensors work at a certain period of time. This kind of schemes usually runs once at the beginning of a WSN setup and won't be executed again until some event (such as 30 percent of the sensors' out of power, or some message delivery failed) drives it. The exempt rules and the grouping policies are analogues in the two classes of strategy. They usually consist of criteria such as coverage, or connectivity (criteria can mix and match with each other). The exempt rules and grouping policies could be quite different from scheme to scheme. They depend on the scheme's perspective. We listed some of the scheduling schemes and some brief review below to get familiar with the concepts.

In [46], a probing-based density control algorithm is proposed to ensure long-lived, robust sensing coverage by leveraging unconstrained network scale. In this protocol, only a subset of nodes are maintained in working mode to ensure desired sensing coverage, and other redundant nodes are allowed to fall asleep most of the time. Working nodes continue working until they run out of their energy or are destroyed. A sleeping node wakes up occasionally to probe its local neighborhood and starts working only if there is no working node within its probing range. Geometry knowledge is used to derive the relationship between probing range and redundancy. In this algorithm, desired redundancy can be obtained by choosing the corresponding probing range. However, this derivation is based on the assumption that all the nodes have exactly the same sensing range. It is hard to find a relationship between probing range and desired redundancy, if nodes have different sensing ranges. Furthermore, the probing-based off-duty eligibility rule cannot ensure the original sensing coverage and blind points may appear after turning off some nodes, which is verified in our experiment.

In [47] Chen et al. proposed an algorithm to turn off nodes based on the necessity for neighbor connectivity. They intend to reduce the system energy consumption without significantly diminishing the connectivity of the network. In [48], Xu et al. proposed a scheme in which energy is conserved by letting nodes turn off their communication unit when they are not involved into sending, forwarding or receiving data phase. Also, node density is leveraged to increase the time that communication unit is powered off. In [49], an algorithm, called Geographical Adaptive Fidelity (GAF) was proposed, which uses geographic location information to divide the area into fixed square grids. Within each grid, it keeps only one node staying awake to forward packets. These three node-scheduling schemes turn off nodes from communication perspective without considering the system's sensing coverage. In fact, in wireless sensor networks, the main role of each node is sensing. Unusual event could happen at any time at any place. Therefore, if we only turn off nodes, which are not participating in data forwarding, certain areas in the deploying area may become “blind points”. Important events may not be detected [46].

In next section we will look at three scheduling schemes in details to demonstrate construction of such schemes. Especially the third one, we will exploit a great deal of details out of it, as we will compare it with our proposals from many perspectives.

3.2 Energy -Efficient Target Coverage Scheduling Scheme for WSN

Mihaela Cardei, My T. Thai, Yingshu Li, and Weili Wu [45] proposed an efficient method to extend the sensor network life time by organizing the sensors into a maximal number of sets, such that each set of sensors is able to cover all the targets in the monitored filed individually. There is a premise for this approach that all the monitored targets are known and stationary. The location of each sensor is also required by the approach. If sensors are allowed to go to sleep mode while same-level of coverage is maintained, the life time of the WSN is surely elongated.

This scheme is based on Target Coverage, so first we will look at the definition of the Target Coverage Problem. [45] defined it as such, Target Coverage Problem (TCP): Given m targets with known location and an energy constrained wireless sensor network with n sensors randomly deployed in the targets' vicinity, schedule the sensor nodes activity such that all the targets are continuously observed and network lifetime is maximized.

This scheme is based on Target Coverage, so first we will look at the definition of the Target Coverage Problem. [45] defined it as such, Target Coverage Problem (TCP): Given m targets with known location and an energy-constrained wireless sensor network with n sensors randomly deployed in the targets' vicinity, schedule the sensor nodes activity such that all the targets are continuously observed and network lifetime is maximized.

The sensor scheduling mechanism can be accomplished as follows:

1) Sensors send their location information to the Base Station (BS).
2) BS executes the sensor scheduling algorithm and broadcast the schedule when each node is active.
3) Every sensor schedule itself for active/sleep intervals.

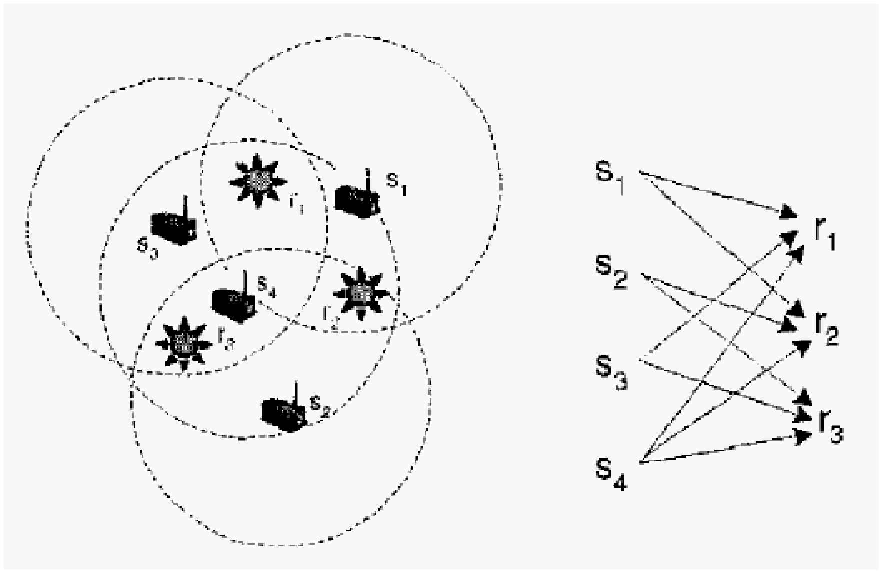
The goal of the scheme is to find the maximum number of sets for the randomly distributed n sensors. [45] defined this problem as the maximum set covers problem (MSC).

3.2.1 MSC Problem Definition

Let us assume that n sensors $s_1, s_2, \dots\dots, s_n$ are randomly deployed to cover m targets $r_1, r_2, \dots\dots, r_m$. The base station (BS) has the coordinates of the sensor nodes and the targets; therefore it is able to compute for each sensor node which targets it covers. One method is to assume that a sensor covers a target if the Euclidean distance between sensor and target is smaller or equal with a predefined sensing range.

Figure 3.2-1 (a) shows an example with four sensor nodes s_1, s_2, s_3, s_4 and three targets r_1, r_2, r_3. In this example, assume a node sensing area being the disk centered *at* the sensor, with radius equal to the sensing range. The coverage relationship between sensors and targets is also illustrated in the Figure 3.2-I (b): $s_1 = \{r_1, r_2\}$, $s_2 = \{r_2, r_3\}$, $s_3 = \{r_1, r_3\}$ and $s_4 = \{r_1, r_2, r_3\}$.Note that a circular sensing area is not a requirement, it only concerns identifying which sensors cover each target.

Assume that all sensor nodes have the same remaining energy. In order to model the network lifetime, also assume that each sensor can be active for a unit time of 1. That is, if all sensors are active continuously, the network lifetime is 1.



(a) (b)

Figure 3.2-1 example with three targets $R = \{r_1, r_2, r_3\}$ and four sensors $S = \{s_1, s_2, s_3, s_4\}$

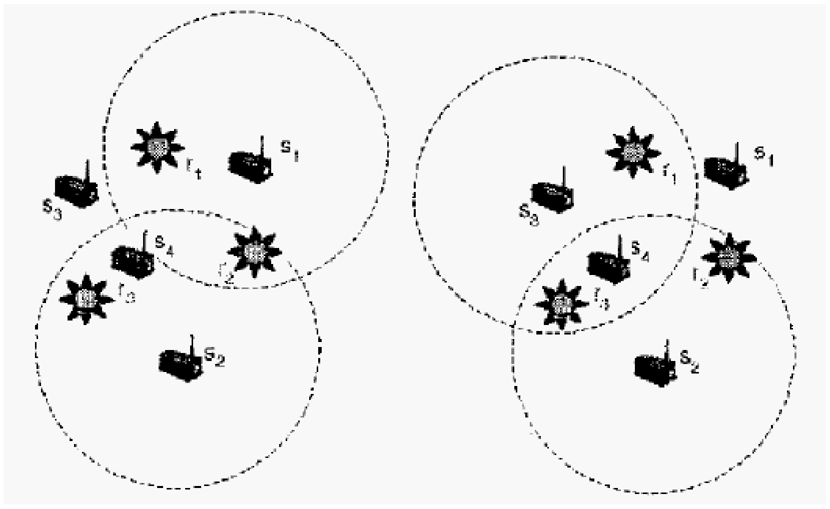
The work in [50], divides the sensors in disjoint sets, e.g. $S_1 = \{s_1, s_2\}$ and $S_2 = \{s_3, s_4\}$. This will result in a network lifetime of 2.

In [46], they improve the scheduling scheme by allowing every sensor to be part of more than one set, and by allowing the sets to be operational for different time intervals. As illustrated in Figure 3.2-2, the sets in this case are: $S_1 = \{s_1, s_2\}$

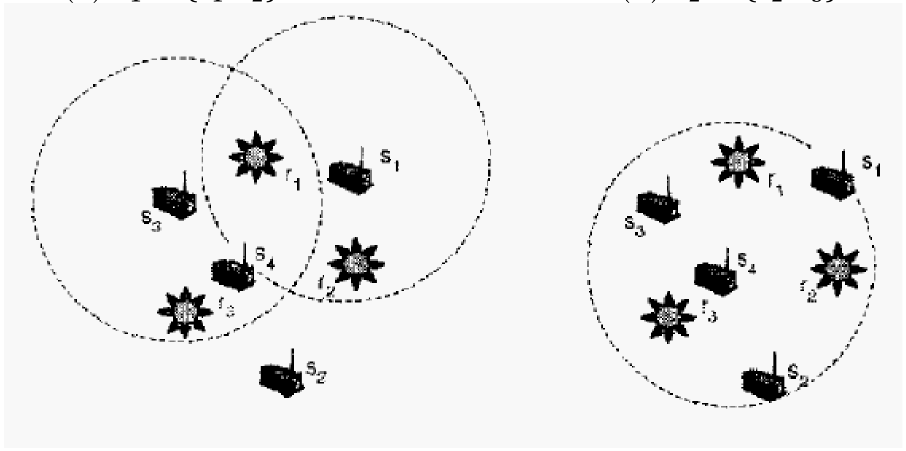
for *0.5* time, $S_2 = \{s_2, s_3\}$ for *0.5* time. $S_3 = \{s_1, s_3\}$ for *0.5* time and $S_4 = \{s_4\}$ for *1* time. This organization results in a network lifetime of *2.5*. This corresponds to 25% increase in network lifetime compared with the disjoint sets solution.

The problem of computing the set covers such that to maximize the network lifetime is formally defined next.

Maximum Set Covers (MSC) Problem: Given a collection C of subsets of a finite set R, find a family of set covers $S_1, \ldots, S_p$ with time weights $t_1, \ldots, t_p$ $(0 \leq t_i \leq 11)$ such that to maximize $t_1 + \ldots + t_p$, and for each subset s in C, s appears in $S_1, \ldots, S_p$ with a total weight of at most 1, where 1 is the life time of each sensor.



(a) $S_1 = \{s_1, s_2\}$ (b) $S_2 = \{s_2, s_3\}$



(c) $S_3 = \{s_1, s_3\}$ (d) $S_4 = \{s_4\}$

Figure 3.2-2 four cover sets: $S_1 = \{s_1, s_2\}$ for *0.5* time, $S_2 = \{s_2, s_3\}$ for *0.5* time. $S_3 = \{s_1, s_3\}$ for *0.5* time and $S_4 = \{s_4\}$ for *1* time

In MSC definition, C is the set of sensors and R is the set of targets, such that each sensor covers (monitor) a subset of targets. We want to determine a number of set covers $S_1, \ldots, S_p$, where each set cover S_i, $i = 1, \ldots, p$ completely covers all the targets, such that to maximize the network lifetime $t_1 + \ldots + t_p$, where t_j , $j = 1, \ldots, p$ is the time interval while the se1 cover S_j is active. Note that if a sensor belongs to more than one cover, then the sum of the time intervals of those covers cannot be greater than 1. This is because each sensor cannot be active more than 1.

3.2.2 Solutions to Compute the Maximum Set Covers and Runtime Complexity Analysis

[46] proved that MSC problem belongs to the class NP and is NP-hard, so we established that MSC is NP-complete. They also provided two heuristics for the MSC problem.

They first model the MSC problem as an Integer Programming in section, and then use the relaxation technique to design a Linear Programming based heuristic. Then, they propose a greedy heuristic, where set covers are formed individually, by covering first the most critical targets.

The first heuristic is called LP-MSC Heuristic. Give the following variables: n is the number of sensors, m is number of targets, and p is the upper-bound for the number of set covers. The total runtime complexity for the heuristic is $O(p^3n^3)$. The other heuristic is named Greedy-MSC Heuristic. The complexity of the Greedy-MSC Heuristic is $O(im^2n)$, where i is the number of set covers. The variable i is upper-bounded by d/w, where d is the number of sensors that covers the most sparsely covered target. Usually w is a constant and $d < n$. Thus the heuristic runtime is $O(dm^2n)$.

3.2.3 Simulation Result for the Heuristics

The simulation is done on a stationary network with sensor nodes and target points randomly located in a 500m x 500m area. Assume the sensing range is equal for all the sensors in the network. The following are the tunable parameters:

- *n,* the number of sensor nodes. It varies between 25 and 750 to study the effect of node density on the performance.
- *m,* the number of targets to be covered. It varies between 5 and 15.
- *r,* the sensing range. It varies between 100m to 300m.

The linear programming is solved by optimization toolbox in Matlab. The simulation results are illustrated in the following diagrams (Figure 3.2-3 – Figure 3.2-5 and table 3.2-1).

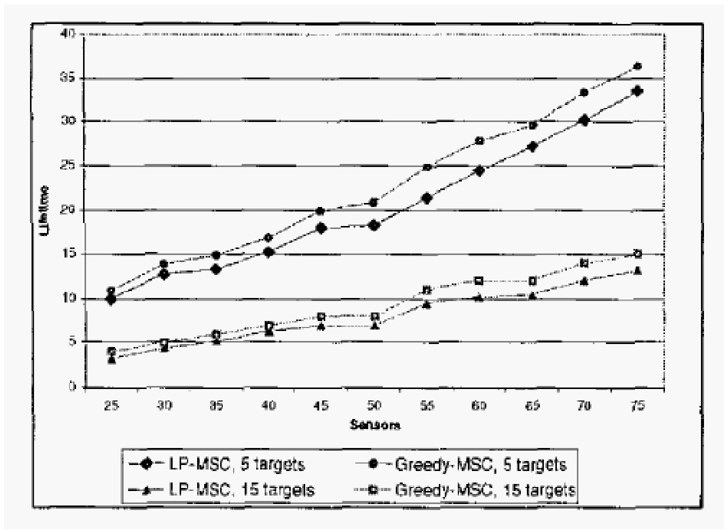


Figure 3.2-3 Network lifetime with number of sensors when range $r = 250m$

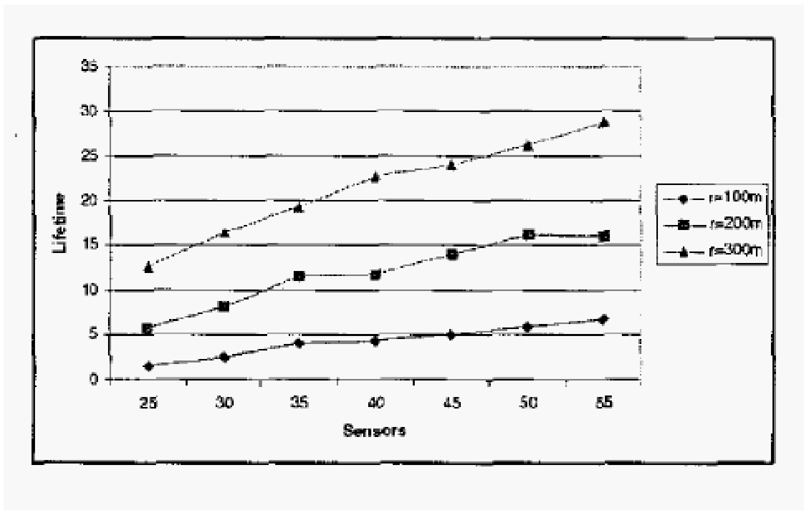
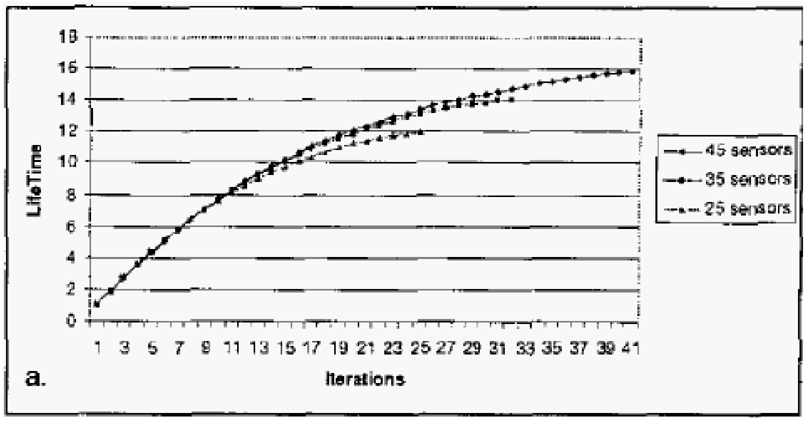


Figure 3.2-4 LP-MSC heuristic, network lifetime with number of sensors for 10 targets

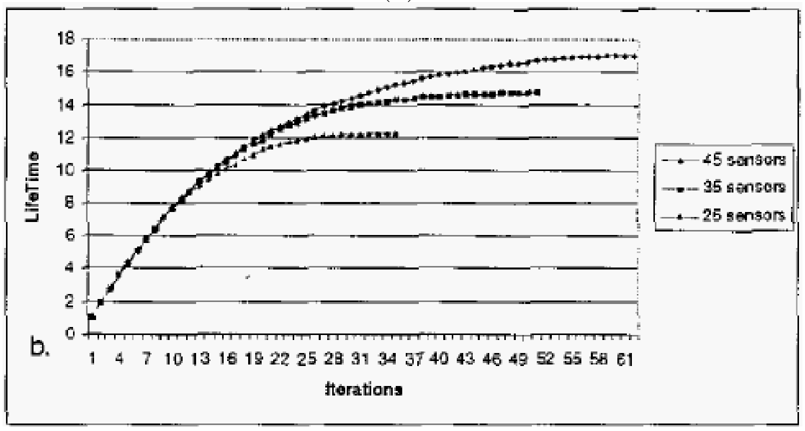
The simulation results can be summarized as follows:

- For *a* specific number of targets, the network lifetime output by our heuristics increases with the number of sensors and the sensing range.
- For a specific number of sensors and sensing range, the network lifetime increases as the number of targets to be monitored decreases.
- For smaller tolerance values. The lifetime value increases over time as result of additional execution of steps 1 and 2 of the LP-MSC heuristic. There is a trade-off between the higher lifetime value and the increase in the runtime, triggered by additional LP-solver calls.

- Greedy-MSC has a lower running time, thus it is more scalable to large sensor networks.

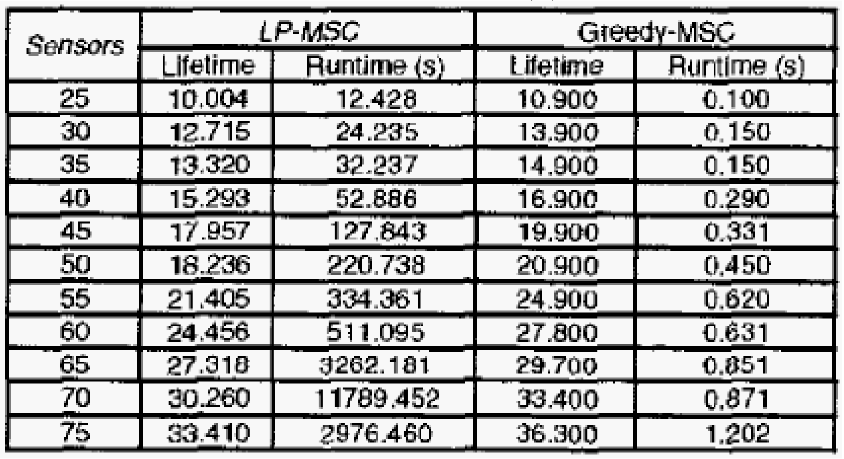


(a)



(b)

Figure 3.2-5 LP-MSC heuristic, lifetime and number of iterations for tolerance 0.1 (a) and tolerance 0.01 (b)



Sensors	*LP-MSC*		Greedy-MSC	
	Lifetime	Runtime (s)	Lifetime	Runtime (s)
25	10.004	12.428	10.900	0.100
30	12.715	24.235	13.900	0.150
35	13.320	32.237	14.900	0.150
40	15.293	52.886	16.900	0.290
45	17.957	127.843	19.900	0.331
50	18.236	220.738	20.900	0.450
55	21.405	334.361	24.900	0.620
60	24.456	511.095	27.800	0.631
65	27.318	3262.181	29.700	0.851
70	30.260	11789.452	33.400	0.871
75	33.410	2976.460	36.300	1.202

Table 3.2-1 Runtime of LP-MSC and Greedy-MSC heuristics

3.2.4 Limitation of the Target Coverage Scheme

There are a few limitations for the above scheme that need to be point out. First one is that this scheme assumes the targets in the monitored field are stationary and their location is known to the system. Actually, not many cases meet this requirement. A wireless sensor network is usually expected to watch the entire monitored field and targets with mobility. In those cases, this scheduling scheme is not applicable. Second, this scheme also requires the geo-location of each sensor in the WSN. This requirement, as we discussed in chapter 2, is also not so easy to fulfill. GPS is convenient but expensive and has accuracy issue. Locating algorithm is not so easy to implement and also not so accurate. Last but not least, the scheme only considered the coverage of the WSN but not connectivity. That is the sensor data from each sensor might not be able to send back to the base station for processing. Then the collected data is meaningless in this case. Our proposed approach has measurements that response to all the problems above.

3.3 A Coverage-preserving Node Scheduling Scheme for Large WSN

Di Tian and Nicolas D. Georganas [51] proposed a node-scheduling scheme, which can reduce system overall energy consumption by turning off some redundant nodes, therefore increasing system lifetime. The coverage-based off-duty eligibility rule and backoff-based node-scheduling scheme guarantees that the original sensing coverage is maintained after turning off redundant nodes. The simulation of the scheme is done in NS-2 as an extension of the LEACH protocol. The comparison of the power consumption between the cases of with and without the scheme showed that the scheme can preserve the system coverage to the maximum extent. In addition, after the node-scheduling scheme turns off some nodes, certain redundancy is still guaranteed, which can provide enough sensing reliability in many applications.

Generally, the node-scheduling problem consists of two sub problems. First, what is the rule that each node should follow to determine whether it should turn itself off or not? Second, when should nodes make such decision? In this section, these two questions are answered respectively.

3.3.1 Coverage-based Off-duty Eligibility Rule

As discussed above, the main objective of this algorithm is to minimize the number of working nodes, as well as maintain the original sensing coverage. To achieve this goal, need to calculate each node's sensing area and then compare it with its neighbors'. If the whole sensing area of a node is fully covered by the union set of its neighbors', this node can be turned off without reducing the system overall sensing coverage.

Some notes need to take before the introduction to the Off-duty Eligibility Rule. First of all, to simplify the problem, we assume that all nodes have the same sensing range and each node knows its sensing range *r*. Later the situation of sensors with different range will be briefly discussed. Second, a node's sensing area is a circle centered at this node with radius r, if all nodes lie on a 2-dimensional plane. (The scheme we will describe is also applicable to a 3-dimensional space.) A node i's sensing area is denoted as $S(i)$. Neighbors are nodes whose distance from the current node is equal to or less than the sensing range r as shown in the following definition.

The neighbor set of node i is defined as

$$N(i) = \{\, n \in \aleph \mid d(i,j) \le r, n \neq i \,\}$$

where $\aleph$ is the node set in the sensing field, $d(i,j)$ denote the distance between node i and node j. Thus, for node i, the off-duty eligibility rule can be expressed as $\bigcup_{j \in N(i)} S(j) \supseteq S(i)$. The expression is equivalent to $\bigcup_{j \in N(i)} (S(j) \cap S(i)) \supseteq S(i)$. By observation, we know that the crescent-shaped intersection $S(j) \cap S(i)$ in Figure 3.3-1(a) includes sector as illustrated in 3.3-1(b). Although the area of the sector is smaller than that of the crescent, it is much easier to calculate the area of the sector rather than that of the crescent, because the area of a sector can be represented by its central angle accurately and uniting two sectors is equivalent to merging two central angles. Therefore, although node j can cover a crescent-shaped region within node i's sensing area (Figure 3.3-1(a) shadow region), node i will only "admit" that node j can help it monitor a sector-shaped region (Figure 3.3-1(b) shadow region) if node i is turned off. To help the further description, we define this sector as a sponsored sector. As shown in Figure 3.3-1(b), sponsored sector by node j to node i is denoted as $S_{j \to i}$. It's not hard to prove the following is true:

If $\bigcup_{j \in N(i)} S_{j \to i} \supseteq S(i)$, then $\bigcup_{j \in N(i)} (S(j) \cap S(i)) \supseteq S(i)$.

$\bigcup_{j \in N(i)} S_{j \to i} \supseteq S(i)$ ensures that investigating whether the neighbors can cover the current node's sensing area is equivalent to checking whether the union of sponsored sectors (called sponsored coverage) contains the current node's sensing area, which in turn, is equivalent to calculating whether the union of central angles can cover the whole 360 degree as illustrated in Figure 3.3-1(e).

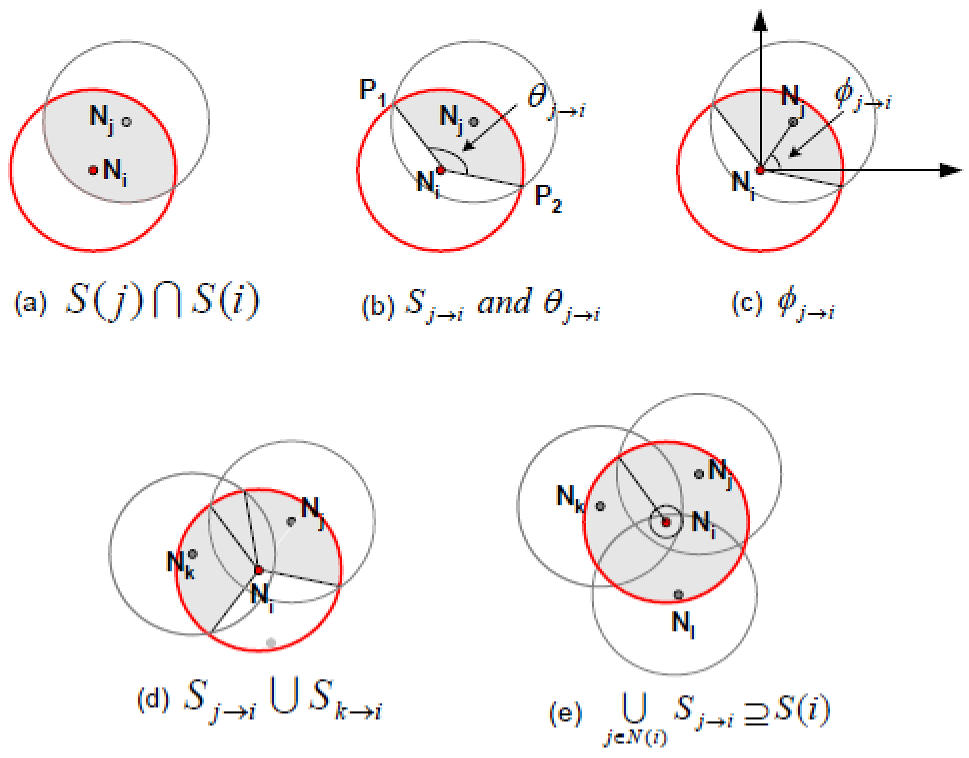

N_j
N_i
(a) $S(j) \bigcap S(i)$
P_1
P_2
$\theta_{j\rightarrow i}$
(b) $S_{j\rightarrow i}$ and $\theta_{j\rightarrow i}$
$\phi_{j\rightarrow i}$
(c) $\phi_{j\rightarrow i}$
N_k
(d) $S_{j\rightarrow i} \bigcup S_{k\rightarrow i}$
N_l
(e) $\bigcup_{j\in N(i)} S_{j\rightarrow i} \supseteq S(i)$

Figure 3.3-1 Sponsored Coverage Calculation-Basic Model

If the condition $\bigcup_{j\in N(i)} S_{j\rightarrow i} \supseteq S(i)$ is satisfied, we call the neighboring nodes are off-duty sponsors of node i.

