DISTRIBUTED ENERGY-EFFICIENT SOLUTIONS FOR

AREA COVERAGE PROBLEMS IN WIRELESS SENSOR NETWORKS

by

CHINH TRUNG VU

Under the Direction of Yingshu Li

ABSTRACT

Wireless sensor networks (WSNs) have recently attracted a great deal of attention due to their numerous attractive applications in many different fields. Sensors and WSNs possess a number of special characteristics that make them very promising in a wide range of applications, but they also put on them lots of constraints that make issues in sensor network particularly challenging. These issues may include topology control, routing, coverage, security, data management and many others. Among them, coverage problem is one of the most fundamental ones for which a WSN has to watch over the environment such as a forest (area coverage) or set of subjects such as collection of precious renaissance paintings (target of point coverage) in order for the network to be able to collect environment parameters, and maybe further monitor the environment.

In this dissertation, we highly focus on the area coverage problem. With no assumption of sensors' locations (i.e., the sensor network is randomly deployed), we only consider distributed and parallel scheduling methods with the ultimate objective of maximizing network lifetime. Additionally, the proposed solutions (including algorithms, a scheme, and a framework) have to be energy-efficient. Generally, we investigate numerous generalizations and variants of the basic coverage problem. Those problems of interest include *k*-coverage, composite event detection,

partial coverage, and coverage for adjustable sensing range network. Various algorithms are proposed. In addition, a scheme and a framework are also suggested to solve those problems. The scheme, which is designed for emergency alarming applications, specifies the guidelines for data and communication patterns that significantly reduce the energy consumption and guarantee very low notification delay. For partial coverage problem, we propose a universal framework (consisting of four strategies) which can take almost any complete-coverage algorithm as an input to generate an algorithm for partial coverage. Among the four strategies, two pairs of strategies are trade-off in terms of network lifetime and coverage uniformity. Extensive simulations are conducted to validate the efficiency of each of our proposed solutions.

INDEX WORDS: Wireless sensor network, Coverage problem, Localized and distributed algorithm, Parallel algorithm, Energy-efficiency, Fault-tolerance, Composite Event Detection, Robustness, Connectivity, *k*-coverage, Partial coverage, Adjustable sensing range WSN, Heterogeneous Network, Homogeneous Network

DISTRIBUTED ENERGY-EFFICIENT SOLUTIONS FOR AREA COVERAGE PROBLEMS IN WIRELESS SENSOR NETWORKS

by

CHINH TRUNG VU

A Dissertation Submitted in Partial Fulfillment of the Requirements of the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2009

DISTRIBUTED ENERGY-EFFICIENT SOLUTIONS FOR

AREA COVERAGE PROBLEMS IN WIRELESS SENSOR NETWORKS

by

CHINH TRUNG VU

Major Professor: Dr. Yingshu Li

Committee: Dr. Rajshekhar Sunderraman
Dr. Anu G. Bourgeois
Dr. Xiaojun Cao
Dr. Yi Zhao

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
August 2009

to my parents, my wife and my late brother

ACKNOWLEDGMENTS

I would like specially to thank my advisor, Dr. Yingshu Li for her expert guidance, advisement, and support throughout my academic study.

I would also like to thank committee members, Dr. Rajshekhar Sunderraman, Dr. Anu G. Bourgeois, Dr.Yi Zhao, and Dr. Xiaojun Cao for serving on my committee, for their comments and for taking precious time to review this dissertation.

I am personally grateful to Dr. Rajshekhar Sunderraman - Director of Graduate Studies - for his invaluable helps on my various (both academic and non-academic) issues throughout my graduate program in this department.

I am thankful to Dr. Yi Zhao and Dr. Guantao Chen from the Department of Mathematics and Statistics for sharing their valuable knowledge and spending time working with me.

I greatly appreciate the Project 322 of Vietnamese government for constant financial support on both my master and doctorate degrees.

I would especially like to thank my parents and my late brother who instilled in me a sense of curiosity and wonder at a very young age, and they deserve much credit for my accomplishments. Above all, I am grateful to them for continuously encouraging me in my academic pursuit. Special thanks with all my heart to my wife for being the greatest support during my Ph.D life which makes my time as a PhD student as smooth as possible. Without my family’s support and beliefs, I would not have been able to achieve any of this today.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PSEUDOCODES

LIST OF ACRONYMS

Base Station	BS
Detection set	DS
Dominating Set	DS
Integer Linear Programming	ILP
Sensor Identification	ID
Linear Programming	LP
Wireless Mobile Ad Hoc Network	MANET
Minimal Dominating Set	MDS
Micro-Electro-Mechanical System	MEMS
Nondeterministic Polynomial time	NP
Quality of Service	QoS
System-on-Chip	SoC
Wireless Ad hoc Sensor Network	WSN, WASN

CHAPTER 1.

INTRODUCTION

Despite the exceptionally fast grow of Internet, various other networks, and their applications, there is still a gap that those networks have not filled up: physical world. For example, there is no traditional network which works and collects information from a toxic area or an active volcano. Wireless sensor networks (WSNs) is invented to fill that gap. A WSN, which is a special kind of ad-hoc networks composed of a large number of sensing, communication, computation, and storage capable devices called sensors, is expected to make a bridge between human world and physical world. Inside a WSN, the sensors autonomously collaborate to sense, collect, process data and to transmit those sensed/processed data to some specific data processing centers which usually are base stations (BSs). That means sensors need to be able to cooperatively accomplish assigned tasks without the intervention or control from outside. These characteristics along with self-organization and self-configuration capabilities of sensors make WSNs very promising for applications in many different fields. Thanks to the advance of supporting technologies such as wireless communication, system-on-chip (SoC), micro-electro-mechanical systems (MEMS), wireless communication, and digital electronics, those tiny devices are becoming cheaper and more and more commercially available. A sensor is a tiny battery-equipped device capable of sensing, communication and computation. Sensors may be static or mobile. The minuscule size, light weight, and portability attributes are a sensor‘s special characteristics which make WSNs to be the best and/or a unique choice in many existing and promising applications. However, there are also numerous constraints on sensors such as limit on energy supply, bandwidth, computational capability, the uncertainty of sensed data and

communicated information, and the vulnerability of sensors to environment. These constraints require thorough and prudent researches to overcome.

1.1. Unique constraints of sensors and WSNs

A sensor network is a wireless network that comprises many sensing devices scattered in an area as guards to record and eventually to control surrounding environment conditions such as temperature, humidity, sound, vibration, and pressure. In a sensor network, sensors cooperatively work to sense, process and route data. The recent development in many supporting technologies has enabled the productions of low-cost, low-power, multifunction, and tiny sensors [AKY02], so that a redundant number of sensors can be densely deployed to a monitored area to prolong network lifetime and to enhance the surveillance quality. Although a huge number of protocols have been devised and applied to wired and traditional wireless networks (such as wireless LAN), those protocols cannot directly be employed to sensor network since sensor network possesses some special characteristics and restrictions that distinguish it from the other types of networks. Those particularities of sensor network may include:

- *Limited support for networking*. The sensor nodes can only communicate with very low quality, high latency and variance, limited bandwidth, and high failure-rate. A sensor's transmission range is short and greatly affected by energy. In a WSN, the communication mainly relies on broadcasting. Moreover, the network is peer-to-peer, with a mesh topology and dynamic, mobile, and unreliable connectivity.
- *Energy constraint*. The most precious resource of a sensor is energy. In most cases, the battery is irreplaceable, while all the sensor's operations consume a certain amount of energy. Therefore energy conservation is always the most critical requirement on designing a sensor network protocol. It has been shown that the power to transmit 1 bit is enough for executing

at least 3000 instructions [POT00]. That means a sensor consumes much more energy on communication than on computation, thus the designed protocols for WSN usually try to make good uses of computation to get the jobs done to compensate for expensive and unreliable communication.

- *Dynamic topology*. The topology of a WSN changes very frequently due to the movement of sensors, the sensors' temporary or permanent failure, the death of sensors, the addition of sensors to the network, or even the temporary sensors' malfunction.

- *Scalability and heterogeneity*. WSN may consist of a large number of different sensors in terms of sensing units, communicational ability, computational power, memory size, and manufacturer. Huge number of sensors is deployed into hostile environment under tough condition, thus it is very difficult to maintain and manage the network. Due to the same reason, sensor nodes may not have global ID in the network.

- *Limited support for software development*. The tasks are typically real-time and massively distributed. They require involved dynamic collaboration among nodes and they must handle multiple competing events. Also, sensors are limited in computational capability and memory sizes. This limits the types of algorithms and results processed on a sensor.

- *Prone to failure.* A tiny sensor node tends to fail to operate due to numerous reasons such as depletion of energy and environmental interference [AKY02] making it very vulnerable to the environment, e.g., easy to be physically damaged. Sensing data is prone to bias under the environment effects such as noise and obstacle. Moreover, sensors have to operate unattended, since it is impossible to service a large number of nodes in remote, possibly inaccessible locations.

1.2. Challenging issues in WSNs

Before a WSN can be brought into real life, many problems need to be carefully resolved. More and more problems are discovered and solved over time. In this section, we itemize some typical ones that draw the most attention from researchers. Keep in mind that the following issues are needed to be attacked under the various constraints and limitations as mentioned in Section 1.1, which makes them exceedingly challenging.

- *Algorithm type*: Energy is the most critical resource of a sensor since every operation requires a certain amount of energy while sensor is battery-driven but battery is not always replaceable. Thus, energy-efficiency should be and have been the foremost concern of any protocol designed for a WSN. Other limitations of a sensor that require thorough awareness when designing a WSN protocol include sensor's limited memory size, communication and computation capability, thus, algorithms for WSNs need to be simple but robust and fault-tolerant. That is also the reason why decentralized algorithm is always preferable (if it is not the only suitable ones) in WSNs. Some requirements that a "good" protocol aims to are simplicity, energy-efficiency, localized, distributed, and parallel type, scalability and flexibility to the enlargement of the network, robustness, fault-tolerance, and low communication overhead.

- *Topology control*: For a prone-to-failure network as WSN, the sensors may malfunction at any time or any place for various reasons. It follows that the topology of a WSN is highly dynamic and unpredictable. For each kind of applications, an appropriate topology may be required.

- *Routing*: After sensors collect the information, enormous streams of information need to be made available to some data consuming centers. The question of how to efficiently, reliably,

and securely route the data through a high-density network is also a challenging issue for sensor networks.

- *Data Management*: A WSN is supposed to frequently collect information about the physical world, e.g., surrounding environment or objects. Information is exchanged on a multiple-source-multiple-destination basis and the number of sensors in a WSN is in the order of hundreds, thousands or even more. Thus, the amount of data collected by a WSN is extremely large. How to manage, process and route the data is truly a challenge. Researchers have considered the following sub-problems for this kind of issue: in-network data processing, data dissemination (multicast, unicast, broadcast) and aggregation (or convergecast).

- *Coverage*: The primary function of a WSN is to watch over the physical world. To accomplish this function, it is compulsory to schedule, organize the network in such a way that it can effectively observe the interested environment or set of objects, and then collect information that it is desired and supposed to gather. This problem is thoroughly investigated in this dissertation.

- *Security*: It is no use if the sensed data of a WSN is illegally modified, blocked, or redirected to some illegal data centers. It is the responsibility of security protocols to protect the WSNs from such undesired actions. Because a WSN is usually an ad hoc wireless network and is usually deployed to an unattended and hostile region, attacks in sensor networks are relatively easy to carry out, but are exceptionally difficult to defend. Also, types of attacks in WSN are very multiform. Some aspects of security issues in WSNs are to guarantee the integrity, confidentiality of the data or to verify the authenticity of entities exchanging the data.

- Besides the above issues, there are numerous other important issues that are being worked on such as time synchronization, localization, positioning and location tracking, sensor management protocol [ILY05], link-layer protocols (e.g., MAC), and transport-layer protocols (e.g., real-time traffic, reliable transfer).

1.3. Coverage variants and our main focus

We devote this section to address various variants of coverage problem which is our main focus in this dissertation:

- *Area coverage vs. target coverage:* Sometimes, the objective of coverage problem is to cover some areas such as a forest. In some other cases, its objective is to cover a set of subjects such as a collection of precious renaissance paintings. The former case is called area coverage while the later is called target or point coverage. The difference between these two kinds of coverage is discussed in more detail in Section 1.4.
- *Fixed sensing range vs. adjustable sensing range:* In most of the work, sensors could not change their sensing radii, i.e., they have fixed sensing ranges. To give the sensors one flexible choice to accomplish their optimization job, sensors are assumed to adjust their sensing ranges in some of the recent work [DHA06], [VUC09], [VUT09].
- *Complete coverage vs. partial coverage:* Some applications do not require to completely cover the whole area. In situations such as monitoring forest fire in rainy season, one good and cheap way to save energy consumption and to extend network lifetime is to cover only a portion of the area instead of the whole area. The size of that portion of the covered area should be a user-specific parameter.

- *1-coverage vs. k-coverage*: Due to the uncertainty of sensors, 1-coverage is not enough for applications that require highly accurate and reliable data. For such applications, each point in the area may need to be covered by at least *k* sensors at the same time where $k > 1$ and *k* is a user-specific parameter.
- *Directional vs. omni-directional sensing region*: Some types of sensors such as temperature or humidity sensors can sense the data in all directions. For some other types such as visual sensors (e.g., camera), they could only be able to catch the image within some certain beams or directions.

In this dissertation, under our focus on area coverage, energy-efficiency is also the foremost requirement. Besides a framework and a scheme, the algorithms we design are fully localized, distributed and parallel, and some of above variants are taken into account (which we will outline in Section 1.7).

1.4. Area coverage vs. target coverage

In order to clarify our concentration on area coverage, we dedicate this section to represent a comparison between target coverage and area coverage. From our personal points of view, the area coverage problem is *not easier* than but *more practical* than the target coverage problem due to following reasons:

- To date, area coverage has been proposed for use in the most practical applications such as habitat monitoring [SZE04], underwater surveillance [CAY06], volcano monitoring network [WER06], forest fire application [YUN05], [SON07], [HEF07], and birds habit study [KUG04].

- For target coverage problem, a target coverage status can be easily verified by comparing the distance from it to sensors and sensors' sensing range. However, for the area coverage, it is challenging to verify the coverage status of every single point inside an area (since there are infinite numbers of such points). More specifically, with the target coverage problem, the objective is to cover a *limited* set of targets or points, but for the area coverage problem, the objective is to coverage an *unlimited* number of points, i.e., every point in an area. In this sense, area coverage problem is not easier than target coverage one.
- When designing distributed algorithms which is the only practical algorithms for WSN, the verification process for area coverage becomes more challenging than target coverage.
- Area coverage can be converted to target coverage in [BER04], [SLI01]. The idea is to map each sub-region which is covered by the same sub-set of sensors to a point. However, to the best of our knowledge, there is no reverse conversion. That is, there is no suggestion on how to convert target coverage to area coverage. For this reason, one may say that "the area coverage problem" is more general than "target coverage problem"

Those are the reasons why we focus on area coverage rather than target coverage.

1.5. Protocol design requirements

As being explained in Section 1.1, a WSN possesses a lot of constraints, protocols designed for them must satisfy many special requirements to overcome those constraints. Some of the most critical requirements are:

- *Energy-efficiency:* A sensor is a battery-driven device and in most cases, the battery is irreplaceable. However, every operation of a sensor consumes a certain amount of energy. Thus, to lengthen network lifetime, a sensor network protocol must take energy into account, or in other words, it must be energy-efficient. To date, the best way to energy-efficiently maximize network lifetime is to balance energy consumption among all the sensors in the network. That is, the more energetic sensors must have more chance to be active, and the more exhausted ones should have more chance to go to sleep.

- *Robustness:* Sensors are unreliable devices. Besides, they are usually deployed in big regions under tough conditions. Thus, a sensor may unpredictably die, may be temporarily go out of service at any time for various reasons, or a sensor's battery may deplete at any time. It is also possible that some new sensors join the network. The protocols for sensor networks must be able to cope with these situations.

- *Fault-tolerance:* Sensor network is a prone-to-failure network [AKY02], and a sensor is an unreliable device equipped with a communication system with high failure-rate and unreliable sensing components. Thus, to guarantee the correctness and integrity of sensed data, protocols designed for WSNs must provide high fault-tolerance.

- *Distributed and parallel algorithms:* Sensors have very limited computational ability and a very small memory size. Therefore, they are not able to execute a complex algorithm. The burden of running any algorithm should be shared among sensors in the network. Besides, a WSN is a scalable network comprising of a large number of sensors, and the topology of a sensor network may change very frequently. Thus, the converge-time (i.e., time complexity) of any algorithms for WSN needs to be short enough to keep up with the changes in the networks. For these reasons, in most cases, only localized, distributed and parallel

algorithms are suitable for WSNs. Notice that there exist decentralized but non-parallel (i.e., sequential) algorithms (see Section 6.1 for example of such algorithms) for which sensors even require only local information, but they have to wait for neighbors before making decisions, usually sleep/active decisions. This kind of algorithms is clearly not suitable for WSN since their time complexity is no smaller than sequential algorithms.

1.6. WSNs applications

[SOH07] gives an extensive list of sensor network and some examples of applications in its 2nd chapter. [BUL05] dedicates all of its content to list various types of sensor network applications. [AKY02] gives a nice survey on applications of WSNs. Various other publications are dedicated to specifying separate applications of WSNs and systems for those applications such as [KUG04], [SZE04], [YUN05], [CAY06], [WER06], [SON07], [HEF07]. Here, we briefly itemize some typical and promising applications of WSNs. More details can be found in [BUL05], [ILY05], and [AKY02].

- *Military applications:* The autonomy, self-organization, self-configuration and portability characteristics of a WSN make it very suitable for military applications. It can be used:
    - For commanders to monitor the status (position, quantity, availability) of their troops, equipment and ammunition.
    - For battlefield surveillance or reconnaissance of opposing forces and terrain.
    - For battle damage assessment.
    - To target the enemy, to detect biological and chemical (NBC) attack.
- *Environmental applications:* It can be used:

- To monitor the condition/status of environment such as humidity, temperature, pressure, and pollution in soil, marine, and atmosphere.
- To detect a disaster such as forest fire, flood, tsunami, volcano activities that is about to happen.
- To track the movement, health condition of animal/insects etc.

- *Health applications:* It can be used:
  - To remotely monitor/track/diagnose the condition/status (position, quantity, heart rate, blood pressure) of doctor, patient or drug, equipment, etc.
  - To tele-monitor human physiological data (e.g., patient behavior), and the data will be collected and analyzed to detect early symptoms of a disease, and to find new treatment, etc.
- *Commercial applications:* It can be used to detect/track/monitor vehicles, to manage/control inventory/warehouse, to support interactive devices, or to control environment of a building.
- *Scientific exploration:* WSNs can be deployed under the water or on the surface of a planet for scientific research purpose.

1.7. Outline of this dissertation

As shown in Figure 1.1, in this dissertation, we consider several generalizations and variants of *basic coverage*. The central part of Figure 1.1 is the basic coverage problem for which the area is required to be completely covered (complete coverage), every point in the area needs to be covered by at least one live sensor (*1*-coverage problem), the network is assumed to be homogeneous (i.e., all the sensors are the same type), and the sensors cannot change their

sensing ranges (fixed-sensing-range network). The outer parts of Figure 1.1 are generalizations and variants of basic coverage which are considered in this dissertation. More specifically, they are:

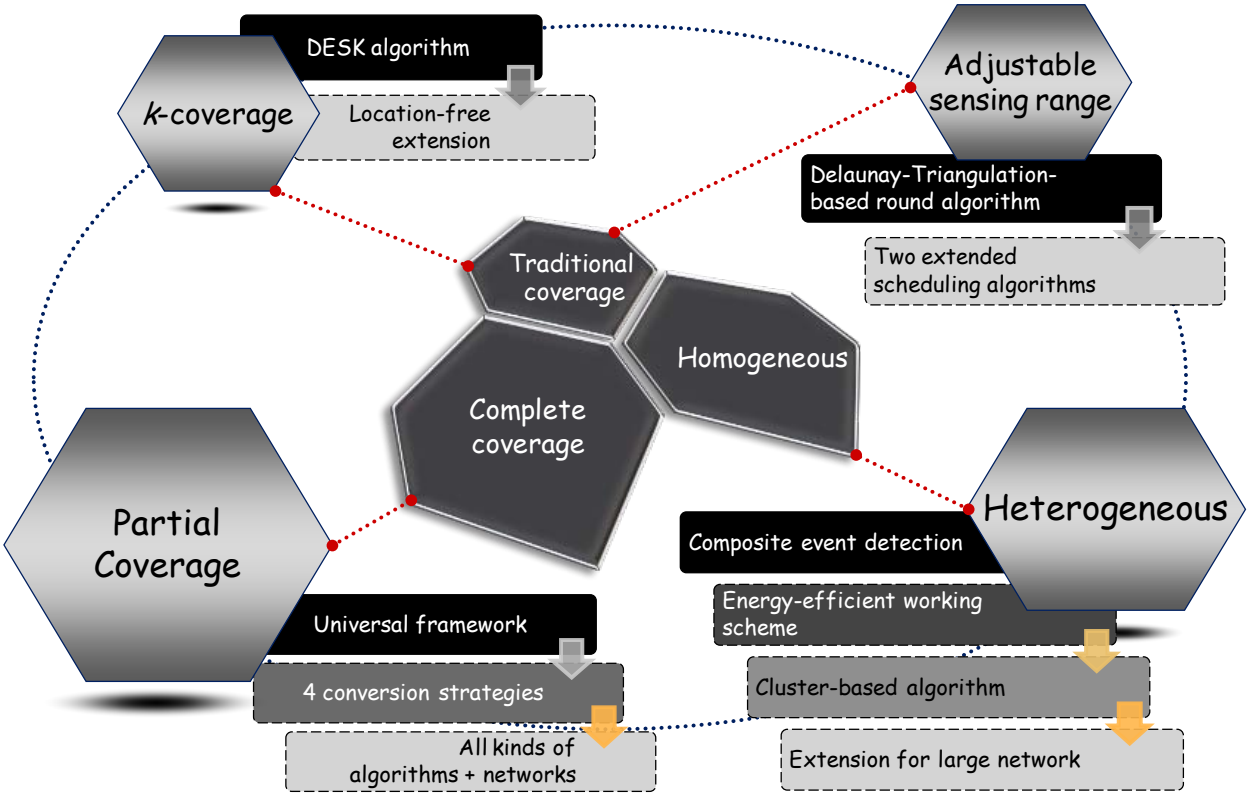


Figure 1.1. Our contribution

- Firstly, in CHAPTER 2 we conduct a brief survey of the existing work in literature that focuses on the general coverage problem. For easy reading, the related work is well categorized.
- Next, in CHAPTER 3, we define the relatively traditional area coverage problem named Sensor Energy-efficient Scheduling for *k*-coverage (SESK) which has strong requirements of being energy-efficient and fault-tolerant. The basic coverage problem is a special case of SESK where *k=1*. We prove that SESK is a NP-complete problem. We then solve the

problem by proposing a novel, completely parallel, localized, and distributed scheduling approach named Distributed Energy-efficient Scheduling for *k*-coverage (DESK) so that the energy consumption among all the sensors is balanced, and network lifetime is maximized while the *k*-coverage requirement being maintained. DESK satisfies all the requirements that a sensor network protocol should have which are shown in Section 3.2.3. The correctness as well as its performance and efficiency are confirmed by our theoretical analysis and simulation results. At the end of CHAPTER 3, we also succinctly explain the way that can help DESK function without relying on sensors' locations.

- In CHAPTER 4, we consider the area coverage problem for WSN where sensors can arbitrarily change their sensing ranges under some upper bounds. We first improve the work in [WAN07] so that the boundary effect is ruled out and the monitored area can be completely covered in all cases. Next, we further extend that improved algorithm by introducing two distributed scheduling algorithms which are trade-off in term of network lifetime and algorithms' time complexity. The major objective of each of the three proposed algorithms is to balance energy consumption and to maximize network lifetime. Among three proposed algorithms, two support connectivity at all cases and the last one has to rely on an assumption to do the same thing. At the end of CHAPTER 4, we explain how to make the last one always provide connectivity as well.
- Regarding alarming application where the occurrence of an emergency event such as a fire is strictly required to be reported in a timely manner, we suggest a practical coverage problem where the network has to collect several kinds of simple environmental data (e.g., not only temperature but also light, smoke density and humidity), and then the sensors locally combine the data into a more meaningful information – which is called *composite event* (e.g.,

there is a fire). We propose a novel scheme which minimizes the delivery delay to base station (BS) (i.e., transmission delay of event occurrence to BS) and significantly reduces the exchanged data by allowing sensors to send compressed data and only do so when necessary. The scheme is able to work with heterogeneous networks where the heterogeneities are in terms of sensors' types (sensors in the network may each contain a number of sensing units, and sensing units may be different for different sensors), sensors' energy, and sensing/communication ranges. Clearly, the scheme provides basic coverage service when the "heterogeneous network" and "low notification delay" requirements are relaxed. The scheme will be discussed in detail in CHAPTER 5. Our analysis shows that the amount of energy could be saved is up to several tens percents when compared with traditional scheme.

  - To illustrate the proposed scheme, we further examine the Timely Energy-efficient *k*-Watching Event Detection problem (TEKWED) in CHAPTER 5. The notification delay, energy efficiency and fault-tolerance are all explicitly taken into consideration in TEKWED. A topology-and-routing-supported algorithm based on the scheme is proposed to construct a set of detection sets that satisfy the short (practically bounded) notification time, energy conservation, and tunable quality of surveillance requirements for event alarming applications.
  - An extension of the abovementioned cluster-based algorithm to large network is also briefly discussed in the end of CHAPTER 5.

- The final generalized problem we consider in this dissertation is the partial coverage problem. For some applications, the requirement to cover the whole area is not always mandatory. To save the network energy, only a portion of an area (whose ratio over the whole area is usually represented by parameter α) is required to be covered. The basic

coverage problem, which, in term of coverage quality, is also referred to as complete coverage, is a special case of partial coverage when the coverage ratio $\alpha=1$. Partial coverage appears to be more challenging than complete coverage due to the difficulty to verify when the area is adequately covered (i.e., the total amount of covered sub-areas occupies no less than α-portion of the whole monitored area). As a consequence, most of the existing works for partial coverage are non-parallel algorithms. Besides, most of them strongly depend on a parameter that is very hard to compute exactly. In CHAPTER 6, we solve the partial coverage problem by developing a completely novel approach. Instead of designing a few separate algorithms for partial coverage, we propose a framework which utilizes plenty of the existing complete-coverage algorithms and converts them for the partial coverage problem. The proposed conversion framework consists of four strategies which are very universal and applicable to conversion of almost any complete-coverage algorithms.

