國立交通大學

資訊工程系

碩 士 論 文

無線感測網路中省電並維持覆蓋程度之分散式協定

Decentralized Energy-Conserving and Coverage-Preserving Protocols for Wireless Sensor Networks

研 究 生：羅立竹

指導教授：曾煜棋　教授

中 華 民 國 九 十 四 年 四 月

Decentralized Energy-Conserving and Coverage-Preserving Protocols for Wireless Sensor Networks

Student: Li-Chu Lo
Advisor: Prof. Yu-Chee Tseng

A Dissertation Submitted to
the Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Chiao-Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science and Information Engineering

Hsinchu, Taiwan

April 2005

無線感測網路中省電並維持覆蓋程度之分散式協定

學生：羅立竹　　　　　　　　　　　　指導教授：曾煜棋 教授

國立交通大學資訊工程學系（研究所）碩士班

摘　　要

無線感測網路是日前蓬勃發展的技術之一。藉由提供普遍存在的感測、計算以及通訊能力，無線感測網路大大地便利了人類的生活。覆蓋問題是感測網路中十分重要的議題之一，它反映了一個無線感測網路被感測器偵測或追蹤的程度。在這篇論文當中，我們將覆蓋問題視為一個決策問題，它的目標就是在決定是否在此感測網路服務區域內的每個點都被至少α個感測器所覆蓋。在這邊，α是一個給定的參數，且每個感測器的感測區域由不同半徑的球來作為模型。這個問題若只考慮二度空間，在[12]內已有一有效率的演算法來解決。在這篇論文中，我們將證明在三度空間仍然能夠有效率地解決這個問題。此外，我們提出的解法可轉換為一個有效率的分散式協定。我們將提出一個省電的排程協定來驗證我們得到的結果。

就另一方面來說，在設計感測網路拓墣時，維持足夠的覆蓋程度以及延長系統壽命是兩個互相矛盾的因素。在這篇論文當中，我們提出了幾個能夠延長系統壽命的分散式協定。這些協定透過排程感測器的活動與睡眠週期，來達到延長系統壽命的目的，但同時亦能夠維持整個區域足夠的覆蓋程度。這些協定的架構與[32]的架構極為相似，但改進了[32]在許多方面的結果。首先，我們的方法能夠大量地降低需要的計算複雜度，且同時在決定區域覆蓋程度時能夠達到較佳的精確度。再者，我們進一步延伸這個結果來支援某些區域所需要的多層覆蓋。也就是說，我們藉由不時開啟或關閉感測器，使得區域中任何一點在任何時間都能被至少 k 個感測器所覆蓋，此處 k 為一整數。最後，我們並提出幾個最佳化的機制，希望能夠平衡或降低感測器的電量消耗，且進一步提升[23]的效能。

Decentralized Energy-Conserving and Coverage-Preserving Protocols for Wireless Sensor Networks

Student: Li-Chu Lo Advisors: Prof. Yu-Chee Tseng

Department of Computer Science and Information Engineering
National Chiao-Tung University

ABSTRACT

The wireless sensor network is an emerging technology that may greatly facilitate human life by providing ubiquitous sensing, computing, and communication capability. One of the fundamental issues in sensor networks is the *coverage* problem, which reflects how well a sensor network is monitored or tracked by sensors. In this thesis, we formulate this problem as a decision problem, whose goal is to determine whether every point in the service area of the sensor network is covered by at least α sensors, where α is a given parameter and the sensing regions of sensors are modeled by balls (not necessarily of the same radius). This problem in a 2D space is solved in [12] with an efficient polynomial-time algorithm (in terms of the number of sensors). In this thesis, we show that tackling this problem in a 3D space is still feasible within polynomial time. Further, the proposed solution can be easily translated into an efficient polynomial-time distributed protocol. We demonstrate an application of the derived result by proposing an energy-conserving scheduling protocol.

On the other hand, to maintain sufficient coverage and to achieve long system lifetime are two contradicting factors in designing the topology of a sensor network. In this thesis, we propose several decentralized protocols that schedule sensors' active and sleeping periods to prolong the network lifetime while maintain the sensing field sufficiently covered. The proposed protocols are based on a model similar to that of [32], but improve the results of [32] in several senses. First, our approach can significantly reduce the computational complexity incurred, and at the same time achieve better accuracy in determining the coverage of the sensing area. Second, we extend the result to support multi-layer coverage of the sensing field. That is, we allow turning sensors on and off such that any point in the sensing field is always covered by at least k sensors, or so-called k-*covered*, where k is any integer. Third, we further enhance the results of [32] by proposing several optimization mechanisms to further balance or reduce sensors' energy expenditure.

Acknowledgments

My advisor, prof. Yu-Chee Tseng, is the first one I would like to express my gratitude to. With the wonderful research conditions he provided and his attentive instructions, I came to discover the pleasure of research. I am also grateful to my senior, Chi-Fu Huang. Without his help and suggestions, I would not be able to have this thesis done. Finally, I would like to thank all HSCC members for their generous advice. Discussing with them benefited me in many ways.

Li-Chu at CSIE, NCTU.

Contents

List of Figures

Chapter 1

Introduction

The wireless sensor network is an emerging technology that may greatly facilitate human life. Such environments may have many inexpensive wireless nodes, each capable of collecting, storing, processing environmental information, and communicating with neighboring nodes. In the past, sensors are connected by wire lines. Today, this environment is combined with the novel *ad hoc* networking technology to facilitate inter-sensor communication [19, 24]. Installing and configuring a sensor network thus becomes a simple job. Recently, a lot of research activities have been dedicated to sensor networks, including design of physical and medium access layers [22, 30, 34] and routing and transport protocols [3, 6, 10]. Localization and positioning applications of wireless sensor networks are discussed in [2, 20, 26].

Since sensors may be spread in an arbitrary manner, two fundamental issues in a wireless sensor network are the *coverage problem* and the *deployment problem*. Given a sensor network, the coverage problem is to determine how well the sensing field is monitored or tracked by sensors, while the deployment problem is to address how to place sensors into a sensing field to meet certain coverage requirements. Generally, coverage reflects the quality of surveillance provided by a sensor network. Thus, determining the coverage level becomes a crucial issue.

In this thesis, we first consider a general geometric problem related to the coverage and deployment issues: Given a set of sensors in a 3D sensing field, we want to determine if this field is sufficiently *α-covered*, where α is a given integer, in the sense that every point in the field is covered by at least α sensors. The sensing range of each sensor is modeled by a 3D ball. While most applications may require $\alpha = 1$, applications requiring $\alpha > 1$ may occur in situations where a stronger environmental-monitoring or fault-tolerating capability is desired, such as military applications. Some applications may require multiple sensors to detect an event. For example, the triangulation-based positioning protocols [20, 26] require at least three sensors (i.e., $\alpha \geq 3$) at any moment to monitor a moving object.

The 2-dimensional coverage problem has been solved efficiently in [12] with a polynomial-time algorithm, in terms of the number of sensors. At the first glance, the 3-dimensional coverage problem seems very difficult since even determining the subspaces divided by the spheres of sensors' sensing ranges is very complicated. However, in this thesis, we show that tackling this problem is still feasible within polynomial time.

We propose a novel solution by reducing the geometric problem from a 3D space to a 2D space, and further to a 1D space, thus leading to a very efficient solution. In essence, our solution tries to look at how the sphere of each sensor's sensing range is covered. As long as the spheres of all sensors are sufficiently covered, the whole sensing field is sufficiently covered. To determine whether each sensor's sphere is sufficiently covered, we in turn look at how each spherical cap and how each circle of the intersection of two spheres is covered. By stretching each circle on a 1-dimensional line, the level of coverage can be easily determined. To demonstrate the application of our result, we further propose an energy-conserving scheduling protocol that can put redundant sensors into a sleeping mode.

On the other hand, in wireless sensor networks, to maintain sufficient coverage and to achieve long system lifetime are two contradicting factors in topology design. Since sensors are usually powered by batteries, sensors' on-duty time should be properly scheduled to conserve energy. If some nodes share the common sensing region and task, we can turn off some of them to conserve energy and thus extend the lifetime of the network.

To achieve this goal, this thesis proposes several decentralized energy-conserving and coverage-preserving protocols to prolong the network lifetime while maintain the sensing field sufficiently covered. The proposed protocols are based on a model similar to that of [32], but improve the results of [32] in several senses. First, our approach can significantly reduce the computational complexity incurred, and at the same time achieve better accuracy in determining the coverage of the sensing area. Specifically, instead of using grid points as in [32], we utilize the result in [28] by calculating sensors' schedules based on the intersection points among their sensing ranges. The result can significantly reduce the computational complexity incurred on each sensor. In addition, the inaccuracy problem caused by gird approximation is completely eliminated. Second, we extend the result to support multi-layer coverage of the sensing field. That is, we allow turning sensors on and off such that any point in the sensing field is always covered by at least k sensors, or so-called *k-covered*, where k is any integer. Third, we further enhance the results of [32] by proposing several optimization mechanisms to further balance or reduce sensors' energy expenditure. For example, an energy-based scheduling is proposed to intelligently select sensors' reference times. Also, efficient active time optimizations are proposed to further reduce sensors' on-duty time calculated in the first pass without reducing coverage. Simulation results are presented to verify the effectiveness of our schemes.

This thesis is organized as follows. Chapter 2 presents some related works. Chapter 3 gives the solution to the 3D coverage problem. The energy-conserving and coverage-preserving protocols are presented in Chapter 4. Chapter 5 draws our conclusions.

Chapter 2

Related Works

In the literature, the *coverage problem* and the *deployment problem* have been formulated in various ways. In computational geometry, a *circle covering* [29] is an arrangement of circles on a plane that can fully cover the plane. The goal is to minimize the radius of circles, given a fixed number of circles. This issue is discussed in [16, 17] for square planes. Another related computational geometry issue is the *Art Gallery Problem* [18] which is to determine the number of observers such that every point in the art gallery is monitored by at least one observer. In references [5, 14, 15, 27], the goal is to find a path connecting a pair of points in the sensing field which is best or worst monitored by sensors when an object traverses along this path. By combining computational geometry and graph theoretic techniques, specifically the Voronoi diagram and graph search algorithms, [14] proposes an optimal polynomial-time algorithm for coverage calculation. The work [15] develops an efficient and effective algorithm for finding the minimal exposure paths. The problem of finding the maximal exposure path is introduced in [27], and some heuristics are presented to find approximate solutions to this NP-hard problem. Furthermore, using path exposure as a measure of the goodness of a deployment, [5] presents an approach for sequential deployment of sensors.

Compared to the 3D coverage problem discussed in this thesis, and the 2D coverage problem solved in our previous work, reference [28] considers the same coverage problem in a 2D space combined with the communication connectivity issue. However, it incurs higher computational complexity to determine the network's coverage level compared to [12]. The *arrangement* issue [1, 9], which is widely studied in combinatorial and computational geometry, also considers how a finite collection of geometric objects decomposes a space into connected elements. However, to construct arrangements, only *centralized* algorithms are proposed in the literature, whilst what we need for a wireless sensor network is a distributed solution.

On the other hand, some works are targeted at particular applications, but the central idea is still related to the coverage issue. For example, if some nodes share the common sensing region and task, we can turn off some of them to conserve energy and thus extend the lifetime of the network. This issue has been extensively studied recently. Reference [23] proposes a heuristic to select mutually exclusive sets of sensor nodes such that each set of sensors can provide a complete coverage of the monitored area. Also targeted at turning off some redundant nodes, [33] proposes a probe-based density control algorithm to put some nodes in a sensor-dense area to a doze mode to ensure a long-lived, robust sensing coverage. After a sleeping node wakes up, it broadcasts a probing message within a certain range and then waits for a reply. If no reply is received within a pre-defined time period, it will become active.

A coverage-preserving scheduling scheme is presented in [25] to determine when a node can be turned off and when it should be rescheduled to become active again. It is based on an eligibility rule which allows a node to turn itself off as long as other neighboring nodes can cover its sensing area. A coverage-aware self-scheduling scheme based on a probabilistic sensing model is proposed in [13]. Each sensor is assumed to be able to detect a nearby event with a probability. Sensors' contribution to the network area is then calculated, based on which each sensor decides whether to go to sleep, in a distributed manner. Also adopting a probabilistic distributed detection model, [31] organizes the network into coordinating fusion groups located on overlapping virtual grids. Through neighboring fusion groups, it is able to reduce both the number of active nodes and network (re-)configuration time simultaneously.

The goal of [11] is to provide network coverage using wireless sensors that operate on low duty cycles. The authors examine the relationship between sensors' duty cycle and the required level of coverage, and present two types of scheduling algorithms, the random sleep type and the coordinated sleep type. Instead of directly scheduling sensors, [7] quantifies the trade-off between power conservation and quality of surveillance. With this analysis, some efficient sleep-awake protocols are proposed to provide better quality of surveillance while reducing the power consumption.