From the geometry calculation, the central angle is given by $\theta_{j\rightarrow i} = 2arccos\left(\frac{d(i,j)}{2r}\right)$. Since $0 < d(I,j) \leq r$, it is easy to know that the range of the central angle $\theta_{j\rightarrow i}$ is on $[120, 180)$. So a node need at least 3 neighbors to be eligible for off-duty.

3.3.2 Sponsored Coverage Calculation for extension model

In the above discussion, each node's geographical location is required. As discussed in chapter GPS is not a favorable solution due to the cost. [51] also provided an extension to the original scheme with the directional information from the incoming signal.

According to Figure 3.3-1 and the off-duty Eligibility Rule, a node needs to know both $\theta_{j\rightarrow i}$ and $\emptyset_{j\rightarrow i}$ in order to decide whether it has off-duty sponsors. Since it has been established that $\theta_{j\rightarrow i}$ is on $[120, 180)$, we can always use 120 as the safe value for $\theta_{j\rightarrow i}$. Now the only question is how to find out the value of $\emptyset_{j\rightarrow i}$. Techniques to estimate direction from incoming signals have already been discussed in the IEEE antennas and propagation community as the Angle-Of

Arrival (AOA) problem. According to [46], this can be accomplished by using more than one directional antenna. In this case, the location information of sensors is no longer required. However, availability of directional information from directional antennas is not currently practical in wireless sensor networks.

The scheme also tried to cope with sensors with different sensing ranges. Unfortunate, according to the discussion, the scheme can only address some of the situations (i.e., the scheme cannot deal with the case as shown in Figure 3.3-2(b)). So we consider the scheme is not applicable to WSN consists of sensors with different sensing range in practice. And do not intend to do further discussion.

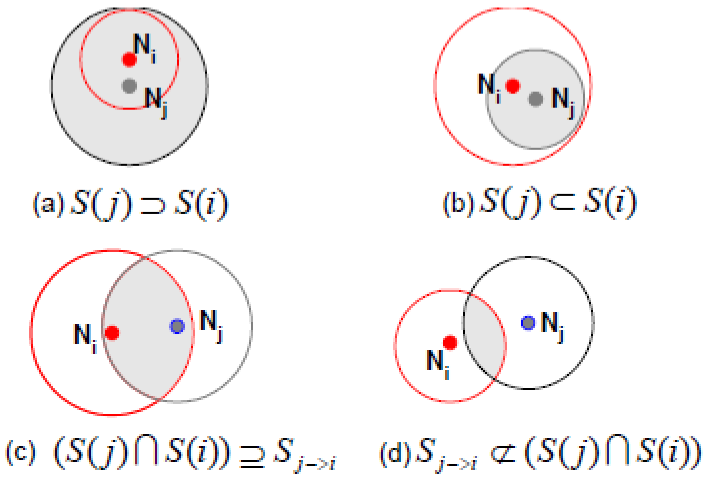

N_i
N_j
(a) $S(j) \supset S(i)$
(b) $S(j) \subset S(i)$
(c) $(S(j) \cap S(i)) \supseteq S_{j->i}$
(d) $S_{j->i} \not\subset (S(j) \cap S(i))$

Figure 3.3-2 layouts of neighboring nodes with different sensing range

3.3.3 Node Scheduling Scheme Based on Eligibility Rule

In the scheduling scheme, the operation is divided into rounds. Each round begins with a self-scheduling phase, followed by a sensing phase. In the self-scheduling phase, nodes investigate the off-duty eligibility rule described in the previous section. Eligible nodes turn off their communication unit and sensing unit to save energy. Non-eligible nodes perform sensing tasks during the sensing phase. To minimize the energy consumed in the self-scheduling phase, the sensing phase should be long compared to the self-scheduling phase.

The self-scheduling phase consists of two steps. First, each node advertises its position and listens to advertisement messages from other nodes to obtain neighboring nodes' position information. Second, each node calculates a neighbor sponsoring sensing area compares it with its own and decides whether it is eligible for off-duty or not. The details of these two steps are introduced as follows.

Step one is to obtain neighbor node information, a simple approach is that each node broadcasts a Position Advertisement Message (PAM), which contains node ID and its current location, at the beginning of each round. Because only neighbors within a node's sensing range are considered in the eligibility rule, in order to minimize energy consumption, each node transmits PAM with the minimum power as long as it reaches its sensing range. Such transmission power control scheme ensures that only nodes within the transmitter's sensing range can receive its PAM. If nodes have different sensing ranges, PAM should also include the current sensing range of the transmitter as well.

After finishing the collection of neighbor information, each node evaluates its eligibility for turning off by calculating the sponsored coverage, as described in the previous section. However, if all nodes make decisions simultaneously, blind points may appear, as shown in Figure 3.3-3. Node 1 finds its sensing area can be covered by node 2, 3 and 4. According to the off-duty eligibility rule, node 1 turns itself off. While at the same time, node 4 also find its sensing area can be covered by node 1, 5 and 6. Believing node 1 will keep working, node 4 turns itself off too. Thus, a blind point occurs after turning off both node 1 and node 4, as in Figure 3.3-3(d).

To avoid such a problem, we introduce a back-off scheme. We let each node start its determination after a random back-off time period *Td* and broadcast a Status Advertisement Message (SAM) to announce its status if it is eligible for turning off. Neighboring nodes receiving a SAM will delete the sender's information from their neighbor lists. Thus, the nodes that have a longer backoff delay will not consider the nodes that have decided to be turned off before.

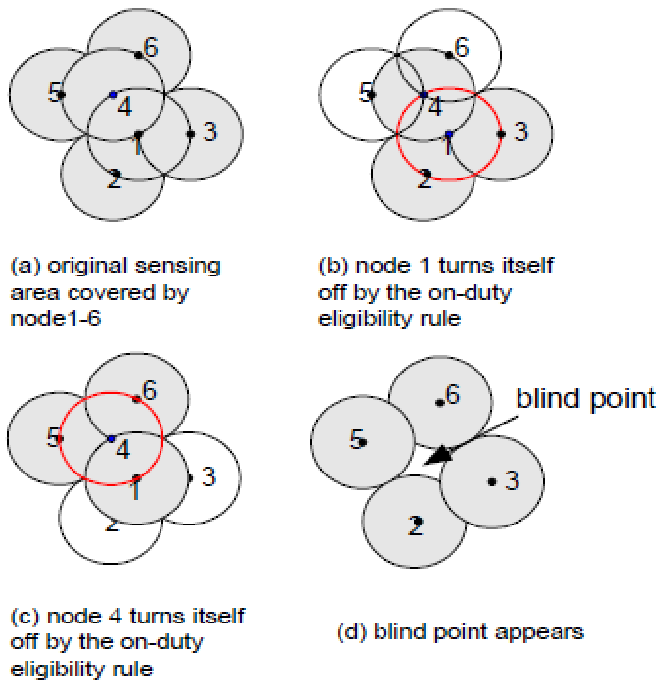

6
5
4
1
3
2
(a) original sensing area covered by node1-6
(b) node 1 turns itself off by the on-duty eligibility rule
(c) node 4 turns itself off by the on-duty eligibility rule
blind point
(d) blind point appears

Figure 3.3-3 Blind Point Occurrence

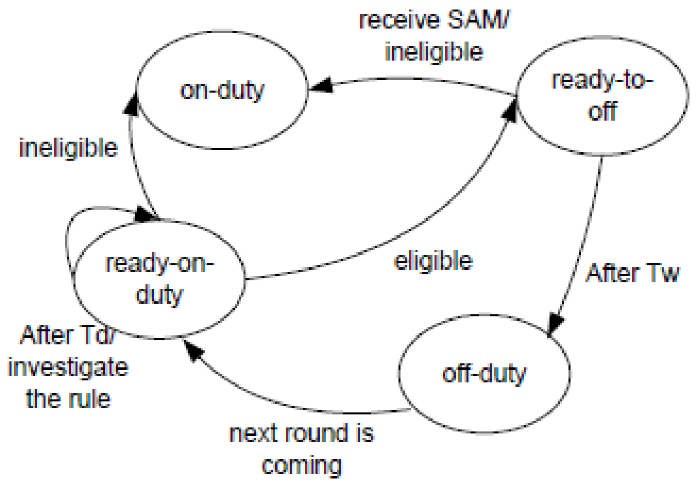


Figure 3.3-4 FSM for Self-scheduling Phase

Assuming W is the size of random back-off time choice, the probability of node 1 and node 4 selecting the same random number is $1/W$. Although a large W can reduce the probability to a sufficient small value, there is still a chance that node 1 and node 4 could select the same random number. To avoid a blind point further, we let each node wait for a short period time T_W after sending the SAM out, if it is eligible for turning off, instead of turning off its communication unit immediately. This ready-to-off period should be enough for node 1 to receive SAM from node 4, or vice verse.

If one SAM is received during the ready-to-off period and the transmitter is one of its off-duty sponsors, the node will reinvestigate its off-duty eligibility. If the eligibility doesn't hold any more, the node returns it status from ready-to-off to on-duty. Otherwise, the node turns itself off after T_W. The nodes, which have decided to serve as on-duty ones, don't re-evaluate their off-duty eligibility once the decision has been made. The status transition graph is shown in Figure 3.3-4.

3.3.4 Simulation results

This section evaluates the ability of the node-scheduling scheme to save energy, therefore, increase system lifetime by comparing the energy consumption per node in the original and extended LEACH.

Figure 3.3-5 illustrates the energy dissipation curve per node in the original LEACH and the extended LEACH in random network topology when Ng =20. The energy dissipation in the extended LEACH is slower than the original one.

Figure 3.3-6 and Figure 3.3-7 show an increase of the system lifetime in the same simulation setting. Here we use two metrics to evaluate the system lifetime: the total number of nodes alive over time and the system sensing coverage over time (the ratio of the area monitored by on-duty nodes to the deployed region). As illustrated in Figure 3.3-6 and Figure 3.3-7, although the extended LEACH does not outperform the original one in term of first node dead time, the number of nodes alive and the system sensing coverage drop more quickly in the original LEACH than in the extended one. In the result, it takes approximately 4378 seconds for the last node to die in the extended LEACH, while 1412 seconds in the original LEACH. And it takes approximately 2055 seconds for the sensing coverage to drop 20% (reach 80%) in the extended LEACH, while 1285 seconds in the original one.

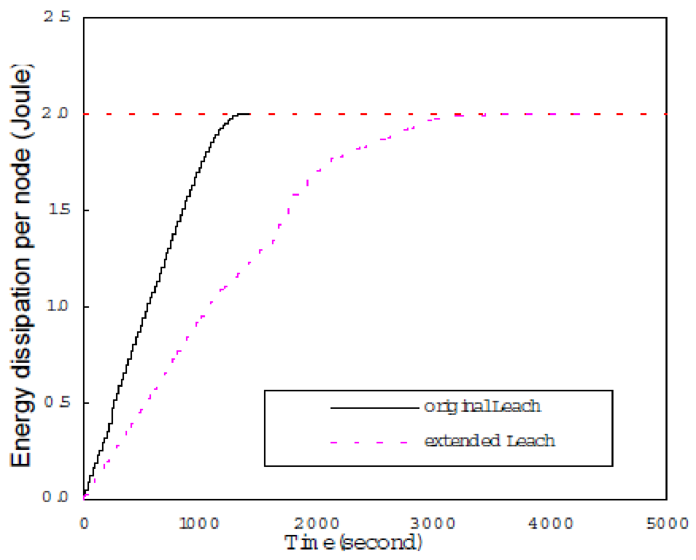


Figure 3.3-5 Energy dissipation curve per node when |ℵ| = 100, R = 50×50, r = 10, $N_g = 20$

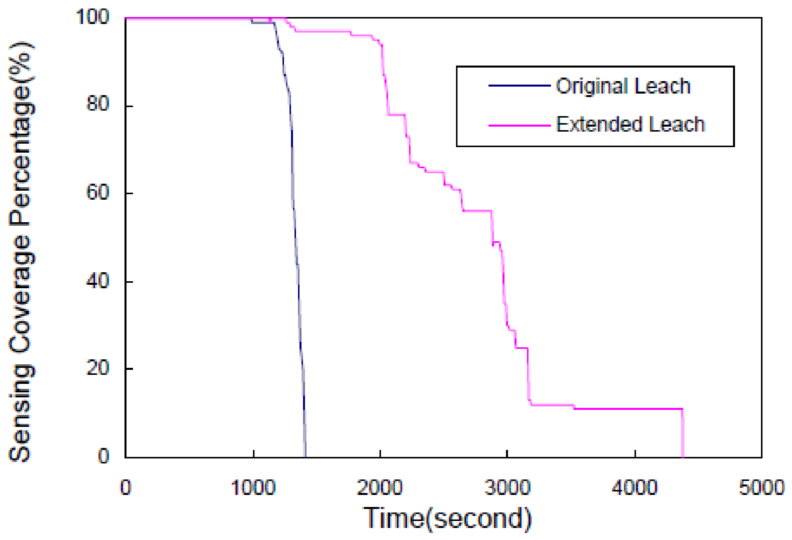


Figure 3.3-6 Sensing Coverage over time when |ℵ| = 100, R = 50×50, r = 10, $N_g = 20$

Furthermore, we also change the number of data gatherings in each round from 4 to 20 with the increment of 4, and compare the time when system coverage drops below 80% in the original and extended LEACH. Figure 3.3-8 showed that the system lifetime with extended LEACH is always longer than, and is about 1.7 times of the original one.

Figure 3.3-9 plots the system lifetime as a function of node density. It can be seen that the system lifetime increases as the node density increases in extended LEACH. In contrast, the system time decreases as the node density increases in original LEACH.

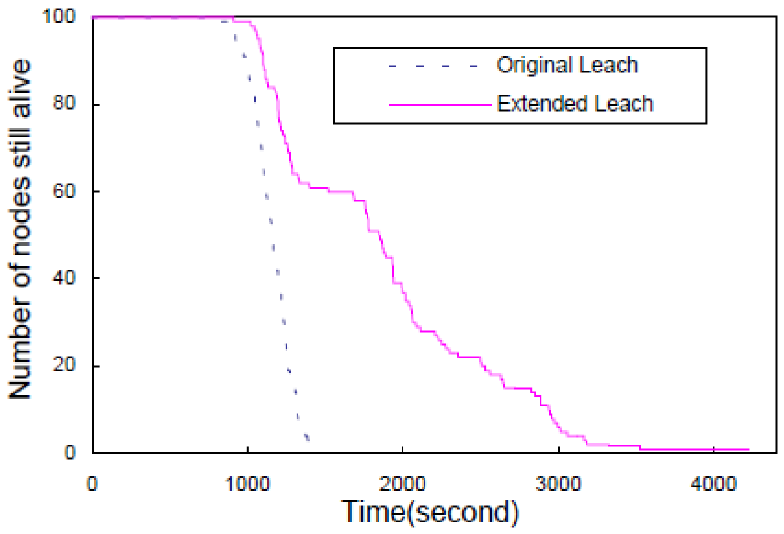


Figure 3.3-7 Number of nodes alive over time when $|\aleph| = 100$, R = 50×50, r = 10, $N_g = 20$

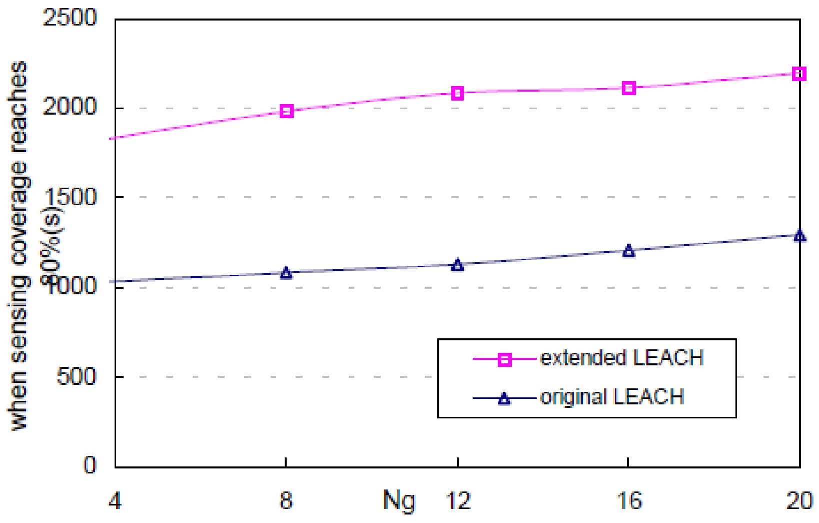


Figure 3.3-8 System lifetime vs. N_g when $|\aleph| = 100$, R = 50×50, r = 10

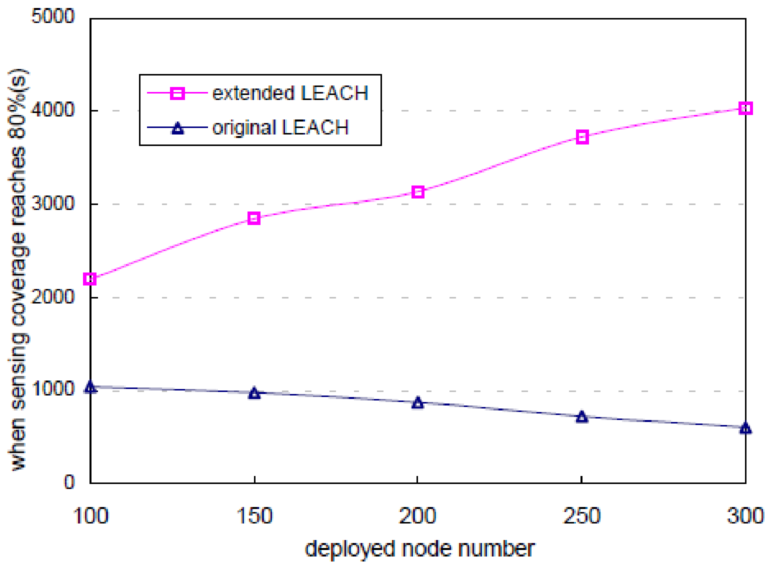


Figure 3.3-9 System lifetime vs. nodes density when $|\aleph| = 100$, R = 50×50, r = 10, $N_g = 20$

3.3.5 Limitations of the Coverage-Preserving Node Scheduling Scheme

As we mentioned before, this scheme requires the location information of the sensor nodes which is not always available for the sensors in a WSN. Even though [51] provided an extended model which try to use the directional information from the incoming signal, the availability and the accuracy of the directional information is still questionable. So this scheme is still considered a sensor-location-aware scheduling scheme. The other concern is that according to the scheduling strategy, after the network has run for a while and some sensors has drained their power, then neither connectivity, nor the coverage is guaranteed. That's because the connectivity is never take into account from the beginning. Even though, a part of the sensor still has much energy left, their sponsored neighboring sensors are out of power. In this case, the sensors left become isolated, and thus their sensor data can reach to the base station. This is a major design problem for the scheme. Coverage is definitely a major perspective for a WSN; however, connectivity issue also cannot be ignored.

3.4 A joint Scheduling Scheme: Random Coverage with Guaranteed Connectivity

Sensor scheduling plays a critical role for energy efficiency of wireless sensor networks. Traditional methods for sensor scheduling use either sensing coverage or network connectivity, but rarely both. In [10], Chong Liu, Kui Wu, Yang Xiao, and Bo Sun addressed a challenging task: Without location information, to schedule sensor nodes for energy saving and guarantee both coverage and network connectivity. Their approach utilizes an integrated method that provides statistical sensing coverage and guaranteed connectivity. They use random scheduling for sensing coverage and then turn on extra sensor nodes to maintain connectivity. The method is distributed and able to dynamically maintain sensing coverage with guaranteed network connectivity. Furthermore, it does not require time synchronization. Some analytical results are presented to reveal the relationship among node density, scheduling parameters, coverage quality, detection probability, and detection delay. Analytical and simulation results demonstrate the effectiveness of the joint scheduling method.

Sensor scheduling plays a critical role for energy efficiency in WSNs. According to the previous discussion, both coverage and connectivity are the most important characters in evaluating the quality of service for a WSN. These two challenge need to be addressed in many applications in WSNs, such as the detection of chemical attacks or the detection of forest fire. We use the following example to stress the importance of the two factors.

Imagine that a wireless sensor network is deployed to detect forest fire. The network should be able to detect the outbreaks of wild fire at any location within the monitored region with a high probability and report the outbreaks to the data collection center (also known as the sink node) with a small delay. In this example, sensing coverage, network connectivity, and energy efficiency are equally important: a large sensing coverage is to meet the users' requirement that an event can be detected with a high probability; network connectivity is to meet the users' requirement that the detected event can be delivered to the sink node; energy efficiency is to meet the users' requirement that the network should keep its operation as long as possible after the deployment. A good scheduling scheme should achieve energy efficiency under the constraints of sensing coverage and network connectivity.

The difficulty in the above joint scheduling problem is that a sensor's sensing range is totally independent of its radio transmission range. It is generally hard to combine and solve the two problems together. This is the reason that although scheduling based on sensing coverage [53], [54], [55], and scheduling based on network connectivity [56] have been studied extensively, very few work has been devoted to solving the joint problem. Since it is usually costly to obtain and maintain the location information of each sensor node, the joint scheduling problem is even more difficult if the location of each sensor node is unknown.

In [10], a goal is set to provide a solution to the joint scheduling problem under the constraints of both sensing coverage and network connectivity without the location information. Specifically, designing a scheduling scheme that has the following features at any given time:

1) The sensing coverage is above a given requirement.
2) All the active sensor nodes are connected.
3) Each active sensor node knows at least one shortest or nearly shortest route to the sink node.

3.4.1 Network Model

There are many ways to organize the communication architecture of a sensor network. A sensor network could be in a hierarchical structure where each sensor communicates with a local cluster head and the cluster head communicates directly with the sink node. Alternatively, it could be in a flat communication structure as well, where each sensor has essentially the same role and relies on other sensors to relay its messages to the sink node via multi-hops radio communication. In this paper, assume the flat communication architecture, where the joint problem of sensing coverage and network connectivity arises.

The environment is considered as a stationary sensor networks in a two-dimensional field and assume that sensor nodes are randomly and independently deployed in a field. Compared to other sensor deployment strategies such as deployment in grids or in predefine positions, random deployment is much easier and cheaper [52]. Also, scheduling for a regular network topology such as grids is simple and may not deserve further investigation.

Another assumption is that a sensor node's radio transmission range is fixed and totally independent of its sensing range because of different hardware components involved. Unlike other work [57] that puts certain constraints on the radio range and the sensing range, our work makes no assumption on the relationship between them. Accurate global time synchronization is not required, which is an extremely hard task for large-scale sensor networks. Instead, our scheduling algorithm permits slight time asynchrony without performance degradation.

3.4.2 Randomized Scheduling for Coverage

The proposed randomized scheduling algorithm for sensing coverage which has several prominent features [58]. The algorithm does not assume the availability of any location or directional information. It is a purely distributed algorithm, thus scalable for large networks. It is also resilient to clock asynchrony and requires only a roughly synchronized clock, which significantly decreases the energy and communication overhead introduced by maintaining network-wide time synchronization. In the following, we briefly summarize the basic idea of the randomized scheduling algorithm.

Assume that the sensor nodes constitute a set S. Given a number k, each sensor node randomly joins one of the k disjoint subsets of set S. Once the k subsets are determined, they work alternatively. At any given time, there is only one subset working, and all the sensor nodes belonging to this subset will turn on. The intuition is that when the network is sufficiently dense, each subset alone will cover most part of the field. This randomized algorithm was stimulated by the work [59] and has been proposed independently at the same time by [53] and [58].

Figure 3.4-1 shows an example. Assume that we have eight sensor nodes (with IDs 0; 1; . . . ; 7) randomly deployed in a rectangular area. Assume that the eight sensor nodes will be assigned into two disjoint subsets, S0 and S1. Each sensor randomly selects a number (i.e., 0 or 1) and then joins the corresponding subset. Assume that sensor nodes 0; 2; 5; 6 select number 0 and, thus, join subset S0, and sensor nodes 1; 3; 4; 7 select number 1 and, thus, join subset S1. Then, subsets S0 and S1 work alternatively, that is, when sensor nodes 0; 2; 5; 6, whose sensing ranges are denoted as the solid circles, are active, sensor nodes 1; 3; 4; 7, whose sensing ranges are denoted as the dashed circles, fall asleep, and vice versa.

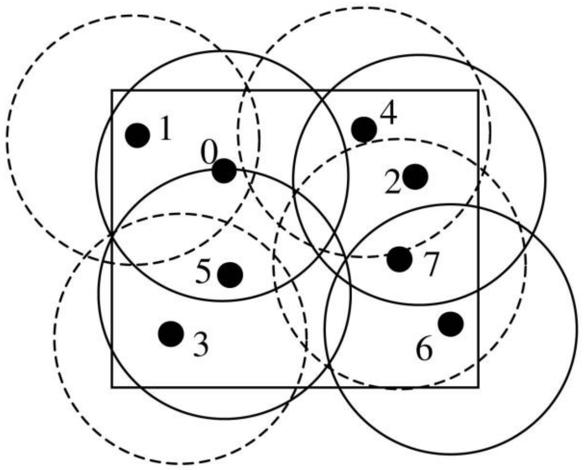


Figure 3.4-1 an example of the randomized coverage-based algorithm

3.4.3 Joint Scheduling: Random Coverage with Guaranteed Connectivity

With the proposed randomized coverage-based scheduling scheme, the coverage quality can be guaranteed statistically by setting an appropriate subset number k. Yet, there is no guarantee on the network connectivity after scheduling. To operate successfully, a sensor network must be connected so that sensor nodes can report the detected events to the sink node. Therefore, in addition to sensing coverage, the sensor network must remain connected, i.e., the active nodes should not be partitioned in any schedule of node duty cycles. We are, hence, motivated to enhance the above randomized scheduling algorithm such that both coverage quality and network connectivity can be met at any given time.

Given that the total number of subsets is k, after the randomized coverage-based scheduling scheme, there are k sub-networks formed, each of which corresponds to a specific subset and consists of all the nodes assigned to that subset. However, there is no guarantee on the connectivity of each sub-network. The following extra-on rule ensures that each sub-network is connected, given that the original network before scheduling is connected. Besides, it also guarantees that the path from any sensor node to the sink node has the global minimum hop count.

Assume that each sensor node knows its minimal hop count to the sink node. A sensor node A is called the upstream node of another sensor node B, if node A and node B are neighboring nodes and the minimal hop count of node A to the sink node is one less than that of node B. Node B is also called node A's downstream node.

Extra-on rule: If a sensor node A has a downstream node B, which is active in time slot i, and if none of node B's upstream nodes is active in that time slot, then node A should also work in time slot i. In other words, besides working in the duty cycles assigned by the randomized coverage-based scheduling, node A is required to work in extra time slots, e.g., time slot i in this case.

This rule requires each sensor node to maintain its minimum hop count to the sink node and the list of its upstream nodes. We stress that the minimum hop count is used to label the relative location information among sensor nodes. Since we focus on static sensor networks only and the failure of certain nodes does not influence such relative relationship, our joint scheduling based on the extra-on rule works correctly in face of network failure without requiring periodical update of the minimum hop count values. The method of collecting the hop count information and its energy cost will be addressed in detail in following paragraphs.

It is natural to realize that the extra-on rule may cause problems of synchronization and large overheads due to the dependency among nodes. These potential problems, however, have been carefully avoided in this protocol design. The details will be given later.

Now let's take a look at the scheme in details. The joint scheduling method ensures the coverage quality and network connectivity simultaneously and has the following steps.

Step 1: Select a Subset Randomly. Initially, each sensor node generates a random number i from $[0, k-1]$ and assigns itself to subset i. This is exactly the same as in the randomized coverage-based scheduling scheme.

Step 2: Propagate Minimum Hop Count. This step starts from the sink node at the time when it broadcasts a HOP advertisement message to its immediate neighboring sensor nodes. Each HOP advertisement message contains the minimum hop count to the sink, the nodeID and its subset decision. In the packet broadcast from the sink, the minimum hop count is set to 0. Initially, the minimum hop count to the sink is set to infinity at each sensor node.