Additionally, at the beginning of each chapter, we may discuss a few works most related (as compared with related work discussed in CHAPTER 2) to the issue we deal with in that chapter and state the motivation. At the end of each chapter, we mention any enhancements or improvements of the work discussed in that chapter for which we already have solutions, but for conciseness reasons we do not go into great detail. For such extensions, we only sketch the general ideas of the solutions and point out possible directions. We dedicate Section 7.2 of CHAPTER 7 for enhancements or related issues that we will consider in our future work.

CHAPTER 2.

RELATED WORK

In this section, we make a classification and brief discussion of work related to the *basic* coverage problem in WSN. For each category (which will be presented in Section 2.2, 2.3, 2.4), we tabulate a brief comparison of work belonging to that category. Since this dissertation focuses on distributed scheduling algorithms, we only discuss the work in "decentralized algorithm" category in more detail. For other categories, we only give the reference where more detail can be found. The related work for generalized problems and variants of coverage problem which we consider in subsequent chapters will be discussed in more detail in each chapter.

2.1. Classification of coverage algorithms in the literature

Even there are many ways to group the work in literature into categories, in this dissertation we classify the existing work into following categories:

- Sensors placement: The works in this category concerns with the following issue and their algorithms are usually carried out prior or at the time of deploying the sensors.
    - Coverage/connectivity conditions: Find out the conditions (e.g., the number of sensors, sensor's sensing range) to provide certain level of coverage (with connectivity) for a sensor network.
    - Deployment schemes (deterministic deployments): Concerned with schemes on how to place the sensors to achieve a number of optimal objectives such as the best coverage quality or maximum network lifetime with least number of sensors possible.

- Sensors scheduling to achieve coverage/connectivity: To discover the schedule for a set of sensors to provide coverage and/or connectivity for a WSN with the assumption that the network has already been (randomly) deployed.
    - Centralized algorithms: Algorithms that require global information (such as sensors' sensing ranges, sensors' location, sensors' residual energy, etc.) of the whole sensor network. Besides, the algorithms are always executed at a powerful center such as BS, after that the result is scattered to each sensor in the network. We further divide this type of algorithms into two smaller groups:
        - Algorithms result in disjoint sets
        - Algorithms result in non-disjoint sets
    - Decentralized (localized and distributed) algorithms: Algorithms that require only the local information (the fix-number-of-hop neighbors' information, usually 1- or 2-hop information) to function and are run at large number of sensor nodes (usually at all nodes). Each sensor then makes it own decision of turning on or off (although its neighbors may have contribution on that decision). We further divide this type of algorithms into two smaller groups:
        - Decentralized algorithms based on back-off (or off-duty) mechanism.
        - Decentralized algorithms work in rounds.
    - Others algorithms: Algorithms that are not in neither of two above sub-categories, for example, verification of coverage problem.
- Quality of service (surveillance) - QoS evaluation: When a sensor network is deterministically or randomly deployed into monitored area, it is desirable to estimate how

well the sensor network can cover the area or a set of objects. In literature, there are two well-known criteria for this issue:

- Maximal breach/ support paths: The goal is to construct the *maximal breach/ support path*, which is the path through sensor-monitored field that an object is most/least likely to be detected in terms of distance from the object to the closest sensor
- Exposure: Another parameter can be used to evaluate coverage quality is exposure which is integral of a sensing function. An example of sensing function is the intensity of the sensed signal (or observability) with the appearance of an object in monitored field. Informally, exposure can be explained as expected average ability that a moving object is observed over a period of time [MEK01].

2.2. Sensors placement

For applications that allow to deploy the sensors manually, the positions of the sensors can be optimized to achieve the best coverage quality, to ensure the connectivity and/or to maximize the total network lifetime. For applications that do not permit to do so, it is desirable to estimate the number of sensors needed to guarantee that the deployed sensor network can provide several requirements (such as *k*-coverage, connectivity). Basically, the work in this category concern with two issues: creating sensors placement scheme such that some optimization objective is achieved or coming up with network condition such that some network requirement (connectivity/coverage) is satisfied. Because of the limitation and the focus in this dissertation, we will not discuss above work in more details. Rather, we only show several work belonging to this category in Table 2.1. Interested readers may find [VUT07] for more detailed information and discussion about a longer list of related papers.

Table 2.1. Work in literature that considers sensors placement stage

Coverage Approach	Problem solved	Coverage type	Approach characteristics
[WAN06]	Network condition	Area	Consider *k*-coverage with 2 kinds of deployments: Poisson and uniform point process; boundary effect.
[KUM04]	Network condition	Area	Consider *k*-coverage with 3 kinds of deployments: unit square grid, Poisson and uniform point process.
[CLO02]	Nodes placement	Target	Objective is to minimize the exposure of deployed network; Minimize the number of deployed sensors.
[ZOU03]	Nodes placement	Target	Algorithm bases on virtual forces (as magnetized objects exert on each other).
[ZOC03]	Nodes placement	Target	Static; provide maximum grid coverage for surveillance and target detection, while minimizing the cost of sensors
[CHA02]	Nodes placement	Target	Static; determine minimum number of sensors and their locations under constraint of imprecise detection and terrain properties
[POD04]	Nodes placement	Area	Maximize area coverage; Algorithm bases on potential field; the number of neighbors of each sensor is required to be at least K.
[KAR03]	Nodes placement	Area/ Target	Consider connectivity; Assume that $r_c=r_s$ and they are the same for all the sensors.
[WAH05]	Nodes placement	Area	Consider connectivity; The algorithm works for arbitrary-shaped region and with any ratio of r_c/r_s.
[DHI02]	Nodes placement	Area	Static, minimize the number of sensors and communication traffic. Allow modeling of obstacles and preferential area.
[HEO02]	Node placement	Area	Distributed self-deployment algorithm. Spread nodes from an initial random to maximize coverage and maintain uniformity.
[WAN04]	Node placement	Area	Mobile; reduce or eliminate coverage holes by relocating mobile sensors. Distributed algorithms, extensible to large deployment because communication and movements are local.

table continues next page »

[WAC03]	Node placement	Area	Mixed; reduce or eliminate coverage holes and minimize cost by relocating mobile nodes. Distributed protocol; provides cost balance by using a combination of static and mobile nodes.
[HOM02] [HOW02]	Node placement	Area	Mobile; deploy mobile nodes in an unknown environment to maximize coverage while retaining line of sight. Incremental and greedy algorithm; not dependent on prior environment models; global maps are built from live sensory data

2.3. Sensors scheduling to achieve coverage/connectivity

With a centralized approach, the algorithm usually runs at a special and powerful center (usually a BS) where the energy, communication and computation constraints can be ignored. The advantages are the nearly-optimal final results and the ease in implementing the algorithm. Nonetheless, the usual drawbacks are the mandatory requirement of global information of the whole network, the slow running speed, scalability and the low adaptability to the changes of the network. Oppositely, decentralized algorithms share the burden of executing algorithm to all (or at least a number of) sensors in the networks. What are the advantages of centralized algorithms is the disadvantage of decentralized ones and vice versa.

2.3.1. Centralized algorithms

The centralized algorithms always provide nearly or close to optimal solution since the algorithm has global view of the whole network. However, its disadvantage is very slow speed for collecting the information though the network and scattering the result throughout the network. Another contribution to slowness of this kind of algorithms is that they have to process on huge amount of information. By that reason, they have low-adaptability to the change of the network (for example, when a sensor died or when new sensors are added to network) and are not suitable for large-scalable network. Again, we do not focus on centralize algorithm in this

dissertation, thus we will not go into more detail in this category. Table 2.2 itemizes work that will be considered in this section

Table 2.2. Centralized approaches for coverage problem in literature

Coverage Approach	Energy-efficient	Set-cover type	Coverage type	Approach characteristics
SET K-COVER [SLI01]	NO	Disjoint	Target/ Area	Maximize the number of set-covers.
[ABR04]	NO	Disjoint	Area	Maximize the number of times sub-areas are covered.
[CHE05]	YES	Disjoint	Target	Minimize breach under bandwidth constraint.
[GAO06]	NO	Disjoint	Area	Consider *k*-coverage problem; Maximize the number of set-covers.
[ZHO04]	NO	Disjoint	Area	*k*-coverage and connectivity.
[CAT05]	NO	Disjoint/ Non-Disjoint	Target	The ILP method produces a non-disjoint set of set covers. The greedy method generates a disjoint one.
[CAW05]	YES	Non-Disjoint	Target	Consider discretely adjustable sensing range sensor networks under energy constraint.
[THAI05]	NO	Non-Disjoint	Target	Minimize coverage breach/Maximum network lifetime under bandwidth constraint.
[DHA06]	YES	Non-Disjoint	Target	Maximize network lifetime for smoothly adjustable sensing range sensor network under energy constraint.
[BER04]	YES	Non-Disjoint	Area/ Target	Considering partial coverage (*q*-coverage). Taking communication cost into account.
[GAO08]	YES	Non-Disjoint	Area	Consider partial coverage (p-percentage coverage)
[WUA08]	YES	Non-Disjoint	Area	Consider partial coverage (p-percentage coverage)

2.3.2. Decentralized algorithms

With the distributed & localized algorithms, the decisive process is locally and simultaneously carried out at sensor nodes who need only local information (e.g., the position of itself and its neighbors, their sensing regions, etc.), thus being very adaptable to the dynamic and scalable nature of sensor networks. Obviously, the distributed & localized algorithms are preferred in wireless sensor network. Normally, the localized and distributed algorithms result in non-disjoint set covers. Table 2.3 provides a brief list of work that (not all, however) will be considered in this section along with some of their characteristics.

Table 2.3. Distributed approaches for coverage problem in literature

Coverage Approach	Energy-efficient	Connectivity support	Coverage type	Approach characteristics
Coverage Configuration Protocol (CCP) by Wang et al. [WAN03]	YES	YES	Area	Consider *k*-coverage. Based on a new eligibility rule for a sensor to turn on/off. Employ SPAN [CHE02] to provide connectivity for the network.
Localized, low communication overhead algorithm. [GAL06]	NO	YES	Area	Utilize the rule introduced by [HAL88].
[VUC06]	YES	NO	Area	Consider *k*-coverage problem.
[BER04]	NO	NO	Area	Consider 1-coverage problem with the support of special data structure representing a monitored area.
Coverage preserving protocol [HUL05]	NO	NO	Area	Utilize the rule introduced by [HAL88]. The main idea is similar to that discussed in [YAN03].
[CAW05]	YES	NO	Target	Consider a discretely adjustable sensing range sensor network.

table continues next page »

Location-free coverage maintenance [ZHE05]	Depend	Depend	Area	One of inputs of this algorithm is a DS algorithm. So the characteristic of the algorithm depends on the DS algorithm. Also, the proposed algorithm is location-free if the DS algorithm is also location-free.
Optimal Geographical Density Control (OGDC) [ZHA03]	YES	YES	Area	Choose next sensor to join the current set cover based on its position such that the coverage overlapping area is minimized.
[TIA02]	NO	NO	Area	There are some flaws with the algorithm pointed out by [JIA04].
[JIA04]	NO	NO	Area	The eligibility rule for a sensor to join to or withdraw from set cover is a variation the one discussed in [HAL88].
[ABR04]	NO	NO	Area	The algorithm maximizes the number of time areas are covered.

2.3.2.1. Algorithms that use back-off mechanism

In this type of distributed algorithm, a sensor frequently checks its validity to join the network by an eligibility rule specifying by the algorithm. It is difficult to take the energy into account with this type of algorithm.

To the best of our knowledge, [WAN03] is the first paper investigating the relation between *k*-coverage and *k*-connectivity. It is proven in that paper that if a WSN has all the sensors with the communication range being at least twice of the sensing range then *k*-coverage also means *k*-connectivity. Another big contribution of this paper is an thorough discussion on a very simple rule to convert from verifying the coverage levels of an area into determining the coverage level of all the intersection points (intersection point is the point that the sensing circle of a sensor intersect with those of others or with the borders of monitored region – see Figure

2.1). That rule is a generalization of the rule previously introduced in [HAL88]. This rule states that if all the intersection points between sensors' perimeters; and sensors' perimeters and monitored region boundaries are sufficiently *k*-covered, then the whole region is sufficiently *k*-covered. A sensor can apply this rule to check its eligibility to join/leave the network (or in other word, turning on/off). Besides, the authors propose an algorithm named CCP to schedule turning on/off the sensors in order to assure the *k*-coverage for the whole area while conserving the energy. In CCP, each sensor occasionally verifies the eligibility to join the network. The algorithm initializes by setting all the sensors to be *active*. When a sensor is in the *active* mode and receives a HELLO message, it will use eligibility rule to see if it can turn off or not. If it can, it turns itself off, goes to *sleep* mode and sets a sleep timer to turn on after certain interval. When that sleep timer expires, the sensor turns into *listen* mode. In this mode, the sensor again checks the eligibility to join the network or continue sleeping. If it qualifies to continue turning off, it again sets up the sleep timer and goes to sleep. Otherwise, it turns into *active* mode. The authors further combine the rule of CCP with that of SPAN [CHE02] (a decentralized algorithm that offers the different connectivity for the network) to provide both the coverage and connectivity for the network in the case that the communication range is less than twice sensing range.

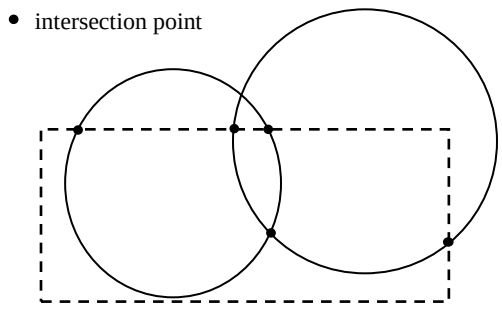


Figure 2.1. Intersection points example

2.3.2.2. Algorithms that work in rounds

In this kind of distributed algorithm, the time line is divided into rounds and each round usually comprises two phases, which are the decision phase (the small interval of time as compared to length of the whole round for sensors to decide to turn on or turn off) and sensing phase (the remaining time of a round for sensors to do their sensing tasks). The algorithm is periodically executed at the beginning of each round. The advantage of this type of algorithm is that the energy consumption and some other constraints can easily be taken into account since the sensors can update and then exchange the information (including their residual energy) each time carrying out the algorithm. However, its disadvantage is that at each round, the sensors must consume the certain amount of energy in decision phase even when it may not join the network that round. In addition, this kind of algorithms usually requires time synchronization among sensors (at least neighboring sensors) to correctly function.

[GAL06] uses a result from [HAL88] as [HUL05] do to verify the coverage condition. The algorithm achieves the low communication overhead by allowing sensors to choose a random waiting time before deciding their status and by limiting the messages exchanged between neighboring sensors. Four variants are introduced corresponding with different limitations in exchanging messages:

- Positive only (PO): Only active sensors send exactly one messages each after they decide to turn on
- Positive and Negative (PN): Every sensor sends exactly one message notifying its decision.
- Positive and Retreat (PR): Only active sensor sends message informing its turning on. If at later time, it learns that others sensors (ones that were already active) can still cover with the

help of newly active sensors, it can change it decision (i.e., turn off). Neighbors are informed about this decision change by a retreat message.

- Positive, Negative and Retreat (PNR): Similarly, a sensor may send Negative message; Active message or Retreat message (after Active message) notifying its decision.

The four variants above are trade-off between communication overhead and the goodness of solution, i.e., the number of active sensors each round.

At the beginning of each round, each sensor sets it own waiting timer with randomly chosen interval. At timeout, it then uses the coverage condition rule to evaluate the coverage status. If all of its intersection points with already-active sensors are covered by other active sensors, it can turn itself off. Otherwise, it has to turn on. After deciding its status, the sensor may need to inform its neighbors by following one of four above policies about exchanging messages. Although connectivity is also considered in the proposed algorithm, no condition on evaluation of the connectivity of active sensors that a sensor node can follow to verify the connectivity of current set cover (as that of coverage) before making decision is clearly specified. This makes this work less convincing.

[HUL05] is an improvement of work in [YAN03]. The major improvement is the utilization of well-known rule of intersection points to check the coverage of an area as being used in [GAL06]. That is, each sensor only needs to check its intersection points to decide to turn on or off. As most heuristic in this category, this algorithm partitions the time line into a number of equal intervals, called *working circle* (which corresponds with *round* in other work). Also as other work, each *working circle* is then divided into two phase: initial phase – for sensors to exchange information and maybe to make decision of turning on or off; and sensing phase. However, this work further divides sensing phase into a number of rounds of length T_{rnd} each,

and each sensor needs to choose on-duty time for each round (this on-duty time needs not be the whole round). At initial phase, each sensor s_i chooses a value Ref_i and exchanges that value (along with sensor's position and sensing range) with its neighbors. Based on *Ref* values of neighboring sensors, it calculates two others value: $Front_i$ and $Back_i$ and it will active from $[(Ref_i - Front_i) \bmod T_{rnd}]$ to $[(Ref_i + Back_i) \bmod T_{rnd}]$. Following is method to calculate $Front_i$ and $Back_i$. Let P be the set of all the intersection points that are inside sensing region of sensor s_i. For each point $p \in P$, s_i creates a circular list L_p (meaning the last element is wrapped around to be right ahead of the first one) of *Ref* values of all the sensors who can cover that point in ascending order of *Ref*. Let $prev(Ref_i)$ and $next(Ref_i)$ be the previous and next element of Ref_i in list L_p. Then for each intersection point $p \in P$, the following values are to be computed:

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

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

At last: $Front_i = \max_{\forall p \in P}\{Front_{p,i}\}$ and $Back_i = \max_{\forall p \in P}\{Back_{p,i}\}$.

A variant of above algorithm which takes energy into account is also discussed in this paper. In this variant, sensors with more and less residual energy may choose Ref_i in different ranges, say $[0, \frac{3T_{rnd}}{4})$ and $[\frac{3T_{rnd}}{4}, T_{rnd})$. Above equations calculating $Front_{p,i}$ and $Back_{p,i}$ are also slightly changed: $\frac{1}{2}$ in above equalities are changed to the ratio of residual energy of sensor s_i to total residual energy of s_i and those of the sensors being at previous/next position in list L_p.

The drawback is that sensor may have to turn on/off too frequently if its active time is only a part of each round – which is usually the case as the result of this heuristic (note that the sensor need non-negligible amount of energy to turn on).

As the title "Location-free coverage maintenance" of [ZHE05] suggests, this work deals with area coverage problem without the knowledge of sensors' locations that is extensively required in other existing work. This work assumes all the sensors have the same sensing range r_s, the same maximum communication range R_c but the communication range can be easily and arbitrarily changed. Firstly, the authors theoretically prove a result claiming the ratio (which will be referred later as *coverage ratio*) of the area that dominators of a minimal dominating set (MDS) can covers on the area that the whole network can cover (which may not be the whole area) is greater than or equal $\frac{r_s^2}{(r_s + r_c)^2}$. The authors further prove another important theorem which states that if the sensor nodes follow Poisson point process rate ρ, then for any $\varepsilon > 0$ if set the communication range to a function $t = f(r_s, \rho, \varepsilon) = s - \min(s, \sqrt{\frac{-\log(\varepsilon + e^{-\pi\rho s^2} - \varepsilon e^{-\pi\rho s^2})}{\pi\rho}})$ then the probability that any MDS has coverage ratio of 1 is *at least* $1 - \varepsilon$. Based on this result, the proposed algorithm just lets each sensor node estimate the node density in its area (through exchanging message, not by the location of sensor's neighbors). Note that this density can be estimate through 1 or several hop(s) neighbors (neighbors here are the ones that sensors are able to reach when they use their maximum communication range R_c). Each sensor then sets it communication range to function *f* above. Finally, any available DS (dominating set) algorithm can be utilized to discover a DS. This last step allows creating a suite of protocols which meets a number of requirements depending on the characteristic of DS algorithm such as balancing power consumption, maximizing network lifetime, etc. Although the result is fairly convincing, the authors however avoid the boundary effect on deriving that result.

In the distributed algorithm discussed in [CAW05] (refer to Section 2.3.1 for the other part of this paper), the idea of so-called waiting time is employed. That is, each sensor maintains its own timer and the sensor has to make decision when this timer expires. The duration of this timer depends on the sensor's residual energy and the minimum sensing range needed to cover all uncovered targets that it is able to cover (i.e., when it uses its maximum sensing range). Each time a neighbor of a sensor becomes active, the set of uncovered target is changed and thus the sensor's waiting time is consequently altered. When the sensor's waiting time expires, if all the targets that it is able to cover have already been covered, then it can turn off. Otherwise, it turns on and uses the smallest possible sensing range to cover all the uncovered targets. Hence, after the sensor which initially has the smallest waiting time is active, all other sensors will eventually turn on or off. The algorithm introduces the communication overhead since targets list exchanged between neighboring sensors may be relatively long.

Based on an observation that an area is covered if there are at least two disks intersecting and their crossing points (or intersection point as is defined in [WAN03]) are covered, [ZHA03] does some trigonometric analysis to determine the most suitable positions for sensors in order to reduce the overlap between their sensing regions. We call such position to be "optimal position". The main idea of OGDC is to add a sensor which locates at place that closest to "optimal position" corresponding with the sensors that were already in the set cover. The timeline is partitioned into rounds. Each round has two phases: The *node selection phase* where the algorithm is executed and the *steady state phase* where active sensors sense the data. At the beginning of *node selection phase*, all the sensors are in UNDECIDED status, which will be in ON or OFF status at the end of this phase. The algorithm begins with a selection for the starting node. This node can be chosen based on its residual energy. After a back-off timer expires

without any other node becoming the starting node, that node will volunteer to be the starting node. It changes its status to ON and broadcasts *power-on message*, which contains the position of the sender and the direction α to the "optimal position" of second working node. The procedure that a node should follow when it receives *power-on message* can be described in Figure 2.2. When a node receives that message, it adds this neighbor to its neighbor list and checks if its sensing region is covered by sensors in its neighbor list or not? If yes, it can change status to OFF and turn itself off after setting a timer for waking up next round. If no, the node can change its state to ON only if it is the one that closest to "optimal position". Otherwise, it continues waiting for another *power-on message.*

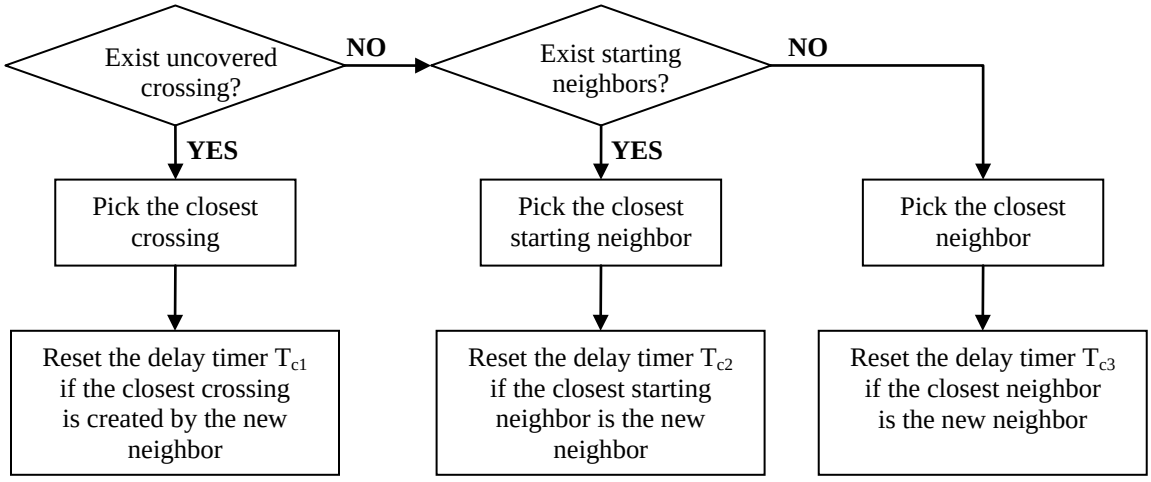


Figure 2.2. The procedure when a node receives a power-on message [ZHA03]

The paper also considers the relationship between coverage and connectivity. The proof for the assurance of connectivity when the network is already covered in the case the sensing range at least twice communication range is formally provided in the paper.

In distributed algorithm of [TIA02], the decision phase consists of two steps: *a)* exchanges position information with neighbors and *b)* decides its status based on that

information. The status decision is made by a rule named *off-duty eligibility rule* which tells a sensor to turn off if its sensing region has been covered by its neighbors. Figure 2.3 explains the status transition of a sensor. When a sensor decides to turn off, it waits for random back-off time T_d (to avoid the case two sensors may turn off at the same time, thus causing the *blind point* – the point in the surveillance area but is covered by no active sensor) and then sends SAM (Status Advertisement Message) to inform all of its neighbors about its new status. To further avoid any possible *blind point*, the sensor waits for T_w time after sending SAM message. If it receives no such message from any neighbor, it can safely turn off.

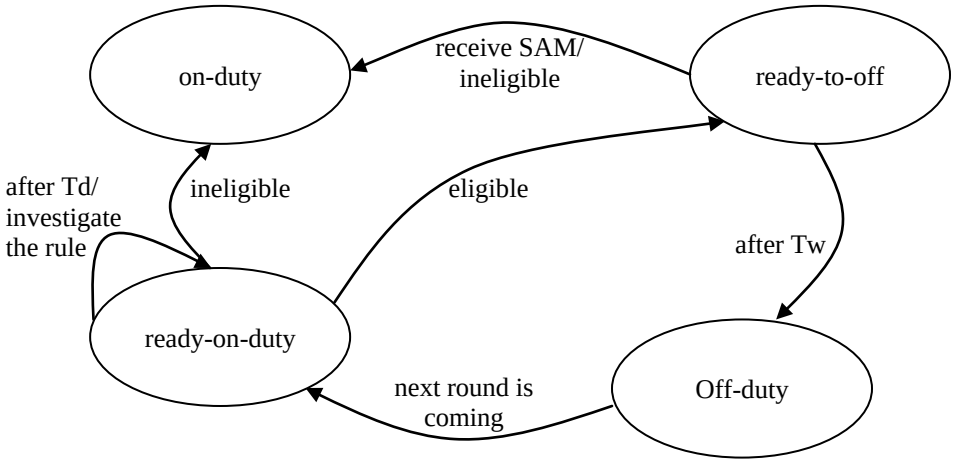


Figure 2.3. The status transition graph [TIA02]

The algorithm expressed in [TIA02] is the base for [JIA04]. It is shown in this paper that the eligibility rules in [TIA02] have some flaws which make the algorithm in [TIA02] not fairly efficient. In this paper, a rule named *effective neighbor rule* is utilized for a sensor to decide to turn off. This rule bases on an observation that the sensing region of a sensor is covered by its neighbors if and only if the segment of each neighbor, which is inside the sensor's sensing range, is also perimeter-covered by other neighbors. The algorithm proposed in this paper employs the idea of working in rounds and *off-duty eligibility rules* proposed in [TIA02]. After having the

information of all of its neighbors, the sensor uses *off-duty eligibility rules* to decide its status. To avoid the *blind point*, the sensor waits for a random back-off time (corresponds with T_w in [TIA02] – see paragraph above), and then the *effective neighbor rule* is applied for a sensor to decide if it can safely turn off or not.