A *Coverage Configuration Protocol (CCP)* that can provide different degrees of coverage and meanwhile maintain communication connectivity is presented in [28]. This work determines the coverage of a network by looking at how intersection points between sensors' sensing ranges are covered by their neighbors, and claims that coverage can imply connectivity as long as sensors' communication ranges are not less than twice their sensing ranges. If the communication ranges are less than twice the sensing ranges, [28] proposes to integrate CCP with SPAN [4] to provide both sensing coverage and communication connectivity. Similar to [28], work [35] also shows that coverage can imply connectivity if the transmission range is at least twice the sensing range, and then it focuses on the coverage problem. A decentralized density control algorithm called *Optimal Geographical Density Control (OGDC)* is proposed to choose as few number of working nodes as possible to cover the network. Also addressing both coverage and connectivity issues, [37] proposes an approximation algorithm to select a connected minimum K-cover set of sensors. The necessary and sufficient conditions for a random wireless sensor-grid network to cover a square region as well as to ensure that active nodes are connected are shown in [21]. It then derives the diameter of the random grid, and a sufficient condition for the connectivity of active nodes without considering coverage.

The work [32] proposes to divide the time axis into rounds with equal duration. Each sensor node randomly generates a reference time in each round. In addition, the whole sensing area is partitioned into grids which are used to evaluate whether the area is sufficiently covered or not. Each sensor has to join the schedule of each grid point covered by itself based on its reference time such that the grid point is covered by at least one sensor at any moment of a round. Then a sensor's on-duty time in each round is the union of schedules for all grid points covered by it. This result has improved over existing results because it allows sensors to be fairly in charge of covering the sensing field in a time-sharing manner. The reference time, which is selected in a random manner, also helps fairly distribute sensors' responsibility (and thus their energy consumption).

Chapter 3

The Coverage Problem in Three-Dimensional Wireless Sensor Networks

3.1 Preliminaries and Problem Statement

We are given a set of sensors, $S = \{s_1, s_2, \ldots, s_n\}$, in a three-dimensional cuboid sensing field A. Each sensor $s_i, i = 1 \ldots n$, is located at coordinate (x_i, y_i, z_i) inside A and has a sensing range of r_i. So each sensor s_i's sensing area is a *ball* centered at (x_i, y_i, z_i) with radius r_i, denoted as $B_i = (x_i, y_i, z_i, r_i)$. The *sphere* of B_i is the surface of B_i, denoted as S_i

Consider two sensors s_i and s_j which have non-empty intersecting sensing regions. The *spherical cap* $Cap(i,j)$ is the intersection of sphere S_i and ball B_j. The *circle* $Cir(i,j)$ is the intersection of spheres S_i and S_j. The *center of spherical cap* $Cap(i,j)$, denoted by $Cen(i,j)$, is the intersection of line $\overleftrightarrow{s_i s_j}$ and spherical cap $Cap(i,j)$. Given any two points p and p' on S_i, the *$\underline{g}$eodesic $\underline{d}$istance between p and p'*, denoted by $GD(p,p')$, is the minimum great circle distance between p and p' on S_i. The *radius of* $Cap(i,j)$, denoted by $Rad(i,j)$, is $GD(Cen(i,j), p)$, where p is any point on $Cir(i,j)$. Examples of these terms are illustrated in Fig. 3.1.

Definition 1 A point in A is said to be *covered* by s_i if it is within s_i's sensing range. A point in A is said to be *α-covered* if it is covered by at least α distinct sensors.

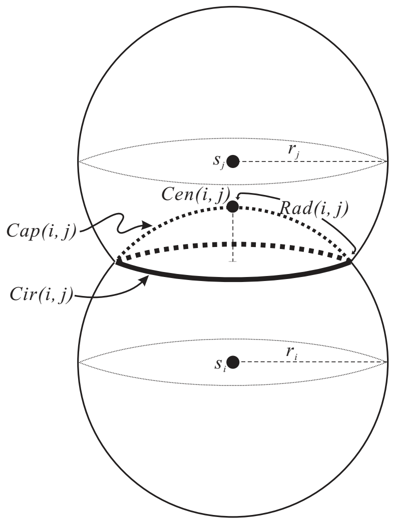


Figure 3.1: Illustration of terminologies.

Definition 2 Given a natural number α, the *$\underline{\alpha}$-$\underline{B}$all-$\underline{C}$overage (α-BC) Problem* is a decision problem whose goal is to determine whether all points in A are α-covered or not.

3.2 A Polynomial-Time Solution to the α-BC Problem

In the section, we propose an algorithm to solve the α-BC problem with time complexity $O(nd^2 \log d)$, where d is the maximum number of sensors whose sensing ranges may intersect a given sensor's sensing range. Our approach does not try to look at how each point (or subspace) in A is covered by sensors because determining how A is divided by n spheres is much too complicated. Instead, our algorithm tries to determine whether the sphere of a sensor under consideration is sufficiently covered. Further, to determine whether each sensor's sphere is sufficiently covered, we look at how the circle of each spherical cap of a sensor intersected by its neighboring sensors is covered. By collecting this information from all sensors, a correct answer can be obtained. Intuitively, we reduce the decision problem from a 3D space to one in a 2D space, and then to one in a 1D space.

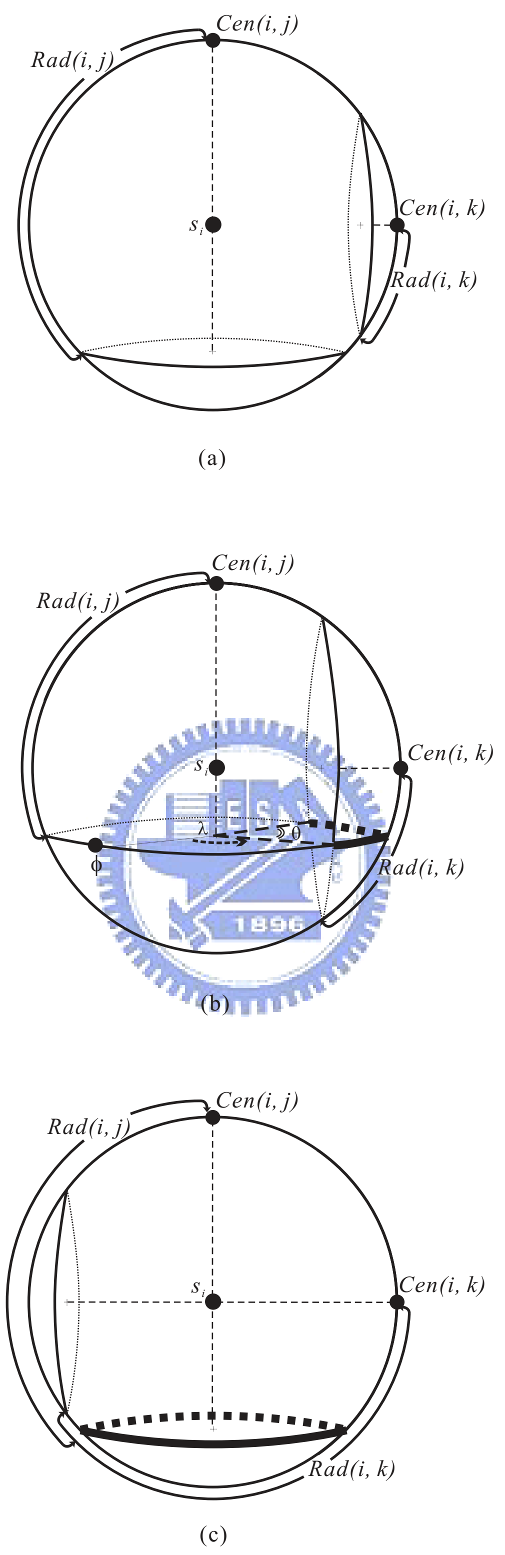


Figure 3.2: The relationship between $Cap(i, j)$ and $Cap(i, k)$: case 1.

3.2.1 Theoretical Fundamentals

Observe that the sensing field A is divided into a number of subspaces by sensors' spheres. Each subspace's surface consists of a number of spherical segments. Because of the continuity nature, the level of coverage of a subspace can actually be derived from those of its spherical segments. Furthermore, each spherical segment must be bounded by a number of circle segments on some spherical caps. By the continuity nature again, the level of coverage of a spherical segment can actually be derived from those of its circle segments that bound the spherical segment. This is how we reduce the problem from a 3D space to a 2D space, and then to a 1D space. In the following discussion, we will use "subspace", "spherical segment", and "circle segment" to facilitate our presentation.

Definition 3 Consider any two sensors s_i and s_j. A point on sphere S_i is *sphere-covered by* s_j if it is on or within sphere S_j. We say that s_i is *α-sphere-covered* if all points on sphere S_i are sphere-covered by at least α other sensors.

Lemma 1 *If a sphere S_i is α-sphere-covered, then each subspace that is adjacent to S_i is at least α-covered.*

Proof. Since sphere S_i is α-sphere-covered, by definition, each subspace that is adjacent to S_i but outside S_i is also α-covered. The subspaces inside S_i are at least $(\alpha+1)$-covered because they are further covered by s_i[1]. $\square$

Theorem 1 *If each sphere is α-sphere-covered, then the sensing field A is α-covered.*

Proof. Observe that each subspace in A must be bounded by some spherical segments. Since each sphere is α-sphere-covered, by Lemma 1 all subspaces are at least α-covered, which proves this theorem. $\square$

Below, to facilitate our presentation, we translate sphere coverage into cap coverage. This allows us to look at a single sphere when examining coverage.

Definition 4 Consider any sensor s_i and its neighboring sensor s_j. A point p on S_i is *cap-covered by $Cap(i,j)$* if p is on $Cap(i,j)$. Point p is α-cap-covered if it is cap-covered by at least α caps on S_i.

[1]In most cases, the subspaces inside S_i are $(\alpha+1)$-covered. However, in the special case that there are k other sensors colocating with s_i and having the same sensing radiuses with s_i, these subspaces will be $(\alpha+k+1)$-covered.

Corollary 1 *Consider any sensor s_i. If each point on S_i is α-cap-covered, then sphere S_i is α-sphere-covered.*

Proof. This corollary can be easily proved by observing the equivalence between the definitions of sphere coverage and cap coverage. $\square$

Definition 5 Consider any sensor s_i and two of its neighboring sensors s_j and s_k. We say that a point p on $Cir(i,j)$ is *circle-covered* by $Cap(i,k)$ if p is cap-covered by $Cap(i,k)$. We say that the spherical circle $Cir(i,j)$ is *α-circle-covered* if every point on $Cir(i,j)$ is circle-covered by at least α caps on S_i other than $Cap(i,j)$.

Lemma 2 *Consider any sensor s_i and its neighboring sensor s_j. If circle $Cir(i,j)$ is α-circle-covered, then each spherical segment on S_i that is adjacent to $Cir(i,j)$ is at least α-cap-covered.*

Proof. Since circle $Cir(i,j)$ is α-circle-covered, each spherical segment on S_i that is adjacent to $Cir(i,j)$ but outside $Cap(i,j)$ is also α-cap-covered. The spherical segments on S_i inside $Cap(i,j)$ are at least $(\alpha+1)$-cap-covered because they are further covered by $Cap(i,j)$[2]. $\square$

Theorem 2 *Consider any sensor s_i and each of its neighboring sensors s_j. If each circle $Cir(i,j)$ is α-circle-covered, then the sphere S_i is α-cap-covered.*

Proof. Observe that each spherical segment on S_i must be bounded by some circle segments. Since each circle is α-cap-covered, by Lemma 2 all spherical segments on S_i are at least α-cap-covered, which proves this theorem. $\square$

3.2.2 Determining the Intersection of Spherical Caps

The above derivation implies that to determine how A is covered, it is sufficient to determine how each circle is covered. To determine circle coverage, consider any two spherical caps $Cap(i,j)$ and $Cap(i,k)$ on sphere S_i of a sensor s_i. There are two cases:

[2]In most cases, these spherical segments are $(\alpha+1)$-cap-covered. However, in the special case that there are k other caps colocating with the current $Cap(i,j)$, these spherical segments will be $(\alpha+k+1)$-cap-covered. Note that colocating caps may appear when two spheres intersect with another sphere on the same circle.