Each node, after receiving a HOP advertisement message, will put the message in its buffer. It will defer the transmission of the HOP message after a backoff time and only rebroadcasts the HOP message that has the minimum hop count. Before the rebroadcast of the HOP message, the hop count value in the HOP message is increased by 1. With this method, HOP message broadcasts with a non-minimal hop count will be suppressed if the HOP message with the actual minimal hop count arrives before the backoff time expires. The number of broadcasts from each sensor node depends on the length of backoff time. Increasing the backoff time will significantly decrease the number of broadcasts. Although a large backoff time value will increase the total time required for the completion of this step, we argue that it is an effective solution because this step is a one-time task for static sensor networks and the energy is the most precious resource for sensor nodes.

If no packets are lost, our method can guarantee that at the end of this step, each sensor node will obtain the minimum hop count to the sink node. In practice, packets may be lost due to collisions or poor channel quality. Nevertheless, packet losses will not impact the successful operation of our joint scheduling scheme, i.e., the network will still be connected even if some nodes may have only a nearly shortest path to the sink node. Our simulation results in Section 6.2 demonstrate that the number of nodes without knowing the actual minimum hop count at the end of this step is negligible even in the presence of packet losses.

Step 3: Exchange information with local neighbors. Each sensor node locally broadcasts its minimum hop count, its nodeID, its subset decision, the nodeIDs of its upstream nodes and their subset decisions. The upstream nodes are the nodes from which the current node receives its minimum hop count. Each sensor node records and maintains all the information it receives from its immediate neighbors.

Step 4: Enforce the extra-on rule. Based on the extra-on rule and the information from Step 3, each sensor node decides the extra time slots it has to remain active to ensure network connectivity and updates its working schedule accordingly. Then the updated working schedule is broadcasted locally to neighboring sensor nodes.

It is easy to see that the update of a sensor node's working schedule can impact the working schedule of its upstream nodes and the neighboring nodes with the same minimum hop count to the sink. To minimize the number of broadcasts of working schedule updates, it is desirable that a sensor node updates its working schedule after it receives all of the latest working schedules from its downstream nodes. This is exactly the reverse process of Step 2. Therefore, a backoff-based broadcast scheme similar to that in Step 2 can be applied here.

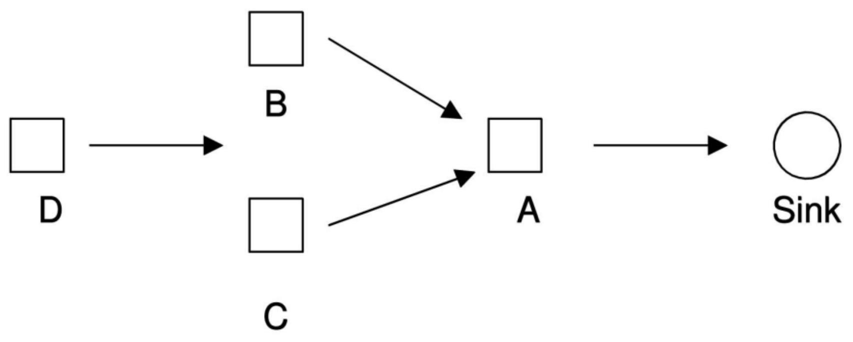


Figure 3.4-2 an example of the extra on rule

As an example, assume that the network consists of one sink node and four sensor nodes, A, B, C, and D as shown in Figure 3.4-2. D is three hops away, B and C are two hops away, and A is one hop away from the sink node. Assume that at the end of Step 1, A, B, C and D are assigned to time slots 1, 2, 3, and 4, respectively. Assume that D broadcasts its updated working schedule first. B and C are its upstream nodes and in Step 3 they know that node D does not have an upstream node working in time slot 4. After they receive D's working schedule update, there are several possibilities:

1. Case 1: Suppose that B and C can hear each other. If B broadcasts its working schedule prior to C, C can hear B's updated working schedule and knows that D has an upstream node working in time slot 4. So, C will not schedule itself to work in time slot 4. In this case, B will work in time slots 2 and 4 and C will work in time slot 3 only. Likewise, if C broadcasts its working schedule prior to B, C will work in time slots 3 and 4 and B will work in time slot 2 only.
2. Case 2: Suppose that B and C cannot hear each other. B and C will both work in time slot 4 to ensure the network connectivity, no matter which node broadcasts first. Therefore, B will work in time slot 2 and 4 and C will work in time slot 3 and 4.

In both cases, based on the latest working schedules received from nodes B and C, node A will know that it has to work in time slots 2, 3, and 4 to ensure network connectivity. Therefore, A will work in time slots 1 to 4.

Step 5: Work according to the new working schedule.

The overhead analysis of joint scheduling in terms of the average number of broadcasts from individual sensor node is presented below.

Step1: No broadcast is needed in this step.

Step2: The number of broadcasts from each sensor node depends on the length of the backoff time. The relationship is studied via simulation. 1500 sensor nodes are randomly deployed in a $200m \times 200m$ area and place the sink node at the center of the area. The radio range of each sensor node is fixed to $10m$. The CSMA MAC layer protocols are extended to adopt the scheduling scheme. To broadcast a HOP advertisement message, Assume that each sensor node has to capture the channel for 1 ms. Figure 3.4-3 illustrates the results with the backoff time from 1ms to 120ms.

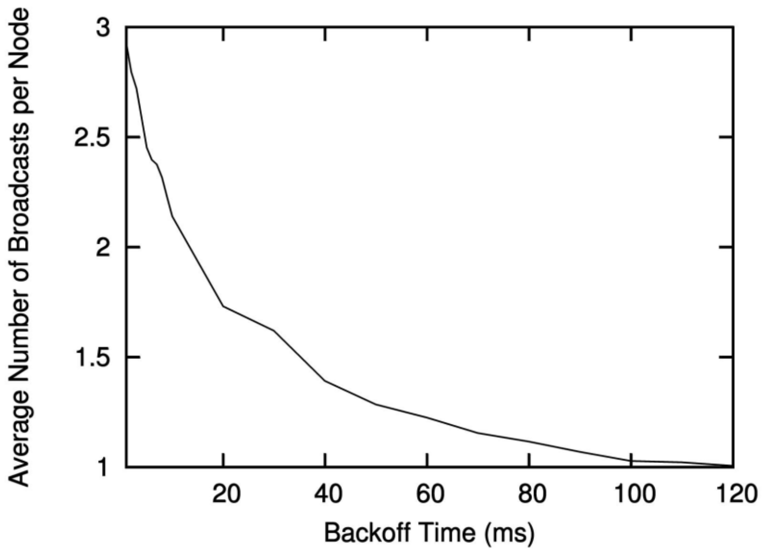


Figure 3.4-3 Average number of broadcasts per node in Step 2

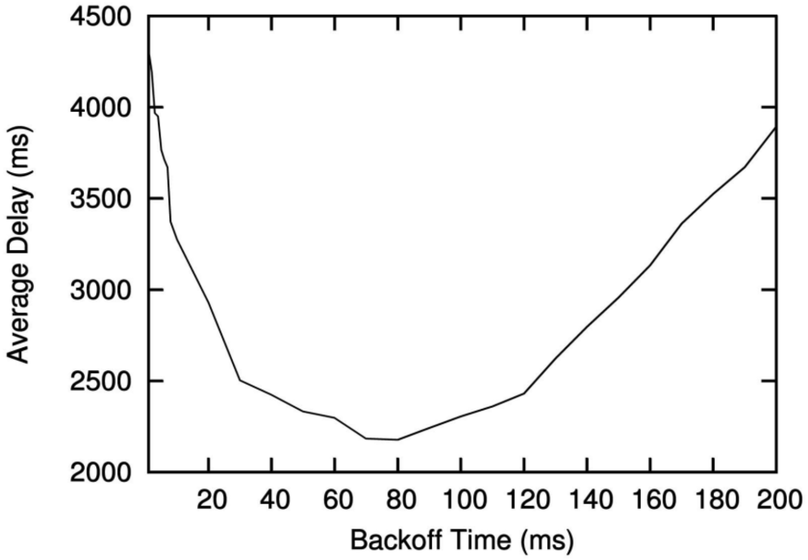


Figure 3.4-4 Average delay to complete Step 2

From the figure, we can see that if the backoff time is large enough, each sensor node almost broadcasts only once in this step. The energy saving is at the cost of longer delay to complete this step, as shown in Figure 3.4-4. Nevertheless, since this is only a one-time task, the delay should not be a big concern. Actually, our simulation results indicate that if the backoff time is set to 120 ms, the time from the moment when the sink node broadcasts the HOP advertisement message to the moment when the last sensor node finishes the HOP advertisement message is below 2; 500 ms, which is acceptable. Fig. 4 also indicates that using a too small backoff time does not necessarily reduce the total delay since a node may need to broadcast multiple times if HOP messages with a smaller hop value arrive later.

Step3: Each sensor node needs to broadcast only once to notify their neighbors.

Step4: Since the propagation of the extra-on decisions is actually the reverse process of Step 2, we can expect that most of the sensor nodes will broadcast only once if the backoff time is appropriately set.

Step5: No broadcast is needed in this step.

In conclusion, the overhead for most of the sensor nodes is three local broadcasts if the backoff time in Step 2 and Step 4 are large enough. The whole system setup process can be finished within several seconds, given a sensor network system similar to the one in our simulation.

3.4.4 Advantages of the Joint Scheduling Algorithm

First, the joint scheduling method can guarantee that the resulting coverage quality is above a given requirement, since the extra-on nodes actually increase the coverage quality provided by the randomized coverage-based scheme. Later, performance evaluation demonstrates that the number of extra-on sensor nodes is not large in any time slot; hence, good coverage with guaranteed connectivity is not at the cost of large energy waste.

Second, the route from each sensor node to the sink node has the minimum or nearly minimum hop count. This feature of our joint scheduling method roughly eliminates the extra energy consumption on data delivery with unnecessarily longer paths.

Third, with the joint scheduling method, the routing problem is simultaneously addressed with the connectivity problem. A sensor node not only has a route to the sink node, but also knows the upstream node in this route. Therefore, no additional routing protocols are needed.

Fourth, the overhead to set up the system is small. As demonstrated in the previous section, most sensor nodes need only three local broadcasts if the backoff times in Step 2 and Step 4 of the joint scheduling algorithm are set appropriately.

Finally, since each sensor node uses only local information to make its scheduling decision, the joint scheduling method is purely distributed and is scalable well to large and dense networks.

Note that our joint scheduling scheme exploits the redundancy in both sensing coverage and network connectivity. However, we cannot exclude the possibility that some nodes are critical nodes for connectivity, i.e., the network will be partitioned if these nodes are turned off.

Therefore, to maintain network connectivity, our joint scheduling method has to turn on these critical nodes all the time. Therefore, these critical nodes might die sooner than other nodes due to their heavier workload. We point out that there is no algorithm to solve this problem unless more nodes are deployed nearby these critical nodes to increase redundancy. Also, note that the joint scheduling is decoupled from the MAC layer protocol. Although the network topology is different from time slot to time slot, within a single time slot, the network topology is fixed and any MAC layer protocol can be used.

3.4.5 Simulation evaluation

Symbol	Description
n	the total number of deployed sensor nodes
T	the time duration of each time slot
a	the size of the whole field
r	the size of sensing area of each sensor
k	the number of disjoint subsets
s	the number of sensor nodes that cover a specific point inside the field
S	the set of sensor nodes that cover a specific point inside the field
s_i	the number of sensor nodes that belong to subset i and cover a specific point inside the field
S_i	the set of sensor nodes that belong to subset i and cover a specific point inside the field
C_p	coverage intensity for a specific point
C_n	network coverage intensity

Table 3.4-1 Table of Notions

A. Simulation Settings

We evaluate the performance of the joint scheduling algorithm on a simulator we implemented in Java. We use the CSMA MAC layer protocol. In our simulation, we deploy sensor nodes randomly in a $200m \times 200m$ square region. The sink node is located at the center of the region. The total number of sensor nodes is selected to meet any given network coverage intensity. The sensing range of each sensor node is fixed to $10m$. We normalize the communication range with the sensing range and use the ratio of the communication range over the sensing range as the measure of the communication range. The traffic load is very light such that packet losses are mainly caused by network partition or channel errors. Under each simulation scenario, 100 runs with different random seeds are executed.

The following metrics are used to evaluate the joint scheduling algorithm:

Packet Delivery Ratio: It is defined as the ratio of total number of packets received at the sink node over the total number of transmitted packets from sensor nodes. Because the traffic load is very light, this metric is an indicator of network connectivity.

The Ratio of Nodes Having the Shortest Path: It is defined as the number of nodes that have the shortest path to the sink node over the total number of nodes. It is an indicator of path optimality.

The Ratio of Extra-On Sensor Nodes: It is defined as the ratio of the number of sensor nodes, which must remain active beyond their regular working shifts assigned by the randomized coverage-based scheduling algorithm, to the total number of deployed sensor nodes. A small ratio of extra-on sensor nodes indicates that small extra energy is required to maintain connectivity after the coverage requirement is granted.

Network Coverage Intensity: It is a measure of coverage quality.

B. Network Connectivity and Path Optimality

To demonstrate that the extra-on rule can assure network connectivity, The packet delivery ratio with and without the extra-on rule applied to the network are compared. The number of deployed sensor nodes is determined by the network coverage intensity C_p and the number of disjoint subsets k. In this test, to preclude the packet losses due to broadcast collision or channel errors, we adopt a perfect radio channel without medium contention. In all the simulated scenarios, the networks are connected if all the deployed sensor nodes are active. After randomly turning off redundant sensor nodes without applying the extra-on rule, the networks are partitioned. This is evident from the fact that the packet delivery ratio cannot achieve 100 percent without using the extra-on rule as shown in Figure 3.4-5. However, with the extra-on rule, the networks can always achieve a 100 percent packet delivery rate, indicating that the extra-on rule provides guaranteed network connectivity.

As discussed in Section 3.4, if there are packets losses exist, some nodes may not have the shortest path to the sink node. Nevertheless, from Figure 3.4-6, we can see that at the end of Step 2 of the joint scheduling scheme, even if the packet loss rate is as high as 10 percent, the ratio of nodes having the shortest path to the sink node is no less than 95 percent.

C. Ratio of Extra-On Sensor Nodes

Apparently, there are three factors influencing the ratio of extra-on sensor nodes: network coverage intensity, the number of subsets, and the ratio of the communication range over the sensing range.

To investigate the influence of network coverage intensity, we fix the number of subsets and vary the communication range and the network coverage intensity. As shown in Figure 3.4-7, the ratio of extra-on sensor nodes drops with the increase of the coverage intensity and the communication range. This is because when the coverage intensity increases, more sensor nodes will remain active for coverage, hence few extra nodes are needed for network connectivity. In addition, the increase of communication range enhances the connectivity of the original networks, resulting in the decrease of the number of extra-on sensor nodes.

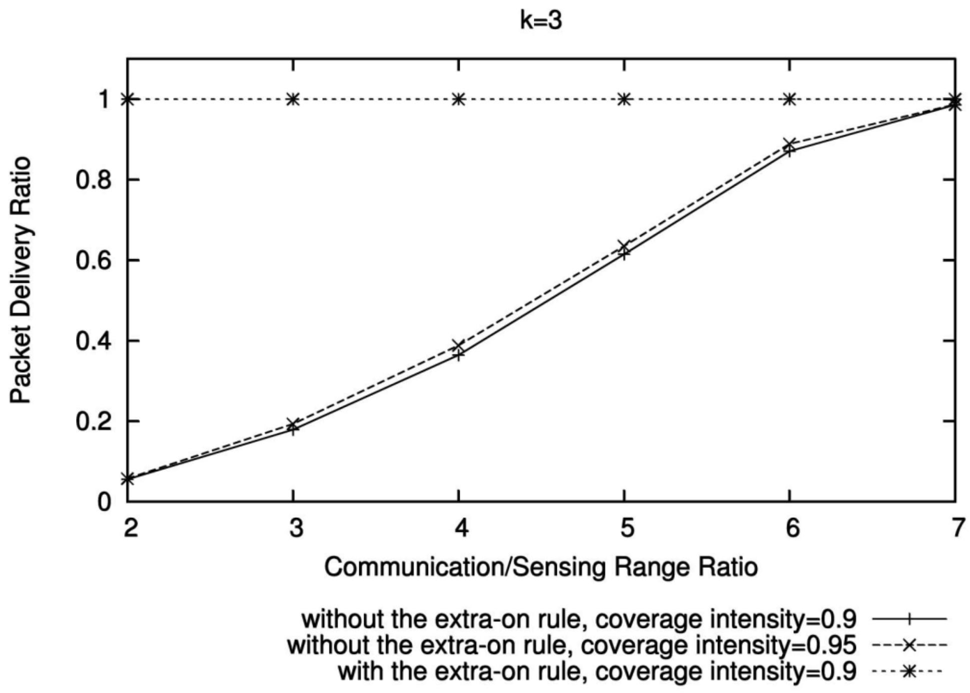


Figure 3.4-5 The Packet delivery ratio

Similarly, to investigate the influence of the number of subsets, we fix the network coverage intensity and vary the communication range and the number of subsets. As shown in Figure 3.4-8, increasing communication range decreases the ratio of extra-on sensor nodes, due to the same reason mentioned above.

From Figure 3.4-8, we can see that the ratio of extra-on nodes decreases with the increase of k. Given fixed coverage intensity, the number of simultaneous active sensor nodes scheduled by the randomized scheduling algorithm is roughly the same and is independent of the value of k. That is, the density of the network after the randomized scheduling only depends on the coverage intensity. Since network connectivity is mainly determined by network density, the number of extra-on nodes should be roughly the same in order to maintain network connectivity. Since for a given coverage intensity a larger k value means a larger total number of deployed sensor nodes, the ratio of extra-on sensors decreases with the increase of k.

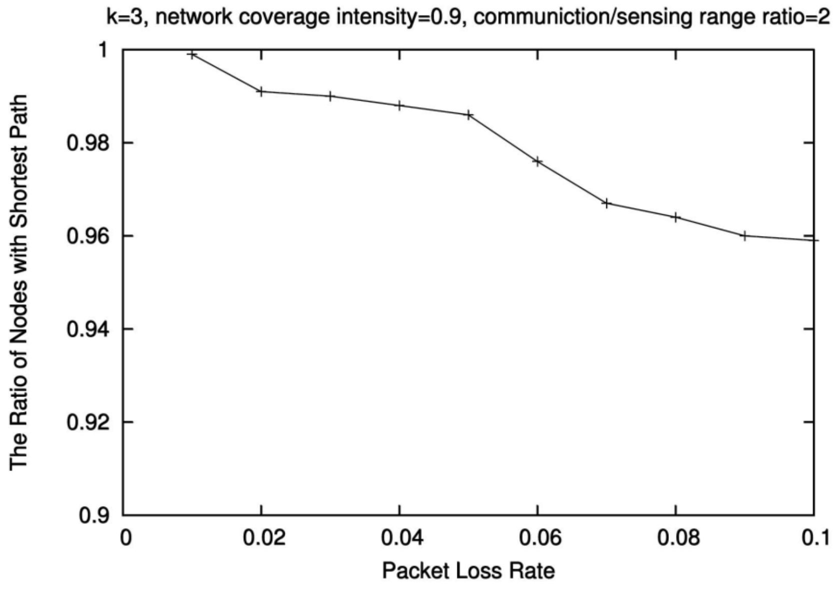


Figure 3.4-6 The ratio of the nodes having the shortest path

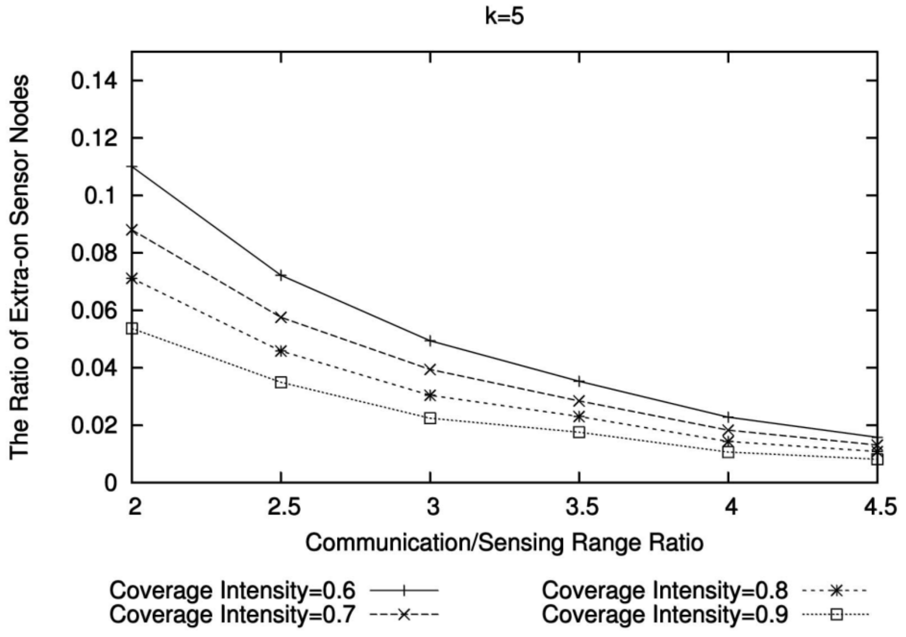


Figure 3.4-7 Influential factors of average number of extra-on nodes (k is fixed)

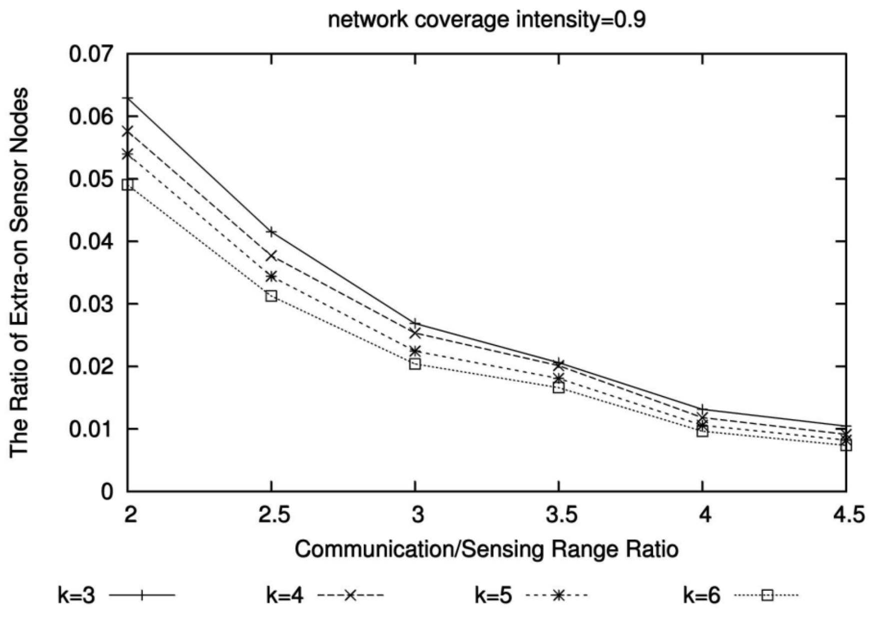


Figure 3.4-8 Average number of extra-on nodes (coverage intensity is fixed)

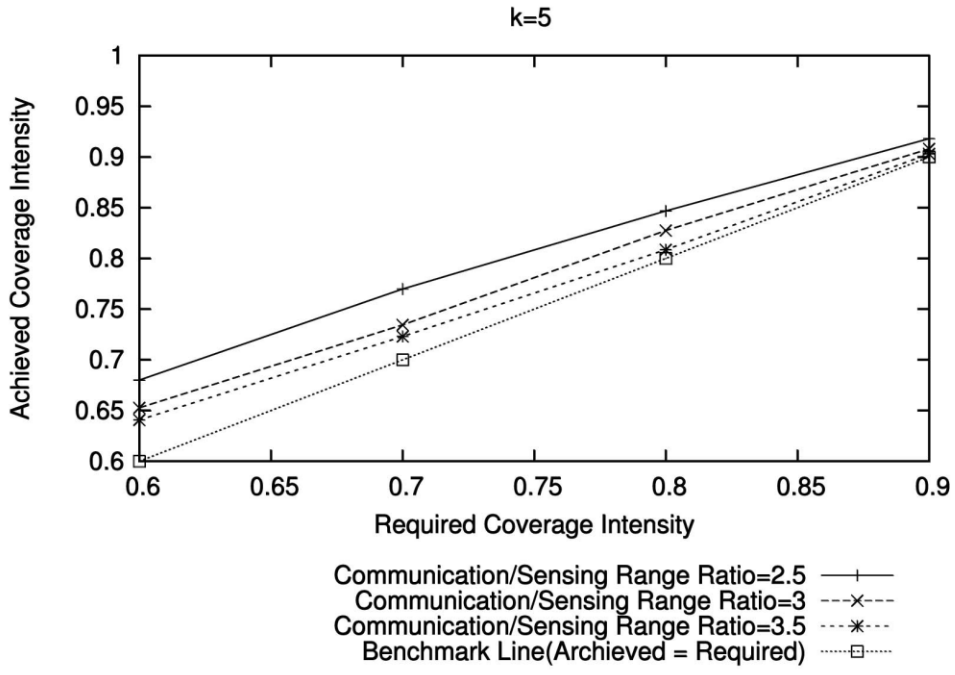


Figure 3.4-9 Achieved coverage intensity versus required coverage intensity

In Figure 3.4-7 and Figure 3.4-8, when the coverage intensity is sufficiently high and the communication range is sufficiently large, the number of extra nodes needed to turn on for connectivity maintenance is very small, compared to the total number of active nodes for coverage. Therefore, with our joint scheduling approach, the extra energy consumption on connectivity maintenance is small and the network coverage intensity does not unnecessarily exceed a given requirement too much.

D. Network Coverage Intensity

The achieved network coverage intensity by our joint scheduling method is illustrated in Figure 3.4-9. As expected, the achieved coverage intensity is always slightly higher than the required coverage due to the fact that extra sensor nodes need to stay on to maintaining network connectivity.

3.4.6 Conclusion

Sensor scheduling plays an essential role for energy efficiency of wireless sensor networks. Traditional sensor scheduling methods usually use sensing coverage or network connectivity, but rarely both. Some research [60,61] has dealt with the joint problem of sensing coverage as well as network connectivity, but requires specific network topology such as grid or strong constraints on the relationship of sensing range and radio transmission range. The randomized scheduling method provides statistical sensing coverage and then switch on extra sensors, if necessary, for network connectivity. Analytical and simulation results demonstrate the effectiveness of the above joint scheduling method. This scheme is by far the most optimal scheduling scheme we have discussed. However it is necessary to point out that as mentioned before this scheme only provides "statistical sensing coverage", which means only with certain probability the sub-network sets of sensors grouped by the scheduling scheme could provide qualified coverage and fewer extra-on nodes. There is no guarantee that the scheme will deliver quality service. For some of the critical applications of the WSN, such as forest first watching or volcano activity monitoring, we cannot take chances.

So in next chapter, we will present our proposed scheduling scheme which guarantees better coverage and fewer extra-on nodes. Details of the scheme and the simulation result are. The comparisons are mostly done against the joint scheduling scheme in section 3.4, simply because it has outperformed all the other schemes we have discussed so far.

Chapter 4

New Proposal: A Clique Based Sensor Scheduling Scheme with Guaranteed Connectivity

The chapter is the main body of our research. Our proposed new scheduling scheme will be described in details, along with the performance analysis and finally simulation results.

Our research is regarding the node scheduling problem of m-covered and connected sensor networks, a new concept of Clique is proposed firstly in this paper, and based on which, a new distributed node scheduling algorithm CCMS is designed, which can allocate all nodes in the sensor network into $k(k \leq m)$ different groups $\{0, 1, \ldots, k-1\}$ without requiring location information, and at the same time, it can guarantee that each group will be still connected and maintain the coverage ratio as high as possible. Theoretical analysis and simulation results show that the newly proposed Coverage and Connectivity Maintaining Scheduling scheme (CCMS) has better performance than previous randomized joint scheduling scheme and elongate the lifetime of the sensor network effectively.

With the capabilities of sensing, data processing, GPS positioning and communication, sensors have great potential in a wide variety of commercial and military applications including environmental and military monitoring, earthquake and weather forecast, underground, deep water and outer space exploration. Because of the surrounding uncertainty, usually hundreds or thousands of sensor nodes need to be deployed simultaneously. Those nodes are self-organized to collect special information of a sensory field. A wireless sensor network (WSN) is a collection of sensor nodes deployed over a region of interest for sensing or monitoring certain conditions or events, and collect data for further analysis in order to achieve the goal of such deployment [1,2,3]. Due to their extremely small dimension, sensor nodes have very limited energy supply, and it is usually hard to recharge the battery after deployment, either because the number of sensor nodes is too large, or because the deployment area is hostile. One challenge in the research of sensor networks is to obtain a maximum system lifetime by using the limited energy sources. Because of the densely distribution of sensor nodes, node scheduling plays a critical role for energy efficiency in WSNs [4,5,6].