2.3.3. Others

Similar to intersection points rule introduced in [HAL88] and [WAN03], [HUT05] suggests another rule to evaluate the coverage degree of an area by doing a check at each sensor. Two rules named *k*-Unit-disk Coverage (*k*-UC) and *k*-Non-unit-disk Coverage (*k*-NC) for uniform and non-uniform sensing range sensor networks, respectively, are proposed in this paper. With the assumption that the sensing region of each sensor is a disk centered at the sensor with radius of its sensing range, those rules state that the whole area is *k*-covered if and only if the perimeter of sensing regions of all sensors are *k*-covered. In fact, *k*-NC is the generalization of *k*-UC and it can easily and simultaneously be applied at each sensor with the requirement of only 1-hop-sensing-neighbor information.

To determine the coverage level of perimeter of a sensor s_i, ones can calculate the angle corresponding to the arch that each of its neighbors covers its perimeter. Figure 2.4.a illustrates such arches. The angles corresponding with those arches, which were flattened into the range $[0,2\pi]$, are shown in the Figure 2.4.b. From Figure 2.4.b, the coverage level of sensor s_i perimeter can easily be calculated by traversing the range from 0 to 2π.

Surprisingly, this statement can as well be applied for sensors with irregular-shaped sensing regions. Figure 2.5 shows an example of a network comprising such sensors in which the number inside each sub-region is coverage level of that region. It is easy to validate that the

whole area is perimeter-1-covered since the perimeters of some sensors are covered by the same level (i.e., equal to 1). Nonetheless, no further scheduling approach is proposed in this paper.

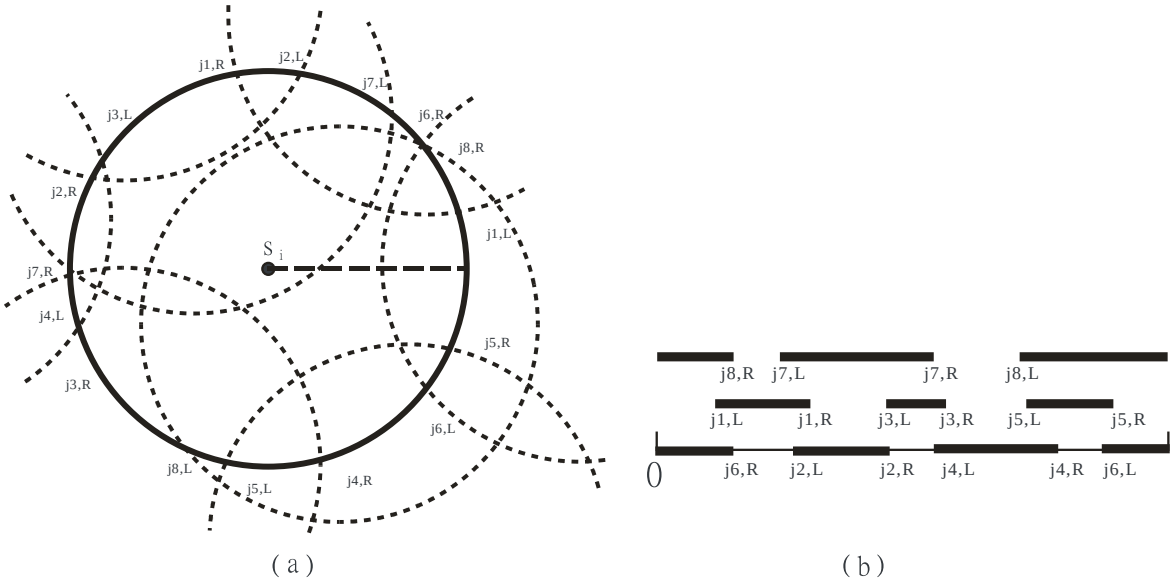


Figure 2.4. Transform to perimeter coverage [HUT05]

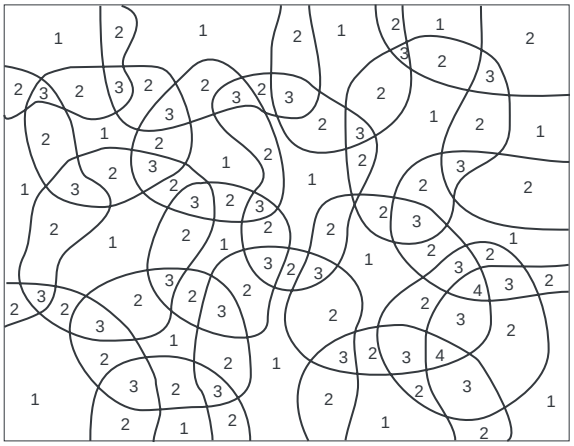


Figure 2.5. Irregular sensing regions [HUT05]

Beside two centralized algorithms discussed in Section 2.3.1, [ABR04] also proposes a distributed one. Recall that each sensor has a unique ID. Each sensor executes the algorithm only one time, when the network initiates, to assign itself into a set covers. The basic idea of the

greedy distributed approach is relatively similar to the centralized greedy one. That is, among k set covers, each sensor chooses the one in which it has the biggest contribution. As also was defined, the contribution here is the number of uncovered areas inside sensor's sensing region (i.e., sensor can cover). The point of time when sensor s_i makes the decision is $t=i$, which guarantees that no two sensors make decision at the same time. The time complexity of this algorithm is $nk|s_{max}|$ where $|s_{max}|$ is the biggest number of areas that a sensor can cover. Though the algorithm is somewhat simple, it assures the performance ratio of 1/2 in compared with optimal solution. Another interesting point of this algorithm is that even though it is distributed algorithm, the resulting set covers are still disjoint.

2.4. Quality of service evaluation

It is always desirable to estimate how good or how bad a deployed network is. This question is however not easy to answer. In literature, the common method to estimate the quality of a network is as follows:

- Firstly, finding a path through the sensor network which is best/worst observed by the network.
- The resulting path is then evaluated based on some evaluation metrics.

As mentioned before, there are two metrics which can be used to mathematically evaluate the goodness of a path: the distance to the closest sensor and the exposure. Due to extensive use of the minimum/maximum distance from or to a sensor, some well-known graph structure such as *Delaunay triangulation* and its dual *Voronoi diagram*, *Grabriel graph*, or *relative neighborhood graph* are extensively employed by the work in this category. Some works considering evaluating a sensor network are listed in Table 2.4 (see next page). Notice that the

proposed solutions in this category can be helpful to answer reverses questions: “How to active sensors and/or decide sensing range to reduce breach/exposure”. However, to the best of our knowledge, there are no work in literature concerning the reversed questions.

Once again, we will not discuss above work in more details because of the length and our concentration in this dissertation. Please refer to [VUT07] for discussion and comparison in greater detail.

Table 2.4. QoS work in literature

Coverage Approach	Algorithm type	Evaluation metric	Approach characteristics
[MEG05], [MEK01]	Centralized	Maximal breach/ support path	Use Voronoi diagram/Delaunay triangulation structure to find Maximal breach/support path, respectively.
[LIW03]	Distributed, Localized	Best coverage path	Use Relative Neighborhood Graph and Gabriel graph.
[HUR05]	Distributed, Localized	Best/Worst coverage radius	The algorithms utilize some complicate data structures to dynamically determine sensor radius such that extreme Worst/Best coverage path exists.
[MEQ01]	Centralized	Exposure	Calculate exposure by partitioning the monitored area into numbers of square grids
[MES01]	Distributed, Localized	Exposure	Utilize Voronoi diagram to partition the monitored area. The exposure path is then searched along the edges of this diagram.
[XUH05]	Distributed, Localized	Worst coverage path	The scheme is relatively similar to those in [LIW03].

CHAPTER 3.

THE *k*-COVERAGE PROBLEM

In this chapter, we define and solve a generalization of basic coverage problem that requires a certain level of fault tolerance and energy balancing. Specifically, the problem requires that every point inside the monitored area is always covered by at least *k* active sensors.

3.1. The SESK problem

This section is dedicated to first define and then formulate the Sensor Energy-efficient Scheduling for *k*-coverage (SESK) problem – our main problem in this section. We further confirm that SESK is an NP-complete problem. The heuristic for this NP-complete problem is given in the next section.

Definition 3.1. *A location in an area A is said to be covered by sensor s_i if it is within s_i's sensing range. A location in A is said to be k-covered if it is within at least k sensors' sensing range. Area A is said to be k-covered if every point within it is k-covered.*

In this text, *k* is called the *coverage level* or *coverage degree*. The SESK problem is defined as follows:

Definition 3.2. *Sensor Energy-efficient Scheduling for k-coverage* (SESK)*: Given a two-dimensional area A and a set of N sensors S ={s_1, s2, …, s_N}, derive an active/sleep schedule for each sensor such that:*

1) *The whole area A is k-covered.*
2) *The energy consumption among all the sensors is balanced.*
3) *The network life time is maximized.*

Our objective is to find the maximum number of non-disjoint sets of sensors such that each set cover can assure the *k*-coverage for the whole region. In [SLI01], the "SET K-COVER" problem, whose goal is to discover *K* disjoint set covers satisfying that each set cover can 1-cover the whole area, is proven to be NP-complete. Since disjoint set is a special case of non-disjoint set and 1-cover is also a special case of *k*-cover, "SET K-COVER" is definitely a special case of SESK. Thus, it follows that the following theorem holds:

Theorem 3.1. *SESK is a NP-complete problem.*

One of our optimization goals is to maximize network lifetime, which is defined as following:

Definition 3.3. (Network lifetime) *Network lifetime is the duration during which the whole monitored region is k-covered.*

To mathematically formulate the SESK problem, the following notations need to be stated:

- m: The number of discovered non-disjoint set covers.
- k: The desired coverage level specified by users.
- $C_j (j = 1..m)$: The j^{th} set cover.
- $cov_j (j = 1..m)$: The coverage level that set cover C_j can provide for the whole monitored area.
- $E_i (i = 1..N)$: The initial energy of sensor i.
- $e_{j,i} (i = 1..N; j = 1..m)$: The amount of energy that sensor i consumes when the set cover Cj is active. $e_{j,i} = 0$ if set cover C_j does not contain sensor i.
- e_i^{die} : The residual energy of sensor i at the time the network dies.

The SESK problem can be mathematically formulated as follows:

Objective:

$$Max\ m \tag{3.1}$$

$$Min \sum_{i_1;i_2}^{N} (e_{i_1}^{die} - e_{i_2}^{die})^2 \tag{3.2}$$

Subject to:

$$\bigcup_{j=1}^{m} C_j \subseteq S \tag{3.3}$$

$$\mathrm{cov}_j \geq k \ for\ all\ j = 1..m \tag{3.4}$$

$$\sum_{j=1}^{m} e_{j,i} \leq E_i \ \ for\ all\ i = 1..N \tag{3.5}$$

Formulation explanations and remarks:

1. Eq. 3.1 claims that our objective is to find as many subsets as possible. Since DESK works in rounds, to maximize the number of subsets is to maximize the lifetime of the network.
2. By minimizing the difference residual energy of sensors at the point of time network dies, Eq. 3.2 is an effort to balance the energy consumption among all the sensors
3. Eq. 3.3 guarantees that all the set covers are the subsets of the set of all sensors.
4. Eq. 3.4 assures that the whole region is continuously *k*-covered by forcing the coverage level of any set cover must be at least *k*.
5. By forcing the total energy consumption of each sensors in all set covers is no bigger than its initial energy, Eq. 3.5 ensures that sensors cannot overspend their initially supplied energy.
6. No relation between set covers is specified since they are non-disjoint. Furthermore, they are possibly identical.

3.2. The DESK algorithm

In [GAO06], we solved a problem similarly to SESK by a centralized algorithm. However, to find an approximate solution to the SESK problem, a NP-complete problem, we introduce a completely localized and distributed heuristic named DESK (Distributed Energy-efficient Scheduling for *k*-coverage) that *a)* requires only 1-hop-sensing neighbors' information to *b)* discovers and schedules the non-disjoint subsets of sensors which can guarantee the *k*-coverage over the working area where *k* can be changed by users. *c)* We as well take energy into consideration. *d)* We mathematically model the time a sensor needs to wait before deciding its status using parameters α, β which can dynamically tune the algorithm corresponding to user's requirement on energy's priority. That is, if the energy consumption is a very critical issue, the user can assign α a very high value and β a low value. In contrast, if energy is not the major concern, the value of α may be small and β may be large. *e)* For the sake of evaluating the DESK's efficiency, we develop a simple energy model that takes sensing, communication and computation energy consumptions into account.

3.2.1. Main idea

DESK operates in rounds. By that, the network is capable of automatically adjusting coverage level until the number of live sensors is not enough to *k*-cover the whole surveillance area. Also by working in rounds, some sensors may frequently have a chance to deactivate. Thus, their battery can take the advantage of the relaxation effect mentioned in [CHI99]. This helps a sensor to live longer than its pre-defined longevity.

Firstly, we introduce the *k*-perimeter-coverage which is stated in [HUT05] as following:

Definition 3.4. *A sensor is said to be k-perimeter-covered if all the points on its perimeter are covered by at least k sensors other than itself.*

Our work is based on the result from [HUT05] which is formally stated in the following theorem:

Theorem 3.2. *Suppose that no two sensors are located in the same location. The whole network area A is k-covered if and only if each sensor in the network is k-perimeter-covered.*

This theorem indicates the rule to validate the coverage levels of each sub-region of the monitored area. Based on that, our algorithm schedules the sensors to be active/sleep with the consideration of each sensor's residual energy and its contribution to the coverage level of the whole network.

3.2.2. Assumptions

We assume that all the sensors have a clock with a uniform starting time t_0, so that their activities can be synchronized. This is realistic since some work has investigated the global synchronization and both centralized and localized solutions have been proposed ([ELS02], [LIR04]). The second assumption is that the initial network deployment guarantees that every point in the monitored area can be at least *k*-covered. The condition to satisfy this assumption has been addressed in [KUM04] and [WAN06] (see Section 2.2). In our work, the sensing area of a sensor is modeled as a circle centered at the sensor with radius as its sensing range. We further assume that the communication range of a sensor is at least twice the sensing range, i.e., $r_c \geq 2 \times r_s$. Thus, the *k*-coverage can guarantee *k*-connectivity ([WAN03], [ZHA03]). Finally, we assume that no two sensors are located at the same position. However, we have no restriction on a sensor's initial energy and the sensing range.

3.2.3. Algorithm parameters

For the sake of explanation later on, the notations, a sensor's status and message types are introduced as follows.

- Sensor's attributes:
  - w_i: Timer/time duration that decides the time sensor s_i to become active/sleep. w_i refers to both the timer itself and the time duration.
  - R_i: Timer for sleep sensor s_i to wake up at the next round.
  - n_i: The current number of dependent neighbors, i.e., the number of neighbors requesting sensor s_i to become active.
  - N_i: The number of neighbors of sensor s_i.
  - r_i: Sensor s_i's sensing range.
  - E_i: Sensor s_i's initial energy.
  - e_i: Sensor s_i's current residual energy.
  - $e_{threshold}$ (Threshold energy): The minimum amount of energy that a sensor needs to be active in a whole round.
- Exchanged messages:
  - mACTIVATE: A sensor informs others that it becomes active.
  - mASK2SLEEP: A sensor suggests a neighbor to go to sleep due to its uselessness. The concept of uselessness is explained in Section 3.2.4.1.

  - mGOSLEEP: A sensor finds itself useless, i.e., all of its neighbors ask it to deactivate and itself is already *k*-covered.

- Sensor's status:
  - ACTIVE: Sensor is active.
  - SLEEP: Sensor decides to turn off.
  - LISTENING: Sensor has not yet decided.

- Others:
  - *L*: List of non-sleep neighbors.
  - Δ: Maximum number of neighbors that a sensor may have (or degree of the network).
  - Communication complexity: Estimated by the number of sent messages.

On what follows, we discuss necessary parameters and factors used in the proposed algorithm.

- DESK works in a rounding fashion with the round length of *dRound*, meaning that each sensor runs this algorithm every *dRound* unit of time. At the beginning of each round is a decision phase with the duration of *W*. The value of *W* and *dRound* should be chosen such as *W*<< *dRound* (see Figure 3.1). There are several advantages of working in rounds:
  - *k can be dynamically changed:* For some applications, such as forest fire, the value of *k* needs to be changed while the network running. For example, in the dried season, there is more chance of fire happening, thus the value of *k* needs to be high. However, in the rainy season, that chance is small, so the value of *k* needs to be small to save

network energy. Also, the operation of the network needs not be interrupted while *k* is being changed.

- DESK supports robustness: At each round, there is exactly one set cover in responsible for sensing task. In the situation that some sensors in that set cover are out of service (may die, for example), then the sensing data will be effected and network may temporarily not provide *k*-coverage for some interval of time. However, this problem will not affect long since the new set cover will be discovered at the next round to take charge of sensing task.
- Besides, DESK is an energy-efficient distributed algorithm which requires only 1-sensing-hop-neighbor information and DESK also provide *k*-coverage for the whole network (which is a kind of fault-tolerance). Thus, DESK satisfies all the requirements of a sensor network protocol shown in Section 1.2.

- All the sensors have to decide its status in the decision phase. At this phase, each sensor needs to temporarily turn on to decide its status.
- Every sensor s_i decides its status (active/sleep) after waiting for w_i time. The value of w_i may be changed anytime due to the active/sleep decision of any of its neighbors. Besides, the value of w_i depends on s_i's residual energy e_i and its contribution c_i on coverage level of the network. Sensor's contribution c_i can be defined in terms of some parameters, such as the perimeter coverage p_i which is the summation of perimeter coverage (in radian) that s_i covers its neighbors' perimeters. However, in this section we define c_i as the number of the neighbors n_i who need s_i to be active.

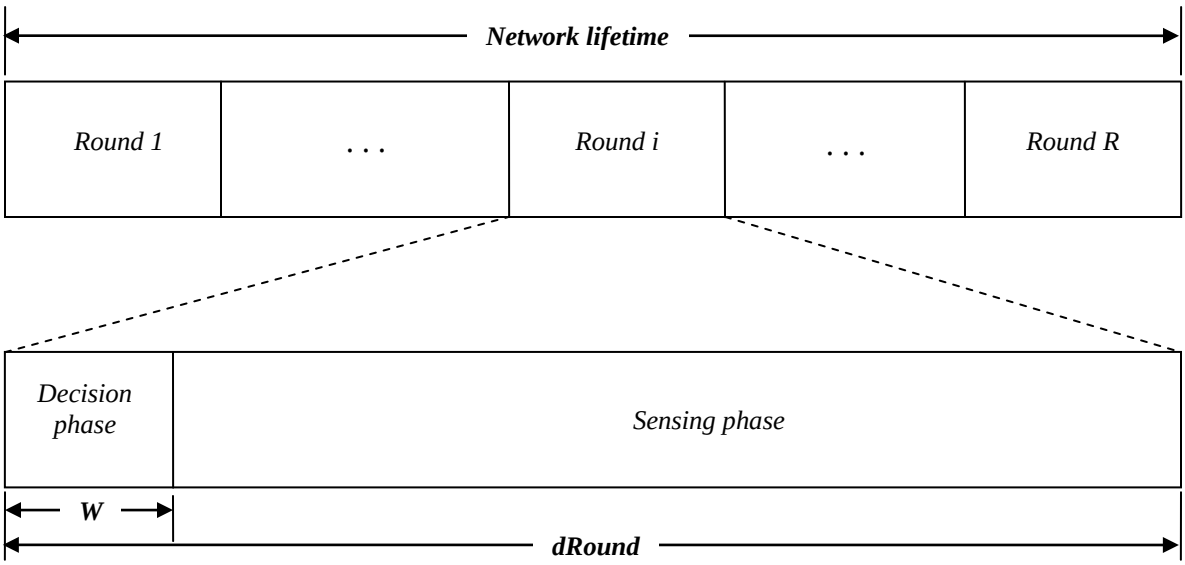


Figure 3.1. DESK works in rounds

The waiting time for sensor s_i can be formulated as follows:

$$w_i = \begin{cases} \dfrac{\eta}{n_i^{\alpha} l(e_i, r_i)^{\beta}} \times W + z & \text{if } e_i \geq e_{threshold} \\ W & \text{otherwise} \end{cases} \quad (3.6)$$

Where: α, β, η are constants, z is a random number between [0; d], where d is a time slot, to avoid the case where two sensors having the same w_i to be active at the same time. $l(e_i, r_i)$ is the function computing the lifetime of sensor s_i in terms of its current energy e_i and its sensing range r_i. This function may be linear, quadratic or anything else. The function $l(e_i, r_i)$ will be discussed in detail in Section 3.3.2.

$e_{threshold}$ and η are network-dependent parameters. η is chosen to make sure $w_i \leq W$ and $e_{threshold}$ guarantees that a sensor can live for a whole round:

$$e_{threshold} \text{ satisfies } l(e_{threshold}, r_i) = W \quad (3.7)$$

$$\eta = dRound^{\beta} \quad (3.8)$$

3.2.4. The algorithm

In this section, we first show the pseudo-code of DESK and then describe it in more detail. To better illustrate DESK, we also present a simple example which shows DESK step by step at the end of this section.

3.2.4.1. Pseudo-code

The pseudo-code for DESK is illustrated as in Pseudocode 1 (next page).

In the pseudo-code, the term "useless neighbor" or "redundant neighbor" is used to refer to one that does not contribute in the perimeter coverage of the considered sensor. That is, the portion of the perimeter of the considered sensor overlapping with that neighbor is already *k*-covered by already active sensors.

It is worth noting that although DESK works in rounds, no interruption in executing sensing task exists. As being stated in Section 3.3.2, a sensor can still sense data while being in LISTENING mode. Thus, by entering the LISTENING mode at the beginning of each round, sensors still perform the sensing job while participating in the decision phase. This guarantees the continuous and smooth operation of the whole network.

Line 6 may not be necessary. However, it can help improve the algorithm’s performance and result. We further clarify this in the example later in the next section. Notice that in line 17, waiting time w_i is updated only when the status of sensor has not yet been decided and the residual energy is enough for the sensor to live through the whole round. Ones may also be aware that the sensor continues running DESK even after becoming active.

Usually, a sensor decides its status when its timer expires and its timer keep changing through time, thus DESK works in highly asynchronous manner.

Pseudocode 1: DESK(s_i)

1: ► *Preparation*
2: Update current residual energy e_i
3: Collect information and construct the N_i-element list L of its one-hop neighbors
4: Compute the waiting time w_i and start the decision phase timer t
5: *status*=LISTENING
6: Pre-check redundant neighbors, sends mASK2SLEEP messages to them and move them out of list L if any found.
7: n_i = number of elements of list L
8: **while** $t \leq W$ **do**
9: ► *Receive a message from neighbor* s_j
10: Receive(s_j , MessageID)
11: **if** MessageID==mACTIVATE **then**
12: Update coverage level
13: Check if any sensor in list L is useless to s_i's coverage. If yes, send ASK2SLEEP message to that sensor
14: **else if** MessageID==mASK2SLEEP **then**
15: $n_i = n_i - 1$
16: **if** $n_i > 0$ **and** status == LISTENING **then**
17: Update w_i
18: **end if**
19: **else if** MessageID==mGOSLEEP **then**
20: Remove s_j out of list L
21: **end if**
22: ► *Decide status*
23: **if** ($t \geq w_i$ **and** *status*==LISTENING) **or** n_i==0 **then**
24: **if** n_i == 0 **then**
25: Set the timer R_i for s_i waking up at next round
26: One-hop broadcast mGOSLEEP message
27: *status*=SLEEP
28: Turn itself off ► *Go to sleep, stop running DESK*
29: **else**
30: *status*=ACTIVE
31: Set itself to be *Active* ► *Turn on*
32: One-hop broadcast mACTIVATE message
33: **end if**
34: **end if**
35: **end while**

3.2.4.2. Description

Basically, with DESK, sensors have to wait for any incoming messages from neighbors and act accordingly. More specifically, the algorithm works as follows:

- All the sensors collect coordinates, current residual energy, and sensing range information of its one-hop live neighbors. It stores this information into a list L in the increasing order of $\alpha_{j,L}$ (for the ease of applying *k*-NC later).
- Each sensor sets its timer w_i with the assumption that all of its neighbors need it to join the network, i.e., $n_i=N_i$.
- When a sensor s_j joins the network, it broadcasts a mACTIVATE message to inform all of its 1-hop neighbors about its status change. Each of these neighbors then applies the *k*-NC algorithm to re-compute its coverage status. If it finds any neighbor *u* that is useless in covering its perimeter, i.e., the perimeter that *u* covers was covered by other active neighbors, it will send mASK2SLEEP message to that sensor.
- On receiving mASK2SLEEP message (a neighbor does not need it to be active), this sensor updates (decrease by 1) its counter n_i, contribution c_i and recalculate waiting time w_i. It then check if its counter (n_i) is decreased to 0 or not.
- If a sensor receives mASK2SLEEP message from all of its neighbors ($n_i=0$), meaning it is of no use to all of its neighbors, then it will send message mGOSLEEP to all of its neighbor telling them that it is about to go to sleep, and set a timer R_i for waking up in next round, and at last turn itself off (go to sleep).
- On receiving mGOSLEEP message, a sensor merely removes the neighbor sending that message out of its list L.