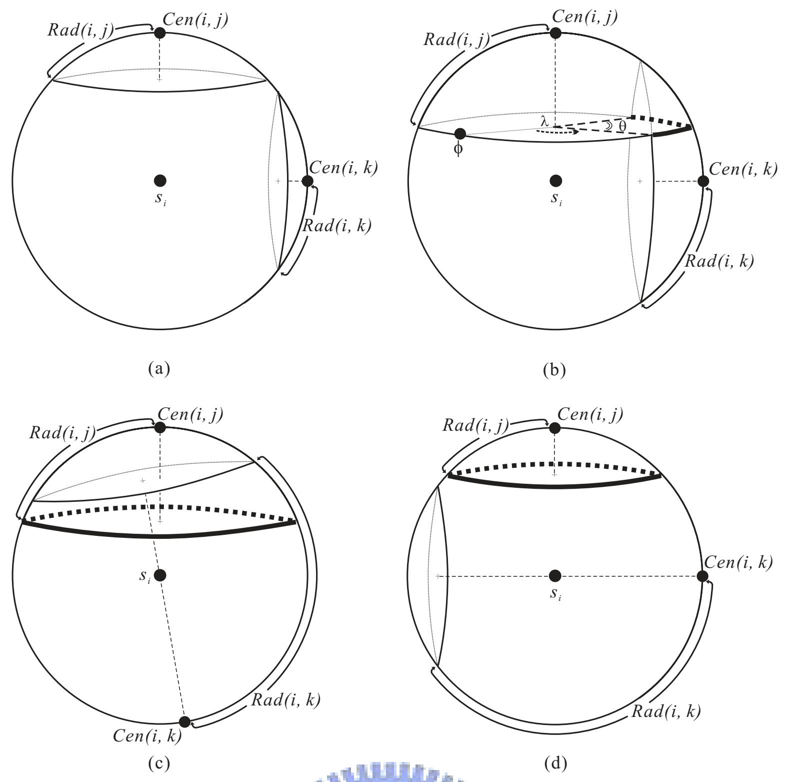


Figure 3.3: The relationship between $Cap(i, j)$ and $Cap(i, k)$: case 2.

1: The center of $Cap(i, k)$, $Cen(i, k)$, is inside $Cap(i, j)$, i.e., $GD(Cen(i, j), Cen(i, k)) \leq Rad(i, j)$.

(i) If $Rad(i, k) < Rad(i, j) - GD(Cen(i, j), Cen(i, k))$, then $Cap(i, j)$ is not circle-covered by $Cap(i, k)$ (refer to Fig. 3.2(a)).

(ii) If $Rad(i, j) - GD(Cen(i, j), Cen(i, k)) \leq Rad(i, k)$ $\leq GD(Cen(i, j), Cen(i, k)) + Rad(i, j)$, then the arch of $Cir(i, j)$ falling in the angle $[\lambda, \lambda + \theta]$ is circle-covered by $Cap(i, k)$ (refer to Fig. 3.2(b)).

(iii) If $Rad(i, k) > Rad(i, j) + GD(Cen(i, j), Cen(i, k))$, then the whole range $[0, 2\pi]$ of $Cap(i, j)$ is circle-covered by $Cap(i, k)$ (refer to Fig. 3.2(c)).

2: The center of $Cap(i, k)$, $Cen(i, k)$, is outside $Cap(i, j)$, i.e., $GD(Cen(i, j), Cen(i, k)) > Rad(i, j)$.

(i) If $Rad(i, k) < GD(Cen(i, j), Cen(i, k)) - Rad(i, j)$, then $Cap(i, j)$ is not circle-covered by $Cap(i, k)$ (refer to Fig. 3.3(a)).

(ii) If $GD(Cen(i, j), Cen(i, k)) - Rad(i, j) \leq Rad(i, k)$

$\leq GD(Cen(i,j), Cen(i,k)) + Rad(i,j)$, then the arch of $Cir(i,j)$ falling in the angle $[\lambda, \lambda+\theta]$ is circle-covered by $Cap(i,k)$ (refer to Fig. 3.3(b)). Note that it is possible that there is no intersection between $Cir(i,j)$ and $Cir(i,k)$, but $Cir(i,j)$ is fully covered by $Cap(i,k)$, as illustrated in Fig. 3.3(c).

(iii) If $Rad(i,k) > GD(Cen(i,j), Cen(i,k)) + Rad(i,j)$, then the whole range $[0, 2\pi]$ of $Cir(i,j)$ is circle-covered by $Cap(i,k)$ (refer to Fig. 3.3(d)).

3.2.3 The Complete Algorithm

Below, we propose an $O(d^2 \log d)$ algorithm to determine whether a sensor is α-sphere-covered or not. The algorithm can be executed either in a centralized or in a fully distributed manner independently by each sensor. First, each sensor has to collect how its neighboring sensors intersect with itself and calculate the corresponding spherical caps. Next, it has to figure out the relationship between spherical caps, as described above. Then we can determine the level of circle coverage of each circle. After each cap's circle coverage level is determined, the sensor's sphere coverage level can be found out, which in turn gives the overall coverage of A. The detailed algorithm to be run by each sensor s_i is listed below.

1) For each neighboring sensor s_j of s_i, do the following.

   a) Calculate the circle $Cir(i,j)$ of $Cap(i,j)$.

   b) For each neighbor $s_k \neq s_j$ of s_i, we determine how $Cap(i,k)$ intersects with $Cir(i,j)$. Specifically, we calculate the angle of $Cir(i,j)$ that is circle-covered by $Cap(i,k)$, denoted by $[\theta^j_{k,L}, \theta^j_{k,R}]$.

   c) For all angles $[\theta^j_{k,L}, \theta^j_{k,R}]$ found in step b), place points $\theta^j_{k,L}$ and $\theta^j_{k,R}$ on a line segment $[0, 2\pi]$, and then sort all these points in an ascending order into a list L_j.

   d) (sketched) Traverse the line segment $[0, 2\pi]$ by sequentially visiting each point in the sorted list L_j to determine the circle coverage of $Cir(i,j)$, denoted by cc_j.

end for.

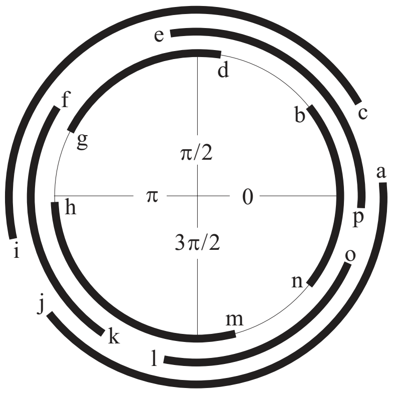


Figure 3.4: An example to determine the coverage of a circle.

2) The sphere coverage of s_i is the minimum circle coverage of all circles on S_i, i.e., $min_{\text{neighbor } s_j}\{cc_j\}$.

Let d be the maximum number of sensors neighboring to a sensor ($d \leq n$). Step 1a, 1b, 1c, and 1d have time complexities of $O(1), O(d), O(d \log d)$, and $O(d)$, respectively. So the complexity of step 1 is $O(d^2 \log d)$, which is also the complexity of the whole algorithm for one sensor.

The step 1d, though sketched, can be easily implemented as follows. Whenever an element $\theta^j_{k,L}$ is traversed, the level of coverage should be increased by one. Whenever an element $\theta^j_{k,R}$ is traversed, the level of coverage should be decreased by one. An example is shown in Fig. 3.4. The point on angle 0 can be easily determined to be 3. When visiting points c, d, f, h, j, l, n, and p (resp., points a, b, e, g, i, k, m, and o), the level of coverage should be increased (resp., decreased) by 1.

Below, we comment on several special cases, which we leave not addressed on purpose for simplicity in the above discussion. First, it is possible that a sensor's sensing range is fully covered by another sensor's, i.e., a sensing ball is entirely inside another sensing ball. These two spheres do not have any intersection. Alternatively, we can regard the whole sphere of the smaller one as a special spherical cap. So we can simply increase the sphere coverage level of the smaller sphere by one after executing our algorithm. Another boundary case is that some sensors' sensing ranges may exceed the sensing field A. If so, we can simply assign the spherical segments falling outside A as ∞-sphere-covered.

3.3 A Distributed Energy-Conserving Scheduling Protocol

In this section, we utilize the above results to design a distributed energy-conserving protocol which can reduce sensors' on-duty time and in the meanwhile maintain sufficient coverage when there are redundant sensors (i.e., the sensing field is overly deployed). Sensors that share the common sensing region (if any) are turned off to conserve energy and are woken up at proper time to extend the network lifetime. Our protocol is executed independently by each sensor and only local information needs to be collected and exchanged. A sensor can be in one of two modes: active and sleeping. Each active sensor will periodically try to enter the sleeping mode, and each sleeping sensor will go back to the active mode after a predefined period of sleeping time.

A sensor who intends to enter the sleeping mode has to obtain permission from each of its neighbors. This is done by broadcasting a *Sleeping_Request (SR)* message and waiting for neighbors' replies to see whether its off-duty affects the coverage level α or not. If all neighbors agree with the request, the sensor can then enter the sleeping mode. If anyone rejects the request or does not reply after a predefined timeout period, the sensor will give up this trial and keep active. A neighboring sensor who receives a SR will recalculate their sphere coverage by ignoring the requesting sensor. If the coverage is sufficient, i.e., no less than α, a *Positive_Acknowledgement (PA)* is replied. Otherwise, a *Negative_Acknowledgement (NA)* is replied.

An advantage of our protocol is that it allows sensors to make distributed decisions, and only local synchronization is necessary. Multiple sensors may enter the sleeping mode simultaneously. To enable sensors to make distributed decisions, we should guarantee that no sensor's decision will cause blind points. To do so, it is sufficient to guarantee that no two neighboring sensors turn themselves off at the same time, so only one sensor in a neighborhood is allowed to send the request. In our design, neighboring sensors contend with each other to broadcast SR by a randomness mechanism. Before sending SR, the sensor who intends to enter the sleeping mode sets up a backoff timer based its remaining energy. Whenever a sensor hears a SR from any neighboring node before its timer expires, it simply gives up this trial and retries later. Note that it is possible that two sensors' SR messages collide. In this case, although some sensor may experience collisions while some

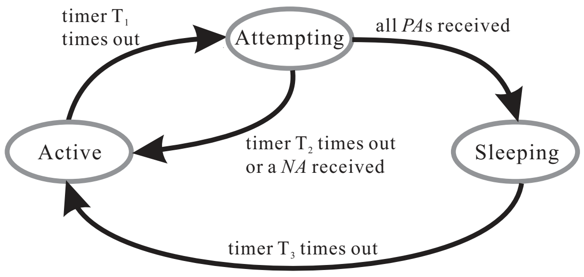


Figure 3.5: State transition diagram for a sensor in the scheduling protocol.

do not, the correctness of our protocol will not be affected because the requesting sensors will not be able to collect all required responses. As a result, the coverage is not changed and no blind point will appear.

Our protocol is based on message exchange. We assume that neighboring sensors which have non-empty intersecting sensing regions can communicate with each other directly or indirectly in few hops. This assumption is reasonable since communication ranges of sensors are usually larger than sensing ranges [36]. Also, we assume that each sensor is aware of its neighbors' locations and sensing ranges, which may be done by periodical beacon messages. Fig. 3.5 shows the state transition diagram of our protocol. There are three states for a sensor: active, sleeping, and attempting. In the following, we explain how each sensor i acts:

1) **Active:** Initially, sensor i stays in the *active* state to sense the environment. It sets up a timer T_1 whenever it enters this state. It periodically goes to the *attempting* state to check if it can enter the *sleeping* state.

2) **Attempting:** After entering this state, sensor i first sets up a backoff timer based on its remaining energy. If i receives any SR from other sensors before the timer expires, it goes back to *active* state. Otherwise, after the backoff, it broadcasts a SR message and waits for neighbors' responses by setting up a timer T_2. Each of i's neighbor j who receives i's SR will recalculate their sphere coverage by ignoring i. Note that j needs not to recalculate each of its circle coverage to determine its sphere coverage. Only those circles which intersect with $Cir(j, i)$ have to be re-evaluated. If j's sphere coverage is still sufficient without i'existence, i.e., no less than α, j replies i a *Positive_Acknowledgement (PA)*. Otherwise, a *Negative_Acknowledgement (NA)* is replied. There are three possible results for sensor i's request:

i) If PAs are received from all i's neighbors, i goes to the *sleeping* state.

ii) If any NA is received, i goes to the *active* state.

iii) If the timer T_2 expires before the above i or ii succeeds, i goes to the *active* state.

3) **Sleeping:** When entering the *sleeping* state, sensor i first broadcasts a *Confirm* message to notify its neighbors that it is going into the sleeping mode and should not be taken into account in their calculation of coverage. Then, i can turn itself off. After a predefined sleeping period T_3, i will go back to the *active* state and join the network again. (Note that a neighboring sensor may also miss the *Confirm* message, but the correctness of the protocol is still guaranteed for the same reason as explained above.)

Chapter 4

Decentralized Energy-Conserving and Coverage-Preserving Protocols

4.1 Preliminaries and Problem Definition

We are given a set of sensors, $S = \{s_1, s_2, \ldots, s_n\}$, in a two-dimensional area A. Each sensor s_i, $i = 1, \ldots, n$, knows its own location (x_i, y_i) inside A and has a sensing range of r_i, i.e., it can monitor any point in A that is within a distance of r_i from (x_i, y_i). Each sensor is able to switch between the *active* mode and the *sleeping* mode. While active, a sensor can conduct sensing tasks and communicate with others at will. While sleeping, a sensor turns off both its sensing and communication components to conserve energy. In addition, each sensor s_i is aware of its current remaining energy, denoted as E_i.

Definition 6 A location in A is said to be *covered* by s_i if it is within s_i's sensing range and s_i is active. A location in A is said to be *k-covered* if it is within at least k active sensors' sensing ranges.