Currently, there are many node scheduling methods proposed, which can be classified into the following two major categories: round-based node scheduling and group-based node scheduling. In a round-based node scheduling method, the sensor nodes will execute the scheduling algorithm during the initialization of each round. According to some kind of competition scheme, some nodes will be keep active in the current round, and other nodes will keep sleep instead [7,8,9]. This kind of methods requires each sensor node to execute the scheduling algorithm for more than once during its lifecycle. But in a group-based node scheduling method, each node will execute the scheduling algorithm only once after its deployment. After the execution of the scheduling algorithm, all sensor nodes will be allocated into some different group. Therefore, in each of the followed time slots, each group of nodes will keep active in turn [10,11].

In a node scheduling method, there are two major problems needed to be considered: the maintenance of coverage and the connectivity for the whole sensory field. One difficulty of the scheduling problem is that a sensor's sensing range is totally independent with its radio transmission range. Therefore, it is difficult to combine and solve the two problems together [10]. Most researchers focus on the research of scheduling based on sensing coverage [12,13,14,15] or scheduling based on network connectivity [16,17,18,19] separately. Since it is usually costly to obtain and maintain the location information of each sensor node, the joint scheduling problem is even more difficult is the location of each sensor node is unknown [10,20,21].

If each point in a sensory field is monitored by at least m sensors, then we call the field is m-covered. Sensor nodes are usually densely deployed, i.e. we can assume that the sensory field is m-covered [22]. In this paper, we deal with the following challenging task: Assume that the sensor network is m-covered and connected. Without location information, how can we schedule the sensor nodes into $k(k \le m)$ different groups $\{0, 1, \dots, k-1\}$, such that each group will be still connected and can at the same time maintain the coverage ratio as high as possible.

In order to solve the above problem, we define a new structure named a Clique in an m-covered and connected sensor network. Using the cliques, we present a new group ID assignment algorithm for a clique. A new algorithm for nodes in the sensor network to find out its local cliques in a distributed way is given then. Finally, we give a connectivity maintenance algorithm to guarantee the connectivity for each group.

4.1 Network Model and Definitions

4.1.1 Network Model

In this paper, we consider stationary sensor networks in a two-dimensional field and assume that sensor nodes are randomly and independently deployed in a field.

Next, we assume that a sensor's sensing range, which is denoted by r_s , is defined as the range beyond which the sensor's sensing ability can be neglected. Furthermore, we assume that sensor's communication range r_c is larger than or equal to $2r_s$, which is usually true in practice. For example, ultrasonic sensors have a sensing range of approximately 0.2 to 6 meters while the transmission range of MICA motes is about 30 meters [62].

We do not assume accurate global time synchronization, which is an extremely hard task for large-scale sensor networks. Instead, our scheduling algorithm permits slight time asynchrony without performance degradation.

Finally, before deployment, we assume that all of the sensor nodes are encoded with Node ID (NID) from $\underbrace{00\ldots001}_{\text{n}}$ to $\underbrace{11\ldots111}_{\text{n}}$according to the coding method of the n-dimensional Hyper-cubes [26], and encode the Sink as $\underbrace{00\ldots000}_{\text{n}}$ especially. So after deployment, each sensor node will have a unique ID randomly chosen from $\{\underbrace{00\ldots001}_{\text{n}}\ \ldots\ \underbrace{11\ldots111}_{\text{n}}\}$ beforehand, and the Sink will have an ID as $\underbrace{00\ldots000}_{\text{n}}$.

4.1.2 Definitions

For ease of use, we give the following definitions and nations which will be used in the rest of the chapter.

Definition *4.1.1* For any t sensor nodes$p_1, p_2, \ldots, p_t$, if they are neighboring to each other, then the set $\{p_1, p_2, \ldots, p_t\}$ is called a t-clique.

Definition *4.1.2* For any two cliques C_1 and C_2, if the nodes in C_1 all belong to C_2 ($C_1 \subseteq C_2$), then we record their relationship as $C_1 \in C_2$, and call C_1 a sub-clique of C_2 . Otherwise, if there exists at least 1 node in C_1 that doesn not belong to C_2, then we record their relationship as $C_1 \notin C_2$.

***Definition** 4.1.3* For any two sensor nodes p_1 and p_2, if the code of p_1 is bigger than that of p_2, then we record their relationship as $p_1 > p_2$.

4.2 Clique Based Node Scheduling Algorithm

We will describe our Clique Base Node Scheduling Scheme in details as follows.

A. GID Assignment for Nodes in Cliques

From the assumption that $r_c \geq 2r_s$, we have the following theorem.

***Theorem** 4.2.1* Suppose that a given point p inside a monitored field is covered by t sensor nodes, then these t sensor nodes form a t-clique.

Proof: Suppose that the t sensor nodes that cover the given point p are $p_1, p_2, \dots, p_t$, then the distance betweenp_j and p equals to or less than r_s for each $p_j \in \{p_1, p_2, \dots, p_t\}$, i.e. $d(p_j, p) \leq r_s$. From the triangle inequalities, $d(p_i, p_j) \leq d(p_i, p) + d(p_j, p) \leq 2r_s \leq r_c$, for any two nodes p_i and p_j in$\{p_1, p_2, \dots, p_t\}$, which means that the two nodes p_i and p_j are neighboring to each other. Therefore these t sensor nodes form a t-clique.

Based on the above theorem, we give a distributed GIDs (Group IDs) assignment algorithm for the sensor nodes in a t-clique for an m-covered and connected sensor network as follows (where $t \geq m$).

Algorithm-1 (GANC: GID Assignment for Nodes in a t-clique):

Assume that the sensor network is m-covered and connected. For any t-clique consists of t ($t \geq m$) sensor node $p_1, p_2, \dots, p_t$ with ID $P_1, P_2, \dots, P_t$, the nodes will be allocated into k ($k \leq m$) different groups $\{0, 1, \dots, k-1\}$ according to the following steps:

Step 1: without loss of generality, assume that $P_1 = min\{P_1, P_2, \dots, P_t\}$, then p_1 claim itself as the clique-head of the t-clique $\{p_1, p_2, \dots, p_t\}$. If all nodes have been assigned GIDs already, then the algorithm is terminated. Otherwise, go to Step 2.

Step 2: suppose that the first a ($a < t$) nodes $\{p_1, p_2, \dots, p_a\}$ in $\{p_1, p_2, \dots, p_t\}$ have been assigned GID $\{\acute{g}_1, \acute{g}_2, \dots, \acute{g}_a\}$ already, and $\{g_1, g_2, \dots, g_b\}$ are all the different GIDs in $\{\acute{g}_1, \acute{g}_2, \dots, \acute{g}_a\}$, where $b \leq a$.

Case 1: if $b = k$, then for each sensor node $p_j \in \{p_{a+1}, p_{a+2}, \dots, p_t\}$, p_j selects $j \in \{0, 1, \dots, k-1\}$ randomly, and assigns GID j to node p_j.

Case 2: If $b < k$, then let

$$U = \{ 0,1,\dots,k-1 \} \backslash \{ g_1, g_2, \dots, g_b \} = \{ u_0, u_1, \dots, u_{k-b-1} \}.$$

Sub-case A: if $t-a \geq k-b$, then p_1 selects $(t-a)-(k-b)$ different GIDs $V = \{ v_0, u_1, \dots, u_{(t-a)-(k-b)-1} \}$ from $\{ 0,1,\dots,k-1 \}$ randomly, and distributes $U \cup V$ to nodes $\{ p_{a+1}, p_{a+2}, \dots, p_t \}$ randomly.

Sub-case B: if $t-a < k-b$, then p_1 selects $t-a$ different GIDs $\{ v_0, u_1, \dots, u_{t-a-1} \}$ from U randomly, and distributes these GIDs to $\{ p_{a+1}, p_{a+2}, \dots, p_t \}$ randomly.

Remark

1) In step 1, a clique-head will be selected first from the clique, and the clique-head will collect the GIDs of all its Clique members to determine which nodes have been assigned GIDs already, and which nodes haven't yet.
2) In step 2, the nodes $\{ p_1, p_2, \dots, p_a \}$ in $\{ p_1, p_2, \dots, p_t \}$ denote all the nodes that have been assigned GIDs in the clique, and the $t-a$ nodes $\{ p_{a+1}, p_{a+2}, \dots, p_t \}$ denotes all the nodes that haven't been assigned GIDs yet. Additional, b represents the number of different GIDs that have been assigned to $\{ p_1, p_2, \dots, p_a \}$ already, and U represents the set of GIDs that haven't been assigned to $\{ p_1, p_2, \dots, p_a \}$ yet. Finally, $t-a \geq k-b$ means that the number of nodes that haven't been assigned GIDs in the clique is not smaller than the number of GIDs that haven't been used yet. And $t-a < k-b$ means that the number of nodes that haven't been assigned GIDs is small than the number of left GIDs.

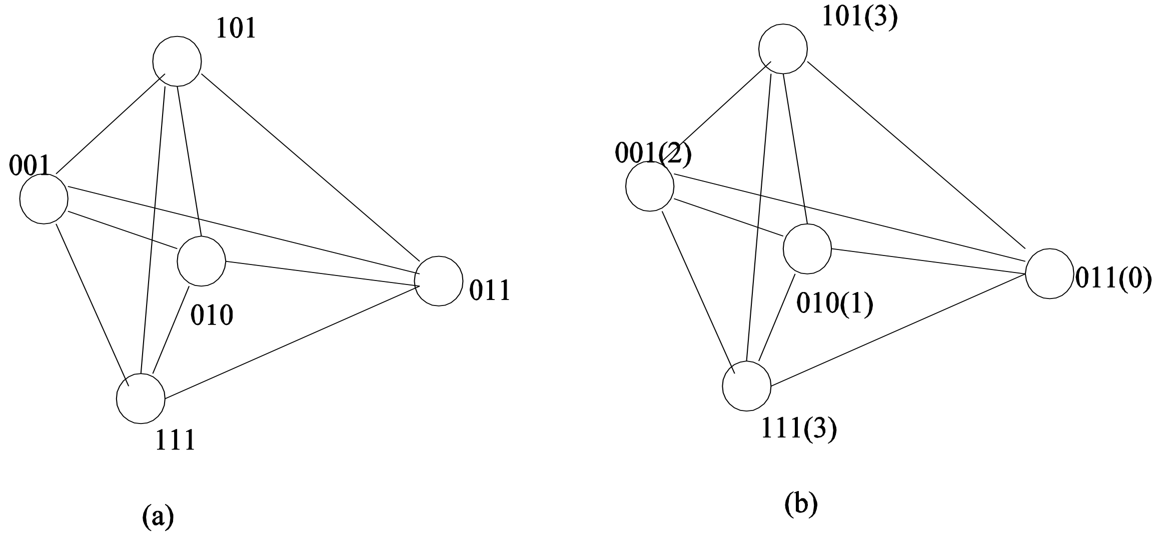


Figure 4.2-1 A 5-clique in a 4-covered Sensor Network

The following example will demonstrate how *Algorithm-1* works.

Example 1: Figure 4.2-1(a) shows a 5-clique. Let $k = 4$, so try to puts the nodes in 4 different groups $\{0, 1, 2, 3\}$. Since node 001 has the smallest ID in the clique, it becomes the clique-head according to the step 1 of *Algorithm-1*.

Suppose that all of the five nodes $\{001,010,011,011,101,111\}$ have not been assigned GIDs yet. We have $a = b = 0$, and $U = \{0, 1, 2, 3\}$. According to case 2 sub-case A, node 001 will select a random number from $\{0, 1, 2, 3\}$, say 3, i.e., $V = \{3\}$. $U \cup V$ then are distributed to nodes as the GIDs uniformly. In this example, nodes 001, 010, 011, 101, 111 are assigned 2, 1, 0, 3, 3 respectively as shown in Figure 4.2-1(b).

B. Finding Local Cliques

Algorithm-1 solved the problem about how to assign k different GIDs most evenly to all sensor nodes in a t-clique, where $t \geq k$. From *Theorem 4.2.1*, we know that if a point in the sensory field covered by t different nodes $p_1, p_2, \ldots, p_t$, then $\{ p_1, p_2, \ldots, p_t \}$ forms a t-clique. In this subsection, we consider how to find out these cliques. First, we look at the following scenario.

Example 2: In Figure 4.22-2, there are 6 sensor node 001, 010, 011, 101, 110, 111 distributed in a local area of a 3-covered sensory field. A circle denotes the sensing range of the corresponding node, and a line between a pair of node denotes that these two nodes are neighbors. In this diagram, a given point p is covered by 4 nodes 001, 011, 101, 111. So the node 001 can compute the 4-clique 001, 011, 101, 111 as follows:

1) During information exchanging with neighbors, each neighbor j of 001 will send the neighboring nodes list of j to node 001. Then 001 will maintain the following information Table 4.2-1, where IDNN denotes ID of Neighboring Node, NNLN denotes Neighboring Node List of Neighbors.

Row ID	IDNN	NNLN	$\{IDNN\} \cup \{NNLN\}$
1	001	001,011,101,110,111	001,010,011,101,110,111
2	010	001,011,110	001,010,011,110
3	011	001,010,101,111	001,010,011,101,111
4	101	001,011,111	001,011,101,111
5	110	001,010	001,010,110
6	111	001,011,101	001,011,101,111

Table 4.2-1 Information Table Obtained by Node 001

2) According to the information in the Table 4.2-1, node 001 computes the 4-clique $\{001,011,101,111\}$ using $\{IDNN\} \cup \{NNLN\}$ recorded in

rows 1, 3, 4 and 6 as follows
$\{001,010,011,101,110,111\} \cap \{001,010,011,101,111\} \cap$
$\{001,011,101,111\} \cap \{001,011,101,111\} = \{001,011,101,111\}$.

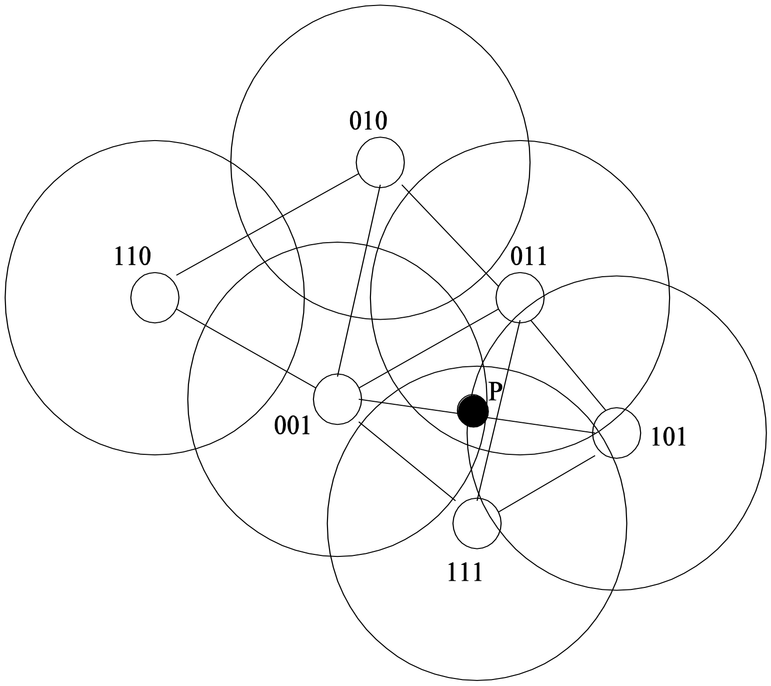


Figure 4.2-2 An Example for Clique Searching

In general, we can bring forward a solution to the problem of t-clique searching in a distributed way in an m-covered sensor network as follows.

Algorithm-2 (DLCS — Distributed Local Cliques Searching):

Suppose that the sensor network is m-covered and connected and each sensor node maintains an information table including IDNN (ID of Neighboring Node), NNLN (Neighboring Node List of Neighbors), and $\{IDNN\} \cup \{NNLN\}$, such as Table 4.2-1. The following algorithm serves the purpose of t-clique searching ($t \geq m$). For each node, say p_1, execute the steps in as List .

Remark

1) It is easy to know from the algorithm that any node in a t-clique will find out the same clique.
2) When s is much larger than m we can let the algorithm start from some number less than s to reduce the load of computation.

Theorem *4.2.2* Suppose a field is m-covered by a sensor network. Then each point of the field is covered by the sensor nodes in some t-Cliques found by Algorithm-2.

Proof: from Theorem 4.2.1, any point in an m-covered sensory field is covered by a t-clique ($t \geq m$). It is easy to check that the Algorithm-2 will find out all the t-clique in the network.

```
Step 1: let $p_2, p_3, \ldots, p_s$ be all the active neighbors
        of $p_1$ and$P_1, P_2, \ldots, P_s$ denote the IDs of
        $p_1, p_2, \ldots, p_s$ respectively. Let $N_1, N_2, \ldots, N_s$
        denote the NNLN and $\overline{N_1}, \overline{N_2}, \ldots, \overline{N_s}$ denote the
        $\{IDNN\} \cup \{NNLN\}$ of $p_1, p_2, \ldots, p_s$ respectively.

Step 2:
for ($t = s;\ t \geq m; t--$) do
      $p_1$ computes $\binom{s-1}{t-1}$ different subsets $S_1, S_2, \ldots, S_{\binom{s-1}{t-1}}$
      of $\{ p_1, p_2, \ldots, p_s \}$
      Let $C_x$ denote the set of all the $x$-cliques
      ($t < x \leq s$) that have been found by node $p_1$.

      for ($j = 1;\ j \leq \binom{s-1}{t-1}; j++$ ) do
          if there is no $c_x \in C_x$ such that $S_j \subseteq c_x$
          then
              let $S_j = S_j \cup \{p_1\}$
              let C = $\bigcap_{i \in S_j} \overline{N}_i$
              if $|C| = t$, record C as a t-clique end if
          end if
      end for
end for
```

List 4.2-1 Algorithm-2: Distributed Local Cliques Searching

We use the same scenario used before (Figure 4.2-2) to demonstrate the Algorithm-2.

Example 3: in Figure 4.2-2, node 001 has 5 neighbors $110, 010, 011, 101, 111$, which means that $s = 6$. Let $m = 3$. Table 4.2-1 shows the result of step 1 of Algorithm-2. The result of step 2 are as follows:

a) There does not exist a 6-clique
b) There does not exist a 5-clique
c) There is a 4-clique $\{001,011,111,101\}$
d) There are 4 2 subsets of neighbors $\{110, 010, 011, 101, 111\}$, that are not included in the found 4-clique. From these subsets, two 3-clique are found: {001,010,011} and {001,110,010}.

C. Distributed GIDs Assignment Scheme for m-covered Sensor Network

By combing Algorithm-1 and Algorithm-2, we present our new distributed GID assignment algorithm of nodes in the whole sensor network as follows:

Algorithm-3 (DGAS: Distributed GIDs Assignment Scheme):

Suppose that an m-covered and connected sensor network has n different sensor nodes, and each node maintains a information table including IDNN, NNLN, and $\{IDNN\} \cup \{NNLN\}$. Then these n sensor nodes can be allocated into k $(k \leq m)$ different groups $\{ 0, 1, \dots, k-1 \}$ in a distributed way according to the following steps:

```
Step 1: for each node $p_i$, run Algorithm-2 to find out all the t-clique
        $(m \leq t \leq s)$, where $s$ is the number of neighbors of $p_i$. Denote all the $t$-
        cliques as $C_t$.

Step 2: for $(t = s; t \geq m; t--)$ do
                for each $c_t \in C_t$ and $p_i$ has the smallest ID in $c_t$ do
                    Node $p_i$ assigns GIDs to all the nodes in $C_t$ using Algorithm-1.
                end for
            end for
```

Theorem *4.2.3* Suppose that the whole sensor network is m-covered and connected. After executing the *Algorithm-3*, each node in the sensor network will be assigned a single GID, which belongs to $\{ 0, 1, \dots, k-1 \}$.

Proof: Suppose p_1 is an arbitrary node in the network. Since the whole sensor network is *m*-covered and connected and the nodes that cover the point p_1 form a clique, there is at least one *t*-clique that includes node p_1, where $t > m$. This clique will be found in Step 1 of Algorithm-3. In Step 2, some node p_1 in the clique with smallest ID will assign a GID to p_1. According to Algorithm-1, each node only can be assigned a GID once.

D. Distributed Maintenance of Connectivity for each Group

In previous subsections we have given algorithms to evenly allocate the nodes in an *m*-covered and connected sensor network into k $(k \leq m)$ different groups in a distributed way. Next, we will consider the problem of connectivity for each group.

Definition *4.2.1* For any sensor node p_i let node p_j be a neighboring node of p_i. Suppose the minimum hops to the Sink from node p_i, p_i are H_i and H_j respectively. If $H_i = H_j + 1$ then node p_j is called an *Upstream Node* of p_i and p_i

is called a Downstream Node of p_j. If $H_i = H_j$, then node p_j is called a *Brother Node* of p_i. If p_j's NID is bigger than p_i's, then we call p_j an *Older Brother Node* of p_i and p_i is a *Younger Brother node* of p_j.

With the definition above, we give the steps of our connectivity maintenance scheme as follows:

Algorithm-4 (CMEG: Connectivity Maintenance for Each Group):

Suppose that the whole sensor network is connected, and each node maintains a list of all Upstream Nodes that are on its shortest paths to the Sink. After the GIDs assignment, each node in the sensor network will update its GID List to maintain the connectivity of the network. An initial GID List just contains one GID obtained from Algorithm-3.

Step 1: For a node p_i, suppose that its GIDs List is L_i. For any $g \in L_i$:

- Step 1.1: If there is neither a Upstream Node nor a Brother Node, which has g in its GID List, then p_i selects one of its Upstream Nodes, p_j, who has the shortest GIDs List (Of course, here we can also consider the surplus energy level of these nodes simultaneously), and sends an AGAC (Appended GID Application for Connectivity) message to p_j.
- Step 1.2: If there is no Upstream Node but there are Brother Nodes, who have g in their GID List, then p_i selects one of its Brother Nodes p_j, who has the shortest GIDs List (Of course, here we can also consider the surplus energy level of these nodes simultaneously), and sends an AGAC (Appended GID Application for Connectivity) message to p_j.

Step 2: After the GIDs assignment, each node in the sensor network maintains a back-off timer.

- Step 2.1: If the node p_j received AGAC messages from any of its Downstream Node or Younger Brother Node p_i before time out, then it will update its GID List according to the AGAC messages.
- Step 2.2: If the node p_j received AGAC messages from its any Older Brother Node p_i before time out, and node p_j finds out that it has no Upstream Node that has g in its GID List, then it will select one of its Upstream Nodes, p_x, who has the shortest GIDs List (Of course, here we can also consider the surplus energy level of these nodes simultaneously), and sends an AGAC (Appended GID Application for Connectivity) message to p_x.
- Step 2.3: After time out, it will broadcast its updated GIDs List to all of its neighbors.

The Algorithm-4 has the following useful result.

***Theorem** 4.2.4* Suppose that the whole sensor network is connected. Then after executing Algorithm-4, each node of the network has a path to the Sink, which consists of nodes with the same GID. I.e. the nodes in each group 0, 1... or *k-1* are connected.

Proof : For any given sensor node pi in the sensory field, let its shortest hops to Sink is H_i. We prove the theorem by induction.

1) When $H_i = 1$, it is obvious that node can communicate with Sink directly.
2) Let $H_i = 2$. Since the sensor network is connected, there exist some Upstream Nodes of p_i, whose hops to Sink is 1. If there is one of those Upstream Nodes that has a GID same as p_i's, then the conclusion follows. Otherwise,
    a) If there is no Brother Node that has a GID same as p_i's, then according to the step 1.1 of Algorithm-4, the node p_i will send an AGAC message to one of its Upstream Nodes, say p_j , and ask p_j to add an appended GID in its GID List. In any case, p_i will have at least one Upstream Node p_j with the same GID with p_i. So, the node p_i has a path of size 2 which consists of nodes with the same GID.
    b) If there is a Brother Node p_{o1} has a GID same as p_i's, then, for po1, if it has an Upstream Node with the same GID, it is obvious that the conclusion follows. Otherwise, if p_{o1} has no Older Brother Node, then according to the step 2.2 of Algorithm-4, it will select one of its Upstream Nodes p_x, and ask p_x to add an appended GID in its GID List. It means that the conclusion follows. If po1 has an Older Brother Node, say p_{o2}, which has a GID same as p_{o1}'s, through recursion and the step 1.2 of Algorithm-4, it is easy to know that the conclusion follows, since there exists certainly a node p_{on},, which will have no Older Brother Node.
3) Suppose the conclusion is true for $H_i = \text{d}$, where d > 2. For $H_i = d + 1$, if it has an Upstream Node that has a GID same as p_i's, or it has neither Upstream Node nor Brother Node, which has the same GID as p_i's, then the Algorithm-4 will find one Upstream Nodes of p_i with d hops to the Sink, which has a GID same as p_j. (This GID is either already existed or added by requesting). So by induction, the conclusion is true. Otherwise, if there is no Older Brother Node that has a GID same as p_i's, then according to the step 2.2 of Algorithm-4, pi will select one of its Upstream Nodes px with d hops to the Sink, which has a GID same as p_i. By induction, the conclusion is true. Finally, if there are Older Brother Nodes that have a same GID as pi's, then according to the step 1.2 of Algorithm-4, p_i will select an Older Brother Node, say p_{o1}, which has a GID same as p_i's. And then, by induction on p_{o1} the conclusion follows.

E. Connectivity and Coverage Maintenance Scheduling Algorithm

Now we are in a position to give our new connectivity and coverage maintenance scheduling algorithm as follows.

Algorithm-5 (CCMS: Connectivity and Coverage Maintenance Scheduling)

Suppose that a sensor network is m-covered and connected. Then all nodes in the sensor network can be scheduled into k different groups $\{0,1,\dots,k-1\}$, $k \leq m$, using the following steps:

Phase (1): GIDs Assignment

Step 1: Each sensor node p_i maintains a information table IT_i which includes information IDNN (ID of its Neighboring Nodes), NNLN (Neighboring Nodes List of its Neighbors), IUN (Identification of Upstream Node), TSHS (The Shortest Hops to the Sink), and the parameter k. Initially, IT_i is an empty table. The Sink broadcasts a Hello message and the parameter k. Each node that received this message record k will set TSHS as 1.

Step 2: Each node with non-empty TSHS broadcasts a Hello message including its ID, k and TSHS. For a node p_i, if it received a message including the TSHS_j and parameter k from node p_j , then it compares the TSHS_j with the records TSHS_i in its IT_i. If the TSHS_i is empty, then it records the ID of p_j (in IUN), the TSHS (= the received value plus 1) and parameter k into IT_i. If the TSHS_i is not empty, then it checks whether $\mathsf{TSHS}_i = \mathsf{TSHS}_j + 1$. p_j is recorded in IUN if the equation is true. (Here, we assume that the TSHS included in the message that arrived at p_i first will not be bigger than that arrived later.) p_i will also updates IDNN in IT_i when it receives messages.

Step 3: Each node generates and broadcasts a message including ID and IDNN. Through information exchanges among neighboring nodes, each node confirms its active neighboring nodes, and forms its NNLN in the table.

Step 4: Run the Algorithm-3 to assign GIDs.

Phase (2): Connectivity Maintenance

Step 5: Run the Algorithm-4 to update the connectivity.

Phase (3): Group Working

Step 6: In the working phase, all nodes work in turns at their given time slots. At the time slot t, if $t \equiv g \bmod k$, then all nodes in group g keep working, while other nodes will hibernate.

4.3 Performance Analysis

In this section, we analyzed the performance of our connectivity and coverage maintenance scheduling scheme (CCMS) theoretically. Some important performance aspects of scheduling scheme are evaluated for our algorithm. The comparison between our scheme and the previously introduced randomized scheduling scheme (RSGC)[10] is also highlighted in this section.

4.3.1 Coverage Performance

We need two lemmas to prove our theorem.

Lemma 4.3.1 Suppose k different colors are used to fill $t\ (k \leq t)$ spots randomly. Then the probability that these spots have $j\ (j < k)$ different colors is

$$\frac{1}{k^t}\sum_{i=0}^{j-1}(-1)^i\binom{k}{j}\binom{j}{i}(j-i)^t$$

Proof: There are total k^t different ways to use k colors to fill t spots. On the other hand, it is well-known that the number of onto functions from t elements to j elements is

$$\sum_{i=0}^{j-1}(-1)^i\binom{j}{i}(j-i)^t$$

And there are $\binom{k}{j}$ different ways to choose j colors from k colors. So the conclusion follows.