3.2.4.3. Example

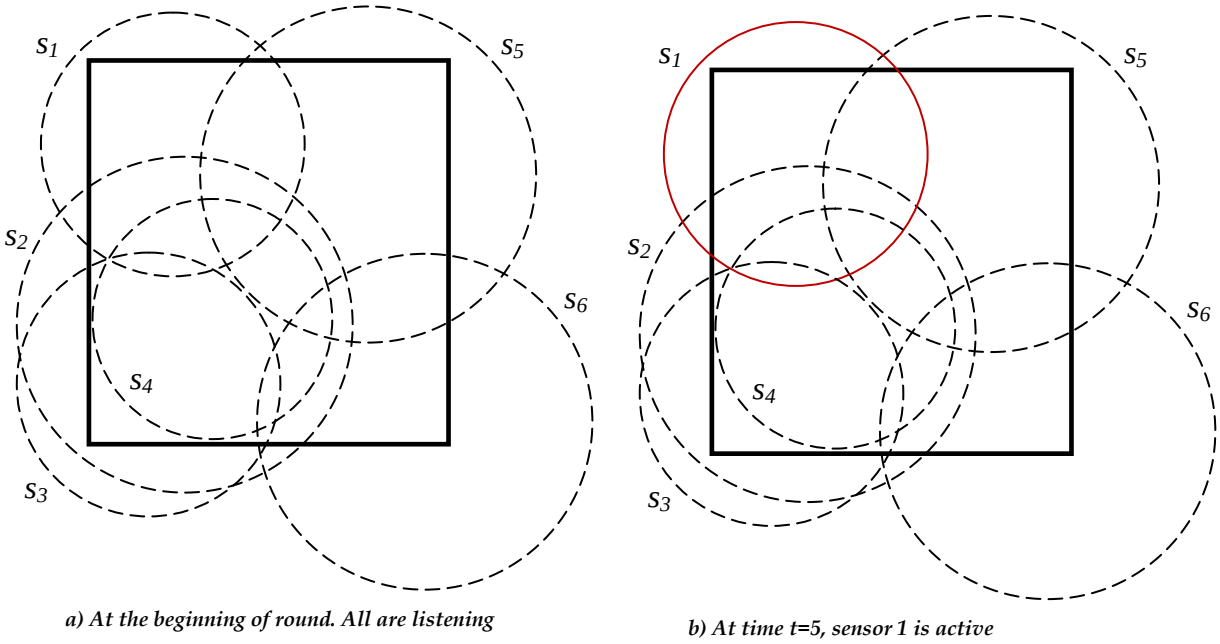


Figure 3.2. The step-by-step operations of DESK. Part 1

In Figure 3.2 and Figure 3.3, we illustrate the working of DESK with a network consisting of 6 sensors. In those figures, the thick border rectangle is the desired monitored area and the circles are sensing regions of sensors. *Solid* circles represent active sensors; *dotted* circles represent sleep sensors; *dashed* circles represent listening sensors. For the sake of simplicity, we choose $k=1$.

- At the beginning of each round, no sensors are active. All sensors are in LISTENING mode, i.e. all wait for the time to make decision while still doing sensing job. Notice that right at the beginning of the round, sensor s_3 is useless to s_2 (Figure 3.2.a). Consequently, s_2 sends mASK2SLEEP message to s_3. So, s_3 accordingly decreases its n_3 and updates waiting time w_3.
- At the time t = 5, s_1 becomes active (Figure 3.2.b).

- At the time t = 15, the waiting timer of s_2 expires. Consequently, s_2 becomes active and then notifies its neighbors about its changing status. Since s_1 had already become active, it is easy to verify that both s_3 and s_4 are useless to all of their neighbors at this time. Thus, s_3 and s_4 go to sleep (Figure 3.3.a).
- At the time t = 45, s_6 and after that, at the time t = 50, s_5 eventually becomes active (Figure 3.3.b). After this point, the area is *k*-covered.
- The discovered set cover for this round is $\{s_1, s_2, s_5, s_6\}$.

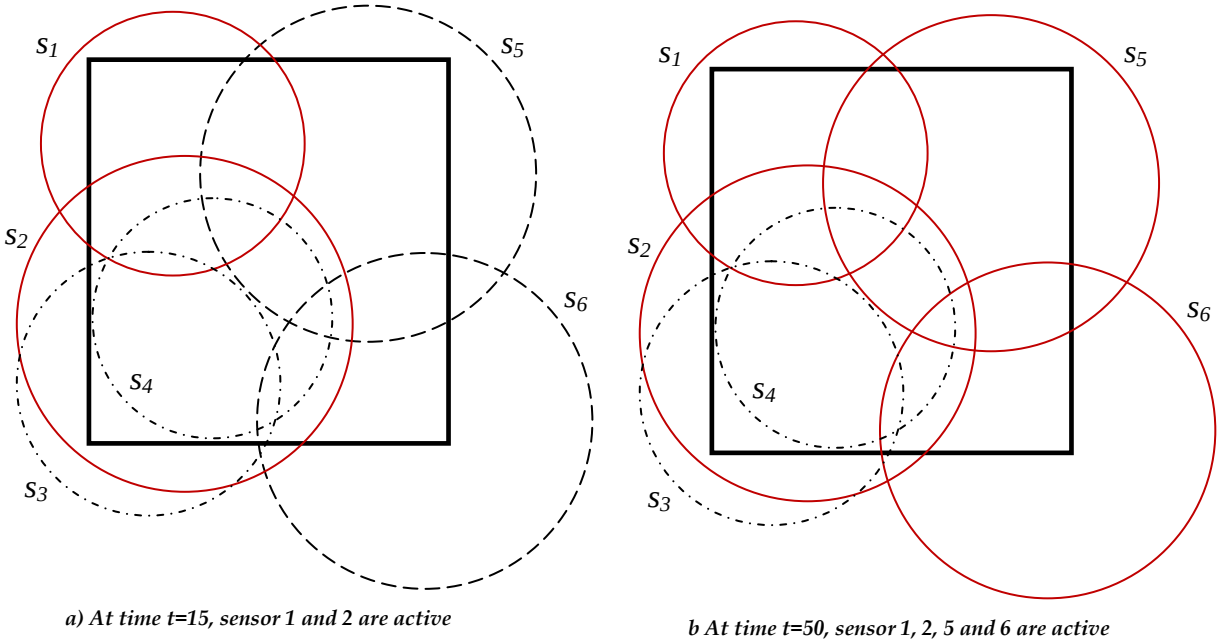


Figure 3.3. The step-by-step operations of DESK. Part 2

3.3. DESK analysis

3.3.1. Theoretical analysis

To theoretically evaluate DESK, we need to give the following definition first:

Definition 3.5. (sub-region [HUT05]) *A sub-region in area A is a set of points that are covered by the same set of sensors.*

Theorem 3.3. *When a sensor is useless to the coverage of all of its neighbors, its sensing region is already k-covered.*

Proof: Consider a sensor s_i. Without loss of generality, assume that sensor s_i has enough live neighbors to *k*-cover its perimeter. These neighbors partition the region inside s_i 's sensing region into some sub-regions. Each sub-region is bounded by the perimeter of one or more s_i 's neighbors. Since all these neighbors ask s_i to sleep, the perimeter segment, which is inside s_i 's sensing region, of each of these neighbors is already *k*-covered. As shown in [HUT05], each sub-region is at least *k*-covered. It follows that this theorem holds. ■

The correctness of DESK can be validated through the following theorem.

Theorem 3.4. *The algorithm ensures that the whole monitored area is k-covered.*

Proof: Without loss of generality, assume that sensor s_i has enough live neighbors to *k*-cover its perimeter. A sensor can go to sleep only when all of its neighbors ask it to do so. Hence, a sensor can allow a neighbor to go to sleep only when the perimeter segment covered by that neighbor is already *k*-covered. Thus, at the end of the decision phase, a sensor allows its neighbor(s) to turn off only when its whole perimeter is already *k*-covered. Furthermore, Theorem 3.3 has as well shown that sleep sensors are *k*-covered. Finally, all the sensors are *k*-covered. According to Theorem 3.2, the whole monitored area is guaranteed to be *k*-covered. ■

Theorem 3.5. *The time complexity of DESK is* $O(\min(\frac{W}{d},\Delta)\Delta)$ *and the communication complexity of DESK is* $O(n\Delta)$.

Proof: Let us investigate the time complexity for the worst case. The length of the decision phase is *W*, and the time slot is *d*. If at each time slot, a sensor receives mACTIVATE messages from one or more neighbor(s), it may receive a maximum of $\frac{W}{d}$ mACTIVATE messages. However, a sensor has no more than Δ neighbors; hence, a sensor may receive at most $\min(\frac{W}{d}, \Delta)$ mACTIVATE messages each round. Besides, it needs $O(\Delta)$ time to run the *k*-NC algorithm to check its perimeter coverage [HUT05]. Moreover, all the sensors simultaneously run DESK. Thus the time complexity is $O(\min(\frac{W}{d}, \Delta)\Delta)$.

Since each sensor has at most Δ neighbors and throughout the decision phase, a sensor sends at most one mASK2SLEEP message per neighbor and only one message to broadcast its status (active/sleep), so each sensor sends at most $O(\Delta)$ messages in the decision phase. This means that the message complexity is $O(n\Delta)$. ■

3.3.2. Simulation

In this section, we evaluate the efficiency of DESK through conducting some simulations measuring the number of sensors per subset, the number of messages sent by each sensor per round and network lifetime with different number of sensors and different values of *k*. We also compare network lifetimes of the networks with different initial energy of sensors.

3.3.2.1. Energy model

We now construct a simple energy model as the guideline to measure energy consumption. The distributed algorithm requires the consideration of various kinds of energy consumption including message transmission/reception, data sensing and computational energy.

To the best of our knowledge, no work has been done to mathematically construct an energy model that takes all the energy consumptions into account. A detailed survey on numerous kinds of energy consumption in wireless sensor networks is given in [RAG02]. Based on their work, we develop a simple energy model for measuring DESK's performance.

Normally, a sensor node has three major units that consume energy: the micro-controller unit (MCU) which is capable of computation, communication subsystem which is responsible for transmitting/receiving messages and the sensing unit that collects data [RAG02]. In our model, each subsystem can be turned on or off depending on the current status of the sensor which is summarized in Table 3.1.

Table 3.1. Energy consumption

Sensor mode	MCU	Radio	Sensor	Power (mW)
Listening	On	On	On	20.05+ $f(r_i)$
Active	On	Off	On	9.72+ $f(r_i)$
Sleep	Off	Off	Off	0.02
Energy needed to send a 2-bit-content message				0.515

In Table 3.1, the function $f(r_i)$ is the energy consumption related to the sensing range r_i of sensor s_i. We consider two kinds of function f:

$$\text{Linear function: } f(r_i) = \frac{1}{\kappa} \times r_s \tag{3.9}$$

$$\text{Quadratic function: } f(r_i) = \frac{1}{\kappa} \times r_s^2 \tag{3.10}$$

where κ is an energy coefficient.

For the sake of simplicity, we omit the energy needed to receive a message, to turn on the radio, to start up the sensor node, etc. We also do not consider the need of collecting sensing

data. Thus, when a sensor becomes active (i.e., it already decides it status), it can turn its radio off to save battery.

Since DESK uses only three different types of messages, two bits are sufficient for the payload of exchanged messages. The value of energy spent to send a message shown in Table 3.1 is obtained by using the equation to calculate the energy cost for transmitting messages shown in [RAG02]. The power consumptions when the sensors are in *Listening*, *Active* and *Sleep* mode displayed in Table 3.1 are acquired from the statistical data of MEDUSA-II node - a sensor node developed at the University of California, Los Angeles [RAG02].

In our model, the remaining lifetime of a sensor is the time that a sensor can live in the active mode. That is, if a sensor works with sensing range of r_i at a point of time, when the residual energy is e_i, then the lifetime can be calculated as:

$$l(e_i, r_i) = \frac{e_i}{\text{Energy consumption in active mode}} \tag{3.11}$$

Thus, in our simulation, the equation for sensor's lifetime function is:

$$l(e_i, r_i) = \frac{e_i}{9.72 + f(r_i)} \tag{3.12}$$

3.3.2.2. Network configuration

All the parameters used for the simulation are provided in Table 3.2. The sensors are randomly deployed in a fixed region of size $800^m \times 800^m$. The energy is randomly generated for each sensor within a range whose lower bound is *200 Joules*. The sensing range of each sensor is as well randomly chosen between 400^m to 500^m. As shown in Table 3.2, the length of a round is much larger than that of the decision phase. We define the network life time as the duration until at least one portion of surveillance area cannot be covered by at least *k* active sensors.

Table 3.2. SESK simulation parameters

Area size	$800^m \times 800^m$	Decision phase	2 *second*
Sensing range	$400^m \rightarrow 500^m$	Slot time	0.5 *ms*
Minimum power	$200J$	Round time	20 *minutes*
α, β	1	κ	8,000

3.3.2.3. Simulation results

In this sub-section, we show the results of various simulation we conduct to evaluate the efficiency of DESK.

In Figure 3.4, Figure 3.5 and Figure 3.6 the upper bound of a sensor's initial energy is $300J$; the energy consumption in terms of sensing range is quadric which is shown in Eq. 3.9; and the number of the sensors range from 50 to 200. Figure 3.7 and Figure 3.8 evaluate the network life time of a 100-node network when the ratio between the upper bound and lower bound of sensors initial energy ranges from 1 to 2.5; and the energy consumption function in terms of sensing range *f* is quadratic and linear as illustrated in Eq. 3.9 and 3.10, respectively.

Figure 3.4 shows number of active sensors each round. For algorithm working in rounds as DESK, the set of active sensors in each round is a set cover, thus we can say that Figure 3.4 shows the average size of set cover. As DESK considers the balance on consuming the energy among all the sensors, it is also valid that the number of sensors per subset increases with the number of sensors. Due to its effort to use as many sensors as possible, the number of unallocated sensors, i.e., sensors which have never become active, are almost equal to 0, which indicates that DESK efficiently makes almost all the sensors in a network to participate in the *k*-

coverage sensing task. It is easy to see that the bigger the value of k, the larger the size of a set cover since it needs more sensor to cover each point inside monitored area.

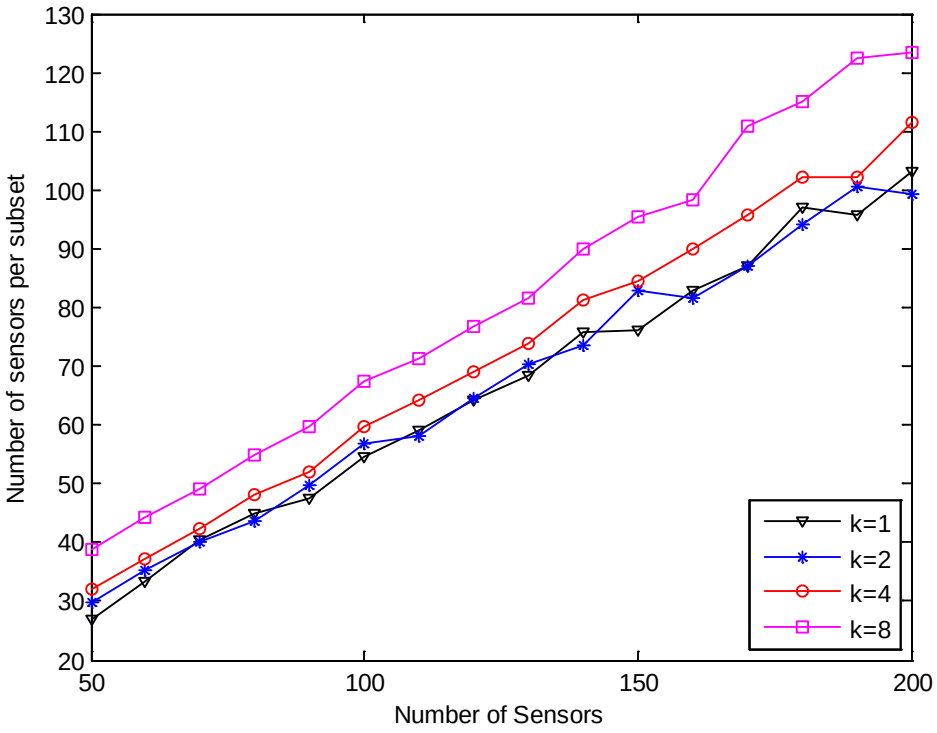


Figure 3.4. Number of active sensors per subset

Figure 3.5 presents the number of messages that a sensor sends during each round, more specifically, during each decision phase. It can be seen that more messages are sent when the number of the deployed sensors increases. It is not surprising since a sensor may have more neighbors. Theoretically, with the same topology, the higher the value of *k* the lower the number of messages needed to be exchanged. However, it can be observed that the number of messages that each sensor sends per round are almost the same for k = 1; 2; 4; and 8. This phenomenon is originated from the random deployment method of our simulation. Furthermore, when investigating the simulation data, we find out that most part of perimeter of each sensor is k^*-

covered, where $k^* > k$. Thus the number of sleep sensors each round and the number of exchanged messages are not so much different for the different values of k. DESK will perform better if a controlled deployment method is employed, which can balance the coverage levels of the boundary parts of the monitored area and its central part.

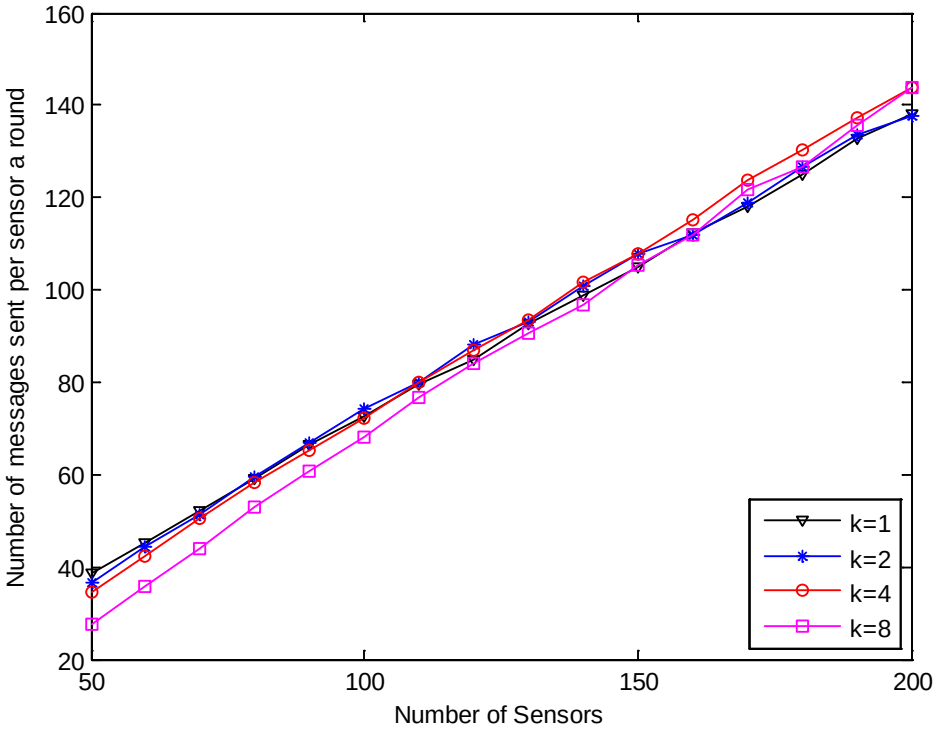


Figure 3.5. Number of messages sent per sensor each round

In Figure 3.6, network lifetime is illustrated. As shown in Figure 3.6, network lifetime decreases when the value of k increases. This is easy to understand since the bigger the value of k, the larger the number of active sensors a round, hence the smaller network lifetime. It also can be seen that with each value of k, the lifetime slightly fluctuates. This fact is due to the random nature of our method to conduct the simulation. We do not put any control on network

deployment and we as well randomly assign the sensor properties (e.g., initial energy, position, sensing range) from a wide range.

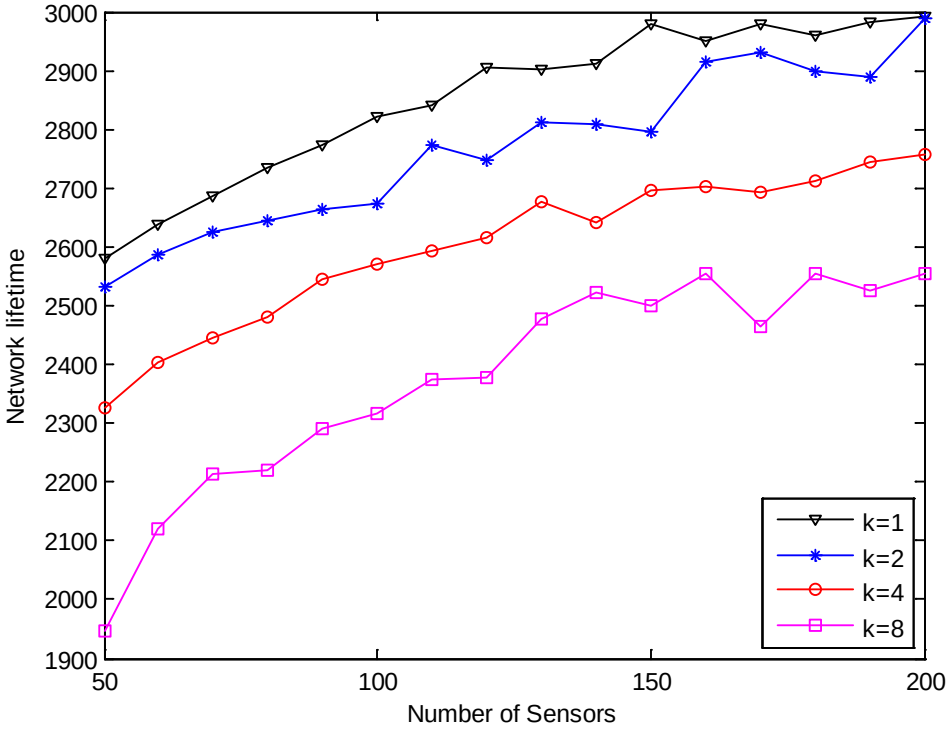


Figure 3.6. Network lifetime

In Figure 3.7 (the energy function *f* is linear) and Figure 3.8 (the energy function *f* is quadratic), we conduct measurements of network lifetime with different power ratio, i.e., the ratio between the upper bound and lower bound of the range in that initial power for each sensor is randomly assigned. As illustrated in Figure 3.7 and Figure 3.8, the lifetime of network significantly increases as the power ratio increases. This phenomenon is relatively logical since some sensors are given more energy when the power range is widened. Compared with linear energy model, the sensor has to consume more energy in the order of its sensing range when the sensor's energy consumption is quadratic energy model. That is why the lifetimes of network are considerably different between Figure 3.7 and Figure 3.8.

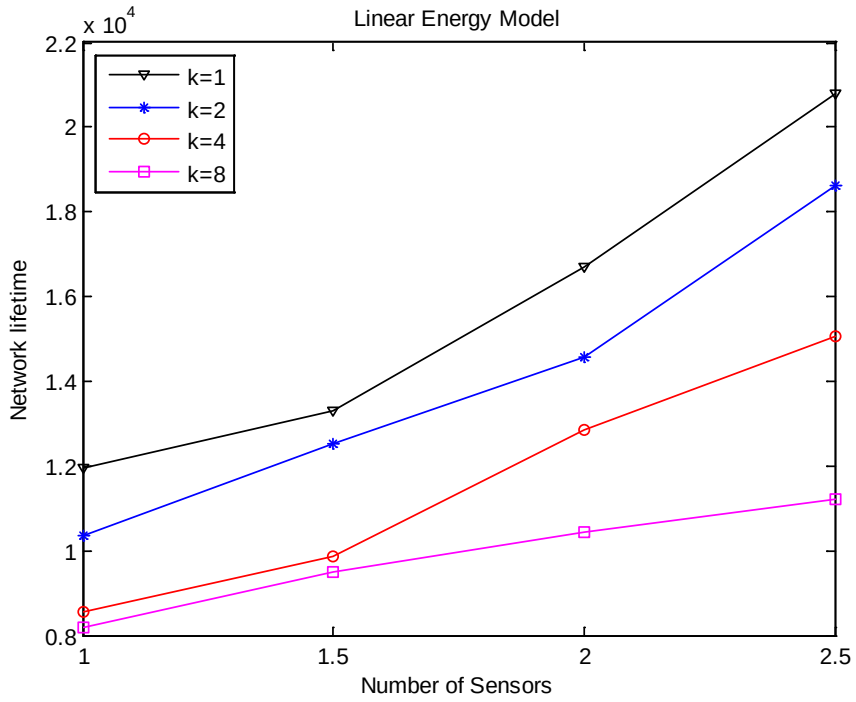


Figure 3.7. Linear energy model: Network lifetime with different power range

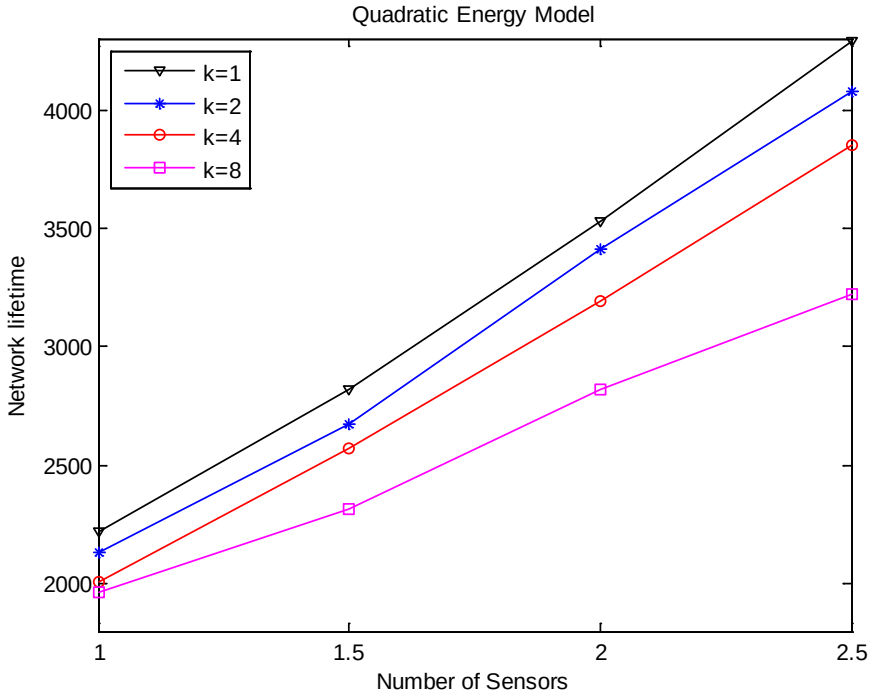


Figure 3.8. Quadratic energy model: Network lifetime with different power range

3.4. Improvement: location free extension

In this chapter we have proposed an efficient and fully distributed, parallel algorithm for *k*-coverage problem. However, our algorithm highly depends on the location of all the sensors in the network. This assumption is, in fact, implicitly employed by most of the work concerning coverage problem even though that information is not always cheap and easy to obtain. Recently, [BEJ08] has proposed a rule which is an extension of the *k*-NC rule [HUT05] we used in Pseudocode 1 and is independent on the sensors' location. That rule in [BEJ08] can directly be utilized in Pseudocode 1 to make it location-free.

CHAPTER 4.

ADJUSTABLE SENSING RANGE IN WSN

In this chapter, we consider the area coverage problem for WSN where sensors can arbitrarily change their sensing ranges under some upper bound. With variable sensing range, the difficulties to cover a continuous space (where number of points is infinity) in the area coverage problem becomes somewhat harder than covering limited number of discrete points in the target coverage problem. Very few papers have paid effort for the former problem. We first improve the work in [WAN07] so that the boundary effect is ruled out and the monitored area can be completely covered at all cases. Next, we extend that improved algorithm by introducing two distributed scheduling algorithms which are trade-off in terms of network lifetime and algorithms efficiency. The first scheduling algorithm adaptively has related sensors increase sensing ranges when there exist any uncovered regions. The second scheduling one carefully allows some sensors to go to sleep (for future use) and has the others cooperatively cover the whole area. The major objective of each of our 3 proposed algorithms in this chapter is to balance energy consumption and to maximize network lifetime. Our proposed algorithm efficiency is shown by algorithms complexity analysis and extensive simulation. In compared with the work in [WAN07], our proposed algorithms are not only better in providing higher coverage quality, but also they greatly lengthen network lifetime and greatly reduce the unnecessary coverage redundancy.