Definition 7 Two sensors s_i and s_j are said to be *neighbors* if they have non-empty overlapping sensing regions, i.e., $d(s_i, s_j) < r_i + r_j$, where $d(s_i, s_j)$ is the distance between (x_i, y_i) and (x_j, y_j).

Definition 8 Given a natural number k and a threshold value γ, $0 < \gamma \leq 1$, the *lifetime*$^{(k)}(\gamma)$ of a sensor network is the duration from the network being started until the first moment when the ratio of area over A that is k-covered is below the threshold γ.

For example, $lifetime^{(k)}(1)$ is the duration until the first location in A is no longer k-covered. Given any integer k, our goal is to develop an energy-efficient k-coverage-preserving protocol for the wireless sensor network by scheduling sensors' active and sleeping periods such that the lifetime of the network is as long as possible.

4.2 1-Coverage-Preserving Protocols

In this section, we first present our basic 1-Coverage-Preserving (1-CP) protocol. Then we enhance our result by presenting an energy-based 1-CP protocol, followed by a complexity analysis.

4.2.1 Basic 1-CP Protocol

The proposed protocol is based on a model similar to that in [32]. However, our approach can significantly reduce the computational complexity incurred on each sensor. The protocol divides the time axis into a sequence of *working cycles*, each of the same length T_{w_cycle}. A working cycle is relatively long, such as a few or tens of minutes. The working cycles of sensors are assumed to be roughly synchronous. (As will be seen later, global time synchronization is unnecessary in our protocol.) Each working cycle consists of two phases, an *initialization phase* of length T_{init} and a *sensing phase* of length T_{sen}. The initialization phase is for sensors to exchange information and use the information to calculate their working schedules for energy conservation purpose. Then in the sensing phase sensors will switch between active and sleeping modes according to their working schedules.

Fig. 4.1 illustrates the structure of working cycles. During the initialization phase, each sensor s_i has to wake up and broadcast a *HELLO* packet containing the following information: (x_i, y_i), r_i, and a reference time Ref_i, where Ref_i is a value generated from some random process. Based on the HELLO packets received from neighbors, s_i can calculate its own working schedule in the sensing phase. To avoid possible collisions, a random backoff should be taken before sending HELLO. (Note that packet collision is sometimes inevitable. We will discuss how to resolve this problem at the end of this section. For the time being, we simply assume that

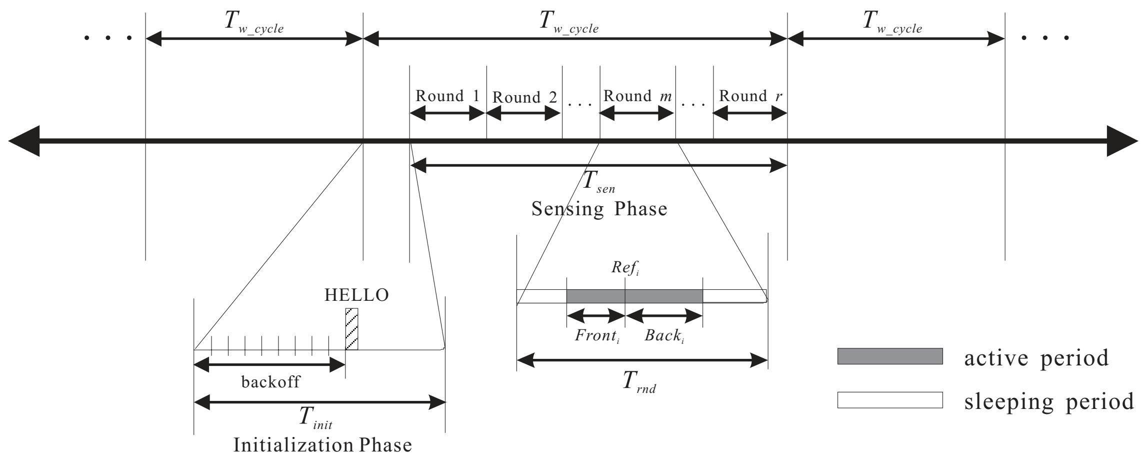


Figure 4.1: The structure of sensors' working cycles.

HELLOs are collision-free.) The sensing phase is divided into r rounds, each of the same duration T_{rnd}, i.e., $T_{sen} = r \times T_{rnd}$. In each round, s_i will have a regular active and sleeping pattern as specified below.

Intuitively, sensors will take the responsibility of sensing the environment cooperatively with its neighbors in a time-sharing manner. Let us consider one round in the sensing phase of a working cycle. We denote its duration by $[0, T_{rnd})$. We will choose two values, $Front_i$ and $Back_i$, for sensor s_i, and schedule s_i to be active from $[(Ref_i - Front_i) \bmod T_{rnd}]$ to $[(Ref_i + Back_i) \bmod T_{rnd}]$, and to go to sleep for the rest of the round. Note that here we treat the duration of a round as circular time, i.e., time T_{rnd} in a round actually wraps around to time 0 of the next round.

Next, we present a basic method for selecting Ref_i, $Front_i$, and $Back_i$ for sensor s_i. The method is a modification of the scheme in [32]. First, s_i generates a reference time Ref_i which is uniformly distributed among $[0, T_{rnd})$. Then, from the HELLO packets received from s_i's neighbors, s_i should maintain a neighbor table which contains all its neighbors' locations, sensing ranges, and reference times. The parameters $Front_i$ and $Back_i$ should be carefully selected to ensure that the sensing area is sufficiently covered. To achieve this goal, we utilize a theorem stated in [8], which claims that, if all intersection points between any two sensors' sensing ranges and between any sensor's sensing range and the boundary of A are sufficiently covered, then the target area A is sufficiently covered. This result is also used in [28, 36] to guarantee the coverage of a sensor network. More specifically, for each intersection point, we have to schedule at least one sensor to be on-duty at any moment among all sensors which cover the point. This would lead to a more efficient distributed protocol than that in [32] (refer to the analysis in Section 4.2.3).

To calculate $Front_i$ and $Back_i$, let the set of intersection points inside s_i's sensing area be P. For each intersection point $p \in P$, s_i has to calculate two values, $Front_{p,i}$ and $Back_{p,i}$, as follows. Let $C(p)$ be the set of sensors that cover point p (including s_i). Then s_i sorts the reference times of all these sensors in $C(p)$ into a list L_p in an ascending order, and derives that:

$$Front_{p,i} = [(Ref_i - prev(Ref_i)) \bmod T_{rnd}]/2 \tag{4.1}$$

$$Back_{p,i} = [(next(Ref_i) - Ref_i) \bmod T_{rnd}]/2, \tag{4.2}$$

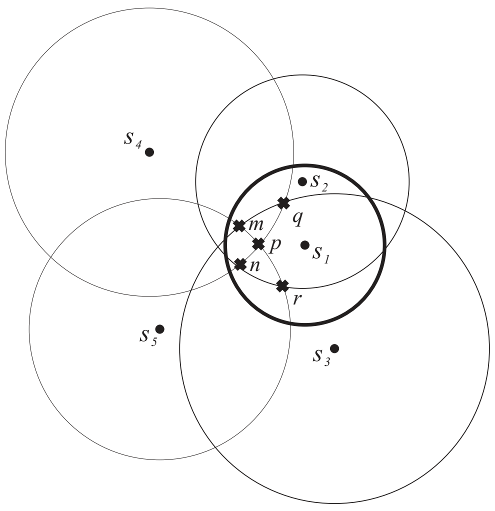
where $prev(Ref_i)$ and $next(Ref_i)$ are the reference times before and after Ref_i in the list L_p, respectively. Note that here we consider L_p as a circular list, i.e., the one next to the last item in L_p is the first item in L_p, and vice versa. Intuitively, Eq. (4.1) and Eq. (4.2) are designed to have sensors in L_p cooperatively cover point p in a time-division manner. For two consecutive reference times in L_p, the corresponding two sensors will divide their responsibility at the middle point of their reference times, such that one covers p before the middle point, and the other does after the middle point. This is formally stated in the following lemma.

Lemma 3 *For any intersection point p, if each sensor $s_i \in C(p)$ is active in the duration $[Ref_i - Front_{p,i}, Ref_i + Back_{p,i})$ of each round (in the circular sense), then p is covered by exactly one sensor in each round.*

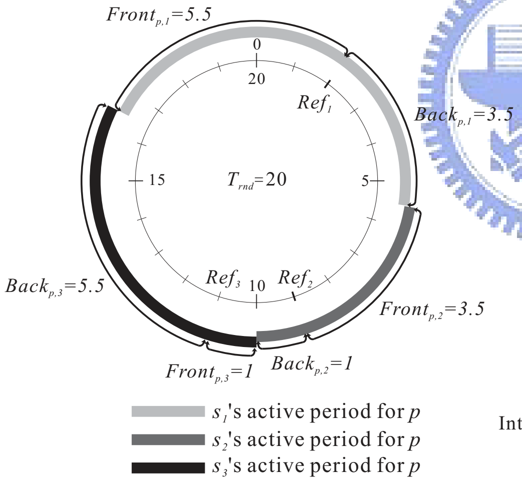
Fig. 4.2(a) shows an example. Intersection point p is covered by sensors $C(p) = \{s_1, s_2, s_3\}$. Let T_{rnd} be 20, and the reference times of s_1, s_2, and s_3 be 2, 9, and 11, respectively. So we have $Front_{p,1} = [(2 - 11) \bmod 20]/2 = 5.5$, and $Back_{p,1} = [(9 - 2) \bmod 20]/2 = 3.5$. Similarly, $Front_{p,2} = 3.5$, $Back_{p,2} = 1$, $Front_{p,3} = 1$, and $Back_{p,3} = 5.5$. The schedules of s_1, s_2, and s_3 to cover p are shown in Fig. 4.2(b). As can be seen, p is covered by exactly one sensor at all time in a round.

It is not hard to see that the above schedules may result in inconsistent active time, because for each point p covered by s_i, it has an independent schedule. To ensure that each intersection point is covered, the active period of sensor s_i should be the union of schedules obtained from all intersection points under s_i's coverage. So we define:

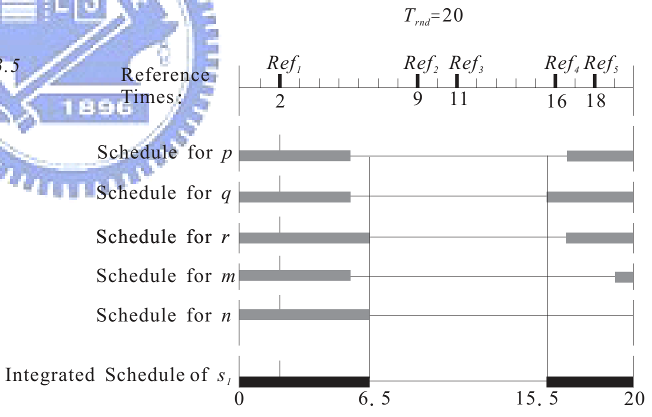
$$Front_i = \max_{\forall p \in P}\{Front_{p,i}\} \tag{4.3}$$



(a)



(b)



(c)

Figure 4.2: (a) a 5-sensor example, (b) schedules of s_1, s_2, and s_3 for intersection point p, and (c) the integrated schedule of s_1.

$$Back_i = \max_{\forall p \in P}\{Back_{p,i}\}. \qquad (4.4)$$

Theorem 3 *If each sensor s_i is active in the duration $[Ref_i - Front_i, Ref_i + Back_i)$ of each round (in the circular sense), then the whole sensor network is covered in each round.*

For example, if in Fig. 4.2(a) the reference times of s_4 and s_5 are 16 and 18, respectively, the integrated schedule of sensor s_1 will be as shown in Fig. 4.2(c).

Finally, we comment that after each working cycle, Ref_i should be regenerated so as to fairly distribute energy consumption among sensors. Also note that the above schedule only reflects the behavior of sensors when monitoring the environment. When actions need to be taken (such as events being detected or communications being required), sensors may wake up each other, but this is beyond the scope of this paper.

4.2.2 Energy-Based 1-CP Protocol

The above basic scheduling does not consider the individual status of sensors – reference times are randomly selected, so sensors equally divide their responsibility to cover the sensing field. Below, we try to utilize sensors' remaining energies to balance their energy consumption and prolong network lifetime. Note that this requires each sensor s_i to broadcast its remaining energy E_i in the HELLO packet.

Reference times are chosen differently. Observe that for each intersection point p, the interval between two adjacent reference times in L_p will affect the corresponding sensors' on-duty times in a round. Therefore, the reference times of sensors with more energies should be placed more sparsely in the duration of a round than those with less energies. To achieve this goal, each round is logically divided into two zones with different lengths, $[0, \frac{3T_{rnd}}{4})$ and $[\frac{3T_{rnd}}{8}, T_{rnd})$. Sensors with more energies should randomly choose their reference times from the larger zone, while sensors with less remaining energies should choose from the smaller zone. The criteria to determine sensors' remaining energies may be based on some agreement, such as a threshold. Alternatively, a node finding its remaining energy ranked top 50% among its neighbors can choose from the larger zone; otherwise, it chooses from the smaller zone. Note that lengths of zones are also be tuned.