Define

$$P(t,k,j) = \frac{1}{k^t}\sum_{i=0}^{j-1}(-1)^i\binom{k}{j}\binom{j}{i}(j-i)^t \tag{1}$$

Then we have the following lemma.

Lemma 4.3.2

$$\sum_{j=1}^{\min\{k,t\}} P(t,k,j) = 1$$

Proof: the conclusion comes from Lemma 4.3.1
Using the second kind Stirling number

$$S(n,j) = \frac{1}{j!}\sum_{i=0}^{j-1}(-1)^i\binom{j}{i}(j-i)^n\,,$$

We can write

$$P(t,k,j) = \frac{j!}{k^t}\binom{k}{j} S(t,j) \ .$$

In the proofs below, we will also use the following formula about Stirling number.

$$S(n + 1, j) = S(n, j - 1) + jS(n, j) \tag{2}$$

***Theorem** 4.3.3* Suppose that the sensor network is m-covered and connected, and Algorithm GANC has been used to allocate the nodes into $k\ (k \leq m)$ different groups. If a given point p in the sensory field is covered by $t(t \geq m)$ sensor nodes, then the probability that at least k of the t sensor nodes having different GIDs is

$$P_{CCMS} = \begin{cases} 1 & if\ t_2 \geq k; \\ \dfrac{1}{k^{t_1}} \displaystyle\sum_{j=(k-t_2)}^{\min\{k,t_1\}} \sum_{i=0}^{j-1} (-1)^i \binom{k}{j}\binom{j}{i} (j-i)^{t_1} & if\ t_2 < k; \end{cases}$$

where $t_1 + t_2 = t$, and t_1 is the number of sensor nodes that have been assigned GIDs before applying Algorithm GANC in this t-Clique.

Proof: Since the point p is covered by t sensor nodes, the t nodes form a t-clique by Theorem 4.2.1. When Algorithm-1 is applied to this t-clique, the Clique Head will assign group numbers to the nodes. Let t_1 denote the number of sensor nodes that have been assigned GIDs before. If $t_2 \geq k$, then according to the Algorithm-1, the Clique Head will select k different sensor nodes randomly from these t_2 sensor nodes that have not been assigned GIDs before, and assign $\{0, 1, \dots, k-1\}$ randomly to these k nodes. So, the possibility that there exist at least k sensor nodes with different GIDs among these t sensor nodes will be 1 in this case.

When $t_2 < k$, by Lemma 4.1 the probability that t_1 sensor nodes that have been assigned GIDs before have j different GIDs is

$$\frac{1}{k^t}\sum_{i=0}^{j-1} (-1)^i \binom{k}{j}\binom{j}{i} (j-i)^t$$

Let

$$P(t_1,k,j) = \frac{1}{k^{t_1}}\sum_{i=0}^{j-1} (-1)^i \binom{k}{j}\binom{j}{i} (j-i)^{t_1}$$

then the probability that there exist k sensor nodes with different GIDs among these t sensor nodes will be $\sum_{j=(k-t_2)}^{\min\{k,t_1\}} P(t_1,k,j)$, because all the t_2 nodes that have not been assigned GIDs before will be assigned different GIDs according to step 2 of the Algorithm-1.

In [10], a randomized scheduling scheme was give. In this scheme, each sensor node will choose a random number from $\{0, 1, \dots, k-1\}$ as its GID. We call this method random GID assignment. We will refer their scheme as RSGC (Randomized Scheduling scheme with Guaranteed Connectivity). In a $t-clique$ $(t \geq k)$ of a sensor network, let P_{RSGC} denote the probability that there are k different GIDs in the random GID assignment. The following theorem compares the coverage of these two schemes.

***Theorem** 4.3.4* Suppose that the sensor network is m-covered and connected, and the nodes will be allocated into k $(k \leq m)$ different groups $\{0, 1, \dots, k-1\}$, Then the new node scheduling algorithm CCMS has better performance of coverage maintenance for each group of sensor nodes than the randomized scheduling scheme RSGC. Moreover, the value of $P_{RCCMS} - P_{RSGC}$ is

$$\begin{cases} \displaystyle\sum_{j=1}^{k-1} P(t,k,j) & if\ t_2 \geq k; \\ \displaystyle\sum_{i=0}^{t_2-1} \frac{1}{k^{t-i}}\left((k-i-1)(k-i-1)!\binom{k}{k-i-1} S(t-i-1, k-i-1)\right) & if\ t_2 < k; \end{cases}$$

Proof: From Theorem 4.3, we know that

$$P_{CCMS} = \begin{cases} 1 & if\ t_2 \geq k; \\ \displaystyle\frac{1}{k^{t_1}} \sum_{j=(k-t_2)}^{\min\{k,t_1\}} \sum_{i=0}^{j-1} (-1)^i \binom{k}{j}\binom{j}{i} (j-i)^{t_1} & if\ t_2 < k; \end{cases}$$

And from the number of onto function from t elements to k elements we have

$$P_{RSGC} = \frac{1}{k^t} \sum_{i=0}^{k-1} (-1)^i \binom{k}{i} (k-i)^t$$

When $t_2 \geq k$ we have $P_{RSGC} < P_{CCMS}$, since $P_{CCMS} = 1$ and $P_{RSGC} < 1$.
Now we consider the case of $t_2 < k$. In this case, we have

$$P_{RSGC} = 1 - \sum_{j=1}^{k-1} P(t,k,j)$$

and

$$P_{CCMS} = 1 - \sum_{j=1}^{k-t_2-1} P(t-t_2,k,j)$$

Define

$$f(x) = \sum_{j=1}^{k-x-1} P(t-x,k,j) = \sum_{j=1}^{k-x-1} \frac{j!}{k^{t-x}}\binom{k}{j} S(t-x,j)$$

Then $P_{RSGC} = 1 - f(0)$ and $P_{CCMS} = 1 - f(t_2)$.

We are going to calculate $f(0) - f(t_2)$. For a non-negative integer b, we have

$$\begin{aligned}
k^{t-b}f(b) &= \sum_{j=1}^{k-b-1} j!\binom{k}{j} S(t-b,j) \\
&= k + \sum_{j=2}^{k-b-1} j!\binom{k}{j} S(t-b,j) \\
&= k + \sum_{j=2}^{k-b-1} j!\binom{k}{j} (S(t-b-1,j-1) + jS(t-b-1)) \\
&= k + \sum_{j=2}^{k-b-1} j!\binom{k}{j} S(t-b-1,j-1) + \sum_{j=2}^{k-b-1} j!\binom{k}{j} jS(t-b-1,j) \\
&= k + \sum_{j=2}^{k-b-2} (j+1)!\binom{k}{j+1} S(t-b-1,j) + \sum_{j=2}^{k-b-1} j!\binom{k}{j} jS(t-b-1,j) \\
&= \sum_{j=2}^{k-b-2} (j+1)!\binom{k}{j+1} S(t-b-1,j) + \sum_{j=1}^{k-b-1} j!\binom{k}{j} jS(t-b-1,j) \\
&= \sum_{j=2}^{k-b-2} j!\,(k-j)\binom{k}{j} S(t-b-1,j) + \sum_{j=1}^{k-b-1} j!\binom{k}{j} jS(t-b-1,j) \\
&= \sum_{j=2}^{k-b-2} j!\,k\binom{k}{j} S(t-b-1,j) + \sum_{j=k-b-1}^{k-b-1} j!\binom{k}{j} jS(t-b-1,j) \\
&= \sum_{j=2}^{k-b-2} j!\,k\binom{k}{j} S(t-b-1,j) + (t-b-1)(k-b-1)!\binom{k}{k-b-1} S(t-b-1,k-b-1)
\end{aligned}$$

$$= k^{t-b}f(b+1) + (k-b-1)(k-b-1!)\binom{k}{k-b-1}S(t-b-1, k-b-1)$$

Therefore,

$$f(b) - f(b+1) = \frac{1}{k^{t-b}}((k-b-1)(k-b-1)!)\binom{k}{k-b-1}S(t-b-1, k-b-1)$$

Now we have

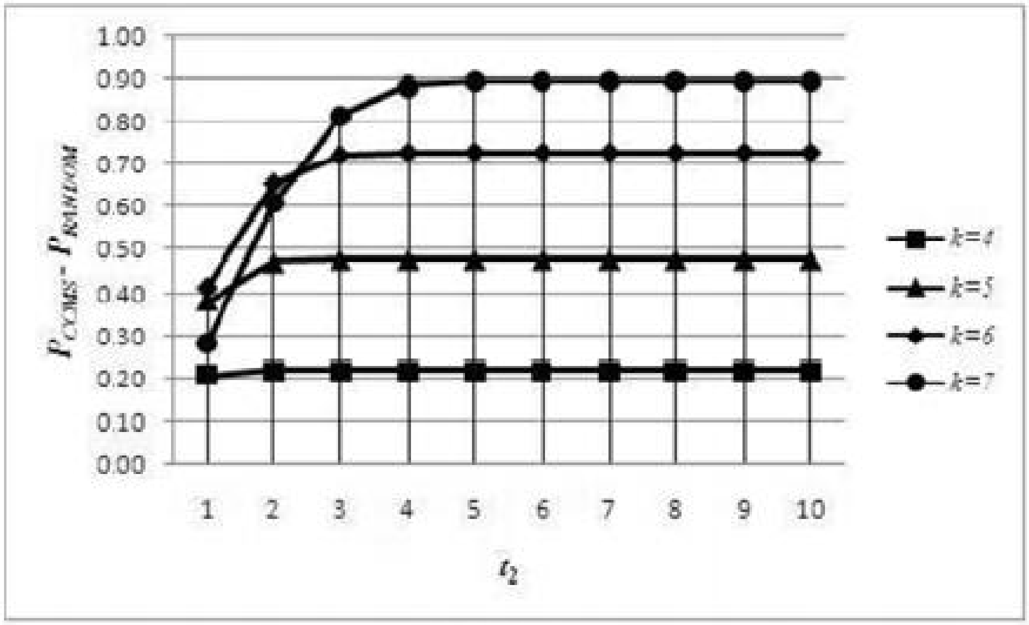
$$\begin{aligned} P_{CCMS} - P_{RSGC} &= f(0) - f(t_2) \\ &= \sum_{i=0}^{t_2-1}(f(i) - f(i+1)) \\ &= \sum_{i=0}^{t_2-1}\frac{1}{k^{t-i}}((k-i-1)(k-i-1!)\binom{k}{k-i-1}S(t-i-1, k-i-1)) \end{aligned}$$

In order to illustrate the results of Theorem 4.3.4, we give some examples below.

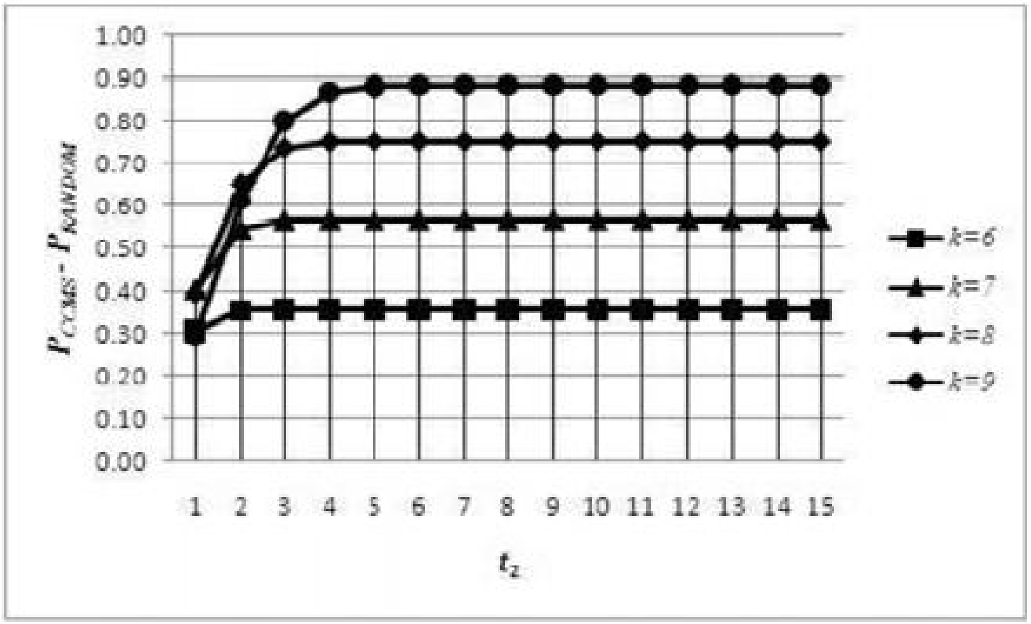
Example 5: From the above theorem 4.3.4, we see that the CCMS has better performance of GIDs assignment than the randomized scheduling scheme RSGC. In this example, we will give the Figure 4.3-1 and Figure 4.3-2 to show the degree of the improvement of CCMS, where we suppose that a given point p is covered by t different sensor nodes and all sensor nodes in the whole sensor network will be allocated into k different groups.

Figure 4.3-1 and Figure 4.3-2 give the results of "$P_{CCMS} - P_{RSGC}$ vs t_2" and " P_{CCMS} vs P_{RSGC}". From the two figures, it is obvious that the performance of coverage maintenance of CCMS is much better than that of the Randomized Scheduling Scheme RSGC. Especially from the Figure 4.3-2, we can see that the coverage ratio of the CCMS can reach to 100 percent in some cases, but that of the Randomized Scheduling Scheme RSGC is very low (about 20 to 30 percent) and cannot reach to 100 percent whatsoever.

In the Figure 4.3-3 and Figure 4.3-4, nodes are distributed in random or in Grid respectively, and in both situations, nodes are partitioned into three different groups by using CCMS and RSGC, where different groups are illustrated by different colors in these figures. From both Figure 4.3-3 and Figure 4.3-4,we can clearly see that after scheduling, comparing with RSGC, the nodes in each group are more evenly distributed in the whole sensory field when using the algorithm CCMS.

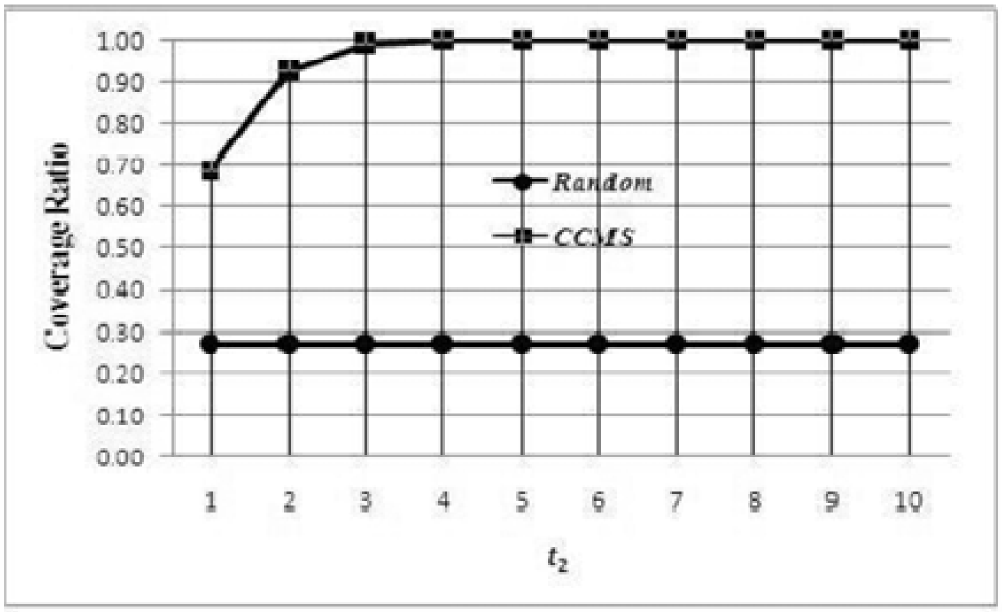


(a) $t = 10$

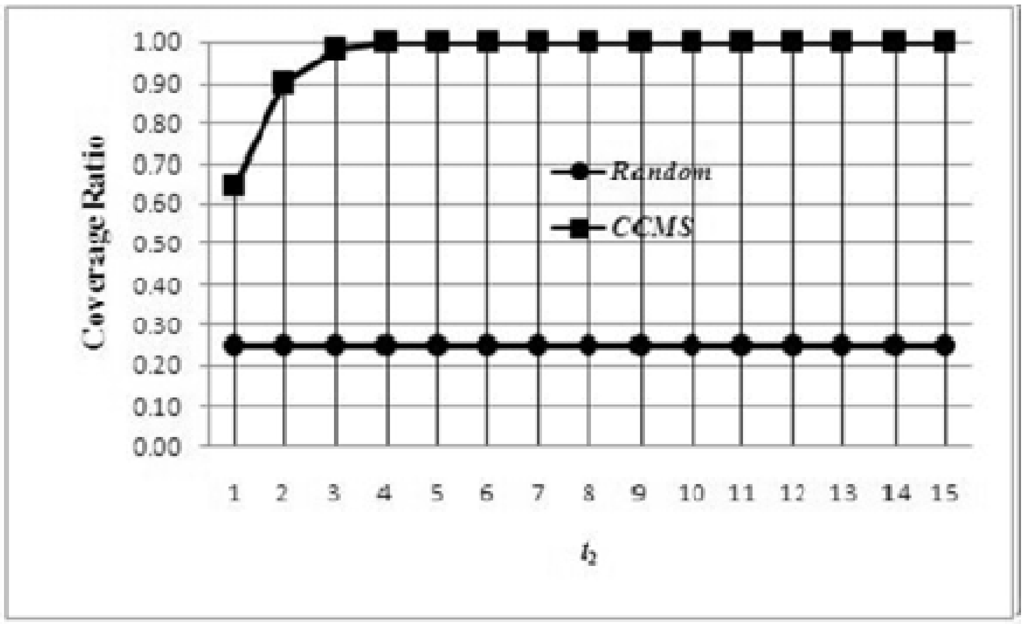


(b) $t = 15$

Figure 4.3-1 Improvement ratio of CCMS

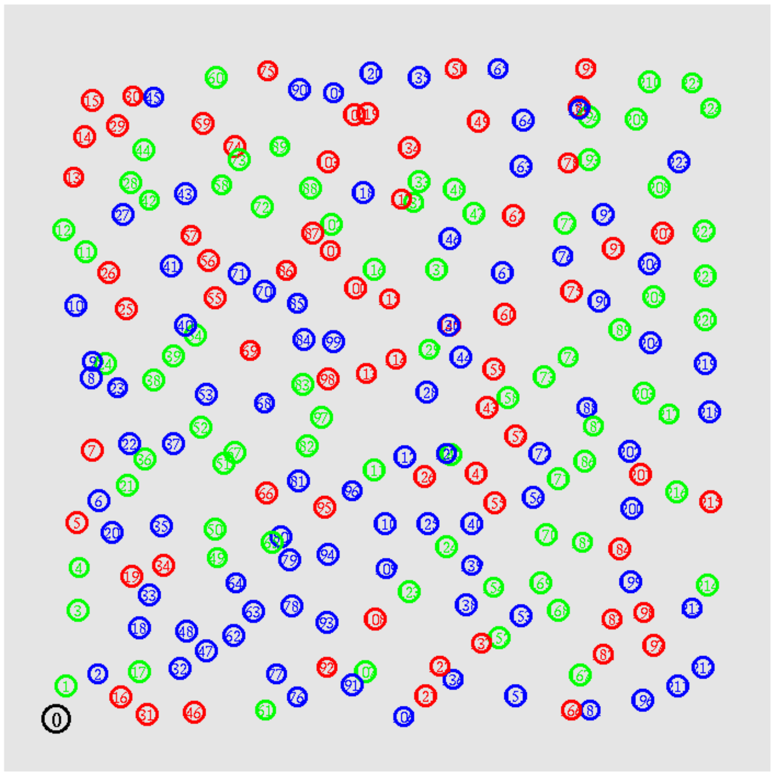


(*a*) t =10, k=6



(*b*) t =15, k=8

Figure 4.3-2 Comparison between CCMS and the Randomized Scheduling Scheme RSGC


(a) RSGC

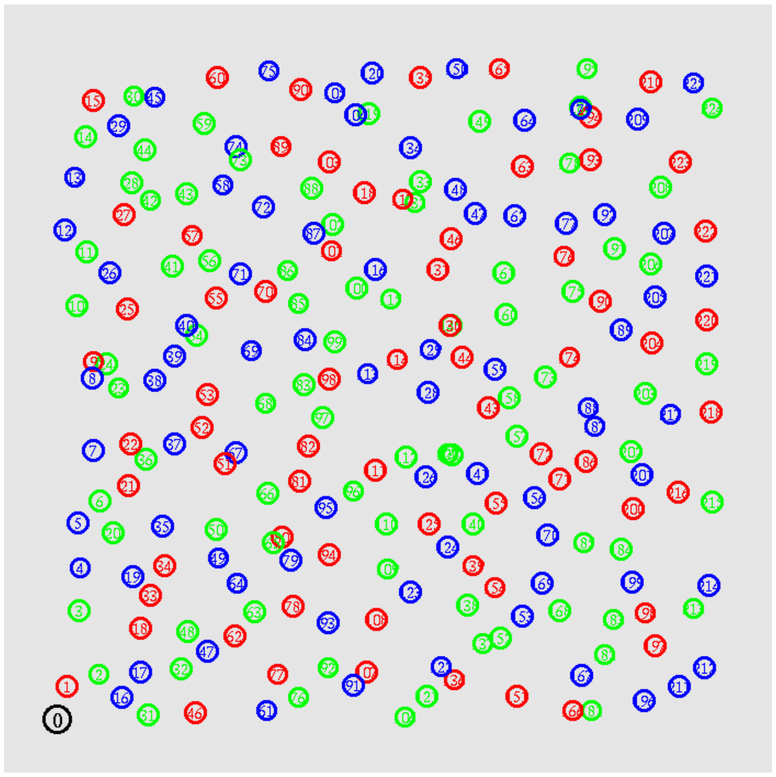
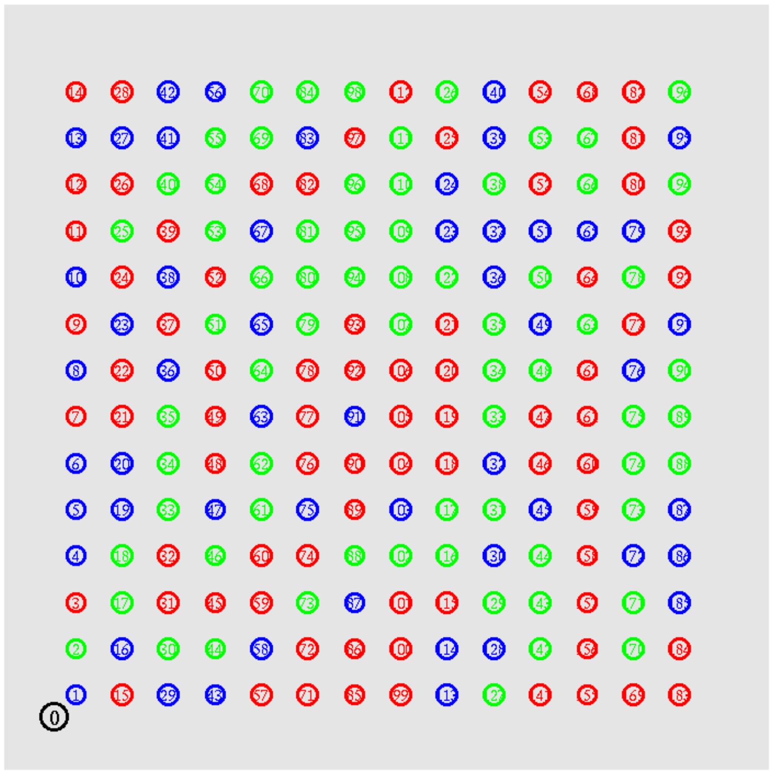
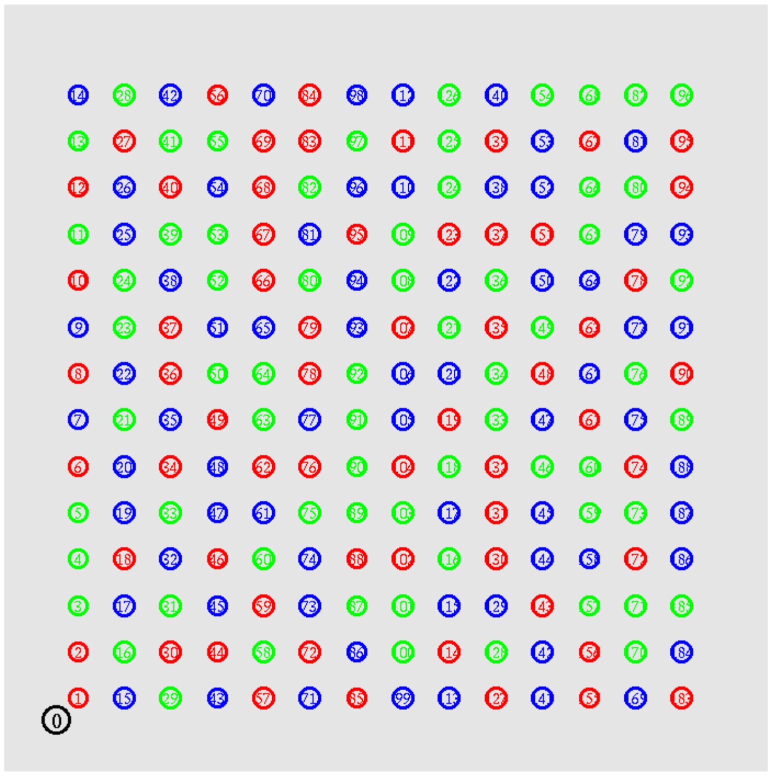

(b) CCMS

Figure 4.3-3 Comparison of Scheduling Results in Randomly Distributed Sensory Field



(a) RSGC



(b) CCMS

Figure 4.3-4 Comparison of Scheduling Results in Grid Distributed Sensory Field

4.3.2 Detection Performance

To consider the detection performance of a sensor network, we define a continuous detection rate (CDR) in time period l as follows.

$$R_{cd}(l) = \int_0^l \frac{1}{l} P(t)dt$$

where $P(t)$ is the probability that the sensor network can detect the event at time t. It is obvious that if a sensor network can detect (trace) the event all the time at a time period l, then the CDR will be 1. Otherwise the CDR will be a real number in [0, 1). The CDR reflects the ability of continuously detecting events of a sensor network.

Theorem *4.3.5* Let l denote the time duration of an event occurring in the sensory field. If at the time point t, there are s_t different sensor nodes that can detect this event, then, with the node scheduling algorithm CCMS, the R_{cd} during the time duration l can be calculated as:

$$R_{cd} = \int_0^l \frac{1}{l} \times \left\{ 1 - \left[1 - \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j) \right]^{st} \right\} dt\,,$$

where $P(s_t,k,j) = \frac{1}{k^{s_t}} \sum_{i=0}^{j-1} (-1)^i \binom{k}{j} \binom{j}{i} (j-i)^{s_t}$.

proof: From the Lemma 4.3.1, it is easy to know that P(s_t, k, j) represents the probability that there are j different GIDs in the s_t different sensor nodes. So, for any given sensor node a in these s_t different sensor nodes, the probability that a will be assigned a given GID g equals to

$$\sum_{j}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)\,,$$

which means that the probability that a will not be assigned the given GID g equals to

$$1 - \sum_{j}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)\,,$$

Therefore the probability that all of these s_t different sensor nodes will not be assigned the given GID g equals to

$$\left[1 - \sum_{j}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j) \right]^{s_t}\,,$$

which means that the probability that there exists at least one node in these s_t different sensor nodes that will be assigned the given GID g equals to

$$1 - \left[1 - \sum_{j}^{\min\{k, s_t\}} \frac{1}{j} \times P(s_t, k, j)\right]^{s_t}.$$

For example, in the gth time slot, the probability that there exists at least one node in these s_t different sensor nodes that will be assigned the given GID g equals to $1 - \left[1 - \sum_{j}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t, k, j)\right]^{s_t}$. It also means that in any given $g - th$ time slot, the probability that the event can be detected is

$$1 - \left[1 - \sum_{j}^{\min\{k, s_t\}} \frac{1}{j} \times P(s_t, k, j)\right]^{s_t}.$$

So, with the node scheduling algorithm CCMS, the P_{cd} during the time duration l can be calculated as:

$$R_{cd} = \int_0^l \frac{1}{l} \times \left\{1 - \left[1 - \sum_{j=1}^{\min\{k, s_t\}} \frac{1}{j} \times P(s_t, k, j)\right]^{st}\right\} dt$$

Theorem 4.3.6 Let l denote the time duration of an event occurring in the sensory field. Suppose at a time point t, there are s_t different sensor nodes that can detect the event. Let R_{cd}^{CCMS} denote the RCD, during the time duration l, by using the node scheduling algorithm CCMS, and R_{cd}^{RSGC} denote the RCD by using the node scheduling algorithm RSGC. Then we have $R_{cd}^{CCMS} \geq R_{cd}^{RSGC}$.