4.1. Existing work and our motivation

Since Wu and Yang in [WUY04] first introduce the coverage problem for the network where sensors may vary their sensing ranges, there are some works dedicate effort for this

problem. Most of them [DHA06], [CAW05], [CAR06], [YAN06] try to solve the target coverage problem since it is easy to verify the coverage status of set of discreet targets. Only a few work [WAN07] study the area coverage problem. Some works [CAW05] , [CAR06], [YAN06], [WUY04] assume that sensors are only capable of adjusting their sensing ranges to the discreet set of some pre-defined options. Some other works [WAN07], [DHA06] deal with network where sensors can smoothly adjust their sensing ranges to any range under some upper threshold. The work in [WUY04] deals with sensors having two or three levels of sensing ranges. The work in [CAW05] extends to allow sensors to vary sensing ranges among P levels. With this relaxed assumption, the authors propose three algorithms - one ILP-based centralized, one greedy centralized and one greedy distributed - with the objective of maximizing the number of set covers, each is able to cover the whole set of targets. The work in [YAN06] makes a step further by providing connectivity for each of those set covers. The work in [DHA06] proposes a centralized algorithm to cover set of targets which schedules sensors based on the Garg-Könemann method [GAR98]. The algorithm has approximation of $(1 + \varepsilon)(1 + \log m)$ for any $\varepsilon > 0$ where m is the number of targets.

Except the work in [WAN07] and [WUY04], most of the works deal with the target coverage problem and does not take energy into consideration. In [WAN07], Wang and Medidi propose two distributed algorithms based on Delaunay-triangulation structure. However, even the authors have proved that the proposed algorithms could provide complete coverage, the proof has flaw because it does not take into account the so-called boundary effect. We will go into detail about this flaw and further correct it by proposing an algorithm named IDT (Improved Delaunay-Triangulation) based on the “energy balancing heuristic” in [WAN07] in Section 4.3. However, in both IDT and algorithm in [WAN07], sensors are not scheduled to be turned-off, to

be re-used or to re-adjust their sensing ranges when the network coverage quality is decreased under some threshold (e.g., less than 100%). Thus, in this work, we extend IDT by two scheduling algorithms. One will adaptively increase related sensors' sensing ranges in order to maintain the coverage quality. The other scheduling algorithm properly turns sensors on/off so that the coverage quality is preserved.

4.2. Preliminary

For the sake of explaining our algorithms, we dedicate this section introduce some preliminary knowledge, definitions and to state our assumptions.

First, it is necessary to mention (even it is already discussed in Section 2.3.3) the *k*-NC rule proposed in [HUT05] which will be employed in IDT and two scheduling algorithms for a sensor to check if there exists any uncovered sub-region adjacent to it. Considering the perimeter circle of sensing region of a sensor, it can be partitioned into some portions, and each portion is covered by the same set of neighbors. The number of neighbors which cover each portion is called the *perimeter coverage level* of that portion. The perimeter coverage level of a node is defined in [HUT05] as the minimum perimeter coverage level of all of its perimeter portions. This parameter usually is denoted by *k* and the sensor is said to be *k*-perimeter-covered. Please refer to Definition 5 in [HUT05] for the formal definition of *k*-perimeter-coverage. In this work, we only consider $k = 1$.

Based on that definition, the authors in [HUT05] propose a rule which allows a sensor to verify if the network is completely covered or not based on only 1-hop neighbor information. The rule claims that if all the sensors in the network are 1-perimeter-covered, then the network is 1-covered.

Next, we introduce some definitions which will be used later:

Definition 4.1. (Uncovered portion) *A portion of a sensor's sensing perimeter which is not covered by any of its active neighbors is called an uncovered portion.*

Definition 4.2. (Uncovered node/sensor) *With current sensing radius assignment, a sensor node is called an uncovered node or uncovered sensor if it is not completely 1-perimeter-covered by its active neighbors.*

Definition 4.3. (Useful neighbors) *A neighbor is useful to a sensor if, when being assigned with the maximum sensing range, it can cover the uncovered portion of that sensor.*

Definition 4.4. (Breach) *A breach is a sub-region of the monitored area which is not covered by any active sensors.*

Definition 4.5. (Sensing neighbors) *Two sensors are said to be sensing neighbors if, when being assigned the maximum sensing ranges, their sensing regions overlap.*

To close this section, we then state some assumptions:

4.2.1. Assumption

We assume that the sensing region of a sensor is a disk centered at the sensor's location and has the radius of sensor's sensing range. We further assume that sensor could smoothly vary its sensing range up to a maximum cutoff range, denoted by $MaxR_s$ and the communication range of a sensor is at least $2MaxR_s$ (so the connectivity is guaranteed when coverage is provided [WAN03]). We also assume that a sensor and its neighbors have different IDs, i.e., within 1-hop vicinity, a node has a unique ID. Based on its ID, the priority of a sensor s is assigned as a pair of its current energy and its ID. i.e., $s.priority = < E(s), s.ID >$. The last assumption we made is that a sensor has information about its 1-hop neighbors.

For the rest of this chapter, we use notation Δ to denote the degree of the network (the maximum node degree).

4.3. Boundary effect issue and IDT algorithm

In this section, we will first explain some key aspects of energy balancing heuristic of [WAN07] in Section 4.3.1. We name this heuristic as "Original Delaunay-Triangulation" (ODT) algorithm. We then show the so-called "boundary effect" and how it makes ODT ineffective in Section 4.3.2. In Section 4.3.3, we explain our heuristic named "Improved Delaunay-Triangulation" (IDT) algorithm which is able to eradicate the boundary effect.

4.3.1. Original algorithm proposed in [WAN07]

Firstly, the authors in [WAN07] introduce a completely distributed algorithm which requires only one hop neighbor information to construct an approximate Delaunay-triangulation. It would be necessary to emphasize that the simple algorithm proposed in [WAN07] only constructs an approximate Delaynay-triangulation, not exact Delaunay-triangulation. Thus all the Voronoi cells or Delaunay triangles built by that algorithm are approximate. The approximate algorithm works as follows. Each sensor constructs its Voronoi cells by drawing bisector of the line connecting it with a neighbor. Depending on the relative location of itself, the bisector line and the Voronoi cells constructed so far, the Voronoi cells will be modified accordingly. The algorithm to construct Voronoi diagram is straightforward and intuitive. After constructing Voronoi diagram, the Delaunay triangulation can easily be derived by connecting pair of sensors whose Voronoi cells share a common border.

In the next step, each sensor steps through all Delaynay triangles which have it as a vertex. For each triangle, the sensor calculates a weighted centroid based on the location of three

vertices (which include that sensor) and their residual energy. Having the coordinate of this weighted centroid, the sensor calculates the Euclidean distance from itself to the weighed centroid. The sensor radius will be the larger one of its currently assigned sensing radius and that distance.

This way, each the Delaunay triangle is guaranteed to be covered by its three sensors locating at its three vertices (corners). Since the whole area is partitioned into a set of Delaunay triangles, the whole area is covered too. However, there is still an issue which we will discuss in detail next.

4.3.2. The boundary effect issue

For the coverage problem, researchers sometimes have to avoid a so-called “boundary effect” (e.g., [ZHE05]) which is caused by the fact that the coverage quality in the boundary region is different from that of the center region. Especially, for algorithms which are based on the geographical locations of sensors, the boundary effect becomes even more severe since the sensors closed to the boundary area are not covered by neighbors in all directions.

As discussed in Section 4.3.1, we have explained the intuition behind ODT. Naturally, several questions arise:

a) what happens for sensors locating at boundary region whose Delaunay triangles are not completely around them, i.e., there exists an angle around the sensor that does not belong to any Delaunay triangle?

b) Is it possible that those sensors cause any coverage breach?

c) If yes, then in what situation, coverage breach may appear?

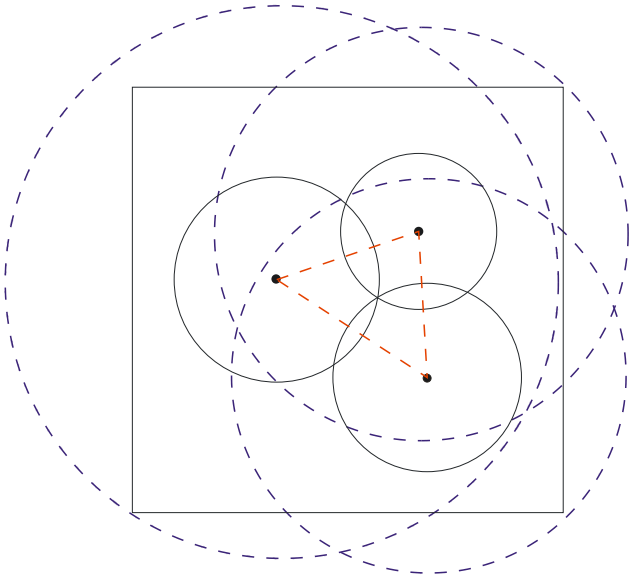


Figure 4.1. Boundary effect example

To answer the above question, let us take an example shown in Figure 4.1. The dashed-lined circles represent sensing regions of three sensors with their maximum sensing ranges. It can easily be seen that these three sensors completely cover the whole monitored area (monitored area is represented by the solid-bordered square). The dashed-bordered triangle is a Delaunay triangle created by these three sensors. The solid-lined circles are their sensing regions being assigned by ODT. Apparently, these sensors cannot cover the whole area with the new assigned sensing ranges. The reason is that the Delaunay triangle forces those sensors to reduce their ranges, therefore, they cannot reach and cover the boundary area. As a result, the region around the border is un-covered (or breached). Thus, with sensing range assignment scheme made by ODT algorithm, it is still possible that there exists an uncovered region. Thus the answer for the question (b) is “yes” and a coverage breach appears (question c) when the Delaunay triangulations adjacent to a sensor at boundary region are too small which cause that sensor its sensing range to adjust to a too small range. Having those answers, next, we explain our improved IDT algorithm which eliminates any coverage breaches.

4.3.3. IDT Algorithm

In this section, we explain our IDT algorithm which could overcome the boundary effect so that it could provide 100% coverage for the network for all cases. The pseudo-code of IDT is shown in Pseudocode 2:

Pseudocode 2: **IDT algorithm running at sensor *s***

1: Use the algorithm in [LIE02] to build the local Delaunay triangulation.
2: **foreach** Triangle *T* with 3 sensors *s; t; u* as its vertices **do**
3: ► *Determine weighted centroid m of T*
4: $m_x = \frac{E(s) \times s_x + E(t) \times t_x + E(u) \times u_x}{E(s) + E(t) + E(u)}$
5: $m_y = \frac{E(s) \times s_y + E(t) \times t_y + E(u) \times u_y}{E(s) + E(t) + E(u)}$
6: *s.R = max*(*s.R;*|*sm*|) ► *Assign sensing range*
7: **end for**
8: ► *Uncovered sensor increases sensing range*
9: **if** *s* has received messages from all its higher priority neighbors **then**
10: **if** *s* is NOT completely perimeter- covered **then**
11: Adjust its sensing range to $MaxR_s$
12: Ask all useful neighbors to adjust their sensing ranges to $MaxR_s$
13: **else**
14: *s* tells its neighbors that the checking is done ► *Neighbors do nothing*
15: **end if**
16: **end if**

Different from ODT where the approximate Delaunay triangulation is deduced from approximate Voronoi diagram, IDT directly builds the exact Delaunay triangulation by employing the approach in [LIE02]. In [LIE02], the Delaunay-triangulation is locally built based on "*locally equiangular*" property [SIB77]. Also, [SIB77] shows that Delaunay-triangulation is the only triangulation which possesses that property. The *locally equiangular* property guarantees that for any two rectangles sharing a common edge, if their union is a convex quadrilateral, then the replacement of their common edge by the alternative diagonal (of convex

quadrilateral) does not increase the minimum of the six interior angles (of the two triangles). In[LIE02], a sensor builds its local triangulations by checking its neighbors one by one, if the line (a diagonal of a convex quadrilateral) of the sensor to a neighbor violates the *locally equiangular*, that line will not belong to the local Delaunay triangulation. Thus, the time complexity of local Delaunay-triangulation construction is $O(\Delta)$ where Δ is the degree of the network.

After Delaunay triangulation is built, each sensor may belong to several Delaunay triangles. Consider a sensor *s*. It scans through all triangles that are incident to it (line 2 to line 7 of Pseudocode 2). For each triangle, it will calculate a virtual point named "weighted centroid". The coordinates (*x* and *y*) of this point are calculated by line 4 and 5 in the Pseudocode 2 where $E(s)$, s_x, s_y are sensor *s*'s current residual energy, *x* coordinate, and *y* coordinate, respectively. *t* and *u* are the other two vertices of the triangle. Having the weighted centroid's coordinates, *s* may change its sensing range to the Euclidean distance from itself to that weighted centroid. Since there are several triangles having *s* as a vertex, each will have it own weighted centroid. The final sensing range of *s* will be the maximum Euclidean distance to the furthest weighted centroid. This step to assign sensor sensing range corresponds with ODT algorithm (explained in Section 4.3.1). ODT algorithm, however, has ignored the boundary effect. The ODT algorithm may leave some region uncovered as explained in Section 4.3.2. We take into account the boundary effect by letting a sensor which is not completely perimeter-covered and all of its neighbors who could help to cover the uncovered portion at the boundary region to increase sensing ranges to the maximum value ($MaxR_s$). This step works as follows. After being assigned sensing ranges in the previous step, all the sensors check if they are 1-perimeter-covered or not by using the *k-NC* rule proposed in [HUT05] where $k = 1$ (explained in Section 4.2). For the ones

who already are, they just wait for the requests from their neighbors if there are any. To avoid unnecessary increase of sensing ranges of several sensors at the same time, we adopt a priority scheme. The priority could be decided by the combination of many parameters. In this work, as explained in Section 4.2.1, we define priority of a sensor *s* as < *Es*, *ID* > where E_s is sensor *s*'s residual energy and ID is sensor *s*'s ID. For a sensor that has not been 1-perimeter-covered, it firstly waits for decision (satisfied with sensing range assigned by ODT or has already increased sensing range to $MaxR_s$) from its neighbors who have higher priorities. This way, energy consumption of sensors is balanced by allowing sensors with more energy to have more chances to work. After all those higher-priority neighbors have made their decisions, the node uses *k-NC* to verify its perimeter coverage one more time. If it is still not 1-perimeter-covered, it increases its sensing range to $MaxR_s$. It then re-evaluates its perimeter coverage status (using *k-NC* rule) to see if it is 1-perimeter-covered. If still not, it asks all the useful neighbors to also increase their sensing ranges to $MaxR_s$.

Next, we analyze the time and message complexities of IDT and prove the correctness of IDT.

Theorem 4.1. *The total message in the whole network and the complexity of IDT are O(n) and* $O(\Delta^2)$*, respectively*.

Proof: After deciding sensing range based on local (Delaunay) triangles, each sensor has to send exactly one message to notify its neighbors if it increase its sensing range to $maxR_s$ or not. Thus, the message complexity of IDT is *O(n)*.

Clearly, each time a sensor has to adjust its sensing range, that sensor takes $O(\Delta)$ time to construct the local Delaunay triangulation. It takes $O(T)$ time to go through all the triangles where *T* is maximum number of (Delaunay) triangles that a sensor node may belong to. Since

$T<\Delta$, the total time complexity for one time running the algorithm is $O(\Delta)+O(T)=O(\Delta)$. Since a sensor has at most Δ neighbors, so it could wait for at most Δ higher-priority neighbors to verify their perimeter coverage status (by using the *k*-NC rule [HUT05]).

Since the time complexity of *k*-NC rule is $O(\Delta)$, the total time complexity running algorithm in a sensor lifetime is $O(\Delta^2)$. ■

Next, we will prove that our improvement indeed guarantees complete coverage for the network.

Theorem 4.2. *If the monitored area can be completely covered when all the sensors are active and are assigned the maximum sensing ranges, then IDT ensures that the whole monitored area is completely covered.*

Proof: Prove by contradiction

Assume that after all the sensors are assigned sensing range by IDT algorithm, there still exists an uncovered region *G*. Theorem 4.1 in [WAN07] has proven that the part of the area which is covered by Delaunay triangles will be covered by the sensing. ranges assignment scheme of ODT, thus, *G* is a region that is not covered by any Delaunay triangle and *G* shares at least one border with the monitored area.

Clearly, *G* shares the border with some sensors' sensing region, denoted by s_1, s_2, .., s_l where each s_i is a neighbor of some others. Since s_i, $i = 1..l$ is not 1-perimeter-covered, line 11, 12 of Pseudocode 2 will increase *si*'s sensing range to $MaxR_s$.

Consider a point *p* inside G, meaning $\|s_i p\| > MaxRs$ for all $1 \le i \le l$. According to our assumption that the whole area can be covered if all the sensors are assigned with the maximum sensing range, there will exist a sensor *t* which can cover *p*, meaning $\|tp\| \le MaxR_s$. *t* is assigned

a sensing radius smaller than $MaxR_s$, and t is a sensing neighbor of at least one sensor of set s_1, s_2,.., s_l (otherwise, even with the maximum sensing range, t cannot cover p). Clearly, $t \notin G$, otherwise, part of G will be covered by a Delaunay triangle. That means if being assigned sensing range of $MaxR_s$, t will cover part of uncovered perimeter portion of at least one of s_1, s_2,.., s_l. In other words, t is a useful neighbor of at least one of s_1, s_2,.., s_l. According to line 12 of Pseudocode 2, t will increase its sensing range to *MaxRs*, thus p is covered by t which leads to a contradiction.

Thus, the correctness of the IDT algorithm is proved. ■

To extend network lifetime, in the next two sections, we propose two scheduling algorithms using IDT as a basic building block.

4.4. Adaptive Scheduling of IDT (ASIDT)

The pseudo-code of ASIDT algorithm is shown in Pseudocode 3.

With ASIDT, the network is able to adaptively fill up the breaches (uncovered portions of monitored area) which are created because of the death of active sensors (as their energy is fully depleted). When a sensor s is about to die, it first checks if any of its neighbors t might be not completely perimeter-covered, i.e., there is any portion of t's perimeter that is uncovered without sensor s. If there is such a neighbor t, then s, by sending REMOVE-ME-AND-UPDATE message, will tell its neighbors that it is about to die (turn off permanently), so all those neighbors will have to remove s from their neighbor list and re-run IDT. Otherwise, s, by sending REMOVE-ME message, just merely tells all of its neighbors to remove s from their neighbor list. In this way, only *related* sensors, i.e., sensors "near" the breaches, would be involved in adjusting sensing ranges process. The others sensors which are "far" from the

breaches would not be affected. The block of code from line 18 to line 20 of Pseudocode 3 may not be necessary since IDT already takes care of it. We, however, keep it there to make the algorithm clearer.

Pseudocode 3: **ASIDT algorithm running at sensor *s***

```
1: Running IDT algorithm
2: Neighbor list = all s's neighbors
3: if sensor s is about to run from energy then
4:      s notifies all the neighbors
5:      if there is a neighbor that might be uncovered sensor after s turns off then
6:              Send REMOVE-ME-AND-UPDATE message
                ► Neighbors have to remove s from neighbor list and run IDT
7:      else
8:              Send REMOVE-ME message
                ► Neighbors just have to remove s from their neighbor list
9:      end if
10: end if
11: ► Receiving message from a neighbor
12: if receive REMOVE-ME from neighbor v then
13:     Remove v from neighbor list
14: end if
15: if receive REMOVE-ME-AND-UPDATE from neighbor v then
16:     Remove v from neighbor list
17:     Run IDT again
18:     if New sensing range decided by IDT is bigger than current sensing range then
19:             Set to new sensing range
20:     end if
21: end if
```

Theorem 4.3. *In the worst case, the time complexity of the ASIDT algorithm is $O(\Delta^3)$ and the message complexity is O(n) for lifetime of a sensor.*

Proof: Since a sensor node sends exactly one message during its *lifetime* (to notify its neighbors that it is about to die), thus the total message complexity of ASIDT is *O(n).*

Clearly, a sensor has at most Δ neighbors, thus in the worst case, it can receive at most Δ REMOVE-ME-AND-UPDATE messages in its *lifetime*. Thus during its lifetime, it runs IDT at most Δ times. Since the time complexity of IDT is $O(\Delta^2)$ (Theorem 4.1), thus a sensor has to spend at most $O(\Delta^3)$ time to run the ASIDT algorithm. ■

Theorem 4.4. (Proof of correctness) *If the monitored area can be completely covered when all the sensors are active and are assigned maximum sensing ranges, then ASIDT ensures that the whole monitored area is completely covered.*

Proof: This can directly be derived from the correctness of IDT (Theorem 4.2) ■

ASIDT is really robust as it is able to adaptively cover up any breaches as soon as there appears one(s). The algorithm also has very low time and messages (communication) complexity overhead. However, its drawback is that all the sensors have to be "active" at all time, which may cause some redundancy and waste some energy. Thus, we propose another scheduling algorithm which could reduce that redundancy.

4.5. Scheduling by Pruning IDT (SPIDT) algorithm

This algorithm turns off some redundant sensors to save network's energy. Since not all the sensors are active at all time, we need some mechanism to wake up sleeping sensors if there appears a breach. To accomplish this, we have all the sensors work in rounds. More specifically, the network time-line is divided into a number of equal-length and fixed-size rounds. The length of a round is pre-defined. Each round is further partitioned into two phases: decision phase (where sensors decide status) and sensing phase (where active sensors collect data). The length of a decision phase must be much smaller than that of a round. All the sensors have to turn on at the beginning of each round, running SPIDT to adjust their sensing ranges and locally work

together to decide who to goto sleep and who to turn on for the rest of the round. In literature, many heuristics are working in rounds such as the one in [VUC06]. Most of them require a fixed-length decision phase. Different from those, the length of decision phase of SPIDT is not necessarily fixed and pre-defined. As soon as a sensor decides its status, the decision phase of that sensor ends. The pseudo-code of SPIDT is given in Pseudocode 4:

Pseudocode 4: **SPIDT algorithm running at sensor *s***

```
1: s runs IDT to decide its sensing range
2: Neighbor list = all s's neighbors
3: ► Node s makes decision (sleep/active)
4: repeat
5:      Waiting for a message from a neighbor
6:      if it is TURN -OFF message then
7:           Remove that neighbor our of Neighbor list
8:      end if
9: until receiving messages from all neighbors with lower priorities
10: ► Evaluate if it is redundant to all of its active neighbors.
11: Redundancy = True
12: foreach neighbor v in s's Neighbor list do
13:      if portion of v's border which is inside s is NOT covered by other neighbors of v and s
         then
14:           Redundancy = False
15:           Exit Foreach
16:      end if
17: end for
18: if Redundancy == True then
19:      Send TURN-OFF message to neighbors
20:      Turn itself OFF.
21: else
22:      Send TURN-ON message to neighbors
23: end if
```

At the beginning of a round, a sensor *s* first runs IDT algorithm to decide its sensing range. After this step, there might be redundant sensors that could be turned off without violating coverage requirement (i.e., 100% coverage). In order to avoid sensors simultaneously turning off

which may cause breaches, the sensors evaluate their redundancy in the non-decreasing order of their priorities. The ones with lower priorities (and thus having less residual energy) can make decision first. In this way, we can balance the energy consumption in the network by letting the sensors with less energy have a bigger chance to sleep. A sensor evaluates its redundancy by checking that if it turns off, none of its neighbors in *Neighbor list* would be uncovered. That is, without that sensor, the borders of all those neighbors could still be covered by other *active* sensors and sensors which have not yet evaluated their redundancy (i.e., neighbors in current neighbor list). If that is the case, meaning the sensor is redundant to neighbors in its current *Neighbor list*, it notifies its neighbors by TURN-OFF message and can then go ahead to turn off. Otherwise, it has to turn on (its neighbors are notified by TURN-ON message in this case). Note that, a sensor could evaluate its redundancy only when all of its neighbors who have lower priority already evaluated their redundancy and already decided their status. Also, when evaluating redundancy, the sensors do not take into consideration the neighbors who already decided to turn off (because its neighbor list no longer contains such neighbors). The pierce of codes from line 12 to line 17 of Pseudocode 4 is to check the redundancy of sensor *s* by scanning through every the sensor *v* in its neighbor list. *s* only considers the portion of *v*'s border that is inside *s*. If this portion can be covered by other neighbors of *v* (since this portion is inside *s*, those neighbors are also neighbors of *s* which means *s* does not need 2-hop neighbors information), then *s* knows that it is redundant to *v*. Otherwise, *v* needs *s* to be perimeter-covered, thus *s* cannot turn off. The time complexity for this process is $O(\Delta^2)$. The complexities of DESK is given in following theorem:

Theorem 4.5. *In the worst case, the time complexity of the SPIDT algorithm is* $O(\Delta^3)$ *and the message complexity of SPIDT is O(n) for each round.*

Proof: Since a sensor node sends exactly one message each round (to notify it neighbor that it will turn ON or turn OFF), the message complexity of the algorithm is *O(n).*

First, a sensor needs $O(\Delta^2)$ time to run IDT. A sensor has at most Δ neighbors. In the worst case, a sensor *s* has the highest priority among its Δ neighbors. Thus it has to wait the messages (*TURN-ON* or *TURN-OFF*) from those Δ neighbors. Since each of those neighbors takes $O(\Delta^2)$ time to evaluate its redundancy (line 12 to line 17 of Pseudocode 4), sensor *s* has to wait for $O(\Delta^3)$ time to decide its own status. Thus, the time complexity of SPIDT is $O(\Delta^2) + O(\Delta^3) = O(\Delta^3)$. ■

Theorem 4.6. (Proof of correctness) *If the monitored area can be completely covered when all the sensors are active and are assigned maximum sensing ranges, then ASIDT ensures that the whole monitored area is completely covered.*

Proof: This can directly be derived from the correctness of IDT (Theorem 4.2) ■

4.6. Simulation

4.6.1. Simulation settings

For energy consumption, we employ the quadratic energy model as in Section 3.3.2.1. That is, when a sensor is in active mode, the amount of energy consumption for 1 unit of time is proportional to square of the sensor's sensing range. The formal formula is as follows:

$$energy\,consumption = \frac{1}{\kappa} \times R_s^2 \qquad (4.1)$$

where:

- κ is an energy parameter which depends on the characteristic of a sensor.
- R_s is the sensor's sensing range.