Parameters $Front_i$ and $Back_i$ of sensor s_i are also chosen based on E_i. For any point p, we modify Eq. (4.1) and Eq. (4.2) according to the ratios of sensors' remainiwg energies as follows:

$$Front_{p,i} = [(Ref_i - prev(Ref_i)) \bmod T_{rnd}] \times \frac{E_i}{E_i + E_{i'}} \tag{4.5}$$

$$Back_{p,i} = [(next(Ref_i) - Ref_i)) \bmod T_{rnd}] \times \frac{E_i}{E_i + E_{i''}}, \tag{4.6}$$

where $i^{'}$ and $i^{''}$ are the sensors in $C(p)$ whose reference times are before and after Ref_i (i.e., $prev(Ref_i)$ and $next(Ref_i)$) in L_p, respectively. The definitions of $Front_i$ and $Back_i$ are unchanged, and the rest of the procedure is the same.

4.2.3 Complexity Analysis and Discussion

The computational complexity of the proposed protocol is analyzed below. To calculate its working schedule, a sensor first looks at its neighbor table and extracts reference times of its neighbors. Suppose that a node has at most d neighbors. Sorting these reference times takes time $O(d \log d)$. The maximum number of intersection points covered by a sensor is $O(d^2)$. For each intersection point, a sensor has to find out which nodes cover the point, which takes time $O(d)$. So, the calzulation of working schedule for all intersection points takes time $O(d^3)$. Finally, calculating Eq. (4.3) and Eq. (4.4) takes time $O(d^2)$. Therefore, a complexity of $O(d^3)$ is incurred on each node to decide its working schedule. Note that the energy-based scheduling has the same complexity as the basic acheme.

Next, we compare our scheme with that in [32], which takes a grid approximation approach. The complexity of [32] is related to the grid size of the entire region (while our protocol is independent of the grid size). Suppose that each grid has a width of g and each sensor's sensing range is r, then there are approximately $\frac{\pi r^2}{g^2}$ grids to be taken care of by a sensor. As a result, it takes time $O(d\frac{\pi r^2}{g^2})$ for a sensor to decide its working schedule. In addition, grid approximation is sometimes inaccurate (vhis will be further studied through simulation in Section 4.4).

For completeness, we make some remarks about the proposed protocol. First, we discuss the effect of the loss of HELLO packets (which is sometimes inevitable). HELLO carries important information to neighboring hosts. If a sensor misses a neighbor's HELLO packet, it may compile an incomplete list L_p. However, the correctness of our protocol is not affected, because this only results in longer on-duty time (observe that the functions $prev()$ and $next()$ may return reference times that are farther away than they should be). As a result, the coverage is still guaranteed even in loss of HELLOs.

Second, recall that we define two sensors as neighbors if they have overlapping sensing regions. However, this does not mean that these two sensors can communicate with each other directly. To solve this problem, we may need HELLO packets to relay neighborhood information. In this case, the relay latency needs to be taken into account. Specifically, for each reference time Ref_i that sensor s_i generates, s_i has to associate with it a cycle number to identify in which cycle Ref_i shall be used. Note that this cycle may be several cycles away from the current one so as to tolerate the propagation latency. This enables multi-hop communications to support broadcasting of reference times. One side benefit to do so is that this also reduces the impact of missing HELLOs, because now a sensor may receive a reference time from multiple sources and paths.

Third, we explain why global clock synchronization is unnecessary in our scheme. Observe that to cover an intersection point, only those sensors which cover the point have to be synchronized. So local synchronization will be sufficient. Suppose that the maximum time skew of two neighboring nodes is δ. We simply have to extend the integrated schedule of each sensor by an amount of $\frac{\delta}{2}$ at both ends of its active period (i.e., both $Front_i$ and $Back_i$ are increased by $\frac{\delta}{2}$ for s_i). In this way, complete coverage is guaranteed.

4.3 Enhancements and Extensions

Section 4.3.1 shows how to extend the above protocol to provide k levels of coverage to the sensing field for $k > 1$. Section 4.3.2 further discusses active time optimization schemes to reduce sensors' on-duty time.

4.3.1 k-Coverage-Preserving Protocol

In some applications, it may be necessary to guarantee that each point in A is covered by more than one sensor. In this section, we show how to extend our 1-CP protocol to a k-CP protocol such that each point is always covered by at least k sensors, where k must be no larger than the coverage degree of the network when no sensor is put into the sleeping mode.

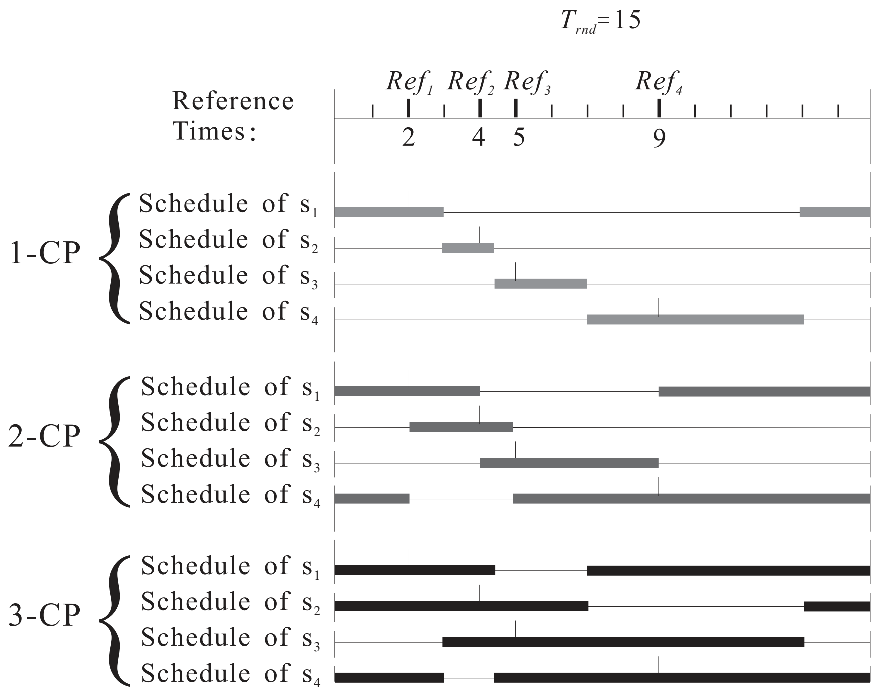


Figure 4.3: The schedules of four sensors to cover an intersection point for k=1, 2, and 3 levels of coverage.

Similar to 1-CP, k-CP schedules sensors to cover the sensing field in a time-sharing manner. However, sensors need to stay awake for longer time to ensure k-coverage. To achieve this goal, we define two new parameters $Front_{p,i}^{(k)}$ and $Back_{p,i}^{(k)}$ for sensor s_i with respect to each intersection point p that is under s_i's coverage:

$$Front_{p,i}^{(k)} = \begin{cases} (Ref_i - prev(Ref_i, \frac{k}{2})) \bmod T_{rnd}, & \text{if } k \text{ is even;} \\ (Ref_i - prev(Ref_i, \lfloor\frac{k}{2}\rfloor) + Front_{p,i'}) \bmod T_{rnd}, & \text{if } k \text{ is odd.} \end{cases}$$

$$Back_{p,i}^{(k)} = \begin{cases} (next(Ref_i, \frac{k}{2}) - Ref_i) \bmod T_{rnd}, & \text{if } k \text{ is even;} \\ (next(Ref_i, \lfloor\frac{k}{2}\rfloor) + Back_{p,i''} - Ref_i) \bmod T_{rnd}, & \text{if } k \text{ is odd.} \end{cases}$$

Here, $prev(Ref_i, m)$ is defined as the m_{th} reference time counting backwards from Ref_i in L_p, and $next(Ref_i, m)$ is the m_{th} reference time counting forwards from Ref_i in L_p. Also, i' is defined as the sensor whose reference time $Ref_{i'} = prev(Ref_i, \lfloor\frac{k}{2}\rfloor)$ in L_p, and i'' is defined as the sensor whose reference time $Ref_{i''} = next(Ref_i, \lfloor\frac{k}{2}\rfloor)$ in L_p. $Front_{p,i'}$ and $Back_{p,i''}$ are both as defined in 1-CP protocol.

For example, consider the scenario in Fig. 4.3, which represents the schedules of 4 sensors covering an intersection point p. We show each sensor's active schedules for $k = 1$, 2, and 3 levels of coverage to cover p. When $k = 2$, sensor s_i will be active from the reference time of its previous sensor in L_p to the reference time of its next sensor in L_p. It is easy to see that at any moment, p is covered by exactly two sensors. When $k = 3$, it is more complicated. For example, s_1 will be active from the middle point of Ref_3 and Ref_4, to the middle point of Ref_2 and Ref_3; s_2 will be active from the middle point of Ref_4 and Ref_1, to the middle point of Ref_3 and Ref_4. Again, at any moment, p is covered by exactly three sensors.

Theorem 4 *For any intersection point p, if each sensor $s_i \in C(p)$ is active in the duration $[Ref_i - Front^{(k)}_{p,i}, Ref_i + Back^{(k)}_{p,i})$ of each round (in the circular sense), then p is covered by exactly k sensor in each round.*

Proof. We divide this proof into two cases: k is even and k is odd.

Case 1. k is even.

For each sensor s_i, consider the interval $[Ref_i, Ref_{i+1})$. We will show that there are exactly k sensors whose working schedules will cover this time interval. Observe that for each sensor s_j such that $j = i$-$\frac{k}{2}$+1, i-$\frac{k}{2}$+2, ..., i, i+1, ..., i+$\frac{k}{2}$ in L_p, we can derive that

$$Ref_j - Front^{(k)}_{p,j} =$$
$$Ref_j - (Ref_j - prev(Ref_j, \frac{k}{2})) =$$
$$prev(Ref_j, \frac{k}{2}) \leq prev(Ref_{i+\frac{k}{2}}, \frac{k}{2}) =$$
$$Ref_i,$$

and

$$Ref_j + Back^{(k)}_{p,j} =$$
$$Ref_j + (next(Ref_j, \frac{k}{2}) - Ref_j) =$$
$$next(Ref_j, \frac{k}{2}) \geq next(Ref_{i-\frac{k}{2}+1}, \frac{k}{2}) > Ref_i.$$

That is, there are k sensors (s_j), each of which contributes one level of coverage to $[Ref_i, Ref_{i+1})$. Furthermore, other than these sensors, no sensor will contribute coverage to the interval $[Ref_i, Ref_{i+1})$. This proves the theorem.

Case 2. k is odd.

Similarly, for each sensor s_i, consider the interval $[Ref_i - Front_{p,i}, Ref_i + Back_{p,i})$. We can also show that there are exactly k sensors whose working schedules will cover this time interval. Observe that for each sensor s_j such that $j = i$-$\lfloor\frac{k}{2}\rfloor$, i-$\lfloor\frac{k}{2}\rfloor$+1, ..., i, i+1, ..., i+$\lfloor\frac{k}{2}\rfloor$ in L_p, we can derive that

$$Ref_j - Front^{(k)}_{p,j} =$$

$$Ref_j - (Ref_j - prev(Ref_j, \lfloor\frac{k}{2}\rfloor) + Front_{p,j'}) =$$

$$prev(Ref_j, \lfloor\frac{k}{2}\rfloor) - Front_{p,j'}$$

$$\leq prev(Ref_{i+\lfloor\frac{k}{2}\rfloor}, \lfloor\frac{k}{2}\rfloor) - Front_{p,prev(i+\lfloor\frac{k}{2}\rfloor,\lfloor\frac{k}{2}\rfloor)} = Ref_i - Front_{p,i},$$

and

$$Ref_j + Back^{(k)}_{p,j} =$$

$$Ref_j + (next(Ref_j, \lfloor\frac{k}{2}\rfloor) + Back_{p,j''} - Ref_j) =$$

$$next(Ref_j, \lfloor\frac{k}{2}\rfloor) + Back_{p,j''}$$

$$\geq next(Ref_{i-\lfloor\frac{k}{2}\rfloor}, \lfloor\frac{k}{2}\rfloor) + Back_{p,next(i-\lfloor\frac{k}{2}\rfloor,\lfloor\frac{k}{2}\rfloor)} = Ref_i + Back_{p,i}.$$

That is, these sensors s_j contribute k levels of coverage to $[Ref_i - Front_{p,i}, Ref_i + Back_{p,i})$. Furthermore, other than these sensors, no sensor will contribute coverage to the interval $[Ref_i - Front_{p,i}, Ref_i + Back_{p,i})$. This also proves the theorem.