Proof: It is easy to prove that R_{cd}^{RSGC} can be calculated as follows:

$$R_{cd}^{RSGC} = \int_0^l \frac{1}{l} \times \left\{1 - \left[1 - \frac{1}{k}\right]^{s_t}\right\} dt.$$

So we need only to prove that

$$\sum_{j=1}^{\min\{k, s_t\}} \frac{1}{j} \times P(s_t, k, j) \geq \frac{1}{k}.$$

This can be proven as follows.

$$\sum_{j=1}^{\min\{k, s_t\}} \frac{1}{j} \times P(s_t, k, j) \geq \frac{1}{\min\{k, s_t\}} \times \sum_{j=1}^{\min\{k, s_t\}} P(s_t, k, j)$$

$$\geq \frac{1}{k} \times \sum_{j=1}^{\min\{k,s_t\}} P(s_t,k,j) = \frac{1}{k}$$

since $\sum_{j=1}^{\min\{k,s_t\}} P(s_t,k,j) = 1$ according to Lemma 4.3.2

The two theorems above has proven that the given node scheduling algorithm CCMS has good ability to detect moving objects in a sensory field. The next two theorems demonstrate the good ability of CCMS to detect the stationary objects.

***Theorem** 4.3.7* Let l denote the time duration of an event occurring at a point covered by s_t sensor nodes. Then, with the node scheduling algorithm CCMS, the probability that the event can be detected during the time duration l can be calculated as:

$$P_{cs} = \begin{cases} 1 & if\ l \geq k \times T; \\ \left[1 - \left(\frac{l}{T} - \left\lfloor\frac{l}{T}\right\rfloor\right)\right] \times \left\{1 - \left[1 - \left\lceil\frac{l}{T}\right\rceil \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)\right]^{s_t}\right\} + & \\ \left(\frac{l}{T} - \left\lfloor\frac{l}{T}\right\rfloor\right) \times \left\{1 - \left[1 - \left(\left\lceil\frac{l}{T}\right\rceil + 1\right) \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)\right]^{s_t}\right\} & Otherwise. \end{cases}$$

where T denotes the time duration of each time slot, i.e. the working duration of each group in one round of scheduling.

Proof : For an event with duration l, l _ k _T means that the event can last a time duration no less than k time slots. Since these s_t sensor nodes belong to at least one group in the k different groups, there exists at least one time slot during the duration l, in which the event can be detected by these s_t sensor nodes. So, if $l \geq k \times T$, we have $P_{cs} = 1$. For $l < k \times T$, obviously, the event can span either $\left\lceil\frac{l}{T}\right\rceil$ or $\left\lceil\frac{l}{T}\right\rceil + 1$ time slots. The probability that the event can span $\left\lceil\frac{l}{T}\right\rceil$ slots can be estimated as $1 - (\frac{l}{T} - \left\lfloor\frac{l}{T}\right\rfloor)$, and the probability that the event can span d $\left\lceil\frac{l}{T}\right\rceil + 1$ slots can be estimated as $\frac{l}{T} - \left\lfloor\frac{l}{T}\right\rfloor$. Furthermore, from the proof of theorem 4.3.5, we know that for any given sensor node a in these s_t different sensor nodes, the probability that a will be assigned a given GID g equals to $\sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)$. So the probability that a will be assigned one of these $\left\lceil\frac{l}{T}\right\rceil$ or $\left\lceil\frac{l}{T}\right\rceil + 1$ different GIDs is

$$\left\lceil \frac{l}{T} \right\rceil \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)$$

or

$$\left(\left\lceil \frac{l}{T} \right\rceil + 1\right) \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)$$

respectively. It means that the probability that a will not be assigned anyone of these $\left\lceil \frac{l}{T} \right\rceil$ or $\left\lceil \frac{l}{T} \right\rceil + 1$

$$1 - \left\lceil \frac{l}{T} \right\rceil \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)$$

or

$$1 - \left(\left\lceil \frac{l}{T} \right\rceil + 1\right) \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)$$

respectively. So, the probability that all of these s_t sensor nodes will not be assigned anyone of these $\left\lceil \frac{l}{T} \right\rceil$ or $\left\lceil \frac{l}{T} \right\rceil + 1$ different GIDs is

$$\left[1 - \left\lceil \frac{l}{T} \right\rceil \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)\right]^{s_t}$$

or

$$\left[1 - \left(\left\lceil \frac{l}{T} \right\rceil + 1\right) \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)\right]^{s_t}$$

respectively. Therefore for these s_t sensor nodes, the probability that there exists at least one node that will be assigned one of these $\left\lceil \frac{l}{T} \right\rceil$ or $\left\lceil \frac{l}{T} \right\rceil + 1$ different GIDs is

$$1 - \left[1 - \left\lceil \frac{l}{T} \right\rceil \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)\right]^{s_t}$$

or

$$1 - \left[1 - \left(\left\lceil \frac{l}{T} \right\rceil + 1\right) \times \sum_{j=1}^{\min\{k,s_t\}} \frac{1}{j} \times P(s_t,k,j)\right]^{s_t}$$

respectively. We have proved beforehand that the probability that the event can span $\left\lceil\frac{l}{T}\right\rceil$ can be estimated as $1-(\frac{l}{T}-\left\lfloor\frac{l}{T}\right\rfloor)$, and the probability that the event can span $\left\lceil\frac{l}{T}\right\rceil + 1$ can be estimated as $\frac{l}{T}-\left\lfloor\frac{l}{T}\right\rfloor$. So we get the conclusion below.

***Theorem** 4.3.8* Let l denote the time duration of an event occurring at a point covered by s_t sensor nodes. Let P_{cs} denote the probability that the event can be detected during the time duration l by using the node scheduling algorithm CCMS, and P_{rs} denote the probability that the event can be detected during the time duration l by using the node scheduling algorithm RSGC, then $P_{cs} \geq P_{rs}$.

Proof: we have

$$P_{rs} = \left[1-(\frac{l}{T}-\left\lfloor\frac{l}{T}\right\rfloor)\right]\times\left\{1-\left[1-\left\lceil\frac{l}{T}\right\rceil\times\frac{1}{k}\right]^{s_t}\right\}+\left(\frac{l}{T}-\left\lfloor\frac{l}{T}\right\rfloor\right)\times\left\{1-[1-\left(\left\lceil\frac{l}{T}\right\rceil+1\right)\times\frac{1}{k}]^{s_t}\right\}$$

We have proved in the proving process of Theorem 4.3.6 that $\sum_{j=1}^{\min\{k,s_t\}}\frac{1}{j}\times P(s_t,k,j) \geq \frac{1}{k}$. It is easy to see that $P_{cs} \geq P_{rs}$.

4.3.3 Analysis on Communication Overheads

Next, we will furthermore evaluate the system communication overhead of our scheduling scheme CCMS in terms of the average number of broadcasts from individual sensor node.

Step 1: No broadcast is needed for sensor nodes except Sink in this step.

Step 2: All sensor nodes need 1 round of information exchange with all its neighbors to confirm its active neighboring nodes in this step.

Step 3: In this step, for each node pi, if it received a new message including the TSHS and parameter k from its neighbors for the first time, then, it needs to exchange 1 round of information IITP with all its active neighboring nodes.

Step 4: In this step, by combing with the Algorithm-3 and Algorithm-2 and Algorithm-1, we can know that, each node needs only to communicate with its Clique Head. So, no broadcast is needed in this step.

Step 5: In this step, by combing with the Algorithm-4, we can know that, each node need to know the GIDs List of its Upstream Nodes. So, 1 round of broadcast of GIDs List is needed in this step.

Step 6: No broadcast is needed in this step.

Therefore, 3 rounds of broadcast are needed in our node scheduling scheme CCMS, which are distributed in the Step 2, Step 3, and Step 5 respectively.

And additionally, we will also evaluate the system storage overhead of our scheduling scheme CCMS as follows:

1) we can know that each node needs to maintain a information table IT consisting of the following information: IDNN, NNLN, IUNN, TSHS, and the parameter k. For any given node in the whole sensory field, let N_{max} denote the maximum number of neighboring nodes, H_{max} denote the maximum hops to Sink, and N denote the number of sensor nodes in the whole sensor network, since we encode the nodes according to an n-dimensional Hypercube, then it is obvious that the length of ID of each node is $\log N$. So, the storage overhead of the information table IT will be no more than $[N_{max} \cdot \log N + N_{max}{}^{2} \cdot \log N + \log H_{max} + \log k]$.
2) for each node, it needs to store the GIDs List after the GIDs assignment, and it is easy to know that the storage overhead of GIDs List will be no more than $[k \cdot \log k]$. Since there are $k \leq m$ and $m \leq N_{max} < N$, we can estimate the storage overhead of CCMS as $O(N_{max}{}^{2} \cdot \log N)$, where N_{max} denote the maximum number of neighboring nodes, and N denote the number of sensor nodes in the whole sensor network.

4.4 Simulation Results

Our simulation is run with Network Simulator version 2 (NS2). We utilized the wireless network extension and the energy model of NS2 to cope with our sensor network environment. Our new scheduling protocol CCMS was implemented as an extension to NS2. The other scheduling protocol RSGC is also implemented for comparison purpose. The simulation mostly concentrates on the following features of the WSN after scheduling:

1) Coverage Ratio: we compare the coverage intensity of the two algorithms in different network density and k (the number of groups).
2) Lifecycle of the Network: we compare the network lifetime of the two algorithms in different network density and k (the number of groups). In the simulation, for any group, if the nodes in the group are not connected any more or the coverage ratio of the group is below 95%, then we regard the group as a dead group.

In simulations, totally N sensor nodes are deployed randomly in a $250m \times 250m$ square sensory field, and for each node, the sensing range r_s is set to be 15, and the communication range r_c is set to be $2r_s$. The simulation results are shown in Figure 4.4-1 to Figure 4.4-3, and all the data in these figures are the average of 10 to 20 times independent simulation results.

4.4.1 Group Coverage Maintenance Performance

Suppose that there are N nodes deployed evenly in a sensory field, which will be partitioned into k different groups by applying the node scheduling schemes. In order to estimate the coverage maintenance performance of each group after applying the node scheduling algorithms, for each group g $(1 \le g \le k)$, we define the Group Coverage Maintenance Ratio R_{GCM} of the group g as follows:

$$R_{GCM}{}^{g} = \frac{Area(g)}{Area[{}^{N}/_{k}]}$$

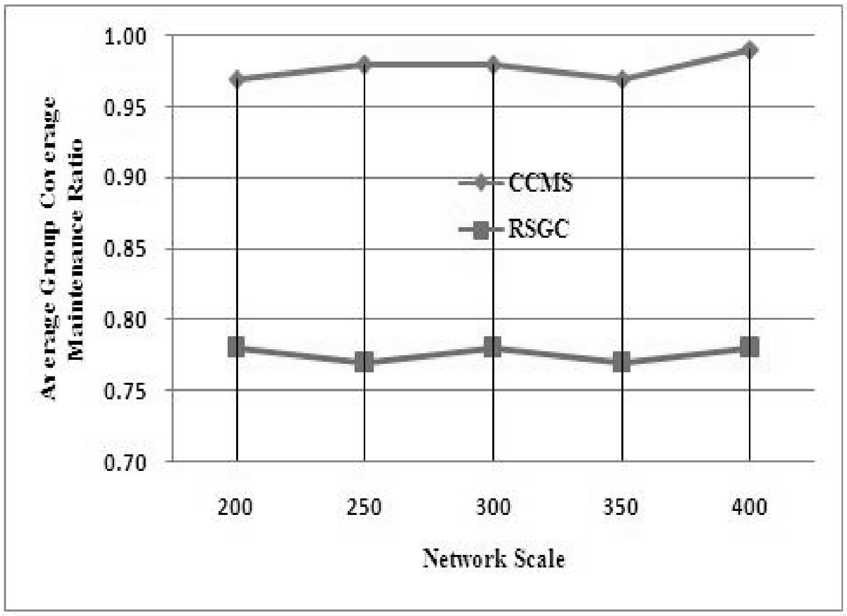
where $Area(g)$ denotes the area covered by the nodes of the group g, and $Area_{\lceil N/_k \rceil}$ denotes the area covered by any $\lceil N/k \rceil$ nodes distributed evenly in the whole sensory field.

Based on the above definition of Group Coverage Maintenance Ratio, we define the Average Group Coverage Maintenance Ratio R_{AGCM} for a node scheduling scheme as follows:

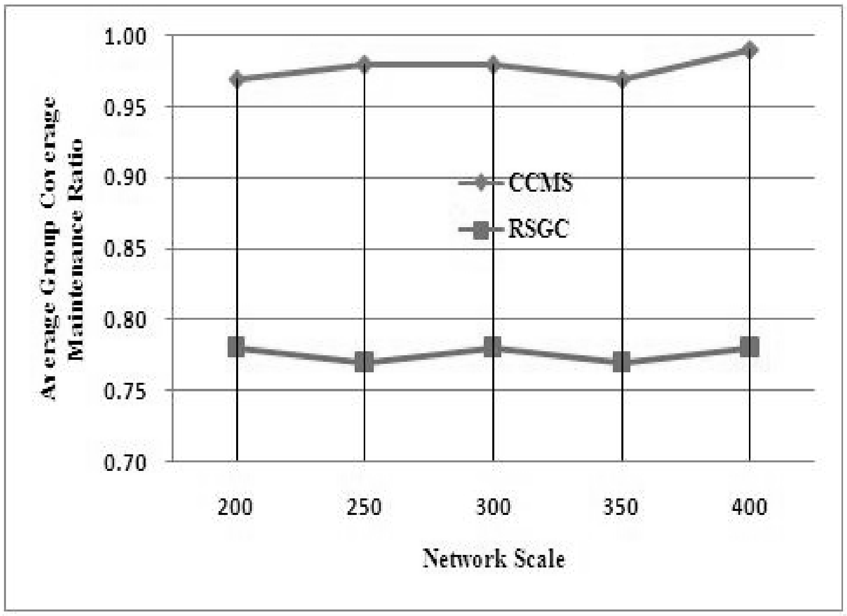
$$R_{AGCM} = \frac{\sum_{i=1}^{k} R_{GCM}{}^{i}}{k}$$

To compare Average Group Coverage Maintenance Ratios, we deployed 200 to 400 nodes evenly in a $250m \times 250m$ square sensory field. The following Figure 4.4-1 illustrates the comparisons of "Average Group Coverage Maintenance Ratio vs Network Scale" between the two algorithms CCMS and RSGC for k = 2 and k = 3 respectively.

From the Figure 4.4-1, it is easy to see that the Average Group Coverage Maintenance Ratio of CCMS is greater than or close to 95%, but that of RSGC is about to 75% only. So CCMS has much better node scheduling performance than RSGC, and using CCMS, each group has much better coverage Ratio in different kinds of network scales.



(a) $k = 2$



(b) $k = 3$

Figure 4.4-1 Average Group Coverage Maintenance Ratio for RSGC and CCMS

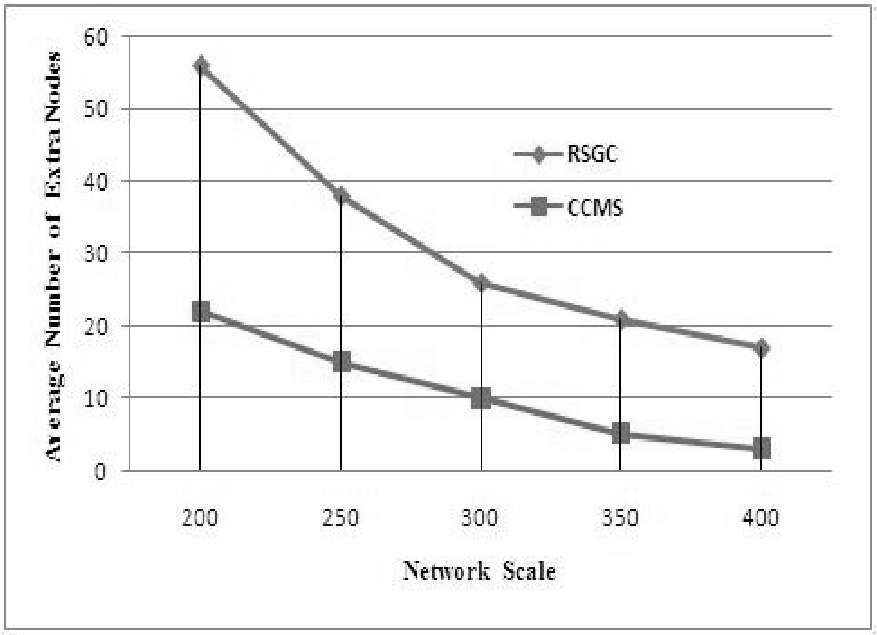
4.4.2 Extra nodes added by using the CMEG Algorithm

From Figure 4.3-3 and Figure 4.3-4, we have found that the algorithm CCMS can guarantee that the nodes in each group are more evenly distributed in the whole sensory field. In the next Table 4.4-1, we give the simulation result of the number of nodes for each group before applying the connectivity maintenance algorithm when using CCMS and RSGC respectively.

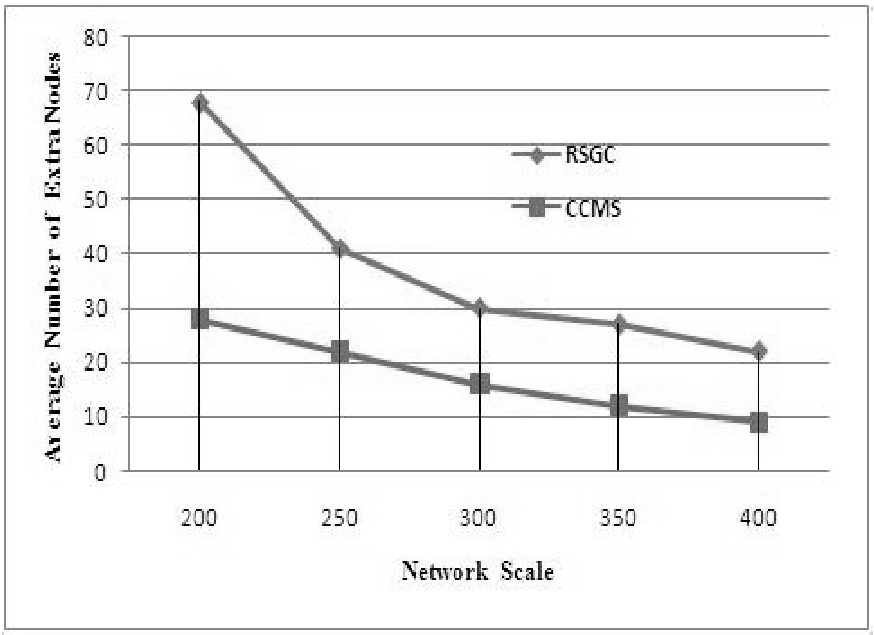
Number of Groups	Number of Nodes in Each Group RSGC	Number of Nodes in Each Group CCMS
K=2	{145,155}	{149,151}
K=3	{98,92,110}	{98,101,101}
K=4	{62,76,76,85}	{73,73,76,78}

Table 4.4-1 Number of Nodes for Each Group before Applying the Connectivity Maintenance Algorithm

From the above Table 4.4-1, we see that the CCMS can partition nodes more evenly into different groups, i.e., after node scheduling by using CCMS, each group will have more similar number of nodes than using RSGC. And obviously, extra nodes may be needed to guarantee the connectivity of the nodes in each group after the nodes are partitioned into different groups. The next Figure 4.4-2 illustrates the comparison of the average number of extra nodes added for each group by applying the connectivity maintenance algorithm CMEG used in CCMS and the connectivity maintenance algorithm used in RSGC.



(a) k = 3



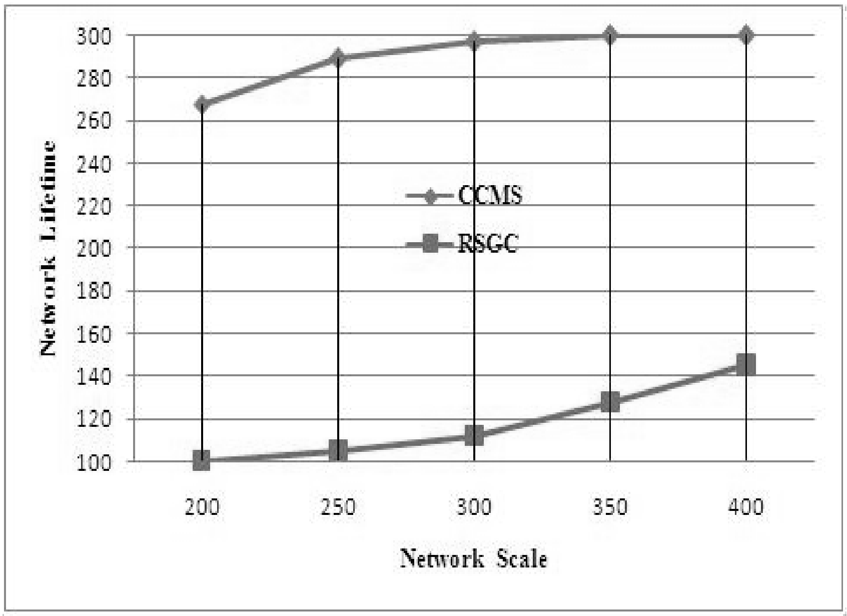
(b) k = 4

Figure 4.4-2 Average Number of Extra Nodes added for Each Group

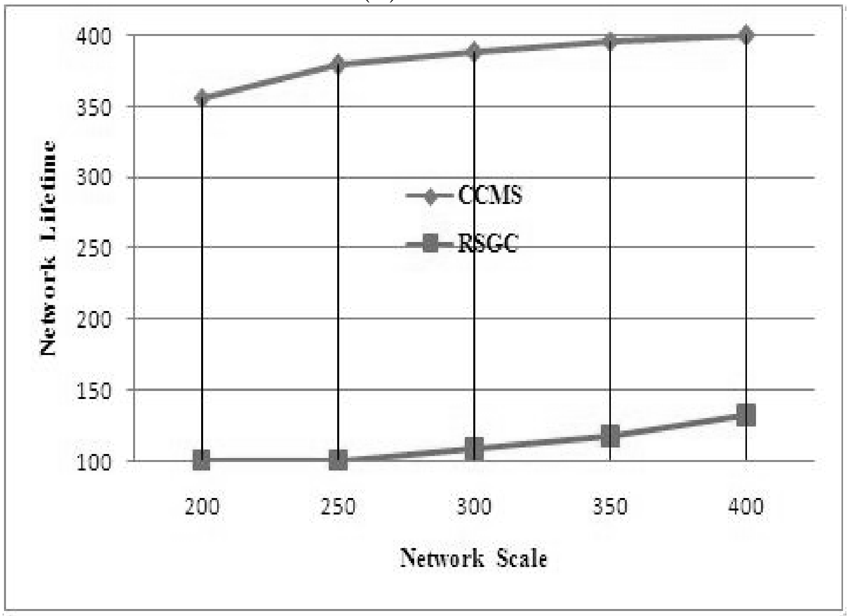
From the above Figure 4.4-2, it is easy to see that, compared with RSGC algorithm. CCMS needs less extra nodes to maintain the connectivity for each group.

4.4.3 Network Lifetime

In order to see the actual working performance of CCMS, next, we illustrate comparisons of the network lifetime of CCMS and RSGC in the following Figure 4.4-3, where each node has been assigned 100 units energy initially. The energy consumption of a working node is set to be 1 unit per time. Furthermore, in the following simulation, we consider the energy consumption in the working period only, and the energy consumption in the period of node scheduling is not included since the process of node scheduling will be executed only once. We note here that it is obvious that the energy consumption of CCMS in the process of node scheduling is larger than that of RSGC, considering that the process of node scheduling of CCMS is more complicated than that of RSGC.



(a) k = 3



(b) k = 4

Figure 4.4-3 Comparison of Network Lifetimes by Applying CCMS and RSGC

The above Figure 4.4-3 illustrates the comparisons of "Network Scale vs Network Lifetime between the two algorithms CCMS and RSGC when $k = 3$ and $k = 4$ respectively. From the Figure 4.4-3, we can see that the CCMS can improve the network lifetime significantly, but the RSGC only can improve a little bit of the network lifetime. According to the analysis, the reasons are: first, RSGC cannot guarantee the nodes are divided evenly into each group and furthermore, cannot guarantee the nodes in each group are evenly distributed in the whole sensory field either. So it cannot guarantee the coverage ratio for each group after scheduling. Second, the connectivity maintenance algorithm of RSGC requires each node to maintain a shortest path to Sink, which will leads to a relative larger number of extra nodes needed to maintain the connectivity of nodes in each group. So there are a lot of nodes that will belong to more than one group.

4.5 Conclusion

Through the performance analysis and the Simulations results we can clearly see that our Clique-based Connectivity Maintenance Scheduling Scheme did have a major impact on the lifetime of the scheduled wireless sensor network. Meanwhile, it retains the same level of coverage as using all of the sensors and guarantees the connectivity of the network even after some of the sensors have run out of power. In addition, this algorithm does not require the location information of the sensors and can monitor the whole sensory field instead of known targets without having to introduce more constraints to the system.

In the comparison with the other scheme Randomized Scheduling with Guaranteed Connectivity, we can tell our proposed scheme outperform the RSGC scheme in all three major perspective: coverage, number of extra-on nodes for connectivity, and lifetime. The only advantage of RSGC is the speed of execution. So in next chapter we proposed another scheduling scheme for WSN, a new approach is used for group assignment, instead of local clique. Without having to find all of the local cliques, the execution speed will be improved with acceptable performance drop from CCMS.

Chapter 5

Second Proposal: An Efficient Node Scheduling Scheme Based on Combinatorial Assignment Code

In this chapter, we will present another sensor scheduling scheme design of ours. This work is almost a completely different approach from the previous chapter in terms of the GIDs assignment. The central concept of this new scheme comes from combinatorics. We used the special characteristic of a combinatorial code, and we named it combinatorial assignment code (abbreviated as CAC). The algorithm is built on top of CAC, so we named the new scheme as CNSA (CAC based Node Scheduling Algorithm). With the help of CAC, sensors can be evenly distributed into sets which are able to monitor the sensory field individually in turns, and thus extend network's lifetime. Compare to CCMS, advantage of this new approach is that the searching for local cliques is not required, so the execution time will be improved significantly. And likewise to CCMS, no location information is required by CNSA.

In this chapter, we first introduce the CAC and it feature. Afterwards, we provide the detailed algorithms of CNSA. In the third section, a brief theoretical performance analysis is given, just to establish that CNSA is superior to RSGC on coverage performance. As this work is still on-going, a more thorough performance analysis and simulations are yet to be done. We don't have much simulation results to show at the time of writing this thesis. The work in this chapter is a basis of our future work.

5.1 Combinatorial Assignment Code

Definition *5.1.1* Let $m \geq k$, an (m, k) combinatorial assignment code (CAC) is set system $(X, \mathfrak{B})$, where $X=\{x_1,x_2,\ldots,x_m\}$ is a set of m elements (called *items*), $\mathfrak{B}=\{B_1,B_2,\ldots,B_k\}$ is a collection of k subsets of X (called *containers*), such that for each k-subset $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\} \subset X$ there exists an item $b_i \in B_i$ such that $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\} = \{b_1,b_2,\ldots,b_k\}$, where $|B_i|$ denotes the number of items in the subset B_i. Let $N=\sum_{i=1}^{k}|B_i|$, then N is called the *size* of the (m, k) combinatorial assignment code, and for convenience, we record an (m, k) combinatorial assignment code with size of N_i as (m,N_i,k)–CAC.

Definition *5.1.2* An (m, k) combinatorial assignment code (m,N_{min},k)-CAC is called the minimum (m, k) combinatorial assignment code $\Leftrightarrow \forall$ (m,N_i,k)-CAC, there is $N_{min} \leq N_i$.

Theorem *5.1.1* If (m,N_{min},k)-CAC is the minimum (m, k) combinatorial assignment code, then there is $N_{min}=km\text{-}k(k\text{-}1)$.