The simulation configuration is listed in the following table:

Table 4.1. ODT, IDT, ASIDT, SPIDT simulation setting

Network size	*800m × 400m*
Energy range	*100 → 120 mJoules*
κ	*8000*
Maximum sensing range	100^m

4.6.2. Simulation results

We first compare lifetime of resulting networks of our algorithms proposed in this chapter (IDT, ASIDT, SPIDT) with that of ODT. However, due to the fact that ODT cannot completely cover the network at all cases (this claim is proved in Section 4.3.2 and is re-confirmed by our simulation results shown in Figure 4.3 and in Figure 4.4), we adopt the following definition of network lifetime, even though that definition is not fair for our proposed algorithms.

Definition 4.6. (Network Lifetime) *Network lifetime is the duration in which the network maintains the same coverage quality as it does at the very beginning*.

Based on Definition 4.6, Figure 4.2 shows the *ratio* of lifetime of resulting network created by IDT, ASIDT and SPIDT over that of ODT. As it can be seen, the lifetime ratio of ASIDT/ODT and SPIDT/ODT are approximately proportional to number of sensor in the network while the ratio of IDT/ODT is almost a constant. This phenomenon results from the way ASIDT and SPIDT re-use sensors and let them re-adjust their sensing ranges after any breaches (uncovered portions) appear. Because the sensors are carefully pruned in SPIDT, network lifetime of SPIDT is longer than that of ASIDT (where all the sensors are active).

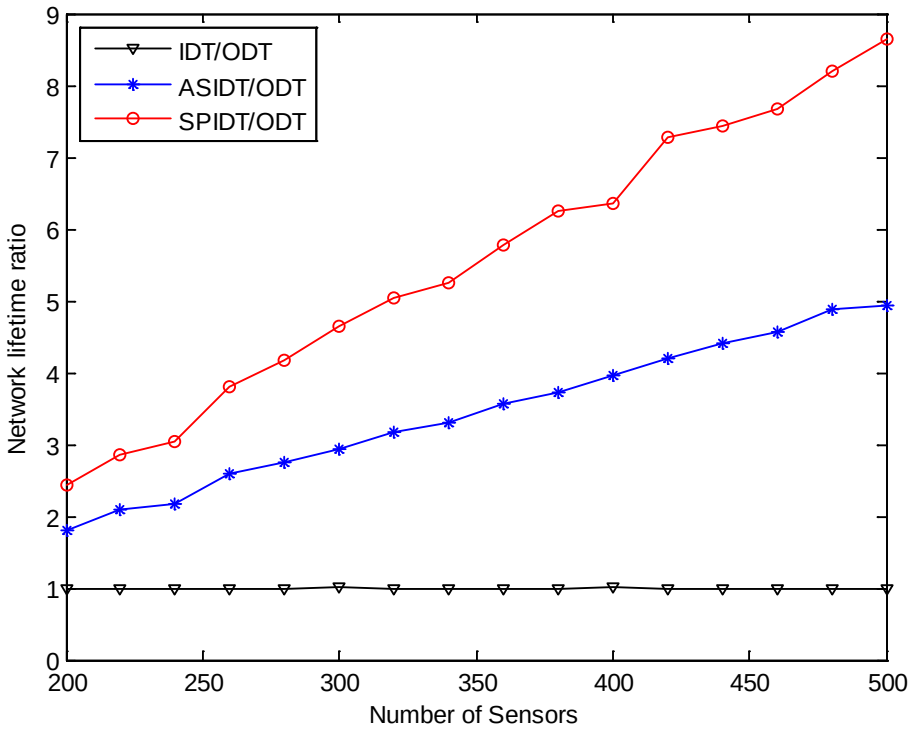


Figure 4.2. Network lifetime ratio

More specifically, network lifetime of ASIDT and SPIDT is bounded by the lifetime of the sensor having longest lifetime while that of IDT, ODT is bounded by lifetime of the *critical* sensors. The lifetime of a sensor is understood in general meaning as the duration from the time that a sensor is deployed until it dies and that sensor may go to sleep (several times) in between. A critical sensor is the one that if it turns off the coverage quality provided by the network will instantly decrease. Since the critical sensors usually locate near the boundary regions of the monitored area, and their sensing ranges are usually large (sometime are $MaxR_s$), the lifetime of a critical sensor might be short. Besides, by letting sensors turn off when they are redundant, thus the lifetime of the most long-lived sensors of SPIDT is longer than that of ASIDT. Those explain the gaps among the three curves shown in Figure 4.2.

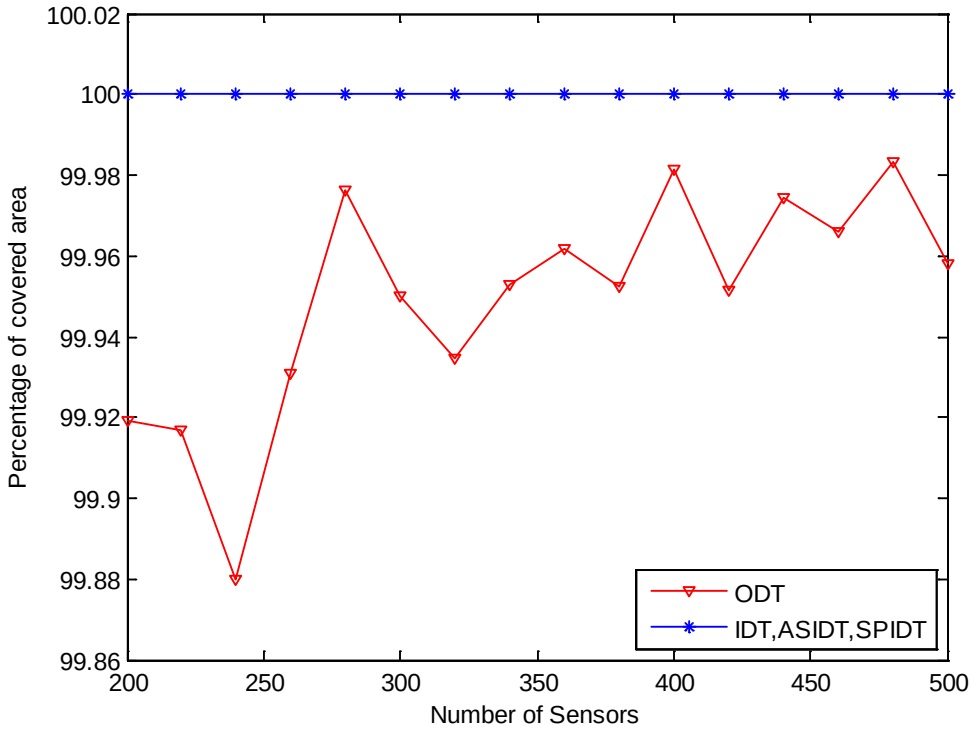


Figure 4.3. Percentage of covered area

To prove our aforementioned claim that ODT does not completely cover the monitored area at all cases and to re-confirm the correctness of Theorem 4.2, Theorem 4.4, and Theorem 4.6, we also measure *percentage of covered area* of the four algorithms in Figure 4.3. Figure 4.4 shows another measurement to confirm the correctness of those theorems. *Percentage of covered area* is defined as the total area of the sub-regions covered by the network over that of the whole monitored area. As can be seen in Figure 4.3, IDT, ASIDT and SPIDT completely cover the monitored area at all cases, thus the average result is always *100%*. On the other hand, ODT sometimes cannot completely cover the monitored area, thus after calculating average, even the average result is really high (more than *99.8%*), but it still cannot reach *100%*. Next, we measure the frequency that ODT cannot completely cover the monitored area.

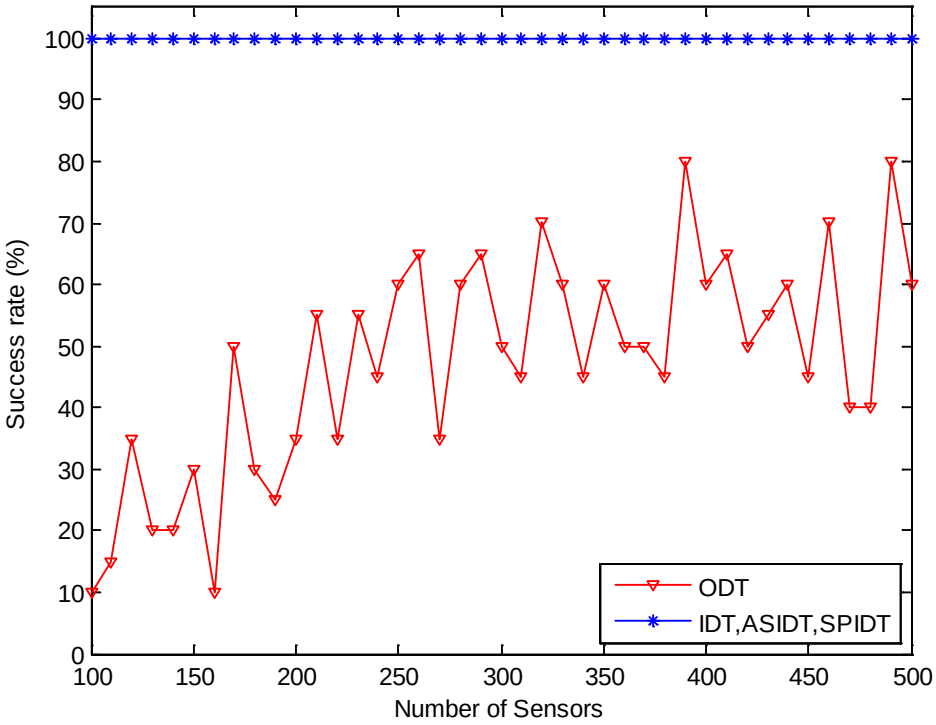


Figure 4.4. Success rate (%)

Figure 4.4 measures the success rate of the four algorithms. If the resulting network of an algorithm can completely cover the monitored area, it is considered as a successful execution. In other words, this plot shows the percentage of successful times of ODT and that of IDT, ASIDT, SPIDT. As can be seen, our three algorithms can always provide complete coverage while ODT cannot. Particularly, when the number of nodes is 100, the chance for ODT to be successful is only 10%. This very low success rate can easily be understood considering two facts:

- The sensors are usually deployed inside the monitored area. If some sensors are to be placed outside the monitored area, the success rate of ODT might increase a bit. Notice that that assumption also highly increases network lifetime of ASIDT and SPIDT in compared with their network lifetime without that assumption.

- The boundary effect explained in Section 4.3.2

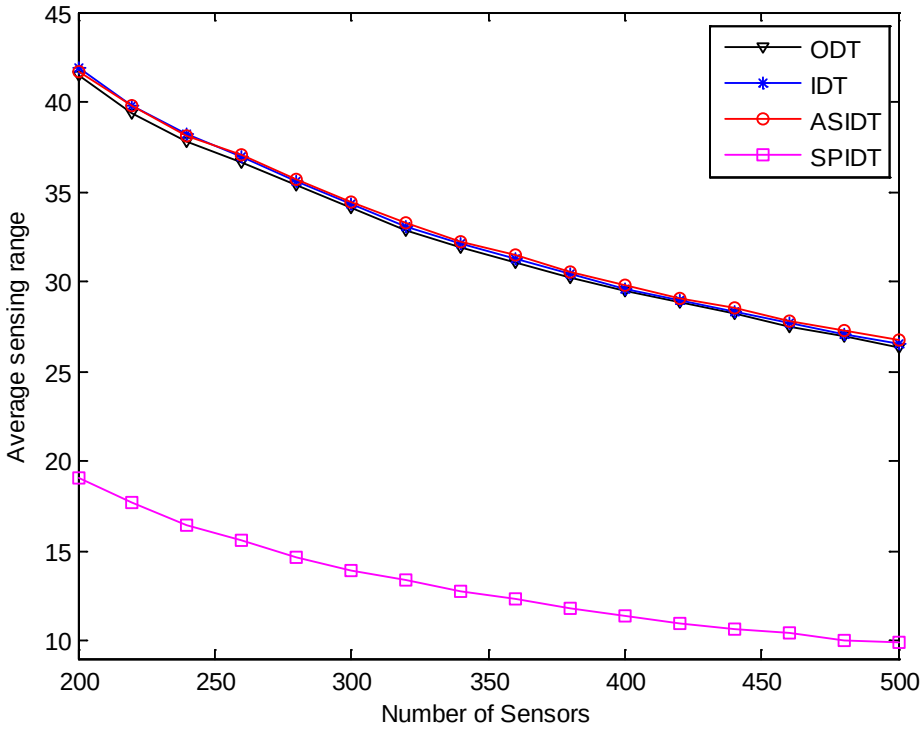


Figure 4.5. Average sensing range.

Besides resulting the longer network lifetime, SPIDT can also diminish the redundancy of unnecessarily assigning too long ranges for sensors' sensing ranges. This is illustrated in Figure 4.5 as the average sensing range of SPIDT is much smaller than that of IDT, ODT and ASIDT. As can be seen, the average sensing range of IDT, ODT and ASIDT are similar due to the fact that those heuristics have all the sensors turn on which causes that redundancy. One reason which contributes to the low average sensing range of SPIDT is that only a subset of sensors are active (thus, only a subset of sensors have non-zero sensing ranges) while the rest (sensors which do not belong to the subset) are considered as having zero sensing ranges.

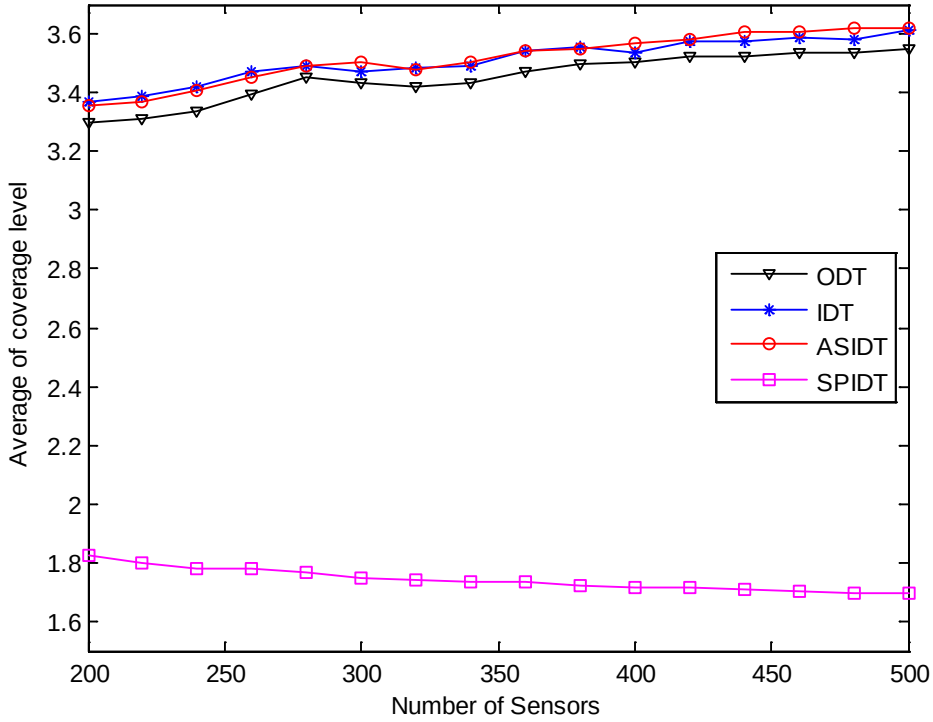


Figure 4.6. Average of coverage level

The last indication we use to verify the efficiency of IDT, ASIDT, and SPIDT is *area coverage level*. To calculate *area coverage level*, we first create a virtual fine-grained grid. We then scan through all the grid points in the area and record the *coverage level* of each point. The *coverage level* of a point is the number of sensors that cover that point. The average of the coverage levels of all points is named as *area coverage level*. Figure 4.6 shows the *area coverage level* of our four comparing algorithms. The reason that average coverage level of ODT is a little bit smaller than that of IDT and ASIDT is that sometimes ODT does not completely cover the area. So, there might exist breaches in the monitored area whose coverage levels are 0 instead of non-zero values for other covered sub-regions. Again, SPIDT beats all the other algorithms in this measurement because SPIDT carefully selects a subset of the sensors to turn off, so that the *area coverage level* is substantially reduced.

As a conclusion, while ASIDT outperforms IDT and ODT in most of the measurements, SPIDT outperforms ASIDT, ODT and IDT in all metrics we conduct simulations on.

4.7. Improvement: SPIDT supports connectivity at all cases.

Even resulting network of ASIDT (Section 4.4) is always connected, that of SPIDT (Section 4.5) might be not for all cases. That is, sometimes SPIDT has to rely on the assumption that the communication range of every sensor is at least twice of its sensing range to guarantee the network connectivity. To overcome this, we can combine the turning on/off rule of SPIDT with some existing rules for constructing routing tree such as Rule 1 or Rule 2 in [WUL99] to provide connectivity for the network without depending on that assumption.

CHAPTER 5.

COMPOSITE EVENT DETECTION

In previous chapters, we implicitly assume that all the sensors have exactly one sensing component (i.e., sensing unit) and components are exactly the same among all the sensors (for example, all the sensors are equipped with only temperature sensing component). In other words, in term of sensing ability, we only consider homogeneous network so far. This assumption is also implicitly employed by most of the work considering coverage problem. So, the designed algorithm merely combines those sensors together without having paid attention of the sensors types. However, in some situations, the network's job is to collect the different kinds of environmental parameters (for example, not only temperature, but also light intensity and air pressure). Moreover, the network needs to understand the meaning of the data reading it measures and if possible, makes some derivations/guesses from that information. Also, the network have to send that derivation to some BS at a timely manner. That are some of the most critical requirements of "event alarming applications" where a complex event (e.g., a *fire*) is expected to be detected by the network and the occurrence of the event has to be notified to BS immediately. In this chapter we concern ourselves with that kind of application, consequently, we only consider heterogeneous network where sensors are different not only in communication/computation capabilities, but also in sensing capabilities. We first point out drawback of existing schemes that make them inappropriate for events alarming application. That is the motivation for us to devise a new scheme that saves significant amount of exchanging messages and energy used for communication.

Based on the proposed scheme, in Section 5.4 we first create a cluster-based algorithm where joint issues of fault-tolerance (*k*-coverage), topology control, routing, and network

connectivity are all taken into account. Notice that within the cluster, the sensors have similar communication/computation abilities but have very different sensing capabilities. In Section 5.5, we then sketch our idea to extend abovementioned cluster-based algorithm to a large network.

5.1. Motivation

Event alarming is an effective function of WSNs which is employed in many applications such as meteorological hazard detection, earthquake-tsunami alerting, and enemy detection in battle fields. Event alarming applications usually involve of detecting a complex event (which is usually referred as "*composite event*") that comprises of several simple events (which is usually referred as "*atomic events*"). Even composite event detection problem is extensively in the field of distributed system such as the works in [JAN05], [ROM04], it has not been considered adequately in field of networking. Thus, there are numerous of sensors and WSNs' characteristics are not properly taken into account in existing works concerning related issue.

In the literature, the most popular scenario is as following: all the sensors periodically send their data to an information processing center, e.g., a BS, a conclusion is then made at the BS to decide whether a pre-defined event has happened based on the reported data. We name this scenario as "data collection". Another more efficient scenario is "data aggregation", where data is processed in-network at some nodes before being forwarded. However, these methods are not suitable for real-time applications, especially for emergency alarming applications, where the alarm is stringently required to be announced in a timely manner. The drawbacks of most of the existing methods are that the real-time requirement is not taken into account, and the amount of the exchanged data may be huge which causes large energy consumption. In addition, at a BS, the received data need to be further analyzed to obtain a conclusion which further delays the alarm to be timely announced.

Because of its “emergency” characteristic, an event alarming application intrinsically differs from data collection/aggregation applications. First, the interested information is an answer to a question which can be derived by a set of predicates, not from the raw data in the form of numerical values. For example, for fire alarming applications, the users are not interested in the (currently) exact values of temperature or the smoke density of the monitored area. Instead, the users expect the quick answer to the concise question “is there any fire in the monitored area?”. Second, to guarantee accuracy and reliability for emergency alarming applications, a conclusion indicating the happening of an event should not rely only on one property of the event. For example, the event fire is a fusion of multiple sensed values of multiple different attributes, i.e., the occurrence of fire should satisfy some conditions such as “temperature $>100^oC$ AND smoke > *100mg/L”*, rather than a simple condition temperature > 100^oC or smoke > *100mg/L* alone. Any change in either temperature or smoke density that makes temperature > 100^oC or smoke > *100mg/L* true is an *atomic event*. The event that is a combination of several atomic events is a *composite event*, e.g., the composite event fire is represented as “temperature > 300^oC AND smoke > *100mg/L”*. The formal definition of an event is given in Definition 5.4 (for atomic event) and Definition 5.5 (for composite event) of Section 5.3. In this chapter, we investigate how to efficiently detect events for emergency alarming applications using WSNs. To detect a composite event, a number of sensing devices (i.e., sensors) with different sensing components need to be involved and then local operations, collaborations are conducted to give out the answer locally. To conserve energy, only the answer is sent to the BS.

Next, we explain our novel scheme which overcomes drawbacks of existing methods.

5.2. The scheme

5.2.1. The basic idea

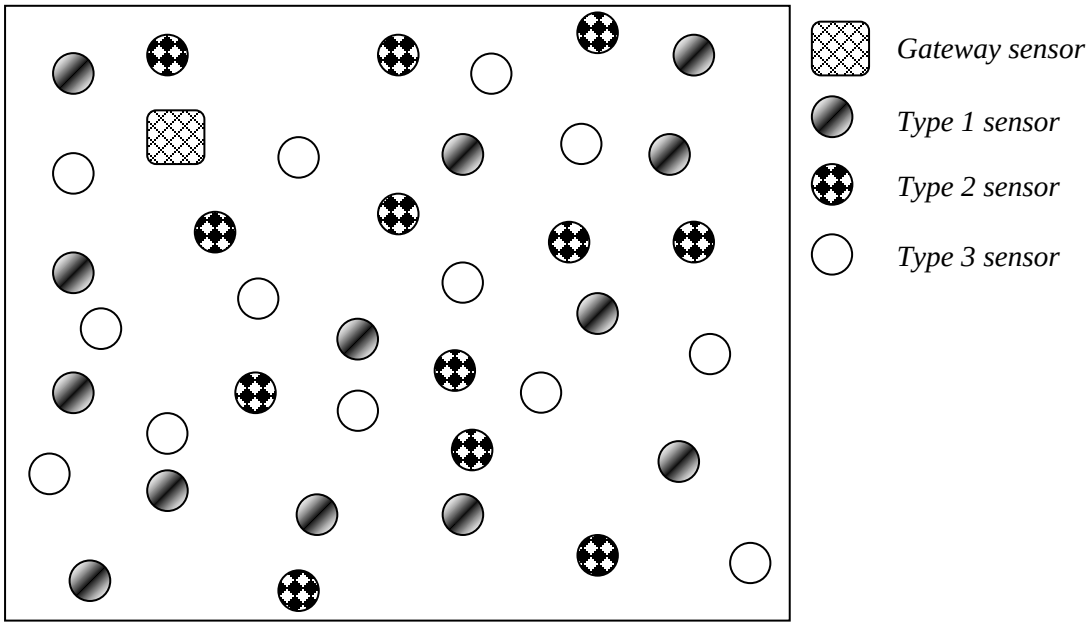


Figure 5.1. Event query in a WSN

The basic idea of our proposed scheme is illustrated in Figure 5.1, where an area is monitored by a WSN. There is a *gateway node* (or several gateway nodes) which is responsible for making a conclusion and reporting it to the users if an event happens. This gateway node is properly selected and *every* sensor in the network has a chance to serve as a gateway node in order to balance the energy consumptions. The different name (gateway node) is used to distinguish the functionality, not the characteristics. To detect events based on different properties, multiple types of sensors are deployed. For example, in Figure 5.1, *type 1*, *type 2*, and *type 3* sensors are used for temperature, smoke density, and light monitoring, respectively. An event E is defined with a compound propositional function as following:

$$E=F\,(P_1(x), P_2(x), \ldots, P_n(x)) \tag{5.1}$$

where $P_1(x)$ through $P_n(x)$ are predicates, F is a function of Boolean algebra operators such as '∧', '∨' or '¬'. For example, an event fire can be defined as $Fire = P_1(x) \wedge P_2(x) \wedge P_3(x)$, where $P_1(x)$ denotes the predicate "*temperature* > 300^oC", $P_2(x)$ denotes the predicate "*smoke* > *100mg/L*", and $P_3(x)$ denotes the predicate "*light* > *500cd*".

The event E and threshold values can be disseminated to gateway and non-gateway nodes by the BS at the initial phase or pre-installed in each sensor. Only the gateway nodes have the information about an event E. Each non-gateway node only knows the threshold values of its monitored properties. During the network operation time, once a sensor detects that the current sensed value is over the threshold of its monitored property, it sends one bit '1' instead of the sensed value to a gateway node. If a gateway node receives a '1', it checks if the compound propositional function which defines an event E derives a TRUE value. If so, it immediately sends a warning to the BS. It is important to emphasize that a sensor need not periodically report its raw sensing-data. Instead, it only reports the predicate '1', that is, it notifies the gateway node only when its sensed value reaches the threshold and refrains from sending data for all other cases.

5.2.2. Communication scheme

Denote r be the number of atomic events comprised to our interested composite event (see Section 5.3.1 for list of notations). For each atomic event, the payload of the packet sent by each sensor has two parts: $\lceil \log_2 r \rceil$ bits (namely event identification bits which depends on the composite event pattern definition) to identify the types of atomic events, and a single bit for the value (which is always '1' showing that the atomic event happens). Thus, if a sensor is able to sense t atomic events, the payload of the packet sent from that sensor occupies $\frac{\lceil \log_2 r + 1 \rceil \times t}{8}$ (the

payload of the packet must be the blocks of 8 bits - bytes). Thus, it is highly practical to employ some sorts of data aggregation mechanism which is extensively employed in database-related applications (the one presented in [UPA03] is an example of such mechanism) in reporting the event to the gateway. The communication overhead would greatly diminish if such a mechanism is utilized. Instead of directly employing that mechanism, we implement a better communication scheme which would be even more efficient when working with our 1-bit scheme. We name this communication scheme of "piggybacking" scheme. To reduce the header overhead of packets, several packets' payloads are piggybacked to a single packet. "Piggybacking" scheme has two variants, and the choice of which variant to implement is completely up to users:

Variant 1: In this variant, only data readings for the same atomic event are piggybacked. More specifically, any intermediate sensor, who forwards packets from its neighbors, will combine those data readings (including its own data readings) belonged to the same type of data reading (each type could detect an atomic event) and create a single packet for each type. It then forwards those packets out. This variant could save the overhead of event identification bits ($\lceil \log_2 r \rceil$ bits). However, a sensor has to forward as many packets as the number of types of sensing data that are sensed by itself and are carried by packets it has to forward.

Variant 2: In this variant, a sensor piggyback whatever it has to forward into one packet. That sensor does not care about the types of data readings that it has to forward. This variant could save the processing time, but it has to include the events identification bits since several types of data readings could be piggybacked in the same packet.