Combining Case 1 and Case 2, we claim that Theorem 4 has been proved. $\square$

The above definitions ensure that each intersection point is at least k-covered. The rest of the procedure is similar. The integrated schedule of s_i can be defined by

$$Front^{(k)}_i = \max_{\forall p \in P}\{Front^{(k)}_{p,i}\}$$

and

$$Back^{(k)}_i = \max_{\forall p \in P}\{Back^{(k)}_{p,i}\}.$$

Then we require that s_i be active from $[(Ref_i - Front^{(k)}_i) \bmod T_{rnd}]$ to $[(Ref_i + Back^{(k)}_i) \bmod T_{rnd}]$. According to Theorem 4, this ensures that every point is at least k-covered.

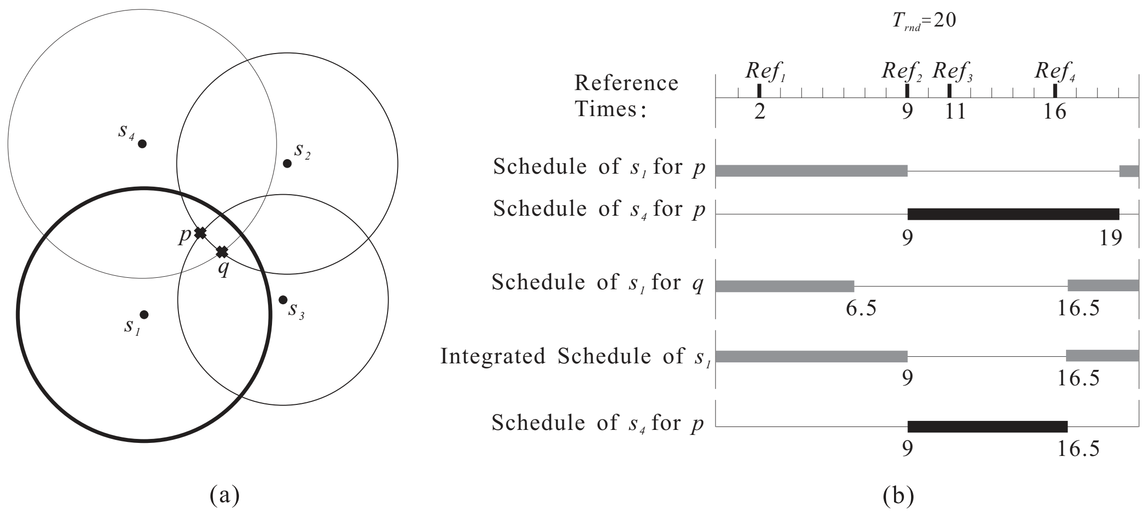

s_4
s_2
p
q
s_1
s_3
(a)
T_{rnd}=20
Reference Times:
Ref_1
Ref_2
Ref_3
Ref_4
2
9
11
16
Schedule of s_1 for p
Schedule of s_4 for p
9
19
Schedule of s_1 for q
6.5
16.5
Integrated Schedule of s_1
9
16.5
Schedule of s_4 for p
9
16.5
(b)

Figure 4.4: An example of redundant schedules.

4.3.2 Active Time Optimization

In this section, we show how to further reduce sensors' on-duty time without hurting coverage of the sensing field. Although the discussion is based on the 1-CP protocol, it can be applied to other schemes, too. Recall Eq. (4.3) and Eq. (4.4). The integrated schedule of a sensor is the union of the schedules to cover all intersection points under the sensor's coverage. This might be too conservative because all sensors are increasing their on-duty time. Below, we first show an example to demonstrate why such an optimization is possible. Then we will develop formal rules for the optimization.

A Motivating Example

Consider the simple example in Fig. 4.4. In this example, both sensors s_1 and s_4 cover the intersection point p, so their schedules in accordance to p should cover p entirely. Here, sensor s_1 should be in charge of duration $[-1, 9]$ (mod 20), and sensor s_4 should be in charge of duration $[9, 19]$. In the meanwhile, s_1 also covers intersection point q, and its schedule in accordance to q is $[-3.5, 6.5]$. Therefore, the integrated schedule of s_1 is $[-3.5, 9]$. Now, because the final schedule of s_1 is lengthened to meet the requirement to cover point q, sensor s_4 can thus shorten its schedule to duration $[9, 16.5]$, while point p is still entirely covered.

Basic Shrinking Rules

The above example has shown the possible redundancy in the calculation of Eq. (4.3) and Eq. (4.4) for sensors' schedules. Below, we first develop some basic rules for an individual sensor s_i to reduce its on-duty time while preserve the coverage of the entire network.

1. To shrink its on-duty time, a sensor s_i has to collect the schedules of all its neighbors. (This will require each sensor to broadcast its integrated schedule to its neighbors.)

2. To avoid two neighboring sensors shrinking their on-duty schedules simultaneously, s_i has to bid for the opportunity with its neighbors. (There are several possibilities to do so. We will address this issue in Section 4.3.2.) Once winning the bidding, s_i can go to step 3.

3. For each intersection point p that is under s_i's coverage, s_i tries to identify the smallest values of $Front'_{p,i}$ and $Back'_{p,i}$ such that p is fully covered in a round even if s_i shrinks its on-duty time to $[Ref_i - Front'_{p,i}, Ref_i + Back'_{p,i}]$. (Note that this can be easily done by putting all integrated schedules of sensors in $C(p)$ on the time axis.)

4. The final integrated schedule of s_i is $[Ref_i - Front'_i, Ref_i + Back'_i]$, where

$$Front'_i = \max_{\forall p \in P}\{Front'_{p,i}\}, \tag{4.7}$$

and

$$Back'_i = \max_{\forall p \in P}\{Back'_{p,i}\}. \tag{4.8}$$

5. After calculating its final schedule, s_i broadcasts this schedule to its neighbors, and it can not change its schedule any more in this cycle.

Note that the above procedure needs more space in the initialization phase. So the parameter T_{init} may need to be increased.

Below we use an example to illustrate the above process. Consider the example in Fig. 4.5(a). The original schedule of each sensor calculated by 1-CP is shown in Fig. 4.5(b). After applying the optimization, two sensors' schedules are shrunk, shown in Fig. 4.5(c). Consider sensor s_3, which covers intersection points p, q, r,

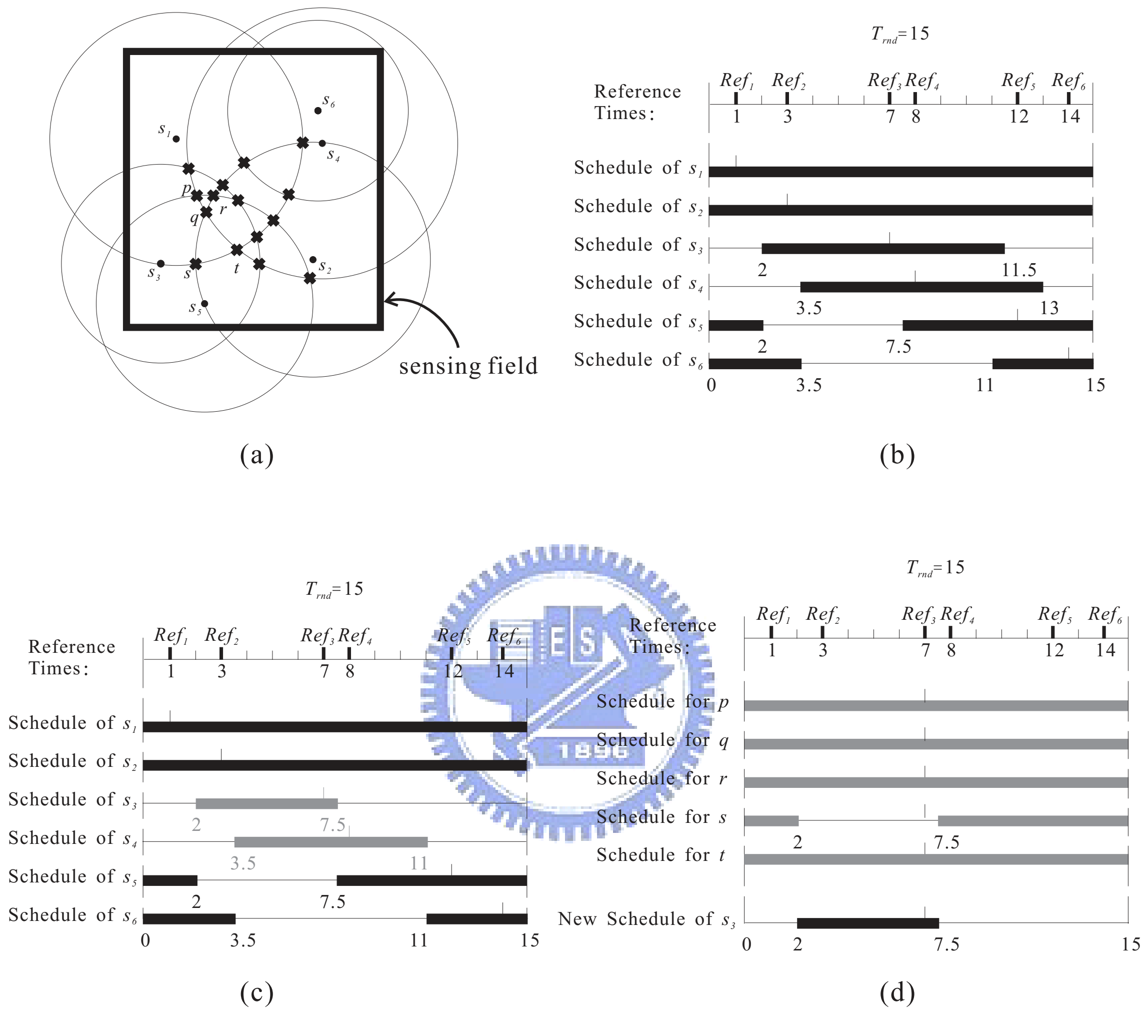


Figure 4.5: An example of the active time optimization process: (a) network topology, (b) original schedules found by the basic 1-CP protocol, (c) the new schedules after optimization, and (d) derivation of the coverage of intersection points provided by s_3's neighbors.

s, and t. Sensor s_3 can easily derive how each intersection point is covered by its neighbors, as shown in Fig. 4.5(d). As can be seen, all intersection points are already covered except s, even if we disregard sensor s_3. So s_3 only needs to be active in duration $[2, 7.5]$. Sensor s_4 can also shrink its schedule, as shown in Fig. 4.5(c), but we omit the details.

Shrinking Orders

One important issue in our design is whether sensors are allowed to shrink their schedules at the same time. To avoid the inconsistency problem, for each intersection point, only one sensor which covers this point is allowed to adjust its schedule at a time. Since sensors covering the same intersection point are all neighbors, we conclude that sensors that are neighbors can not adjust their schedules at the same time. Therefore, a sensor only needs to negotiate with its neighbors in the bidding procedure in step 2. This allows a certain level of concurrency because sensors that are not neighbors may adjust their schedules at the same time.

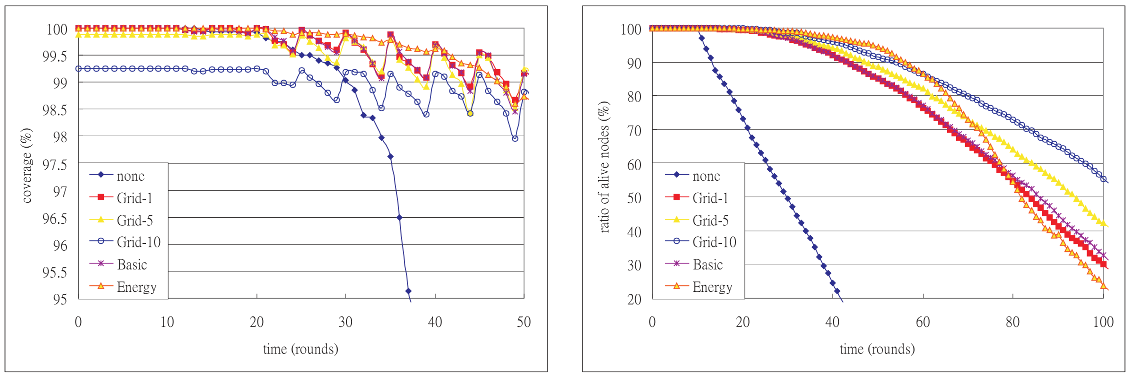
One design issue in the bidding procedure is the order of sensors which try to shrink their schedules. This is also an optimization problem because, for example, weaker sensors may be given higher priorities to do so. Below, we propose two strategies based on different criteria.

- *Longest Schedule First (LSF)*: A sensor which has a longer active duration in a round has a higher priority over another with a shorter active duration. So the former will win the bidding over the latter in the bidding process.

- *Shortest Lifetime First (SLF)*: A sensor with less remaining energy has a higher priority over another with more remaining energy.