Proof: Let $B_1=\{x_1,x_2,\ldots,x_{m-k+1}\}$, $B_2=\{x_2,x_3,\ldots,x_{m-k+2}\}$, $\ldots,B_k=\{x_k,x_{k+1},\ldots,x_m\}$, then it is easy to check that $\{B_1,B_2,\ldots,B_k\}$ is an (m, k) combinatorial assignment code with size of m. So there is $N_{min} \leq k(m\text{-}k+1)=km\text{-}k(k\text{-}1)$. On the other hand, if $N_{min}<km\text{-}k(k\text{-}1)$, then there exists at least one container that has no more than m-k items. So the k items missing from that container cannot be covered.

Definition *5.1.3* Given a set system $(X, \mathfrak{B})$ with $X=\{x_1,x_2,\ldots,x_m\}$ and $\mathfrak{B}=\{B_1,B_2,\ldots,B_k\}$, the incidence matrix of $(X, \mathfrak{B})$ is the $k \times m$ matrix $A=(a_{ij})$, where

$$a_{ij} = \begin{cases} 1: \text{if } x_j \in B_j \\ 0: \text{if } x_j \notin B_j \end{cases}.$$

According to the definition 5.1.3, the actual $k \times m$ incidence matrix $A = (a_{ij})$ can be illustrated as the follows:

$$\begin{array}{c c} & \begin{array}{cccc} x_1 & x_2 & \cdots & x_m \end{array} \\ \begin{array}{c} B_1 \\ B_2 \\ \cdots \\ B_k \end{array} & \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{k1} & a_{k2} & \cdots & a_{km} \end{bmatrix} \end{array}$$

On the basis of definition 5.1.3, we have the following lemma 5.1.1:

***Lemma** 5.1.1* An $k \times m$ 0-1 matrix $A = (a_{ij})$ is the incidence matrix of an (m, k) combinatorial assignment code (m, N, k)-CAC, then in each row of A, there exist at least m-k+1 cells that all contain the entry 1.

Proof: If in one row of A, there exist no more than $m - k$ cells that all contain the entry 1, then it is obvious that the corresponding container will has no more than m-k items. So the k items missing from that container cannot be covered, which is contradict to the definition of the (m, k) combinatorial assignment code defined in definition 1.

Based on the above lemma 5.1.1, we can prove the following theorem.

***Theorem** 5.1.2* An $k \times m$ 0-1 matrix $A = (a_{ij})$ is an incidence matrix of the minimum (m, k) combinatorial assignment code (m, N_{min}, k)-CAC $\Leftrightarrow$ for any k columns of A, the $k \times k$ sub-matrix has a set of k cells in different rows and different columns that all contain the entry 1, and furthermore, in each row of A, there are just $m - k + 1$ cells that all contain the entry 1.

Proof: (Necessity): At first, from lemma 5.1.1 we know that each row of A shall have at least $m - k + 1$ cells that all contain the entry 1. So, if each row has just m-k+1 cells that all contain the entry 1, then the total number of 1s in A is $k(m - k + 1) = km - k(k - 1)$, otherwise, the total number of 1s in A will be more than $km - k(k - 1)$. But according to theorem 1, there is $N_{min} = km - k(k - 1)$. Therefore, it is easy to know that each row of A can have just $m - k + 1$ cells that all contain the entry 1. Secondly, if there are k columns of A such that the corresponding $k \times k$ sub-matrix does not have a set of k cells in different rows and different columns that all contain the entry 1, then it is obvious that the k items corresponding to the $k \times k$ sub-matrix cannot be covered by these k containers, which is contradict to the assumption that (m, N_{min}, k) −CAC is an (m, k) combinatorial assignment code. So, for any k columns of A, the $k \times k$ sub-matrix has a set of k cells in different rows and different columns that all contain the entry 1.

(Sufficiency): Since in each row of A, there are just $m - k + 1$ cells that all contain the entry 1, then the total number of 1s in A is $k(m - k + 1) = km - k(k - 1)$, so there is $N_{min} = \mathrm{km} - \mathrm{k}(\mathrm{k} - 1)$. And on the other hand, since for any k columns of A, the $k \times k$ sub-matrix has a set of k cells in different rows and different columns that all contain the entry 1, which means that for any given k items, each item can be covered by one of these k containers respectively, then, according to the definition 1, it is easy to know that the (m, N_{min}, k)-CAC is an (m, k) combinatorial assignment code. So, based on the definition 2 and theorem 1, the conclusion follows.

From the above theorem 2, we can know also that for any give m and k, the minimum (m, k) combinatorial assignment code is not unique. For example, Let $m = 7$ and $k = 4$, then the following two matrices A_1 and A_2 are incidence matrices of two different minimum (m, k) combinatorial assignment codes respectively.

$$A_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

According to the proof steps of the above theorem 2, we have the following lemma 2 also:

Lemma *5.1.2* An $k \times m$ 0-1 matrix $A = (a_{ij})$ is an incidence matrix of an (m, k) combinatorial assignment code (m, N_{min}, k)-CAC $\Leftrightarrow$ for any k columns of A, the $k \times k$ sub-matrix has a set of k cells in different rows and different columns that all contain the entry 1.

5.2 CAC-based Scheduling Scheme

5.2.1 Network Model and Introduction to the Scheme

We assume stationary sensor networks in a two-dimensional field and assume that sensor nodes are randomly and independently deployed in a field.

Next, we assume that a sensor's sensing range, which is denoted by r_s is defined as the range beyond which the sensor's sensing ability can be neglected. Furthermore, we assume that sensors' communication range r_c is larger than or equal to $2r_s$ which is usually true in practice. For example, ultrasonic sensors have a sensing range of approximately 0.2 to 6 meters while the transmission range of MICA motes is about 30 meters.

Finally, before deployment, we assume that all of the sensor nodes are encoded from $\underbrace{00 \ldots 001}_{n}$ to $\underbrace{11 \ldots 111}_{n}$according to the coding method of the n-dimensional Hyper-cubes, and encode the Sink as $\underbrace{00 \ldots 000}_{n}$ especially. So after deployment, each sensor node will have a unique ID randomly chosen from $\{\underbrace{00 \ldots 001}_{n} \ldots \underbrace{11 \ldots 111}_{n}\}$ beforehand, and the Sink will have an ID as $\underbrace{00 \ldots 000}_{n}$.

On the basis of the above assumptions, the main ideas of our new CAC based Node Scheduling Scheme can be described as follows:

Step1: On the basis of the network model, partition all the nodes in the whole sensor network into clusters through a distributed way.

Step2: By utilizing the CAC model, allocate the nodes in each cluster into the k groups $\{0,1,...,k\text{-}1\}$evenly without requiring the location information of sensor nodes.

Step3: According to some selection rules, each node selects a suitable group ID from its list of group IDs.

Step4: Each node in the whole sensor network executes the connectivity maintenance algorithm independently to guarantee the connectivity for each group.

Step5: Nodes in each group work according to the new working schedule.

5.2.2 Distributed Cluster Approach

Definition *5.2.1* For any m sensor nodes p_1, p_2,..., p_m, if they are all neighboring to each other, then the set { p_1, p_2,..., p_m} is called a *Cluster of Neighbors.*

From the above definition 4, obviously we can know that for any node p_i, it can compute out a unique *Cluster of Neighbors* (although it may belong to more than one *Cluster of Neighbors*, but it can compute out one only). For convenience, we record the *Cluster of Neighbors* computed out by node p_i as CN_i.

Definition *5.2.2* : For any two sensor nodes p_1 and p_2, if the ID of p_1 is bigger than that of the p_2, then we record their relationship as $p_1 > p_2$.

On the basis of the above definition 4 and definition 5.2.1, we can illustrate the *Distributed Clustering Method* (algorithm *DCM*) as follows:

Step1: For each node p, through information among neighboring nodes, node p finds out all its neighboring nodes, and computes out its *Cluster of Neighbors* according to the definition 4.

Step2: If among all the nodes with empty *List of Group IDs* (Obviously, at the initial period, all nodes haven't been assigned any Group IDs yet, then the *List of Group IDs* for each node in the whole sensor network is empty) in its *Custer of Neighbors*, node p has the minimum ID, then node p declares itself as the *Cluster Head* of its *Cluster of Neighbors.*

5.2.3 CAC-based Node Scheduling Scheme for node in cluster of neighbors.

The algorithm *DCM* can partition all the nodes in the whole sensor network into clusters in a distributed way, supposing that there are m sensor nodes in a given *Cluster of Neighbors*, then we need to solve the following problem that how to allocate these m sensor nodes in the *Cluster of Neighbors* into k different groups evenly, and at the same time, we shall guarantee that each group can maintain the coverage ratio as high as possible.

Before we give the detailed solution to the above problem, we give the following lemma 5.2.1 at first.

***Lemma** 5.2.1* Supposing that a given point p inside the monitored sensory field is covered by m sensor nodes $\{p_1,p_2,...,p_m\}$, then, for any node $p_i \in \{p_1,p_2,...,p_m\}$, the node set $\{p_1,p_2,...,p_m\}$ is in its *Cluster of Neighbors*, and furthermore, each pair of nodes in $\{p_1,p_2,...,p_m\}$ are neighboring to each other.

Proof: For any sensor node p_j in $\{p_1,p_2,...,p_m\}$, since there are $d(p_j, p) \le r_s$ and $d(p_i, p) \le r_s$, then $d(p_i, p_j) \le d(p_j, p) + d(p_i, p) \le 2^* r_s \le r_c$. So, it is easy to know that node p_j is neighboring to node p_i and in the *Cluster of Neighbors* of node p_i, and vice versa.

From the above lemma 5.2.1, we can know that if there are t nodes that can monitor a same point p in the sensory field, then these t nodes will all belong to a same *Cluster of Neighbors*, say CN^*. But according to the definition 4, we know that for any node p_i, it can compute out a unique *Cluster of Neighbors* CN_i, so it is easy to see that there is $CN^* \subseteq CN_i$, if node p_i is among these m nodes that can monitor the point p in the sensory field.

From the above analysis, we can know that after partitioning all the nodes in the whole sensor network into clusters by using the algorithm *DCM*, and then each point p in the sensory field can be monitored by the nodes in someone *Cluster of Neighbors* CN_i. So, if we can guarantee that all the nodes in a same *Cluster of Neighbors* CN_i will belong to k different groups, then after scheduling, each of these k groups will be able to monitor the point p. Since there may be more than one point in the sensory field that will be monitored by the nodes in the *Cluster of Neighbors* CN_i only, so if we can guarantee that every k nodes in the *Cluster of Neighbors* CN_i will belong to k different groups, then after scheduling, each of these k groups will be able to monitor all points that can be monitored by the nodes in the *Cluster of Neighbors* CN_i. For a given *Cluster of Neighbors* CN_i, let the number of nodes in CN_i be m, then the problem turns to be how to allocate these m nodes into k different groups such that for any k different nodes in these m nodes, each of these k different nodes will belong to a different group respectively.

Let's consider the definition of CAC proposed in the section 5.1 again, we can find out that the CAC can solve the above problem exactly. So, based on the definition of CAC, we can give our *Node Scheduling Scheme for nodes in a Cluster of Neighbors* (algorithm *NSS_ CN*) as follows:

Step 1: For any sensor node p_i, let the *Cluster of Neighbors* computed out by it be $CN_i=\{p_1,p_2,\ldots,p_m\}$, if p_i is the *Cluster Head* in CN_i, then,

Step 1.1: It collects the GIDs (Group IDs) from all of its *Cluster Members* (i.e. all other nodes in CN_i) at first. Without losing generality, we suppose that the first a nodes $\{p_1,p_2,\ldots,p_a\}$ in $\{p_1,p_2,\ldots,p_m\}$ have been assigned GIDs before.

Step 1.2: And then, node p_i will assign these m sensor nodes $\{p_1,p_2,\ldots,p_m\}$ into the k different groups $\{0,1,\ldots,k\text{-}1\}$ on the basis of the $(m,\ k)$ combinatorial assignment code (m,N,k)-CAC, where the (m,N,k)-CAC has the smallest N and its incidence matrix $A=(a_{ij})$ can be constructed as follows:

Step 1.2(a): If $a\leq m\text{-}k$, then set the entry 1 to all these cells in the first $m\text{-}k$ columns of A, and as for the $k\times k$ sub-matrix that is consisted with the columns from $m\text{-}k+1$ to m in the matrix A, set it to be an identity matrix, and then, set the entry 0 to all other cells in A. I.e. the incidence matrix A will be described as:

$$A=\left(a_{ij}\right)=\begin{array}{r}\\ group(0)\\ group(1)\\ \cdots\\ group(k-1)\end{array}\begin{array}{c}\begin{array}{ccccccccc}p_1 & p_2 & \cdots & p_{m-k} & p_{m-k+1} & p_{m-k+2} & \cdots & p_{m-1} & p_m\end{array}\\ \begin{bmatrix}1 & 1 & \cdots & 1 & 1 & 0 & \cdots & 0 & 0\\ 1 & 1 & \cdots & 1 & 0 & 1 & \cdots & 0 & 0\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & 1\end{bmatrix}\end{array}\quad(1)$$

Step 1.2(b): If $a>m\text{-}k$, then let b denotes the number of nodes that each node has only one GID in its *List of Group IDs* respectively, without losing generality, suppose that these b nodes are $\{p_{a\text{-}b+1},p_{a\text{-}b+2},\ldots,p_a\}$, and furthermore, let c denotes the number of nodes among $\{p_{a\text{-}b+1},p_{a\text{-}b+2},\ldots,p_a\}$ that their GIDs are all different from each other, without losing generality, suppose that these c nodes are $\{p_{a\text{-}c+1},p_{a\text{-}c+2},\ldots,p_a\}$, then

(1) if $c+m\text{-}a\geq k$, then set the incidence matrix A as the formula(1).
(2) if $c+m\text{-}a<k$, then set the incidence matrix A as the formula(2):

$$A=\left(a_{ij}\right)=\begin{array}{r}\\ group(0)\\ group(1)\\ \cdots\\ group(k-1)\end{array}\begin{array}{c}\begin{array}{ccccccccc}p_1 & p_2 & \cdots & p_{a-c} & p_{a-c+1} & p_{a-c+2} & \cdots & p_{m-1} & p_m\end{array}\\ \begin{bmatrix}1 & 1 & \cdots & 1 & 1 & 0 & \cdots & 0 & 0\\ 1 & 1 & \cdots & 1 & 0 & 1 & \cdots & 0 & 0\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & 1\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0\end{bmatrix}\end{array}\quad(2)$$

For example set the entry 1 to all the cells in the first $m\text{-}(c+m\text{-}a)=a\text{-}c$ columns of A, and set the sub-matrix that consists with the first $m\text{-}(a\text{-}c)$ rows and the last $m\text{-}(a\text{-}c)$ columns of A to be an identity matrix, and then, set the entry 0 to all other cells in A.

Step 1.3: After completing the assignment of group IDs for its *Cluster Members* in CN_i, node p_i sends the corresponding *List of Group IDs* to each of its *Cluster Members*.

Step 2: For any node q_i in the whole sensor network, if it receives a *List of Group IDs* from its *Cluster Head*, then it updates its *List of Group IDs* through adding the new *List of Group IDs* together with its previous *List of Group IDs* received before as its current *List of Group IDs*. (I.e. supposing that LGI_i is the *List of Group IDs* stored previously in node q_i, and LGI^* is a newly arrived *List of Group IDs*, then node q_i will use $LGI_i \cup LGI^*$ as its current *List of Group IDs*). And then, it broadcasts its new *List of Group IDs* to its neighbors.

In order to illustrate the actual process of the above algorithm *NSS_ CN*, we give a simple example as follows firstly:

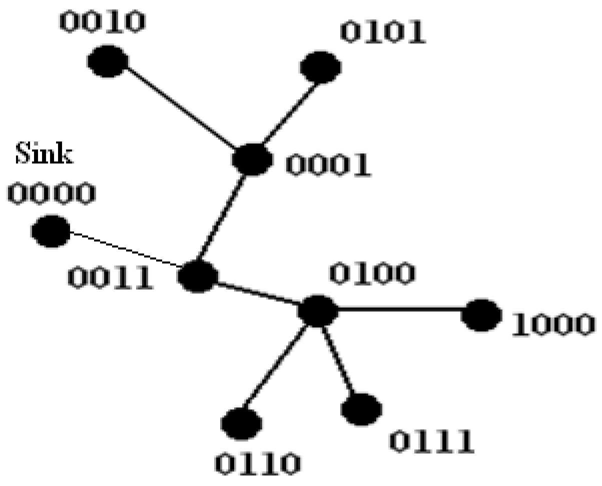


Figure 5.2-1 A Simple Sensor Network

Figure 5.22-1 illustrates a simple sensor network, where the line segment between a pair of nodes denotes that these two nodes are neighboring to each other. Let $k = 3$, combing with the algorithm *DCM*, the algorithm *NSS_ CN* will execute as follows in this occasion:

Step 1: According to the step 1 of algorithm *DCM*, each node computes out its *Cluster of Neighbors*. So, nodes 0001, 0011, and 0100 will compute out their *Cluster of Neighbors* as {0001, 0010, 0011, 0101}, {0001, 0011, 0100} and {0011, 0100, 0110, 0111, 1000} respectively.

Step 2: According to the step 2 of algorithm *DCM*, obviously only the node 0001 will declare itself as the *Cluster Head* in its *Cluster of Neighbors* {0001, 0010, 0011, 0101}.

Step 3: According to the step 1 of algorithm *NSS_ CN*, the node 0001 will allocate different *List of Group IDs* to each of the nodes in its *Cluster of Neighbors* {0001, 0010, 0011, 0101}.Furthermore, according to the step 1.1 and step1.2(a) of the algorithm *NSS_ CN*, node 0001 will assign the *List of Group IDs* {0,1,2},{0},{1},{2} to the nodes 0001, 0010, 0011, 0101 in its *Cluster of Neighbors* respectively.

Step 4: According to the step 2 of algorithm *NSS_ CN*, node 0001 will broadcast its *List of Group IDs* to node 0011, which will trigger node 0011 to declare itself as the *Cluster Head* in its *Cluster of Neighbors* {0001, 0011, 0100}.

Step 5: According to the step 1 of algorithm *NSS_ CN*, the node 0011 will allocate different *List of Group IDs* to each of the nodes in its *Cluster of Neighbors* {0001, 0011, 0100}. Furthermore, according to the step1.2(*b*)(2) of the algorithm *NSS_ CN*, node 0011 will assign the *List of Group IDs* {0,1,2},{1},{2} to the nodes 0001, 0011, 0100 in its *Cluster of Neighbors* respectively.

Step 6: According to the step 2 of algorithm *NSS_ CN*, node 0011 will broadcast its *List of Group IDs* to node 0100, which will trigger node 0100 to declare itself as the *Cluster Head* in its *Cluster of Neighbors* {0011, 0100, 0110, 0111, 1000}.

Step 7: Finally, According to the step 1 of algorithm *NSS_ CN*, the node 0100 will allocate different *List of Group IDs* to each of the nodes in its *Cluster of Neighbors* {0011, 0100, 0110, 0111, 1000}. Furthermore, according to the step1.2(*a*) of the algorithm *NSS_ CN*, node 0100 will assign the *List of Group IDs* {0,1,2},{0,1,2},{0},{1},{2} to the nodes 0011, 0100, 0110, 0111, 1000 in its *Cluster of Neighbors* respectively.

So, after executing the algorithm *NSS_ CN*, the result of GIDs assignment will be as the following Fig.3.

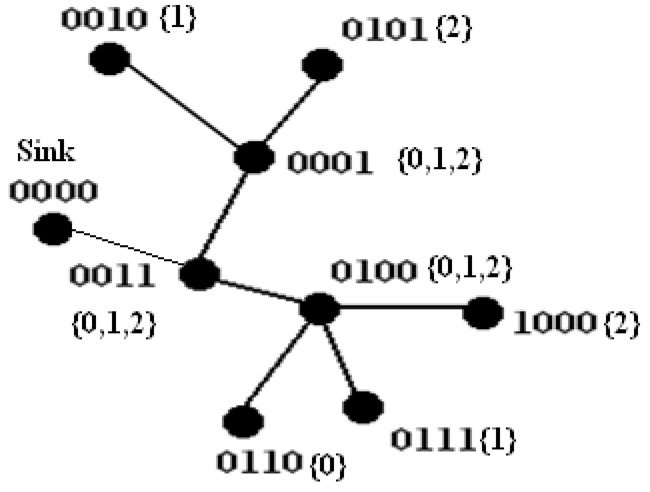


Figure 5.2-2 GID assignment result after executing *NSS_ CN*

According to the definitions and analysis in section 5.1, we can prove that the above given algorithm *NSS_ CN* has the following good characteristic:

***Theorem** 5.2.1* After executing the algorithm *NSS_ CN*, each node in the whole sensor network will be allocated a *List of Group IDs*, and for any sensor node p_i, let the *Cluster of Neighbors* computed out by it be $CN_i=\{p_1,p_2,\ldots,p_m\}$, then for any k nodes in $\{p_1,p_2,\ldots,p_m\}$, each of these k nodes can belong to a different group respectively.

Proof: According to the step 1.2 of the algorithm *NSS_CN*, it is easy to check that the constructed (m,N,k)-CAC is an (m, k) combinatorial assignment code. And additionally, according to the algorithm *DCM*, it is obvious that for each node p_i in the whole sensor network, it will belong to a *Cluster of Neighbors*, and then, by combing with the algorithm *NSS_CN*, it is easy to know that node p_i will be allocated a *List of Group IDs* based on the (m, k) combinatorial assignment code (m,N,k)-CAC. Thereafter, the conclusion follows.

On the basis of the above theorem 3, we can furthermore prove that the following lemma 4 follows.

Lemma *5.2.2* After executing the algorithm *NSS_CN*, each node in the whole sensor network will be allocated a *List of Group IDs*, and for any sensor node p_i, let the *Cluster of Neighbors* computed out by it be $CN_i=\{p_1,p_2,\ldots,p_m\}$, and the length of *List of Group IDs* for each node in these m sensor nodes $\{p_1,p_2,\ldots,p_m\}$ be $\{l_1,l_2,\ldots,l_m\}$ respectively, then for all these nodes whose length of *List of Group IDs* is 1, their GIDs are different from each other. And additionally, for all these nodes whose length of *List of Group IDs* is not 1, then they will include k different GIDs.

Proof: When executing the algorithm *NSS_CN*, if the nodes $\{p_1,p_2,\ldots,p_m\}$ haven't been assigned a *List of Group IDs* before, according to the step 1.2(a), it is easy to check that the conclusion follows. Otherwise, if there are some nodes in $\{p_1,p_2,\ldots,p_m\}$ have been assigned a *List of Group IDs* before, then according to the step 1.2(b), it is easy to check that the conclusion follows also.

5.2.4 ID selection scheme for nodes in the whole sensor network

As for our algorithm *NSS_CN*, compared to the randomized node scheduling method, each node needs to maintain a *List of Group IDs* that includes at least one GID after executing the algorithm *NSS_CN*. However, each node needs only one of the Group IDs to be as the GID before the extra-on rule for the connectivity applys. So, we need to reduce the number of Group IDs stored in each node to be one next.

Assume that the length of *List of Group IDs* for each node in these m sensor nodes $\{p_1, p_2,\ldots,p_m\}$ are $\{l_1,l_2,\ldots,l_m\}$ respectively, then we can give the *Group ID Selection Scheme* (algorithm *GIDSS*) for nodes in the whole sensor network as follows:

For any sensor node p_i in the whole sensor network, assume the length of its *List of Group IDs* is l_i, then if $l_i > 1$, it selects out a unique GID randomly from its *List of Group IDs* as its chosen GID.

After GIDSS executed, each sensor node should have one and only one GID already. Now we start to maintain the connectivity for each group.

5.2.5 Connectivity maintenance for each group

In the above sections we have introduce our scheme to schedule all the nodes in the whole sensor network into k different groups evenly, and through utilizing these schemes, each node will be assigned a unique GID. But considering the nodes in each group, perhaps they are not globally connected. So, in this section we will introduce a method to guarantee that all the nodes in each group will be globally connected.

Definition *5.2.3* For any sensor node p_i, let node p_j be a neighboring node of p_i. Suppose the minimum hops to the Sink from node p_i, p_j are H_i and H_j respectively. If $H_i = H_j + 1$, then node p_j is called an *Upstream Node* of p_i and the node p_i is called a *Downstream Node* of p_j. If $H_i = H_j$, then node p_j is called a *Brother Node* of p_i. If $p_j > p_i$, then node p_j is called an *Older Brother Node* of p_i, otherwise, node p_j is called a *Younger Brother Node* of p_i.

Using the above definition, we present our connectivity maintenance scheme (algorithm *CM*) as follows:

Suppose that a sensor network is connected, and each node maintains a list of all Upstream Nodes that are on its shortest paths to the Sink and all of its Brother Nodes. After the GIDs assignment, each node in the sensor network will update its GID List to maintain the connectivity of each group of the network. An initial GID List just contains one GID obtained from Algorithm *GIDSS*.

Step 1: For a node p_i, suppose that its GIDs List is L_i. For any $g \in L_i$:

Step 1.1: If there is neither an *Upstream Node* nor a Brother Node, which has the GID g in its GID List, then p_i selects one of its *Upstream Nodes*, p_j, who has the shortest GIDs List, and sends an *AGAC* (*Appended GID Application for Connectivity*) message to p_j.

Step 1.2: If there is no *Upstream Node* but there are *Brother Nodes*, who have g in their GID List, then p_i selects one of its *Brother Nodes*, say p_j, that has the shortest GIDs List, and then, p_i sends an *AGAC* message to the node p_j.

Step 2: After the GIDs assignment, each node in the sensor network maintains a back-off timer.

Step 2.1: If the node p_j received *AGAC* messages from its any *Downstream Node* or *Younger Brother Node* p_i before time out, then it will update its GID List according to the *AGAC* messages.

Step 2.2: If the node p_j received $AGAC$ messages from its any *Older Brother Node* p_i before time out, and node p_j finds out that it has no *Upstream Node* that has g in its GID List, then it will select one of its *Upstream Nodes*, p_x, that has the shortest GIDs List, and sends an $AGAC$ message to p_x.

Step 2.3: After time out, it will broadcast its updated GIDs List to all of its neighbors.

The above algorithm *CM* has the following useful result.

Theorem *5.2.2* Suppose that the whole sensor network is connected. Then after executing Algorithm *CM*, each node of the network has a path to the Sink, which consists of nodes with the same GID. I.e. the nodes in each group 0, 1, .., k-1 are connected.

Proof: For any given sensor node p_i in the sensory field, let its shortest hops to Sink is H_i. We prove the theorem by induction.

(1) When H_i =1, it is obvious that node can communicate with Sink directly.

(2) Let H_i =2. Since the sensor network is connected, there exist some Upstream Nodes of p_i, whose hops to Sink is 1. If there is one of those *Upstream Nodes* that has a GID as same as p_i's, then the conclusion follows. Otherwise,

(2.1)If there is no Brother Node that has a GID same as p_i's, then according to the step 1.1 of Algorithm *CM*, the node p_i will send an $AGAC$ message to one of its *Upstream Nodes*, say p_j, and ask p_j to add an appended GID in its GID List. In any case, p_i will have at least one Upstream Node p_j with the same GID with p_i. So, the node p_i has a path of size 2 which consists of nodes with the same GID.

(2.2) If there is a *Brother Node* p_{o1} has a GID same as p_i's, then, for p_{o1}, if it has an *Upstream Node* with the same GID, it is obvious that the conclusion follows. Otherwise, if p_{o1} has no *Older Brother Node*, then according to the step 2.2 of Algorithm *CM*, it will select one of its *Upstream Nodes* p_x, and ask p_x to add an appended GID in its GID List. It means that the conclusion follows. If p_{o1} has an *Older Brother Node*, say p_{o2}, which has a GID same as p_{o1}'s, through recursion and the step 1.2 of Algorithm *CM*, it is easy to know that the conclusion follows, since there exists certainly a node p_{os}, which will have no *Older Brother Node*.

(3) Suppose the conclusion is true for H_i=d, where d>2. For H_i =d+1, if it has an *Upstream Node* that has a GID same as p_i's, or it has neither *Upstream Node* nor *Brother Node*, which has the same GID as p_i's, then the Algorithm *CM* will find one *Upstream Nodes* of p_i with d hops to the Sink, which has a GID same as p_i. (This GID is either already existed or added by requesting). So by induction, the conclusion is true. Otherwise, if there is no *Older Brother Node* that has a GID same as p_i's, then according to the step 2.2 of Algorithm *CM*, p_i will select one of its *Upstream Nodes* p_x with d hops to the Sink, which has a

GID same as p_i. By induction, the conclusion is true. Finally, if there are *Older Brother Nodes* that have a same GID as p_i's, then according to the step 1.2 of Algorithm *CM*, p_i will select an *Older Brother Node*, say p_{o1}, which has a GID same as p_i's. And then, by induction on p_{o1} the conclusion follows.