5.2.3. Scheme advantages

Our scheme significantly reduces the energy consumption by following facts:

- When its reading is over (smaller or greater than) a pre-defined threshold, instead of sending the real, big-sized data reading, the sensor only sends the single bit ‘1’.
- Each atomic event requires only a single bit to be sent, thus the packet’s header overhead will dominate the interested information carried in the payload. By piggybacking packets, the amount of packets sent in the network also significantly reduced while the size of packets is the same or little bit bigger than the normal ones.
- Sensors do not send the data reading all the time, it only sends the data when its reading is over a threshold. This has twofold: the sensors naturally do not waste their energy on unnecessary communication and the network traffic is also moderated.

Since no significant amount of data is sent to the BS, thus each sensor can naturally conserve more energy to extend network lifetime. Section 5.2.4 provides a rough numerical estimation on amount of saved energy. Furthermore, reducing network traffic has a good effect on lowering radio interference. Also, the energy consumption among sensors is well balanced.

Also, our scheme has following advantages:

- Decisions are locally made at special sensors, namely gateway nodes, and then only particular conclusions, e.g., the ones that specify the occurrence of an interested event, are reported to the BS, so that the users can obtain valuable information in a timely manner.
- The pattern definition of an event may consist of multiple properties instead of a single one and can be defined by users.
- By sending a warning from a gateway node to the BS, the BS can know where the event happens (with the assumption that the BS knows all the nodes’ positions).

5.2.4. Amount of data and energy can be saved

Octets:2	1	0/2	0/2/8	0/2	0/2/8	variable	2
Frame control	Sequence number	Destination PAN identifier	Destination address	Source PAN identifier	Source address	Frame payload	Frame check sequence
		Addressing fields					
MAC header						**MAC payload**	**MAC footer**

Figure 5.2. IEEE-802.15.4 general MAC frame format [IEEE4]

In this section, we conduct a rough comparison of amount of exchanging data being saved of our scheme using Variant 1 and the most common scheme in literature. To answer the question of how much is the amount of data can be refrained from being sent by our scheme, we measure the size of packets of traditional method and our novel scheme. To make our comparison practical, we assume the sensors networks using IEEE802.15.4 [IEEE4] (see Figure 5.2) which is used in many commercial sensors products such as those of Crossbow [XBOW09] as MAC+Physical layer protocol. For the sensors' operating system (OS), TinyOS [TINYOS], which is the most advanced and common OS for sensor networks, is our choice.

For variable-size fields of Figure 5.2, we assume 2 bytes for Destination/Source addresses/identifiers. For distributed algorithm where nodes are only required to be identified locally, 2 bytes are enough. Since TinyOS does not automatically support network and transport protocol for any packets sending out, an application layer's payload could be encapsulated directly to MAC and physical headers. Also, the header overhead for IEEE802.15.4 physical layer would be 2 bytes. Consequently, the average size of a packet should be 15 bytes + payload.

Denote $\gamma = \lceil \log_2 r \rceil$ (where r is number of atomic events comprised composite event) which is amount of bits to identify an atomic event. Assume i data readings are piggybacked in

the same packets, we need $\lceil(\gamma+i)/8\rceil$ bytes payload to carry *i* bits '1' with event identification bits. Assume the real data reading has the size of *b* bytes. The amount of data saving for a single packet would be $b\times i+\lceil\log_2\gamma\rceil-\lceil(\gamma+i)/8\rceil$ bytes out of $15+b\times i+\lceil\log_2\gamma\rceil$ bytes. The saving would be $\frac{(b\times i+\lceil\log_2\gamma\rceil-\lceil(\gamma+i)/8\rceil)\times 100}{15+b\times i+\lceil\log_2\gamma\rceil}\%$.

However, the data readings are not over the threshold at all times meaning if the sensors are working under our scheme, they do not always have to send the packets. Thus if the frequency of data readings being over threshold is *f* (*f* ≤ *1*), then we would expect the amount of data saving approximately is $\left[100-\frac{f\times(15+\lceil(\gamma+i)/8\rceil)\times 100}{15+b\times i+\lceil\log_2\gamma\rceil}\right]\%$. For example, if *r=4* (meaning *γ=2*), *f=25%, i=3, b=4* (4-byte integer number) then we could expect the amount of saving is about 85.7%. Notice that *f=25%* means that if sensors sample and sense the data readings (for atomic events) every *2 msec* then the composite event occurs every *8 msec* which would not be the case in most practical situations . In practice, the value of *f* must be much smaller than 25% and the value of both *b* and *i* might be bigger than the value we used in our example, thus the real amount of saving in practice would be (much) bigger than 85.7%

The energy consumption of a sensor is dominated by the energy burned to transmit the data which is proportional to the size of the packet. Network lifetime will increase if the amount of data being transmitted decreases. Thus, by using our scheme, the sensors could save a significantly amount of energy and network lifetime increases as consequence.

5.3. The TEKWED problem definition

In this section, we formally define the TEKWED problem (see Definition 5.7) and the following are some preliminary definitions and notations.

5.3.1. Preliminaries

First of all, we give a set of definitions where some of them are extensions of some definitions discussed above:

Definition 5.1. (Detection set) *A subset of sensors which jointly accomplish the event detection and alarming task.*

Definition 5.2. (Notification time) *Notification time is the summation of the time for all the members within a detection set to report the atomic events to the gateway, the time for the gateway to make a decision, and the time for a gateway to notify the BS that an event happens.*

From [ROM04], we adopt the following definition for an event:

Definition 5.3. (Event) *An event is a change of a real-world state.*

Then, a *k*-watched atomic and a *k*-watched composite event are defined as follows:

Definition 5.4. (*k*-watched atomic event) *An atomic event is said to be k-watched by a set of sensors D if at any time this event occurs at any point within the interested area, at least k sensors in the set D can detect this occurrence.*

Definition 5.5. (*k*-watched composite event) *A composite event is said to be k-watched by a set of sensors D if every atomic event forming that composite event is k-watched by set D.*

The following are some notations that we use in our algorithm.

- *N*: The number of sensors.
- *S*: The set of sensors. i.e., $S = \bigcup_{i=1..N} s_i$.
- *k*: The user-defined fault-tolerance level.

- r: The number of atomic events whose combination forms the composite event of interest. r depends on the pattern definition of the composite event.
- m: The number of detection sets.
- D_j (j = 1..m): The j^{th} detection set.
- t_j (j = 1..m): The assigned active time of set D_j .
- σ_l (l = 1..r): The l^{th} atomic event.
- Σ: The composite event. i.e., $\Sigma = F(\sigma_1, .., \sigma_r)$.
- χ_i (i = 1..N): The sensor s_i's current contribution.

5.3.2. Problem definition

Generally, in k-watching event detection, at least k sensors know the occurrence of an event. So the composite event is ensured to be detected even when any $k-1$ sensors concurrently fail. Also, k-watching helps the BS to be more quickly notified of events than the case where some atomic events are watched by only one sensor. In a primitive and standard form, the k-watching problem can be defined as follows:

Definition 5.6. (k-watching problem) *Given a set of sensors S, a monitored area A, and a composite event Σ which is a combination of r atomic events σ_l, l = 1..r, find a subset D of S such that every σ_l is k-watched by the set D.*

However, practical applications always require several other network constraints. In event detection and alarming applications, the most important issue is to timely report the event to the event consumer, e.g., the BS. Obviously, to ensure the messages are properly routed, a path to a gateway node must be well maintained, i.e., the connectivity for the detection set must

be provided. Additionally, the objective of conserving energy while accomplishing tasks is important. Thus, minimizing the notification delay and energy consumption are our primary concerns in this work. Since atomic event detection and alarming is a special case of composite event detection and alarming, in this work we only consider the latter. Formally, our concentration can be summarized as following:

Definition 5.7. (Timely Energy-efficient *k*-Watching Event Detection - TEKWED) *Given a set of sensors S, a monitored area A, and a composite event Σ which is a combination of r atomic events* σ_l, $l = 1..r$, *find a set of non-disjoint connected subsets (detection sets)* D_j, $j = 1..m$ *of S, and decide their corresponding active duration and subset masters (gateway nodes) such that:*

1. *The composite event Σ is k-watched by* D_j, $j = 1..m$ *at any time.*
2. *Network lifetime is maximized.*
3. *For each detection set, the notification time is minimized.*

5.4. Construction of detection sets

For WSNs, energy conservation is always a primary objective. Additionally, to ensure that events can be properly reported by inclined-to-failure WSNs, the proposed scheme is also required to provide a certain level of fault-tolerance. Different from many traditional solutions for the coverage problem where connectivity is assumed and is ignored, in event alarming applications the connectivity is extremely important since an event needs to be alerted on time. To meet these requirements, our proposed scheme divides the sensors into non-disjoint subsets and each subset can conduct the event alarming process mentioned above with a user defined fault-tolerance level. Instead of requiring all the sensors to be active all the time, only one subset is responsible for the event alarming task at any time. This way, energy can be conserved and

network lifetime can be extended. Our novel scheme has the following contributions and characteristics:

- Even if any limited number of sensors concurrently fails, the BS can still be properly warned in a timely manner if any interested event happens.
- The connectivity is guaranteed among sensors in the current detection set (defined in Section 5.3) by using a topology and routing control scheme.

5.4.1. Assumptions

- Each node has different sensing abilities. That is, each sensor may be equipped with more than one sensing components and both the numbers and types of those sensing components may be different among sensors. For example, a sensor can sense light intensity and/or smoke density, while its neighbor can sense temperature and/or pressure.
- For each type of sensing component, a sensor is equipped with no more than one sensing component. For example, a sensor has only one temperature sensing component and/or one pressure sensing component.
- All of a sensor's sensing components turn on or off simultaneously.
- Compound function *F* and predicates are diffused to all the sensors in the network at deployment phase or a mechanism as discussed in [INT00] is used to accomplish this.
- Sensors may have different communication ranges and different initial battery supplies.

5.4.2. Algorithm description

The algorithm is designed for a cluster. Each cluster is an observation zone. The detection sets construction algorithm starts by choosing a node to be the gateway - a special node in charge

of making the decision about the occurrence of the composite event and notifying the BS if it happens. The gateway can simply be any node with enough residual energy. The gateway node can be chosen by available algorithms as the ones in [MAL00] or in [VAS03]. The gateway is responsible for constructing a set of detection sets by executing the algorithm shown in Pseudocode 5. The **Construct − Leaves** function at line 9 is given in Pseudocode 6. The input of Pseudocode 5 includes the set of sensors in this observation zone and the user-specified fault tolerance level *k*. The output gives a series of Detection Sets (DS) D_j with their activation durations t_j. In this section, we interchangeably use the term sensor and node.

Pseudocode 5: Detection-Sets-Construction(*S*, *k*)

```
1: m=0.
2: while S ≠ Ø do
3:     T = Ø. Set all the counters c_l to k
4:     Color all the nodes WHITE
5:     Choose gateway node gw by algorithms in [MAL00] or [VAS03];  gw.Color = BLACK
6:     T = {gw}
7:     ► Discover detection set
8:     while at least one counter > 0 do
9:         L = Construct-Leaves(S, T)
10:        (Re)Calculate the contribution of each node in L
11:        Color all the nodes with contribution of 0 RED and remove them out of L
12:        if L == Ø then break;
13:        Sort L in descending order of contributions
14:        while L≠ Ø do
15:            Remove the node ρ from the top of list L
16:            Add node ρ into T
17:            ρ.Color = (ρ.Parent is BLACK)? BLACK:GREEN
18:            Decrease all the ρ's correlated counters by 1
19:            if a counter == 0 then
20:                all counters == 0? goto line 25: goto line 10
21:            end if
22:        end while
23:    end while
```

Algorithm continues next page »

```
24:     ► Update the subset and the sensors' energy
25:     if any counter > 0 then
26:         Remove T from S                ► T is isolated
27:     else
28:         if there exists a GREEN node then
29:             foreach GREEN node ρ do
30:                 κ = ρ.Parent
31:                 while κ is RED do
32:                     κ.Color = BLACK
33:                     Add κ to T
34:                     κ = κ.Parent
35:                 end while
36:             end for
37:         end if
38:         m=m+1
39:         gw_m = gw; D_m = T             ► A new detection set
40:         Assign t_m the smallest lifetime of a node in D_m
41:         Recalculate residual energy of sensors in D_m
42:         Remove from S the sensors who ran out of energy
43:     end if
44: end while          ► of "while S ≠ Ø do"
45: return m, {D_j, gw_j , t_j} j=1..m
```

Following is the algorithm to discover all the children of all the leaves of a known tree T.

Pseudocode 6: **Construct-Leaves(S, T)**

Input: A set of sensors *S* and a tree *T*.
Output: *L* - the list of all the children of *T*'s leaves

```
1: Construct a list L consisting of all the WHITE neighbors of T's leaves
2: ► Assign parent for each node in L
3: foreach node ρ in L do
4:     if any neighbor of ρ is BLACK or GREEN then
5:         ρ.Parent = the BLACK/GREEN node with the least number of children
6:     else
7:         ρ.Parent = the RED node with the longest lifetime
8:     end if
9: end for
10: return L
```

The gateway is in charge of building Breath-First-Search-like (BFS-like) trees rooted at itself connecting sensors belonging to a DS. The gateway greedily adds into a DS sensors whose sensing components can help to *k*-monitor atomic events. The gateway will add into the current DS any sensor equipped with a temperature sensing component. These sensors are said to have non-zero contribution because the number of its current helpful sensing components (which will be defined later) is bigger than 0. For sensors which have no contribution to the current DS, it may be added to the current DS later to provide connectivity for routing paths among sensor nodes. After executing this algorithm, the gateway node has knowledge of number of detection sets are constructed; for each detection set, which sensor nodes are involved in this detection set and their topology and routing information; and the duration during which this detection set should remain active to perform the detection task. Based on this information, the gateway node can decide a working schedule for the set of obtained detection sets. This working schedule is then broadcasted to all the sensor nodes so that each of them can know when it needs to be active. During the active time of a detection set, the sensor nodes in this detection set route their messages to the gateway node based on the provided topology and routing information. Essentially, the messages are routed to the gateway node along the constructed BFS-like tree for the current DS. The use of BFS-like trees is motivated by the requirements of topology control and routing paths maintenance for data delivery to the gateway node. Also, a BFS-tree has a similar characteristic as that of a SPT, so the total delay of sending data from active sensors (*i.e.*, sensors belonging to a DS) to the gateway can be minimized.

The detection sets construction algorithm starts by constructing a set of connected BFS-like trees rooted at the gateway node. The sensor nodes in each of these trees form a detection set. For each atomic event σ_l, we maintain a counter c_l recording the number of the currently

needed sensors who can detect σ_l for the current detection set to provide k-watching for σ_l. The initial value of c_l is k for each event σ_l. A sensor may be equipped with several sensing components that can monitor different σ_l. For a sensor s_i, a sensing component $component_{i,l}$ is called a helpful sensing component for counter c_l if it can monitor σ_l and the current value of c_l is greater than 0, and c_l is called a correlated counter for component $component_{i,l}$. At any point of time, a color in the following list, is used to represent a sensor's state:

- **WHITE**: a sensor has not been considered.
- **BLACK**: a sensor has already been added to a detection set and is connected to the gateway through other nodes in the current detection set.
- **RED**: a useless sensor (the one whose all correlated counters c_l are 0) has been considered, but has not been added to a detection set.
- **GREEN**: a sensor has been added to a detection set but its parent is not a BLACK node, thus it is not connected to the gateway through other nodes in the current detection set.

At the beginning of a detection set construction, all the nodes are WHITE except the BLACK gateway node *gw*, thus the tree T contains only *gw*. Denote L as a list of all of T's leaves' WHITE neighbors, which intuitively is the set of nodes in the next level in the BFS tree of the current T's leaves' level.

Our algorithm works in a greedy manner on a sensor's attribute named contribution χ to find a sensor that can be added to the tree. For a sensor s_i, its contribution χ_i can be determined depending on some parameters such as its residual energy, the energy needed to transmit a

message to its parent node, and the number of its helpful sensing components. χ_i can be formulated as follows:

$$\chi_i = f(e_i, d_i) \times \frac{h_i}{sc_i} \quad (5.2)$$

where:

- $f(e_i, d_i)$ is a function to calculate s_i's lifetime depending on its current residual energy e_i and its current communication range d_i. This function is further discussed in detail in Section 5.4.4.
- h_i is the number of si's helpful sensing components.
- sc_i is the number of all the sensing components that s_i is equipped with.

Eq. 5.2 can easily be extended to take any other parameters into account. Notice that the contribution of a sensor becomes 0 when h_i=0, i.e., when all the correlated counters for all its sensing components are 0, it becomes a useless sensor (for current detection set).

The algorithm tries to construct as many detection sets as possible. At each iteration of our algorithm's main loop, a temporary variable *T* is used to store the current tree being constructed. When the tree is completely built, it becomes a new detection set. *T* is gradually constructed by adding the node in *L* who has the biggest contribution. For other nodes in *L* having contribution of 0, they are colored RED. Each time a node is added to *T*, it is removed from *L* and all of its correlated counters are decreased by 1. Also, the node added to *T* is colored BLACK if it can connect with the gateway node through other BLACK nodes. If it cannot, i.e., its parent is a RED or GREEN node, it becomes GREEN. Each time any counter becomes 0, all the remaining sensors in *L* need to recalculate their contribution χ_i, since their h_i values may change. *T*'s construction process finishes when a) all the counters reach 0, i.e., sensors in *T* can

now provide *k*-watching for the composite event or b) there exists no *T*'s neighbors. For the latter case, remove all the sensors in *T* from *S* (line 26, Pseudocode 5) since *T* is isolated from the other sensors in *S*. For the former case, to guarantee connectivity, some RED nodes need to be added to make GREEN nodes connected to the gateway, which is accomplished by the block of code from line 28 to line 37. When *T* is completely built, it contains only BLACK and GREEN nodes. It is then assigned an active time *t* which is the smallest lifetime of a sensor in *T*.

The algorithm keeps constructing more trees using the above process until no more detection sets can be discovered. Finally, the algorithm returns the constructed detection sets with their active durations and the corresponding gateway nodes.

It is worth emphasizing several special points in our heuristic:

- Since the compound proposition is really simple and it is easy to derive the result, any ordinary sensor can make the decision from the reported predicates' values (the '1's sent by the sensors within its detection set) without any difficulty, that means a special type of sensor to work as a gateway node is not necessary.
- While constructing *L* (Pseudocode 6), we explicitly specify the parent for each node in *L*. Thus a topology of each detection set is as also established. Furthermore, when a sensor needs to report an atomic event, it only needs to forward that report to its assigned parent. By that mechanism, no addition routing protocol is needed.
- In our algorithm, we tradeoff between the energy consumption and the notification time while building the tree. By using a BFS-like tree, the notification time is supposed to be smaller. However, if the DFS is used for the BLACK nodes instead of using RED nodes to

connect GREEN nodes to BLACK nodes, the energy consumption should be smaller (since the number of intermediate RED nodes is minimized).

Theorem 5.1. *The algorithm ensures that a composite event is k-watched by every detection set and all the detection sets are connected sets.*

Proof: When a counter for an atomic event reduces to 0, the number of the sensors in the current detection set being able to detect that event is at least *k*, i.e., that event is currently being *k*-watched by the set. In our algorithm, a detection set is claimed to be discovered when all the counters become 0. Thus, the composite event is *k* -watched by any detection set.

If a detection set *D* contains only BLACK nodes, *D* is a connected set since all the BLACK nodes are connected to the gateway node through some sensors in *D*. If *D* contains several GREEN nodes, the block of code from line 28 to line 37 of Pseudocode 5 adds RED nodes to connect GREEN nodes in the lower level to the GREEN or BLACK nodes in the upper level of the BFS tree. The GREEN nodes in the upper level will also be or have already been connected to the BLACK/GREEN nodes in the higher layer of the BFS tree. Thus, all the GREEN nodes and hence all the nodes in the current detection set eventually are connected to gateway node. ■

5.4.3. An example

In order to clarify our proposed algorithms, we represent an example which briefly shows the result after some steps of our algorithms.

As mentioned, a composite event can be understood as the combination of some atomic events, e.g., high temperature, dazzling light and dense smoke indicate a fire event. Figure 5.3, Figure 5.4 illustrates an example of a fire alarming system. Each sensor is equipped with one or

several sensing components where 1 is for temperature, 2 is for light and 3 is for smoke sensing component. The list of sensing components of each sensor is given in pair of square brackets and is shown in italic font. The bold number above each sensor is the sensor ID and the list beside it is the list of its equipped sensing components. Figure 5.3.a shows the initial network with the links between nodes are communication links. From that initial graph, a BFS tree could easily be constructed as shown in Figure 5.3.b. Notice that Figure 5.4.a, each node is assigned a unique parent and in Figure 5.4.b, node 5 is changed to BLACK and is added to the detection set with the purpose of connecting GREEN node 7 and 10 to BLACK node 3. Thus, if node 10 wants to report an event to the gateway node 1, it can send that report along the pre-constructed path: 10-7-5-3-1.

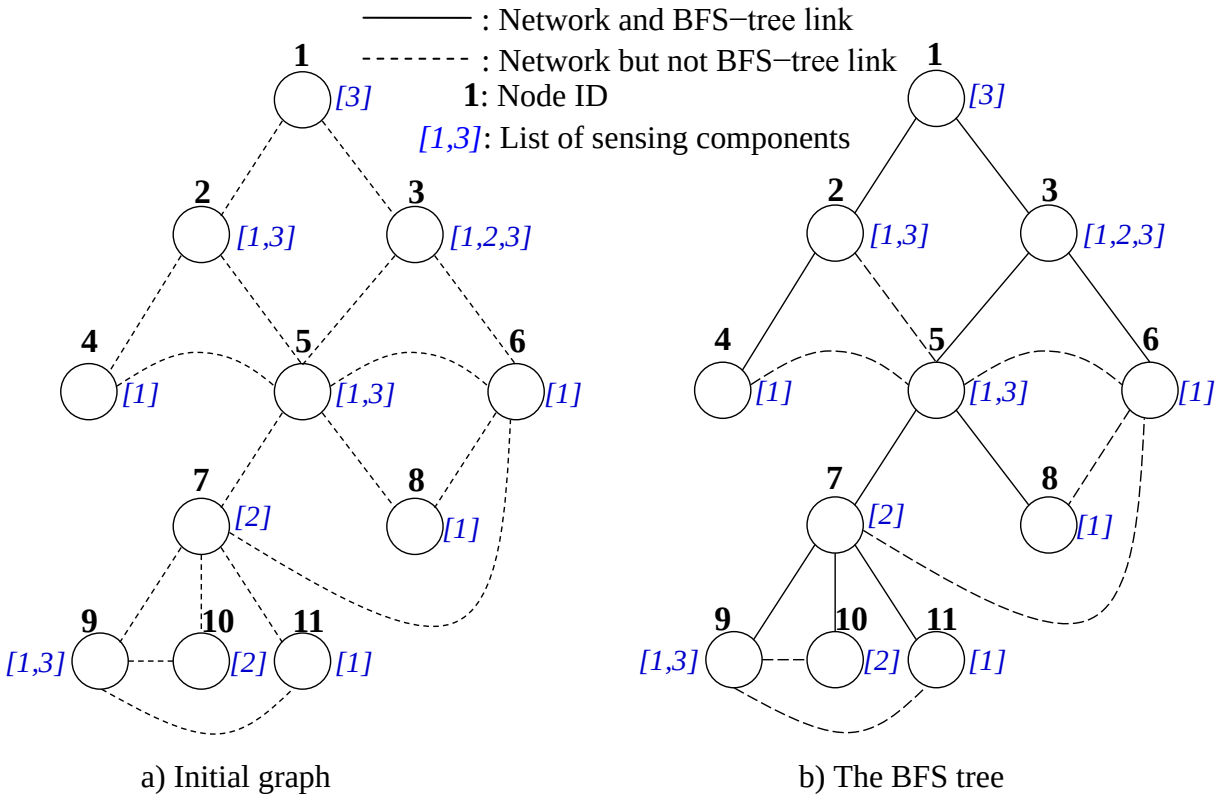
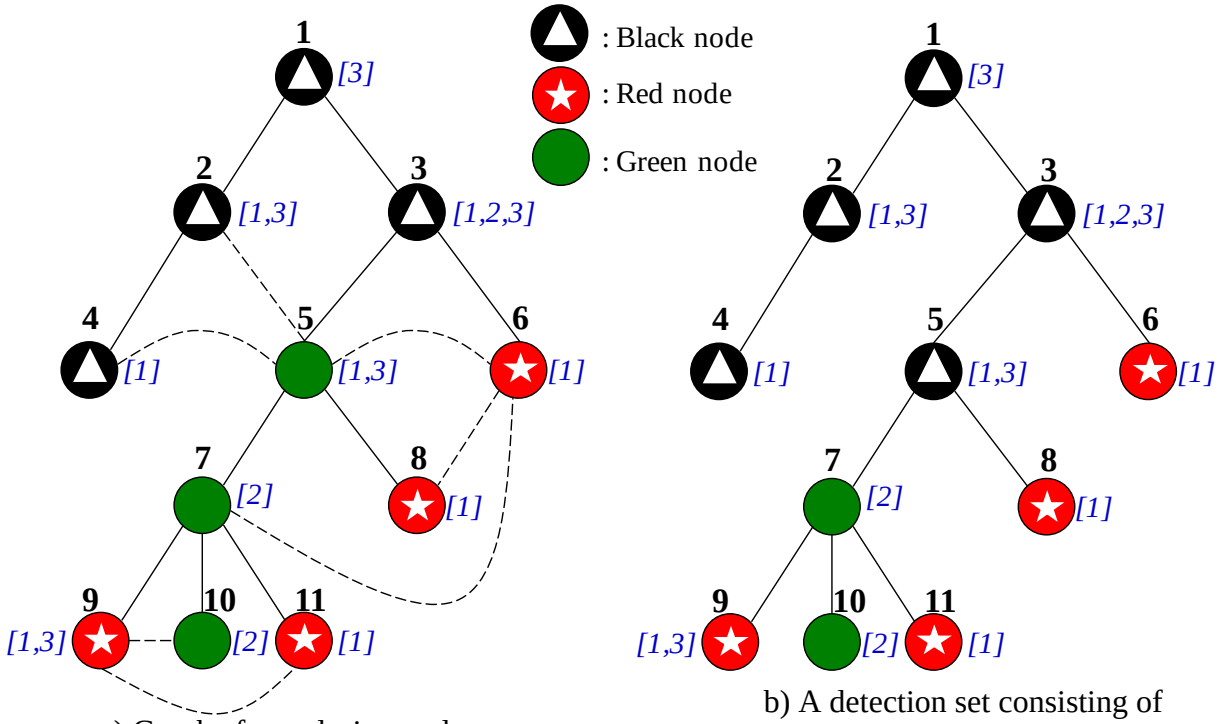


Figure 5.3. The construction of a detection set with $k = 3$, $r = 3$, Node 1 is the gateway.



a) Graph after coloring nodes

b) A detection set consisting of BLACK and GREEN nodes

Figure 5.4. The construction of a detection set with $k = 3$, $r = 3$ (continued).