4.4 Simulation Results

We have developed a simulator to evaluate the performance of the proposed energy-conserving and coverage-preserving protocols. The simulation environment is a 100×100 square area. In Section 4.4.1 and Section 4.4.2, 50 and 100 sensors are randomly generated, and each sensor has a sensing range randomly chosen between 10 to 50 units. In addition, each sensor will be rescheduled every 5 rounds, i.e., each cycle includes 5 rounds. To set up initial energies, we allow a sensor to be able to remain



(a) (b)

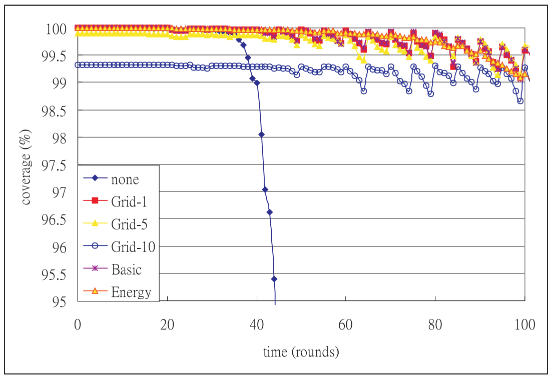
Figure 4.6: Performance comparisons in a 50-sensor network: (a) percentage of coverage, and (b) ratio of alive nodes.

active from 10 to 50 complete rounds in a randomly selected manner. (However, as a sensor is turned off from time to time, it may survive longer.)

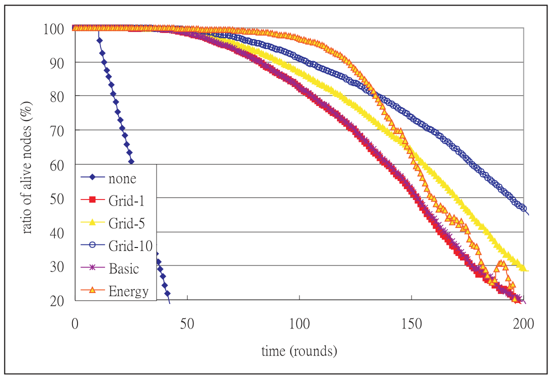
4.4.1 Comparison to the Grid Protocol [32]

In Fig. 4.6, we compare our schemes with the scheme in [32] in a 50-sensor network. The curve "none" means that all sensors keep awake all the time. The curves "Grid-1", " Grid-5", and "Grid-10" are for the scheme in [32] with grid sizes of 1×1, 5×5, and 10×10, respectively. Our basic scheme presented in Section 4.2.1 is labeled by "Basic", and our energy-based scheme presented in Section 4.2.2 is labeled by "Energy". Fig. 4.7 gives the results when 100 nodes are deployed.

In Fig. 4.6(a) and Fig. 4.7(a), we look at the ratios of covered areas achieved by different schemes as time moves on. We observe that using grid points to calculate each sensor's working schedule may cause inaccuracy. For example, in Fig. 4.8, sensor s_1 will consider only the four grid points when applying the scheme in [32], ignoring the grey region in the center. As a result, only relatively smaller grid sizes, such as 1×1, can achieve good coverage. By contrast, our basic scheme can easily achieve similar coverage and longer network lifetime. Our energy-based scheme further outperforms the basic scheme. The energy-based 1-CP protocol can keep almost 100% coverage for longer time while the curve of the basic scheme fluctuates. This is because our basic scheme does not utilize energy information and thus some



(a)



(b)

Figure 4.7: Performance comparisons in a 100-sensor network: (a) percentage of coverage, and (b) ratio of alive nodes.

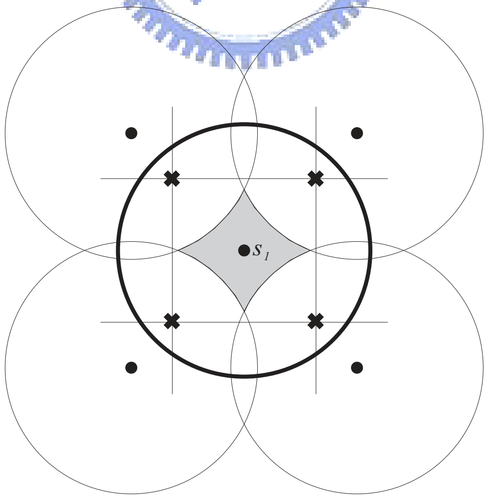


Figure 4.8: An example of inaccuracy incurred by using grid points to calculate nodes' schedules.

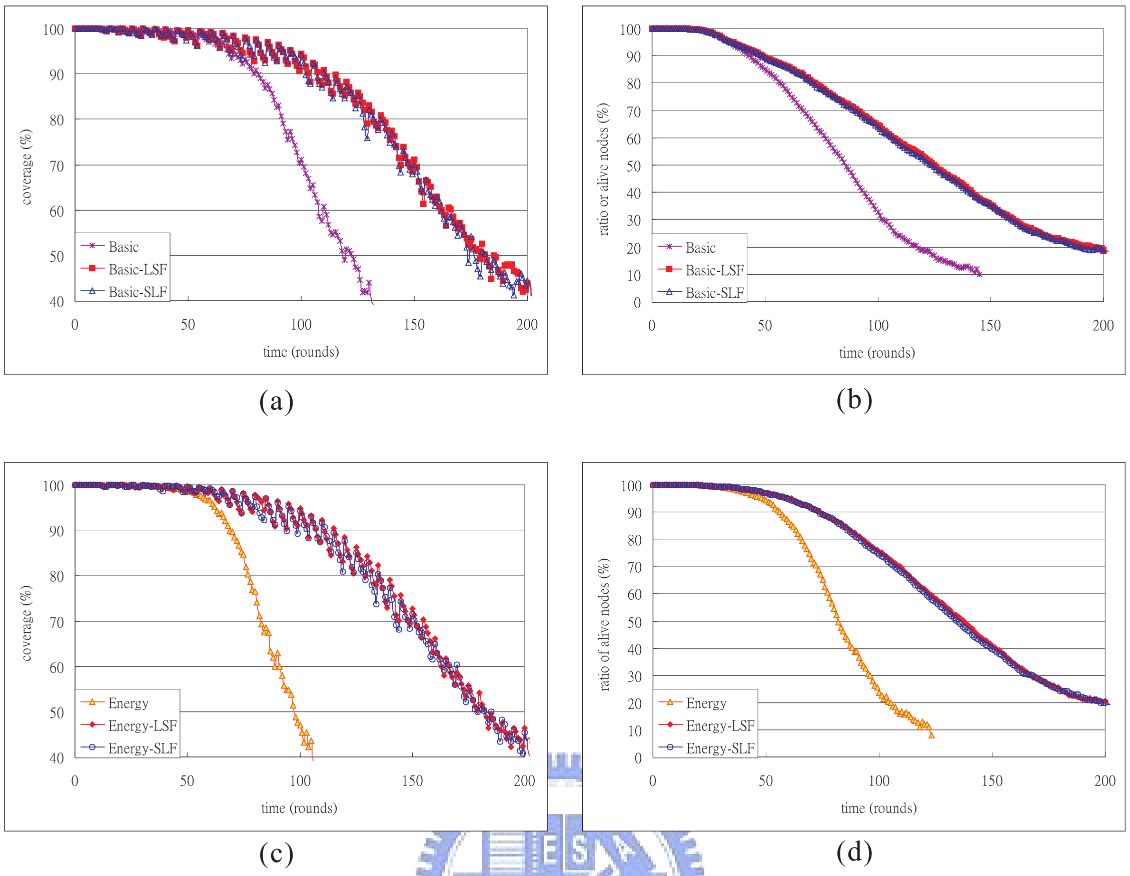
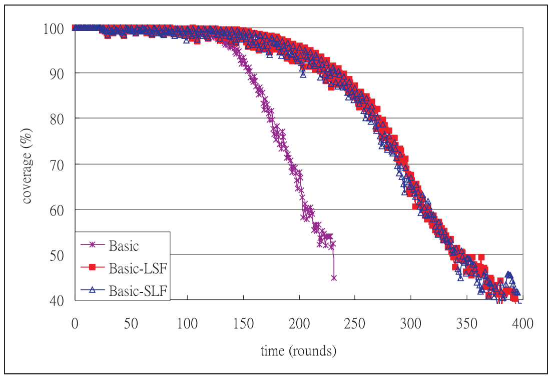


Figure 4.9: Effect of active time optimization in a 50-sensor network: (a) percentage of coverage applying the basic 1-CP protocol, (b) ratio of alive nodes applying the basic 1-CP protocol, (c) percentage of coverage applying the energy-based 1-CP protocol, and (d) ratio or alive nodes applying the energy-based 1-CP protocol.

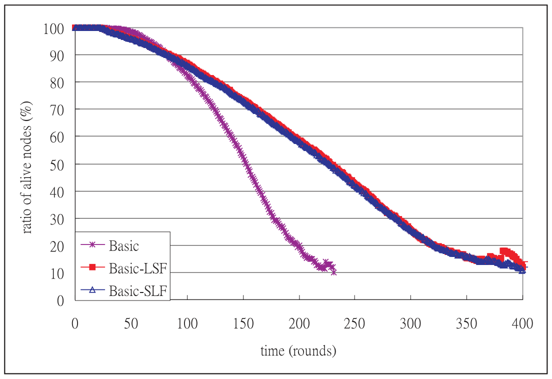
nodes may deplete their energies more quickly, causing uncovered areas. Note that the small-scale fluctuations in the figure are resulted from some sensors running out of energies during a cycle. Fig. 4.6(b) and Fig. 4.7(b) are from the same experiments but use the ratio of alive nodes as the performance metric.

4.4.2 Effect of Active Time Optimization

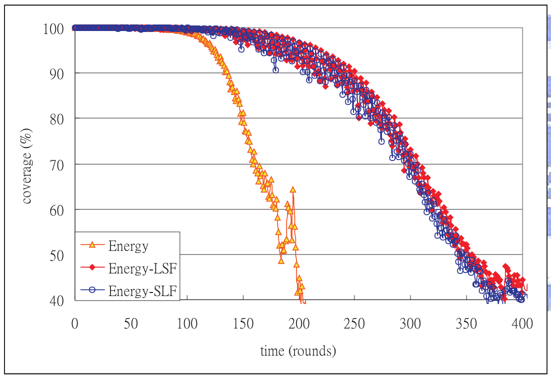
Next, we further evaluate the performance of our active time optimization schemes. The results are shown in Fig. 4.9 and Fig. 4.10. The curves "Basic" and "Energy" represent the performances of the basic 1-CP protocol and the energy-based 1-CP protocol, respectively. "Basic-LSF" and "Basic-SLF" indicate the performances



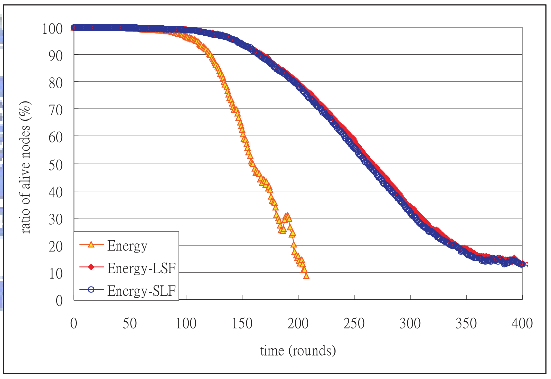
(a)



(b)

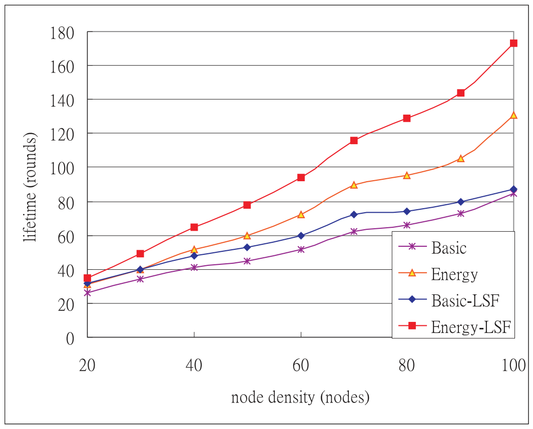


(c)

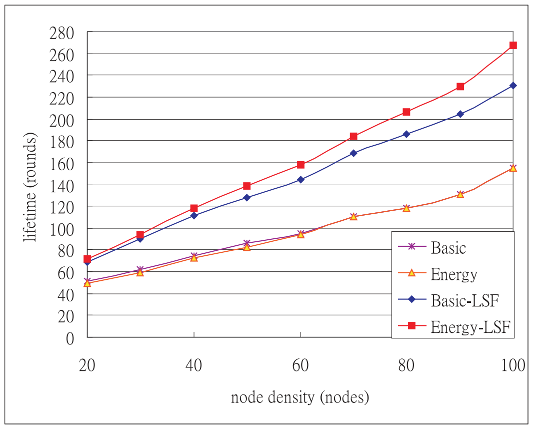


(d)

Figure 4.10: Effect of active time optimization in a 100-sensor network: (a) percentage of coverage applying the basic 1-CP protocol, (b) ratio of alive nodes applying the basic 1-CP protocol, (c) percentage of coverage applying the energy-based 1-CP protocol, and (d) ratio or alive nodes applying the energy-based 1-CP protocol.