5.2.6 CAC based node scheduling scheme

On the basis of the analysis in those previous sections, we can finally give our CAC based Node Scheduling Algorithm (algorithm *CNSA*) as follows:

Phase 1(Initial node scheduling phase):

Phase 1.1: Partition all the nodes in the whole sensor network into a collection of *Cluster of Neighbors* according to the algorithm *DCM* proposed in section 4.2.

Phase 1.2: For each *Cluster of Neighbors*, assign a *List of Group IDs* to each node in the *Cluster of Neighbors* according to the algorithm *NSS_ CN* proposed in section 4.3.

Phase 1.3: For each node in the whole sensor network, select a unique GID from its *List of Group IDs* according to the algorithm *GIDSS* proposed in section 4.4.

Phase 1.4: For each group, maintain its global connectivity according to the algorithm *CM* proposed in section 5.2.5.

Phase 2 (Working phase):

In the working phase, all nodes work in turns at their given time slots. At the time slot t, if $t \equiv g \bmod k$, then all nodes in group g keep working, while other nodes will hibernate.

5.3 Performance analysis

Let's consider about the following problem: For any give point p in the sensory field, supposing that there are m sensor nodes that can monitor it, then, after scheduling, what's the probability that the point p can be monitored by each group? For convenience, we call this kind of probability for a scheduling scheme as its *Probability of Coverage Maintenance* (*PCM*).

As for the randomized node scheduling method introduced in section 4.3.1 Chapter 4, it is easy to know that its *PCM* can be calculated as:

$$PCM_{RANDOM}=\frac{\binom{m}{k}k!}{k^m}. \qquad (3)$$

On the other hand, after executing algorithm *GIDSS* introduced in section 5.2.4, it is obvious that each node in the whole sensor network will also have a unique GID in its *List of Group IDs*. And then, it is easy to prove that the *PCM* of the algorithm *GIDSS* can be calculated as:

$$PCM_{GIDSS}=\begin{cases} 1 & \text{if } m_1 \geq k; \\ \frac{(k-m_1)\binom{m-m_1}{k-m_1}(k-m_1)!}{k(k-m_1)^{m-m_1}}: & \text{otherwise.} \end{cases} \qquad (4)$$

Where m_1 is the number of such kind of nodes in these m sensor nodes, that each of them has a unique GID in its *List of Group IDs* before executing the algorithm *GIDSS*.

Proof: From the lemma 5.2.2, we can know that each node p_i in these m sensor nodes $\{p_1,p_2,\ldots,p_m\}$ will be assigned a *List of Group IDs* on the basis of the *CAC* model, and if the length of its *List of Group IDs* is not 1, then it will be k. Let m_1 denotes the number of such kind of nodes in these m sensor nodes, that each of them has a unique GID in its *List of Group IDs* before executing the algorithm *GIDSS*, then from the lemma 4, we can also know that the GIDs of these m_1 nodes are different from each other. So, if $m_1 \geq k$, there is $PCM_{GIDSS}=1$ obviously, and otherwise, we need be able to select out the other k-m_1 GIDs different from these m_1 GIDs from the other m-m_1 nodes. It is east to check that the probability that we can select out these k-m_1 GIDs from the other m-m_1 nodes is $\frac{\frac{k-m_1}{k}\binom{m-m_1}{k-m_1}(k-m_1)!}{(k-m_1)^{m-m_1}}$. Therefore the conclusion follows.

On the basis of the above analysis, we have the following theorem:

***Theorem** 5.3.1* the *PCM* of the algorithm *GIDSS* is bigger than that of the randomized node scheduling method.

Proof: Since there are $PCM_{GIDSS} = \begin{cases} 1: & \text{if } m_1 \geq k; \\ \frac{(k-m_1)\binom{m-m_1}{k-m_1}(k-m_1)!}{k(k-m_1)^{m-m_1}}: & \text{otherwise} \end{cases}$ and $PCM_{RANDOM} = \frac{\binom{m}{k}k!}{k^m}$, it is obvious that if $m_1 \geq k$, the conclusion will certainly follow, then we need only to prove that the conclusion will follow also, when $m_1 < k$.

Let $f(x) = \frac{(k-x)\binom{m-x}{k-x}(k-x)!}{k(k-x)^{m-x}}$, then we know that there are $PCM_{GIDSS}=f(m_1)$ and $PCM_{RANDOM}= f(0)$. So, if we can prove that $f(x)$ is monotone increasing, and then the conclusion will follow.

Certainly, there is

$$\begin{aligned}
f(x) &= \frac{(k-x)\binom{m-x}{k-x}(k-x)!}{k(k-x)^{m-x}} \\
&= \frac{1}{(m-k)!} * \frac{k-x}{k} * \frac{(m-x)!}{(k-x)^{m-x}} \\
&= \frac{1}{(m-k)!} * \frac{k-x}{k} * \{\frac{m-x}{k-x} * \frac{m-x-1}{k-x} * \dots * \frac{1}{k-x}\} \\
&= \frac{1}{(m-k)!} * \{\frac{k-x}{k} * \frac{m-x}{k-x}\} * \{\frac{m-x-1}{k-x} * \dots * \frac{1}{k-x}\} \\
&= \frac{1}{(m-k)!} * \frac{m-x}{k} * \frac{(m-x-1)!}{(k-x)^{m-x-1}} \\
&= \frac{1}{k(m-k)!} * \frac{(m-x)(m-x-1)!}{(k-x)^{m-x-1}}.
\end{aligned}$$

So we have

$$f(x+1) = \frac{1}{(m-k)!} * \frac{k-x-1}{k} * \frac{(m-x-1)!}{(k-x-1)^{m-x-1}} = \frac{1}{k(m-k)!} * \frac{(k-x-1)(m-x-1)!}{(k-x-1)^{m-x-1}}.$$

And additionally, it is easy to check that for any positive integer α and β, there is

$$(1+\frac{1}{\beta})^{\alpha} > 1 + \frac{\alpha}{\beta} > \frac{1}{\beta} + \frac{\alpha}{\beta}.$$

Let $\alpha = m-x-1$ and $\beta = k-x-1$, then we have

$$(1+\frac{1}{k-x-1})^{m-x-1} > \frac{1}{k-x-1} + \frac{m-x-1}{k-x-1} = \frac{m-x}{k-x-1} \Longrightarrow$$

$$(\frac{k-x}{k-x-1})^{m-x-1} > \frac{m-x}{k-x-1} \Longrightarrow$$

$$\frac{k-x-1}{(k-x-1)^{m-x-1}} > \frac{m-x}{(k-x)^{m-x-1}} \Longrightarrow$$

$$\frac{(k-x-1)(m-x-1)!}{(k-x-1)^{m-x-1}} > \frac{(m-x)(m-x-1)!}{(k-x)^{m-x-1}} \Longrightarrow$$

$$\frac{1}{k(m-k)!} * \frac{(k-x-1)(m-x-1)!}{(k-x-1)^{m-x-1}} > \frac{1}{k(m-k)!} * \frac{(m-x)(m-x-1)!}{(k-x)^{m-x-1}}.$$

Therefore, we have $f(x+1) > f(x)$, which means that $f(x)$ is monotone increasing.

To illustrate the above *theorem 4* in a more visual way, we give two simple examples as follows:

(1) In Figure 5.2.2, let a given point *p* in the sensory field be covered by the node 0100, then according to the lemma 5.2.1, if there are other nodes in the sensor network can cover the point *p*, then all these nodes will be in the *Cluster of Neighbors* of the node 0100, i.e. all these nodes will belong to the set {0011, 0100, 0110, 0111, 1000}. From the former analysis in section 4.3, we have known that these nodes 0011, 0100, 0110, 0111, 1000 will be assigned the *List of Group IDs* {0,1,2},{0,1,2},{0},{1},{2} respectively after executing the algorithm *NSS_ CN*, so before executing the algorithm *GIDSS*, we have m=5, k=3, and m_1=3 in these five sensor nodes {0011, 0100, 0110, 0111, 1000}. Therefore, after executing the algorithm *NSS_ CN*, according to the formula (3) and formula (4), we have $PCM_{RANDOM}= \frac{\binom{m}{k}k!}{k^m} = \frac{\binom{5}{3}3!}{3^5}$=24.69% but PCM_{GIDSS}=1. It is obvious that our new algorithm has much better performance of coverage maintenance in this occasion.

(2) in Figure 5.2.2, let a given point *p* in the sensory field be covered by the node 0001, then according to the lemma 5.2.1, if there are other nodes in the sensor network can cover the point *p*, then all these nodes will be in the *Cluster of Neighbors* of the node 0001, i.e. all these nodes will belong to the set {0001, 0010, 0011, 0101}. From the former analysis in section 5.2.3, we have known that these nodes 0001, 0010, 0011, 0101 will be assigned the *List of Group IDs* {0,1,2},{0,1,2},{1},{2} respectively after executing the algorithm *NSS_ CN*, so before executing the algorithm *GIDSS*, we have m=4, k=3, and m_1=2 in these five sensor nodes {0001, 0010, 0011, 0101}. Therefore, after executing the algorithm *NSS_ CN*, according to the formula (3) and formula (4), we have $PCM_{RANDOM}=\frac{\binom{m}{k}k!}{k^m}=\frac{\binom{4}{3}3!}{3^4}$= 29.63% but $PCM_{GIDSS}= \frac{(k-m_1)\binom{m-m_1}{k-m_1}(k-m_1)!}{k(k-m_1)^{m-m_1}} = \frac{1*\binom{2}{1}(1)!}{3*(1)^2} =$ 66.67%. It is obvious that our new algorithm has much better performance of coverage maintenance in this occasion as well.

In addition, in the connectivity maintenance phase, our CNSA can take advantage of the additional *group IDs* assigned according to CAC code. By assigning nodes extra GIDs which are already in their *CAC group ID table*, CNSA is way more efficient than RSGC's blind need-and-add strategy. This will give CNSA bigger edge on top of the assignment part.

For the execution time compared to our first proposal CCMS, CNSA has a major improvement. As we can see, CNSA's most calculation intensive part is that the cluster heads apply CAC to each neighbor, while, in CCMS, each node need to perform the local clique searching. The calculation intensity is significantly reduced.

5.4 Conclusions

This chapter we first defined a combinatorial code we named CAC. Based on the characteristic of the CAC, we provided our proposed sensor scheduling algorithm CNSA. We also analyzed the performance of CNSA in comparison with RSGC [10]. Theoretical analysis shows that CNSA outmatches RSGC in terms of *Probability of Coverage Maintenance (PCM).* Also with the help of CAC, CNSA can maintain the coverage without the knowledge of local cliques. CNSA executes much faster than CCMS, for there is not local clique searching. Therefore CNSA is also considered an *Efficient Sensor Scheduling Algorithm.*

As we mentioned previously, CNSA is an on-going research, so simulations is yet to be done. However, theoretical analysis already showed some sparkles of CNSA. The design of CNSA we introduced in this thesis will be deemed as the starting point of our further exploration.

Chapter 6

Conclusions and Future Work

6.1 Concluding Remarks

In this thesis, we have studied a good number of works on lifetime and performance optimization for Wireless Sensor Network, along with many aspects requiring more work. We would like to summarize what we have found so far.

We have seen many different strategies serve energy saving. Some of them try to limit the communication between nodes in order to saving transmission and receiving power. Some of the data gathering protocol such as LEACH [42] uses efficient routing and data fusion to reduce the transmission times. However, according to the sensor energy consumption behavior, we know that a sensor consumes almost the same amount of energy in idle mode and in receiving mode. That means keeping the unnecessary nodes in active is already a huge waste of energy. Constrain the transmission power will degrade the connectivity of the network. Besides it still have time problem of idle listening. Using special deployment of sensors to reduce the redundancy sounds promising. However, the applicable case is very tiny in number. Technology has allowed sophisticated inexpensive sensors to be produced. Large range WSN is now available. To carry out a special layout for hundreds of sensors is not practical. In some cases it is even impossible, when the monitored area is too severe or unbearable to human being, such as volcano, or nuclear reaction field. So we established that sensor scheduling scheme is the optimal choice. Especially, nowadays, the computing capability of sensors improves dramatically. Using computing power in exchange of transmission power becomes the central idea for power saving on WSN.

With different scheduling scheme, some features also need to compare. Some scheme that poses limitations to the system is not in favor. For example, [45] proposed target coverage scheduling requires location information of all the sensors in the WSN. Furthermore, it requires the target must be known and stationary. This requirement makes his approach only valid for very specific users. For most of the cases sensors in the WSN monitor the whole sensory field instead of just a few static targets. Others like [51] also overlooked some factor in their scheme. So first of all, their scheme is a round-based scheduling scheme. Some node wakes up from sleep node at a beginning of a round, just to execute the scheduling scheme. We know that switching mode also consume lots of energy. Second, the scheme did not take maintenance of connectivity into consideration. So after the network ran for a while, some nodes' out of power, it is very like to have such situation that some nodes have almost full energy but their neighbors are all dead. They became isolated node. Their energy left energy is useless because none of their data is going anywhere. This is a major design defect. RSGC apparently has an overall better performance than any of other scheduling scheme we discussed. It considered both the coverage and connectivity at the same time. However, as shown in the simulation result our proposal outperformed it in all the tests we performed. The reason is that RSGC is a pure randomized scheduling scheme, which ignores the actually layout of the sensor nodes complete. Therefore, RSGC with some probability will carry out a good group setup. There are some advantages of RSGC too. The execution speed of RSGC is almost instantly done, give that the grouping part of the scheme is simply asking nodes to pick up a random number from [0,k], k is the number of groups. With such a simple algorithm and an acceptable result (receive some improvement in the life time), RSGC is worth some applause.

The simulation result shows that our algorithm has an obvious improvement for the life time of a WSN comparing with RSGC. With the help of the local clique, the clique heads can maximum evenly distributes clique members into each group. The local clique searching takes some time, but with an acceptable delay and much improved life time, we considered it's a fair trade. This is also our motivation for searching for more efficient grouping algorithm. CAC is proposed from this background. Using CAC's feature, our assignment can guarantee any k node from a neighbor cluster are from k different group. However, the cost is more nodes will have several GIDs, which means more working shifts. Therefore, after the GID assignment phase completes, nodes with more than one *Group ID* will pick only one entry from their *Group ID list* as the GID. Theoretical speaking, the result of CNSA would not be as good as our CCNS, but there is a good chance a good grouping will come along with much less execution time. Further verification is needed with simulation. That would be one of our future works.

Finally, please note that, all the comparison done in this thesis is for the evaluation of our new proposals. We know that each research has its context. For example, the round-based scheduling scheme would be more suitable than the group-based ones if the network is very dynamic, and each round the nodes set to sleep would be different from previous rounds. Furthermore, some of the researches introduced can be used jointly for better performance. For instance, the efficient deployment scheme proposed in [25] can guarantee an even longer system lasting time with our scheme. The sensors closer to the sink drain their power faster than others due to the traffic intensity. If our deployment can compensate the node number according to the traffic analysis as [25], we will receive an even better result.

6.2 Future Work

Through all the work we have done for this thesis, we can see there is a bright future for the Wireless Sensor Network study. There is also a lot of room for improvement. So we would like to carry on our work forward.

Here is some outlook for the future works. First of all, as we said we will finish up the simulation part for CNSA. On top of that designing more efficient scheduling algorithm is on our agenda. Also there could be potential security problem with the scheduling schemes. Most of the scheduling schemes are based on the assumption of a friendly environment. Security issues need to be addressed for scheduling schemes, because important information (such as table of neighboring nodes) is released during the protocol execution time. Therefore, secured scheduling scheme is another appealing topic for future research.

Bibliography

[1]. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. IEEE Communications Magazine, 2002,40(8): 102-114.
[2]. A. Vaseashta, S. Vaseashta. A Survey of Sensor Network Security. Sensors & Transducers Journal, 2008, 94(7): 91-102.
[3]. Lei Wang, Haowei Shen, Yaping Lin. Voronoi Tessellation based Rapid Coverage Decision Algorithm for Wireless Sensor Networks. Lecture Notes in Computer Science. (UIC2007) Springer, 2007, 4611: 495-503.
[4]. Tian, D., and Georganas, N. D. Location and calculation-free node scheduling schemes in large wireless sensor networks. Ad Hoc Networks. 2004, 2:
65-85.
[5]. Hussein, Stipanovic. Effective coverage control for mobile sensor networks with guaranteed collision avoidance. IEEE Transactions on Control Systems Technology. 2007,15(4): 642-657.
[6]. Deng J, Han YS, Heinzelman WB, Varshney PK. Scheduling sleeping nodes in high density cluster-based sensor networks. ACM/Kluwer Mobile Networks and Applications (MONET), 2005,10(6):825-835.
[7]. Ye F, Zhong G, Lu SW, Zhang LX. PEAS: A robust energy conserving protocol for long-lived sensor networks. Proc. of the 23nd Int'l Conf. on Distributed Computing Systems. 2003. 28-37.
[8]. Chen BJ, Jamieson K, Balakrishnan H, Morris R. Span: An energy efficient coordination algorithm for topology maintenance in ad hoc wireless networks. ACM Wireless Networks, 2002,8(5):481-494.
[9]. Godfrey PB, Ratajczak D. Naps: Scalable, robust topology management in wireless ad hoc networks. In: Proc. of the 3rd Int?l ACM Symp. on Information Processing in Sensor Networks. 2004. 443-451.
[10]. Liu C, Wu K, Xiao Y, Sun B. Random coverage with guaranteed connectivity: Joint scheduling for wireless sensor networks. IEEE Trans. on Parallel and Distributed Systems, 2006,17(6):562-575.
[11]. C. Liu, K. Wu, and V. King, Randomized Coverage-Preserving Scheduling Schemes for Wireless Sensor Networks, Proc. IFIP Networking Conf. 2005,
May 2005.
[12]. C. Hsin and M. Liu, Network Coverage Using Low Duty-Cycled Sensors: Random & Coordinated Sleep Algorithm, Proc. Third Int?l Symp. Information Processing in Sensor Networks (IPSN 2004), Apr. 2004.

[13]. Li XY, Wan PJ, Frieder O. Coverage in wireless ad hoc sensor networks. IEEE Trans. on Computers, 2003,52(6):753-763.
[14]. TSAI Y R. Coverage preserving routing protocols for randomly distributed wireless sensor networks. IEEE Transactions on Wireless Communications, 2007,6(4): 1240-1245.
[15]. Using boolean reasoning to anonymize databases. A. Ohrn, Ohno-Machado. 1999, A. I. Medicine, pp. 15, 3, 235–254.
[16]. Chen BJ, Jamieson K, Balakrishnan H, Morris R. Span: An energy efficient coordination algorithm for topology maintenance in ad hoc wireless networks. ACM Wireless Networks, 2002,8(5):481-494.
[17]. Yong Ding,Chen Wang,Li Xiao. A Connectivity Based Partition Approach for Node Scheduling in Sensor Networks. Lecture Notes in Computer Science, 2007, 4549: 354-367.
[18]. Godfrey PB, Ratajczak D. Naps: Scalable, robust topology management in wireless ad hoc networks. In: Proc. of the 3rd Int'l Symp. on Information Processing in Sensor Networks. Berkeley: ACM Press, 2004. 443-451.
[19]. Fang, Q.,Liu, J.,Guibas, L.,Zhao, F. RoamHBA: maintaining group connectivity in sensor networks. Proceedings of the Third ACM International Symposium on Information Processing in Sensor Networks (IPSN'04); 2004, 151-160.
[20]. Wang X, Xing G, Zhang Y, Lu C, Pless R, Gill C. Integrated coverage and connectivity configuration in wireless sensor networks. In: Akyildiz IF, Estion D, eds. Proc. of the ACM Int'l Conf. on Embedded Networked Sensor Systems (SenSys). 2003. 28-39.
[21]. D. Tian and D. Georganas, Connectivity Maintenance and Coverage Preservation in Wireless Sensor Networks, Ad Hoc Networks J., 2005, pp. 744-761.
[22]. Kumar S, Lai TH, Balogh J. On k-coverage in a mostly sleeping sensor network. In: Haas ZJ, ed. Proc. of the ACM Int'l Conf. on Mobile Computing and Networking (MobiCom). New York: ACM Press, 2004. 144-158.
[23]. Y.Ishai, E.Kushilevitz,R.Ostrovsky,and A.Sahai. Batch codes and their applications. Proceedings of STOC 2004, ACM Press. 262-271.
[24]. M.B.Paterson, D.R.Stinson, R.Wei. Combinatorial batch code. Advances in Mathematics of Communications (AMC) . 2009,3(1):13-27.
[25]. Chu-Fu Wang, Shu-Chien Huang. Efficient Deployment Algorithms for Prolonging Network Lifetime and Ensuring Coverage in Wireless Sensor Networks. In: Intelligent Systems Design and Applications, 2008. ISDA '08. Eighth International Conference on Volume 2, 26-28 Nov. 2008,196 - 201.
[26]. J.Wu and S. Yang, "SMART: A scan-based movement assisted sensor deployment method in wireless sensor networks", in Proc. INFOCOM, 2005, pp. 2313-2324.
[27]. F. Aurenhammer, "Voronoi Diagrams – A Survey Of A Fundamental Geometric Data Structure," ACM Computing Surveys 23, pp. 345-405, 1991.

[28]. Personal communication with J. Agre, Rockwell Siences Center. DARPA Program Review Meeting, Feb. 2000.
[29]. K. Mulmuley, Computational Geometry: An Introduction Through Randomized Algorithms, Prentice-Hall, 1994..
[30]. Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.B., “Coverage Problems in Wireless Ad-hoc Sensor Networks”, INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol.3, April 2001 pp. 1380 - 1387.
[31]. A. Savvides, F. Koushanfar, M. Potkonjak, M.B. Srivastava, “Location Discovery in Ad-hoc Wireless Sensor Networks,” unpublished, UCLA EE and CS Departments.
[32]. C.W. Kang, M.W. Golay, “An Integrated Method For Comprehensive Sensor Network Developement In Complex Power Plant Systems,” Reliability Engineering & System Safety, vol.67, pp. 17-27, Jan. 2000.
[33]. V. Raghunathan. C. Schurgers, S. Park, and M. B. Srivastava, Energy-Aware Wirdess Microsensor Networks, IEEE Signal ProcesshJg Magwine. 19 (2002), pp 40-50.
[34]. Nancy Lynch, Distributed Algorithms, Morgan Kaufman Publishers, 1996.
[35]. W. Ye, I. Heidcmann. and U. Estrin. 'y\n endrgy-effidzot MAC protocol for wireless sensor networkr," in Proceedings of the IEEE Infocorn. USC/lnfomtion Sciences Institute. New York. NY. USA I=. June 2002. pp. 1567-1576. [Online]. Available: http://www,iai.edul johhnh/pAF'ERS/YeO2a.html.
[36]. D. E. Culler. I. Hill. P. Buonadonna. R. Szewczyk. and A. Ww. "A network-centric approach to embedded software far tiny devices:' in Pmc. of the Fin1 InlernntrOnol Workshop on Embedded Sofhvare (EMSOFT). Oct. 2001.
[37]. A. Woo and D. E. Culler. "A transmission mntrol scheme for media access in sensor networks." in Pmc. dCM MOBICOM. July 2001.
[38]. R. M i n ~M . Bhardwaj. N. Ickes. A. Wag. and A. Chandralwsun. "The hardware and the network Total-system strategies for powcr aware wireless miaosmsors." in Pm. ofthe IEEE C.iS Workshop on wireless Comnicalinnr and Networking, Pasadena. CA. USA. Sep. 2002. [Online]. Available: hrtp://www.~l.cdul--unh-esearch/min-cas0.
[39]. A. Porret, T. Melly, C. C. Enz, and E. A.Vittoz, A Low- Power Low Voltage Transceiver Architecture Suitable for Wireless Distributed Sensors Network, IEEE International Symposium on Ciruits and Systems’00, Geneva, 2000.
[40]. M. J. Dong, K. Geoffrey Yung, and W. J. Kaiser, Low Power Signal Processing Architectures for Network Microsensors, 1997 International Symposium on Low Power Electronics and Design, Digest of Technical Papers (1997).
[41]. C. Intanagonwiwat, R. Govindan and D. Estrin, Directed Diffusion: A scalable and Robust Communication Paradigm for Sensor Networks, ACM MOBICOM’00, 2000.

[42]. W.R.Heizelman, A.Chandrakasan, and H.Balakrishnan, Energy-Efficient Communication Protocol for Wireless Micro Sensor Networks, IEEE Proceedings of the Hawaii International Conference on System Sciences, January 2000..
[43]. M. Cardai. D.-Z. Du. Improving Wireless Sensor Network Lifetims through Power Aware Organization. accepted to appear in ACM Wireless Network**.**
[44]. S. Slijepcevic. M. Potkonjak. Power Efficient Organization of Wireless sensor hetworks. IEEE Inremational Conference on Communications, (Jun. 2001).
[45]. M. Cardei et al., "Energy-efficient target coverage in wireless sensor networks," in IEEE Infocom, 2005.
[46]. F. Ye, G. Zhong, S. Lu, L. Zhang, Energy Efficient Robust Sensing Coverage in Large Sensor Networks, Technical Report.
[47]. B. Chen, K. Jamieson, H. Balakrishnana, R. Morris, Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks, MOBICOM'01, 2001.
[48]. Y. Xu, J. Heidemann, D. Estrin, Adaptive Energy- Conserving Routing for Multihop Ad hoc Networks, Technical Report 527, USC/ISI, Oct.2000.
[49]. Y. Xu, J. Heidemann, D. Estrin, Geography-informed Energy Conservation for Ad Hoc Routing, MOBICOM'01, 2001.
[50]. M. Cardai. D.-Z. Du. Improving Wireless Sensor Network Lifetims through Power Aware Organization. accepted to appear in ACM Wireless Network.
[51]. Di Tian , Nicolas D. Georganas, A coverage-preserving node scheduling scheme for large wireless sensor networks, Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, September 28-28, 2002, Atlanta, Georgia, USA
[52]. S. Tilak, N. Abu-Ghazaleh, and H. W, "Infrastructure Tradeoffs for Sensor Networks," Proc. First Int'l Workshop Wireless Sensor Networks and Applications (WSNA '02), Sept. 2002.
[53]. Z. Abrams, A. Goel, and S. Plotkin, "Set k-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks," Proc. Third Int'l Symp. Information Processing in Sensor Networks (IPSN 2004), Apr. 2004.
[54]. C. Hsin and M. Liu, "Network Coverage Using Low Duty-Cycled Sensors: Random & Coordinated Sleep Algorithm," Proc. Third Int'l Symp. Information Processing in Sensor Networks (IPSN 2004),Apr. 2004.
[55]. C. Liu, K. Wu, and V. King, "Randomized Coverage-Preserving Scheduling Schemes for Wireless Sensor Networks," Proc. IFIP Networking Conf. 2005, May 2005..
[56]. P. Godfrey and D. Ratajczak, "Robust Topology Management in Wireless Ad Hoc Networks," Proc. Third Int'l Symp. Information Processing in Sensor Networks (IPSN 2004), Apr. 2004.
[57]. H. Zhang and J. Hou, "Maintaining Coverage and Connectivity in Large Sensor Networks," Proc. Int'l Workshop Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks, invited paper, Feb. 2004.
[58]. C. Liu, "Randomized Scheduling Algorithm for Wireless Sensor

Neworks," Project Report of Randomized Algorithm, Univ. of
Victoria, Mar. 2004..
[59]. S. Slijepcevic and M. Potkonjak, "Power Efficient Organization of
Wireless Sensor Networks," Proc. IEEE Int'l Conf. Comm. 2001,
June 2001.
[60]. X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill,
"Integrated Coverage and Connectivity Configuration in Wireless
Sensor Networks," Proc. ACM Sensys, 2003, Nov. 2003.
[61]. H. Zhang and J. Hou, "Maintaining Coverage and Connectivity in
Large Sensor Networks," Proc. Int'l Workshop Theoretical and
Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer
Networks, invited paper, Feb. 2004.
[62]. Jun Lu, Lichun Bao, Tatsuya Suda. Probabilistic self-scheduling for
coverage configuration in wireless ad-hoc sensor networks. International Journal
of Pervasive Computing and Communications. 2008, 4(1):26-39..
[63]. Rao A, Ratnasamy S, Papadimitriou C, Shenker S, Stoica I. Geographic
routing without location information. In: Proc. of the 9th Annual Int'l Conf. on
Mobile Computing and Networking. San Diego: ACM Press, 2003. 96-108.
[64]. L. Wang, R. Wei, Z. Tian and B. Wang, A new scheduling method for
wireless sensor networks, submitted.