(a)



(b)

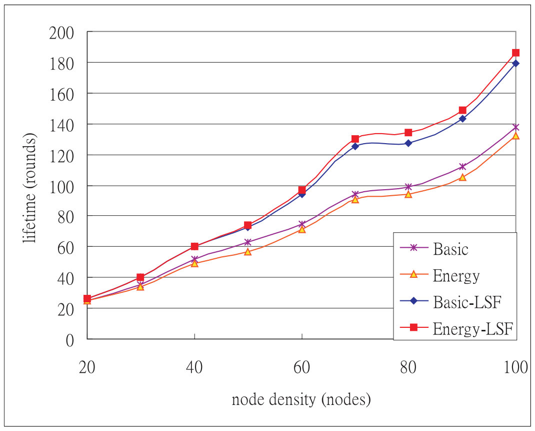
Figure 4.11: The effect of node density on network lifetime: (a) time to 10% failure, and (b) time to 50% failure.

when applying "Basic" using the shrinking orders LSF and SLF, respectively. The performances when applying "Energy" using the orders LSF and SLF are represented as "Energy-LSF" and "Energy-SLF", respectively. As can be seen, the ratio of covered areas decreases at a slower speed when active time optimization is applied. Similar results can be seen when considering the ratio of alive nodes. These validate the effectiveness of the proposed active time optimization.

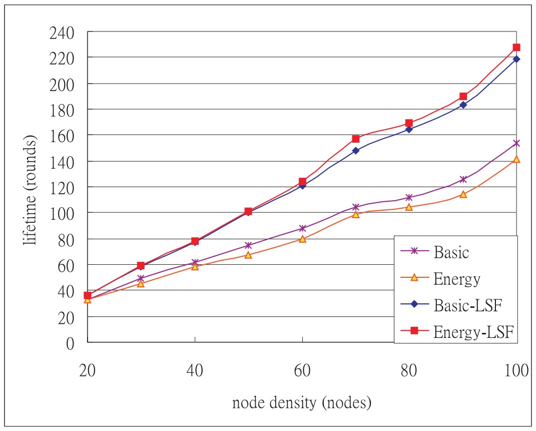
4.4.3 Effect of Node Density

Next we vary node density and examine its effect. Because applying different shrinking orders in the active time optimization has little impact on performances, we only apply the LSF order.

Fig. 4.11 shows the network lifetime when 20 to 100 sensor nodes are deployed. The sensing ranges of sensors are randomly chosen between 10 to 50 units. In Fig. 4.11(a), the network lifetime is defined as the number of rounds that a network can sustain before 10% of nodes die. In Fig. 4.11(b), lifetime is defined as the number of rounds before 50% of nodes die. As can be seen, applying active time optimization always leads to better performance. As the number of nodes increases, the effect becomes even more evident. This is because a higher node density would allow us to achieve better balance in sensors' energy consumption.



(a)



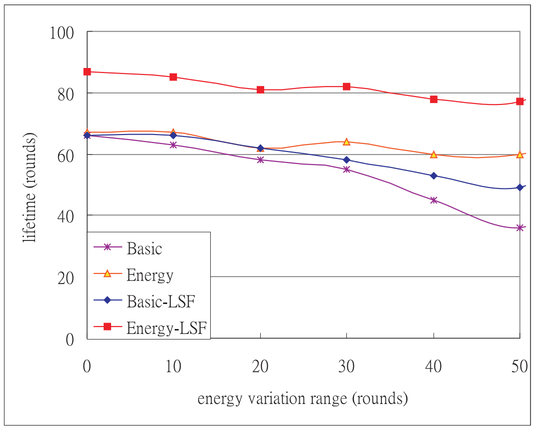
(b)

Figure 4.12: The effect of node density on network lifetime: (a) time to 1% exposure, and (b) time to 5% exposure.

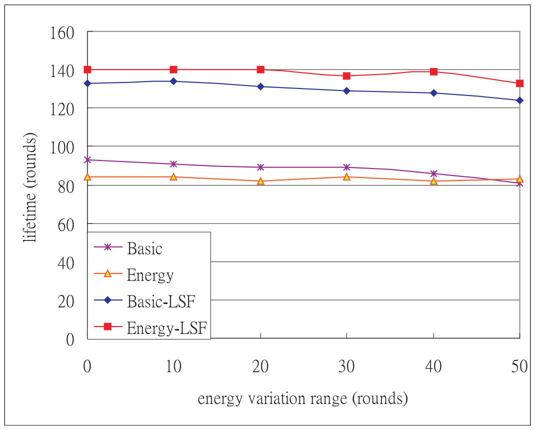
Now consider the ratios of covered areas as a performance metric. In Fig. 4.12(a), the network lifetime is defined as the number of rounds before 1% of the sensing field is uncovered. Similarly, Fig. 4.12(b) before 5% of the sensing field is uncovered. Generally, Fig. 4.12 shows similar results as Fig. 4.11. Thus, node density does have a positive impact to our schemes. Also note that in Fig. 4.12, "Energy" has a slightly shorter network lifetime than "Basic". This trend is also shown in Fig. 4.6 and Fig. 4.7. Because "Energy" provides better coverage guarantee in the beginning, the coverage level will degrade more rapidly than "Basic" after some nodes deplete their energies.

4.4.4 Effect of Energy Variation

Finally, we examine the effects of energy variation. Each sensor's initial energy is uniformly distributed between $[u - \frac{v}{2}, u + \frac{v}{2}]$, where u is the mean and v is called the energy variation range. Here we fix u and vary the value of v to observe its impact. Fig. 4.13 shows the impact on network lifetime in a 50-node network. We see that "Basic-LSF" outperforms "Basic", and "Energy-LSF" outperforms "Energy". Also, as the energy variation range increases, the network lifetime of "Basic" and "Basic-LSF" decreases more rapidly than those of "Energy" and "Energy-LSF". This reflects the importance of using sensors' energies in scheduling, especially when

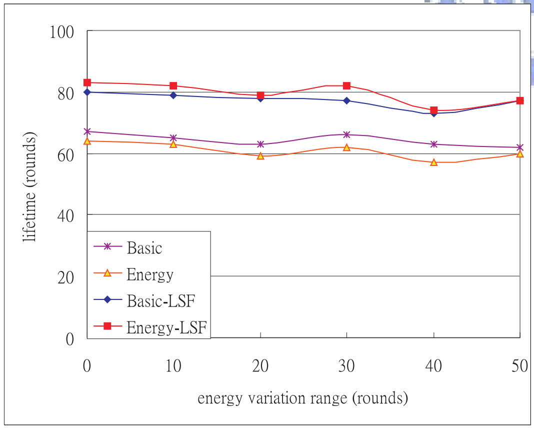


(a)

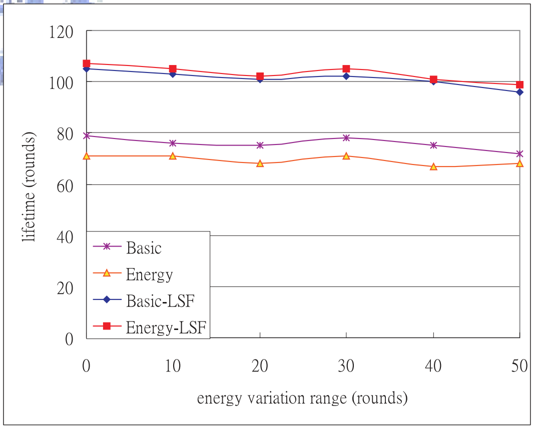


(b)

Figure 4.13: The effect of energy variation range on network lifetime: (a) time to 10% failure, and (b) time to 50% failure.



(a)



(b)

Figure 4.14: The effect of energy variation range on network lifetime: (a) time to 1% exposure, and (b) time to 5% exposure.

sensors' remaining energies vary significantly. The impact reduces when we look at the time to 50% failure because there is less redundancy. Fig. 4.14 considers lifetime from the perspective of the ratios of covered areas. Although “Basic-LSF” and “Energy-LSF” always perform better than “Basic” and “Energy”, the impact is less significant when the energy variation range enlarges.

Chapter 5

Conclusions

In this thesis, we first propose a solution to the 3D coverage problem for wireless sensor networks. The problem seems to be complex at the first glance. However, we have shown that this problem can be solved in polynomial time and in a distributed manner, which is a desirable property in a sensor network. The basic result can be used in deploying sensors in a 3D space. We have also shown an interesting application of the derived result by proposing a distributed active/sleeping scheduling protocol. Only local information exchange is required. The protocol thus may be used in a large-scale, highly dynamic, or detrimental environment where sensors may be easily destroyed and intensively re-deployed at any time.

We also propose several decentralized energy-conserving and coverage-preserving protocols to prolong the network lifetime while maintain the sensing field sufficiently covered. We first present a basic protocol for guaranteeing one coverage of the sensing field. We then extend the result to support multi-layer coverage of the sensing field. Moreover, several optimization mechanisms are proposed to further balance or reduce sensors' energy expenditure. Simulation results do show significant improvement over existing results in both accuracy of coverage and network lifetime.

Bibliography

[1] P. K. Agarwal and M. Sharir. Arrangements and their applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 49–119. Elsevier, North-Holland, New York, 2000.

[2] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE INFOCOM*, pages 775–784, 2000.

[3] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM/Kluwer Wireless Networks*, 8(5):481–494, Sep. 2002.

[5] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja. Sensor deployment strategy for target detection. In *ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[6] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly resilient, energy efficient multipath routing in wireless sensor networks. *ACM Mobile Comput. and Commun. Review*, 5(4):11–25, Oct. 2001.

[7] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, 2004.

[8] P. Hall. *Introduction to the Theory of Coverage Processes.* Wiley, New York, 1988.

[9] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 21, pages 389–412. CRC Press LLC, Boca Raton, FL, 1997.

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *Hawaii Int'l Conf. on Systems Science (HICSS)*, 2000.

[11] C.-F. Hsin and M. Liu. Network coverage using low duty-cycled sensors: Random & coordinated sleep algorithms. In *Int'l Symp. on Information Processing in Sensor Networks (IPSN)*, 2004.

[12] C.-F. Huang and Y.-C. Tseng. The coverage problem in a wireless sensor network. In *ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 115–121, 2003.

[13] J. Lu and T. Suda. Coverage-aware self-scheduling in sensor networks. In *IEEE Computer Communications Workshop (CCW)*, 2003.

[14] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *IEEE INFOCOM*, pages 1380–1387, 2001.

[15] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless sensor networks. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, 2001.

[16] J. B. M. Melissen and P. C. Schuur. Improved coverings of a square with six and eight equal circles. *Electronic Journal of Combinatorics*, 3(1), 1996.

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

[18] J. O'Rourke. Computational geometry column 15. *Int'l Journal of Computational Geometry and Applications*, 2(2):215–217, 1992.

[19] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, May 2000.

[20] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pages 166–179, 2001.

[21] S. Shakkottai, R. Srikan, and N. Shroff. Unreliable sensor grids: Coverage, connectivity, and diameter. In *IEEE INFOCOM*, 2003.

[22] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pages 272–287, 2001.

[23] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE Int'l Conf. on Communications (ICC)*, pages 472–476, 2001.

[24] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Commun.*, 7(5):16–27, Oct. 2000.

[25] D. Tian and N. D. Georganas. A node scheduling scheme for energy conservation in large wireless sensor networks. *Wireless Commun. and Mobile Comput. (WCMC)*, 3:271–290, 2003.

[26] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. In *Int'l Workshop on Information Processing in Sensor Networks (IPSN)*, 2003.

[27] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak. Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In *ACM Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 40–50, 2003.

[28] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *ACM Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 28–39, 2003.

[29] R. Williams. *The Geometrical Foundation of Natural Structure: A Source Book of Design*, pages 51–52. Dover, New York, 1979.

[30] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pages 221–235, 2001.

[31] G. Xing, C. Lu, R. Pless, and J. A. O'Sullivan. Co-grid: An efficient coverage maintenance protocol for distributed sensor networks. In *Int'l Symp. on Information Processing in Sensor Networks (IPSN)*, pages 414–423, 2004.

[32] T. Yan, T. He, and J. A. Stankovic. Differentiated surveillance for sensor networks. In *ACM Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 51–62, 2003.

[33] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Int'l Conf. on Distributed Computing Systems (ICDCS)*, 2003.

[34] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE INFOCOM*, pages 1567–1576, 2002.

[35] H. Zhang and J. C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. In *NSF International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, 2004.

[36] H. Zhang and J. C. Hou. On deriving the upper bound of α-lifetime for large sensor networks. In *ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 121–132, 2004.

[37] Z. Zhou, S. Das, and H. Gupta. Connected k-coverage problem in sensor networks. In *Int'l Conf. on Computer Communications and Networks (ICCCN)*, 2004.

Curriculum Vita

Li-Chu Lo (lolc@csie.nctu.edu.tw) received her B.S. degree in Computer Science from National Chiao-Tung University, Taiwan, in 2003. Her research interests include wireless networks and sensor networks.