5.4.4. The simulation

5.4.4.1. Simulation setting

The function to calculate sensor s_i's lifetime mentioned in Eq. 5.2 can be computed as follows:

$$f(e_i, d_i) = \frac{e_i}{Tx_i + Sx_i} \tag{5.3}$$

Where

- e_i is the current residual energy (in mJ).
- Sx_i is the energy needed to sense an event. Since we do not consider the coverage problem in this work, for the simulation part, we assume that Sx_i is proportional to the number of the

sensing components sc_i equipped on s_i, and each sensing component spends 10 (*mJ/unit of time*) when it is turned on. Thus, $Sx_i = 10 * sc_i$ (*mJ/unit of time*).

- Tx_i is the energy needed to transmit a message which is a function of the sensor's communication range d_i. The following model is widely adopted in the literature: $Tx_i = a \times d_i^{\alpha} + \beta$ where a, α and β are constants, $2 \leq \alpha \leq 4$ and a is usually set to 1 [STO01].

Table 5.1 presents our simulation setting in this section.

Table 5.1. Detection-set-construction simulation setting

Area size	*100m × 100m*
Communication range	*15m*
Initial energy	*20 → 30J*
r	*4*
α	*2*
β	*0*

5.4.4.2. Simulation results

In this section, we evaluate the efficiency of our heuristic through conducting simulations measuring network lifetime and notification time. We also compare network lifetimes for different values of *k* and different composite event pattern definitions. For each measurement, we run the simulations on 50 different completely-randomized networks and report the average results.

Figure 5.5 shows network lifetime with the assumption that a composite event (continuously) happens all the time, so the sensors in each detection set have to keep reporting events all the time, which is the worst case in practice. This assumption makes it easier to

illustrate our algorithm's performance. That means the practical network lifetime resulted from our algorithm must be longer than what is shown in Figure 5.5. As can be observed, network lifetime is relatively proportional to the ratio $\frac{N}{r \times k}$ where N is the number of sensors of the network. This effect is understandable since the bigger the value of the level of fault-tolerance k or the number of the atomic events r, the more sensors need to be involved in the event detection task, thus the more energy that the network consumes in a unit of time, hence the smaller network lifetime. On the other hand, the larger the number of the sensors, the bigger the number of the detection sets, hence the longer network lifetime. The slight fluctuates of curves in in Figure 5.5 are due to the nature of randomly deployed network we employ in simulation implementation.

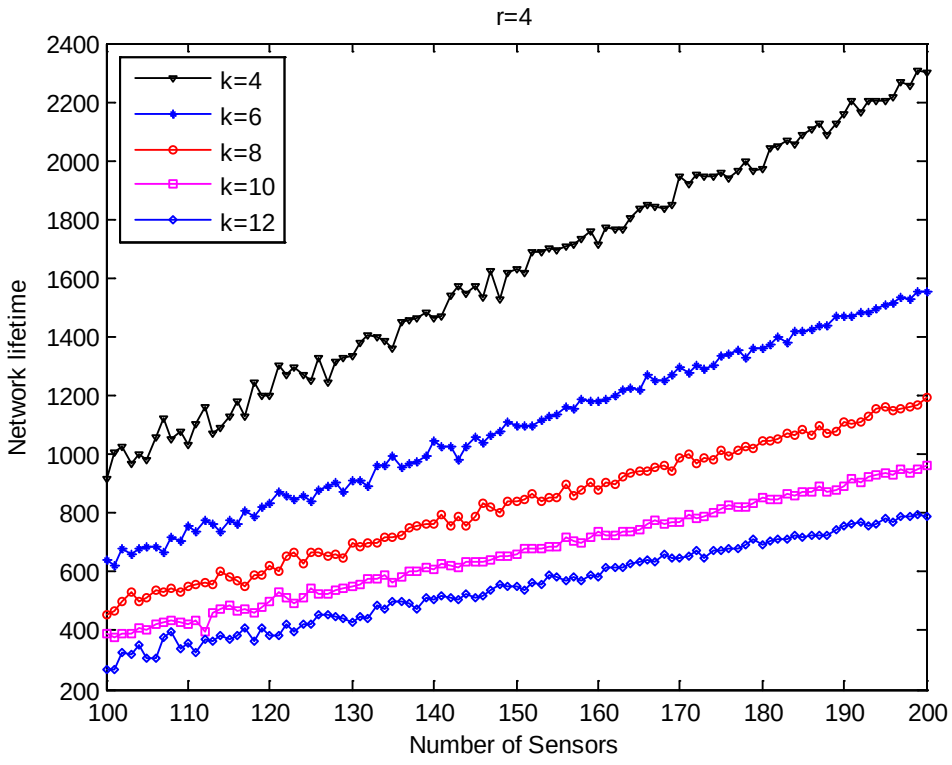


Figure 5.5. Network lifetime

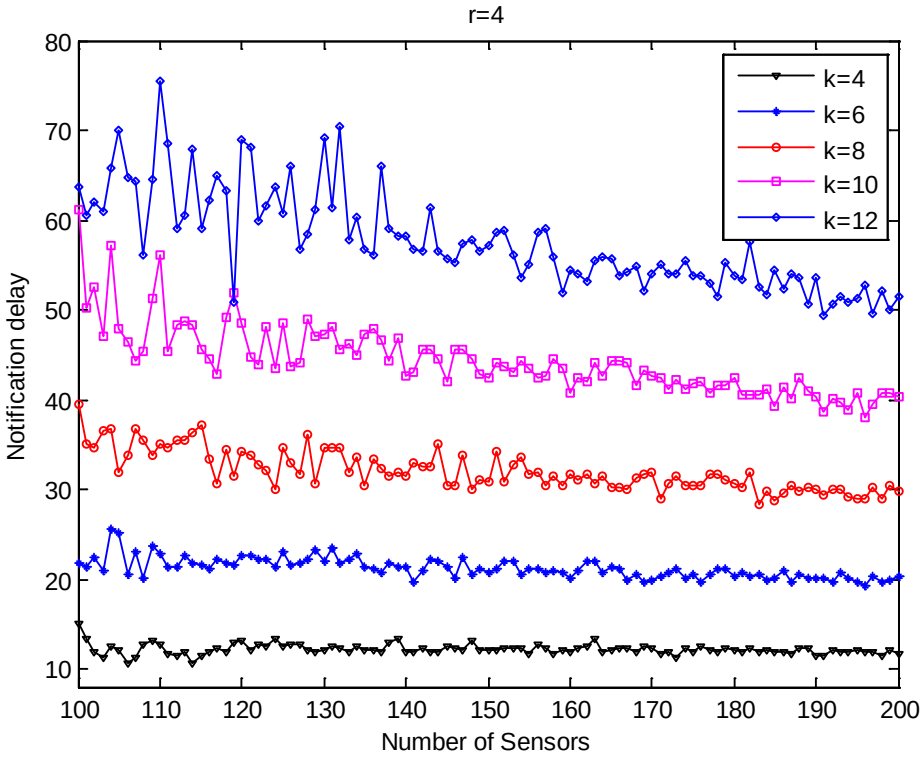


Figure 5.6. Notification delay

In Figure 5.6, we measure the notification time (as being defined in Definition 5.2). Because we have no specific BS, we omit the time from the gateway node to the BS. The larger the number of sensors, the smaller the height of the tree, thus the notification time is consequently smaller. On the contrary, the higher the level of fault-tolerance, the larger the number of sensors involved in detecting the event, consequently the notification time increases. Notice that Figure 5.6 shows the total amount of the time for all the members of a detection set to route their message to the gateway node by using the routing paths created by our algorithm without any piggybacking. The notification time is actually the time that the gateway node needs to know for sure that the composite event indeed occurs, i.e., the time for the gateway to receive *k* reports for each atomic event. However, the gateway can be aware of the occurrence of a composite event if it receives only one report for each atomic event. Which means the gateway

may be notified of the occurrence of the event in much shorter time. To measure this kind of time, we introduce a new simulation parameter named average notification time which is $\frac{Notification\ time}{k}$. This value is from 2.5 to 6.5 in our simulations. It is small enough for an event to be warned in a timely manner as required by TEKWED. Loosely speaking, our heuristic could create detection sets such that the notification delay to the gateway node is bounded.

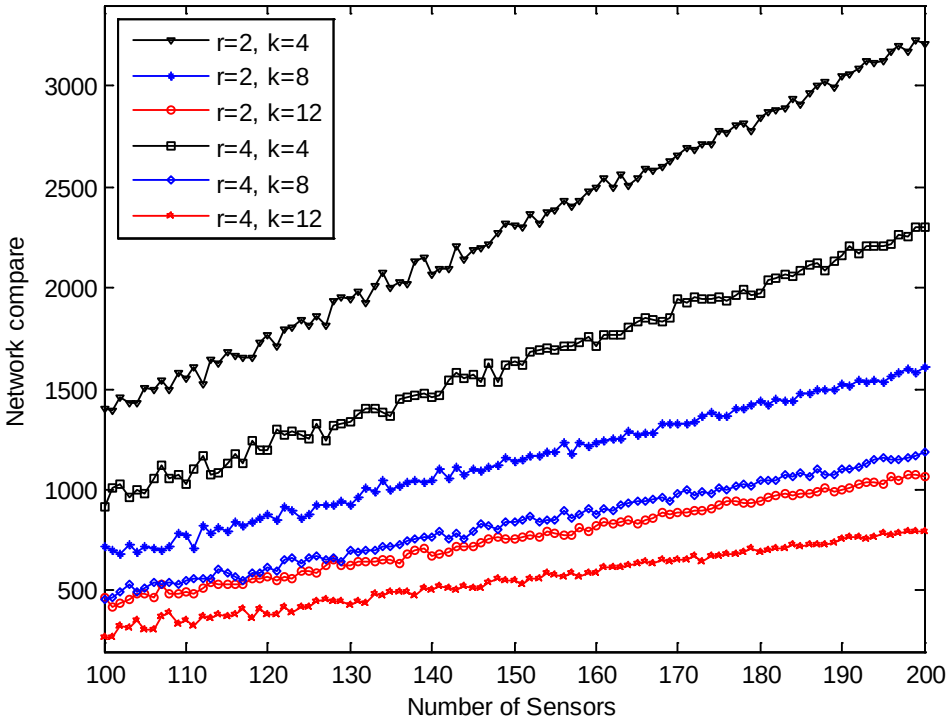


Figure 5.7. Network lifetime for different *r*, *k*

Figure 5.7 compares network lifetimes for several values of *k* and *r*. As can be seen, with the same value of *k*, network lifetime decreases when the value of *r* increases. Similarly, with the same value of *r*, network lifetime also decreases when the value of *k* increases. The effect can be reasoned by the same explanation we have made for Figure 5.5.

5.5. Enhancement: warning delivery for the whole network

In the previous section, we propose an algorithm for a cluster. In [LIC09], we extend the cluster-based algorithm proposed in Section 5.4 for the big network that comprises of a number of clusters.

In this extension, we take advantage of heterogeneous network where the network comprises of not only regular sensors, but also very powerful, resource-rich gateway nodes. In this section, the term "gateway" is referred to powerful gateways and these gateways may or may not be able to sense the data. These powerful gateways are now commercially available [XBOW09] and could help the network to provide better QoS [ITL09].

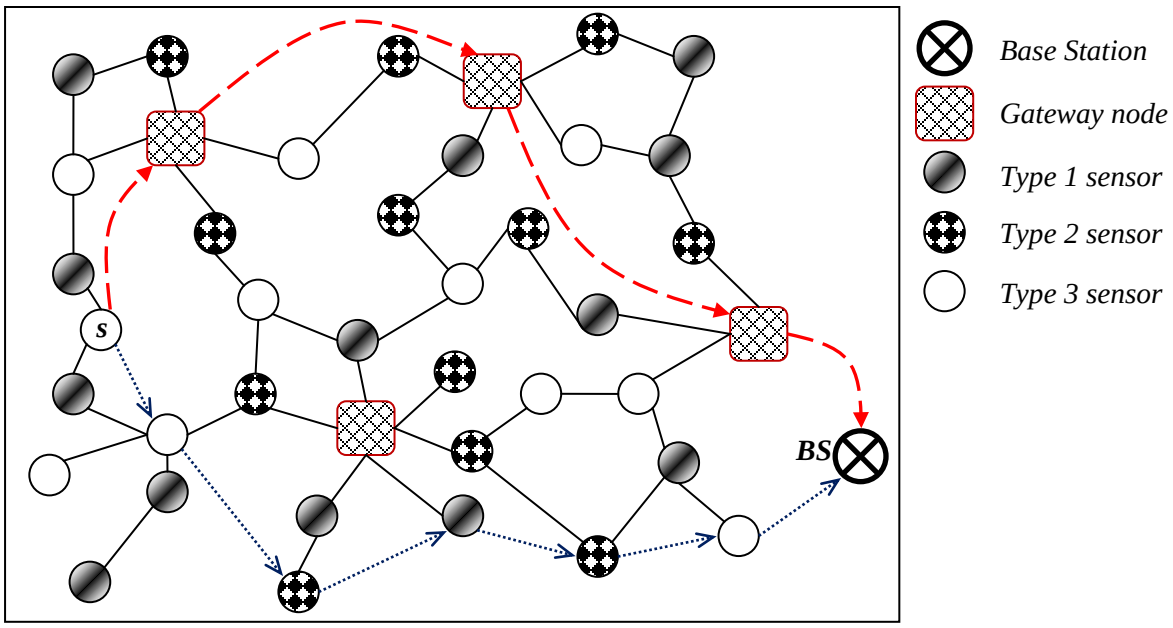


Figure 5.8. A heterogeneous sensor network with powerful gateway

Figure 5.8 is a heterogeneous sensor network where sensors are different in term of sensing, computation and communication capabilities. When a sensor *s* wants to deliver its data to BS, it does not have to relay through many sensors as shown in the dotted-line path in the

bottom of the Figure 5.8. Instead, it sends the data to the gateway, the gateway then forward it through a backbone consisting of long-ranged communication gateways (the dashed line shown in Figure 5.8). Since the gateways are very powerful devices, relaying data through them is the fastest, safest, and most energy-saving way.

We assume that the network is partitioned into a number of clusters and there is at least one gateway in each cluster. The event detection and delivery scheme within cluster is provided in previous section of this chapter. We now only have to create a backbone comprising only powerful gateway. Based on the work in [KHU94], the backbone is a spanning tree which balances the total energy consumption (on transmitting data through the backbone) and the delivery delay. That is, in our spanning tree (the backbone), the distance from a gateway *s* to other gateway *t* cannot be bigger than some constant (denoted by α) times of the distance of the shortest path from *s* to *t*. On the other hand, the total energy consumption to send data between *s* and *t* is at most β (β is another constant) times the total energy consumption to send that data between them in an energy-optimal tree. A tree which satisfies the requirements is referred as "*(α,β)*-Light Approximate Shortest-path Tree" - *(α,β)-LAST*. For a particular value of α, we first derive the feasible value of β such that the problem has solution (i.e., there exists a (α,β)-LAST). We then discuss an algorithm which is employed from [KHU94] to construct a (α,β)-LAST.

To keep this dissertation concise, we will not go into very detail about the tree construction algorithm. Ones may refer to [LIC09] for the whole inter-gateway delivery scheme with its efficiency.

CHAPTER 6.

THE PARTIAL COVERAGE PROBLEM

The *complete area coverage* (or *complete-coverage* problem) in WSNs where every point inside an area is covered by an active sensor has been extensively studied in literature. Most of coverage-related works concern themselves with prolonging network lifetime by different techniques. For many applications that do not require complete coverage at all times, one of the new techniques to extend network lifetime is to reduce the coverage quality to trade for network lifetime. In those applications, the required coverage quality may even be different at different points of time. For example, *forest fire* application [SON07], [HEF07], [YUN05] might require complete coverage in dried seasons while only requires 80% of the area to be covered in rainy seasons. As another example, *birds habit study* [KUG04] application might allow only 70%-coverage at night-time when the birds are sleeping while requires 100%-coverage at daytime when the birds are active. Thus, to extend the network lifetime, we can decrease the coverage quality if it is acceptable. The problem to cover only portion of an area is referred as the "*partial coverage*" problem. The foremost requirement for the partial coverage problem is that the ratio of the area that is covered over the whole monitored area has to be larger than some pre-defined bound. This bound must be a user-specific parameter. In this work, we use the notation α to refer to this parameter. Consequently, the partial coverage problem is also referred as α-coverage problem of which the objective is to cover only α-portion of the area (the formal definition of this problem is given in Definition 6.2**)**. Moreover, when possible it is always desirable to schedule the sensors such that the area is uniformly covered. It is clearly undesired if the network only covers some particular sub-regions of the area while uncovering the other large and continuous sub-regions. To evaluate coverage quality, a metric named *Sensing Void Distance*

(SVD) is used [LIU05], [WUA08] (our formal definition of this metric is given in Definition 6.6).

Due to the difficulty to verify the ratio of covered area over the whole monitored area (coverage ratio), all the existing algorithms for partial coverage are non-parallel, thus have very high time complexity. Besides, all the existing algorithms are intentionally designed for partial coverage, hence they do not utilize the numerous solutions for the complete coverage problem. Differently, in this work, we solve the α-coverage problem by utilizing various well-studied algorithms for the complete coverage problem. Our framework has of four strategies - two general strategies and the other two extended strategies. Among four strategies, two support networks where sensors have fixed sensing ranges (fixed-sensing-range network) and the other two are for networks where sensors can adjust their sensing ranges (adjustable-sensing-range network). Our framework can preserve the characteristics of original algorithms, the conversion process is simple and usually does not change the time complexity of original algorithm. For any particular α, the two general strategies guarantee the constant bound of SVD, i.e., they guarantee some degree of uniformity of covered area over the whole monitored area (coverage uniformity).

6.1. Existing work

The problem of partial coverage is recently investigated in the literature under many alias such as "α-lifetime" [ZHA04], [ZHA05], "p-percentage coverage" [GAO08], [WUA08], "θ-coverage" [LIU05], or "q-portion coverage" [BER04]. The problem requires the network to cover at least "p percent" or "α portion" or "q portion" of an area. In other words, if $\alpha = \theta = q = p/100$, those problems are actually the same. To be consistent with the name when this problem was first proposed [ZHA04], [ZHA05] and to make the term self-explaining, we use the name "α-coverage" (which will be formally defined later in Definition 6.2) to refer to this

problem and α is referred as *coverage ratio*. Trivially, the α-coverage problem is a generalization of complete coverage problem since complete coverage problem is actually 1-coverage problem (that is, α=1.0).

The work in [ZHA04] has provided the upper bound lifetime for a network which is required only to cover α-portion of the whole region. According to the authors of [ZHA04], there are two families of α-coverage algorithms: *regular family* of algorithms that maintain α-coverage from beginning and *special family* of algorithms that initially try to completely cover the area and gradually reduce coverage ratio. The authors have derived the asymptotic upper bound of network lifetime for both families of α-coverage algorithms. And it is shown in [ZHA04] that in compared with *special family* of algorithms, the upper bound of increased lifetime for *regular family* of algorithms is 15% for α=0.99 and 20% for α=0.95. Then in [ZHA05], the authors proposed a centralized algorithm to solve the α-coverage problem of which the increasing of network lifetime is close to the upper bound derived in [ZHA04]. In [BER04], a centralized algorithm based on Garg-Könemann method for q-portion coverage also is proposed. The algorithm has a performance ratio of $(1+\varepsilon)(1+\ln\frac{1}{1-q})$, for any ε > 0.

In [BAI05], percentage coverage rather than complete coverage is selected as the design goal, and a location-based Percentage Coverage Configuration Protocol (PCCP) is developed to assure that the proportion of the sensing area after configuration to the original sensing area is no less than a desired percentage.

Liu et. al. [LIU05] present a centralized algorithm which takes both coverage and connectivity into account. They are the first to analyze partial coverage properties in order to prolong network lifetime. Initially, active sensors are selected randomly. In each iteration, nodes

in a selected candidate path with the maximum gain are chosen. The algorithm continues until the whole area is θ-covered. This method is also employed by [GAO08]. The work in [GAO08] proposed two algorithms (a centralized and a distributed ones for the same problem). To provide different coverage qualities at different locations of the monitored area, the area is partitioned into a number of clusters and the algorithms partially cover the clusters one by one.

For α-coverage problem, ones always want to know how uniformly the sub-regions are covered. To evaluate that, adapting from [LIU05], the work in [WUA08] uses the metric named *Sensing Void Distance* (SVD) which is the distance from an uncovered point to the nearest covered point. The authors of [WUA08] also claim that their CDS-based distributed algorithm CpPCA-CDS can provide a constant-bounded SVD. However, in the case that value of p is too small that even a subset of a CDS can provide p-percentage coverage, the coverage redundancy has to be high to guarantee bounded SVD. In other words, in [WUA08] the coverage redundancy is the price for bounded SVD.

To the best of our knowledge, most proposed algorithms for α-coverage are centralized algorithms. There are distributed algorithms discussed in [GAO08], [WUA08], however, those algorithms work in distributed but not a parallel (i.e., consequential) fashion because the sensors have to wait for the value of α to be calculated by their neighbors to decide to be active or sleep. So the time complexity might be very high. In the worst case, the time complexity of the non-parallel (consequential) algorithms may be of the order of the network size.

6.2. Motivation

Most of the existing algorithms for the α-coverage problem are greedy on a so-called *contribution* (or similar parameters such as *gain*). *Contribution* of a set A of sensors (the set may contain only one sensor) is a parameter that mainly depends on the amount (area) of currently

uncovered region (by the current entire set cover) that can be covered the set *A*. For the simplest case where sensors' sensing regions are assumed to be perfect disks, that region usually is the part of union of disks that is not covered by some other overlapping disks (neighbors). However, calculating the area of that region is not trivial even for the simplest case that all the sensors have the same sensing range. The previous works usually ignore to explain how to do that calculation. To the best of our knowledge, there is no existing method to calculate the exact area for such region.

Besides, most of the current works directly solve the partial coverage problem. The common method is to greedily (on *contribution*) add sensors until at least α portion of the area is covered. As a result, most of the existing algorithms are centralized. The rest, few distributed algorithms have high time complexity due to the fact that they have to scan through all the sensors one by one until α-portion of the area is covered. That means the sensors cannot work in a parallel manner. In that sense, we claim that the α-coverage problem is an *impossible-to-directly-solve* problem in a distributed and parallel manner due to the fact that α is a global parameter which cannot be acquired in a parallel manner.

Moreover, there are a great deal of existing algorithms designed for complete coverage which motivates us to devise a way to utilize them for partial coverage. In the literature, there do exist many fully distributed and parallel algorithms that do not depend on *contribution* such as the ones in [ZHA03], [HUL05], [VUC06], [VUT07], [VUC09], [VUT09]. We should somehow convert those algorithms to the ones that can partially (instead of completely) cover the area. The resulting algorithms have to guarantee some level of coverage quality as required by the users. The conversion should preserve the characteristics of the original algorithms. For example, if the original algorithms provide connectivity along with complete coverage, then the resulting

algorithms should also provide connectivity. Moreover, the conversion should be simple and fast enough to not increase time-complexity too much. The resulting algorithms should uniformly cover the area when it is possible, in other words, the SVD must be bounded. Finally, but very important, the conversion should work with most of the existing complete-coverage algorithms.

In this work, we propose a framework that satisfies all the requirements of an algorithm conversion framework we just have discussed above. The methodology of our framework is to utilize the solutions of special problem (the complete-coverage problem) to solve the generalized problem (the α-coverage problem). The framework consists of four strategies (two general strategies and the other two extended strategies) which are designed for different kinds of networks. Two (one general and one extended strategy) are designed for fixed-sensing-range WSN and the other two are for adjustable-sensing-range WSN. Amazingly, for any certain desired coverage ratio α and a particular WSN, the resulting algorithm for partial coverage of two general strategies guarantee a constant-bounded SVD for all the original algorithms for complete coverage , in other words, the resulting algorithms of two general strategies uniformly cover the area . Also, the general strategies work for almost all complete-coverage algorithms.

6.3. Preliminary

We dedicate this section to introduce some introductory concepts and definitions.

Definition 6.1. (α-cover) *Given a real number α where 0< α <1, a two-dimensional area A and set S of n sensors s_i for i=1..n. Sensor s_i's sensing region is S_i. Notation $\|A\|$ denotes the area of region A. A sub-set $C \in S$ is said to α-cover area A (and thus C is said to be a* α-set-cover*) if:*

$$\left\| \left(\bigcup_{s_i \in C} S_i \right) \bigcap A \right\| \geq \alpha \|A\| \tag{6.1}$$

By this definition, the traditional set cover (which completely covers the area) is also called *1-set-cover* in this work. Next, we define the *α-coverage problem* in its most general form:

Definition 6.2. (α-coverage problem) *Given a real number α where 0< α <1, a two-dimensional area A and set S of n sensors s_i for i=1..n. Find the set of α-set covers $C_1,C_2,..,C_l$ of S.*

Notice that, the above definition includes only the most general and common requirement (inequalities 6.1) for partial coverage. In different situations, some additional requirements may also be included. A requirement to minimize the cardinality of set C_i or to minimize the total energy consumption of sensors in set C_i could be examples. Next, we introduce some supporting definitions:

Definition 6.3. (γ-virtual network) *For a particular real number γ where 0<γ and a WSN S with n sensors $s_1, s_2, .., s_n$.*

- *If the network S is a fixed sensing range WSN where sensor s_i, i=1..n has the sensing range of R_i, then the γ-virtual network of S, denoted by S^γ, is the network where sensor s_i has the sensing range of $\frac{R_i}{\sqrt{\gamma}}$ for i=1..n. This sensing range is called* virtual sensing range.
- *If the network S is an adjustable sensing range WSN where sensor s_i, i=1..n has the* maximum *sensing range of $MaxR_i$, then*:
  - *The γ-virtual network of S, denoted by S^γ, is the adjustable-sensing-range network where sensor s_i has* maximum *sensing range of $\frac{MaxR_i}{\sqrt{\gamma}}$ for i=1..n. This maximum sensing range is called* virtual maximum sensing range.

- *The fixed-γ-virtual network of S, denoted by $S^{\gamma}_{\text{fixed}}$, is the fixed-sensing-range network where sensor s_i has sensing range of $\frac{MaxR_i}{\sqrt{\gamma}}$ for i=1..n. This sensing range is called* virtual sensing range.

Because we have to deal with two types of networks in this paper (real and virtual networks), it is necessary to distinguish two types of set covers corresponding with two types of networks as in the following definition.

Definition 6.4. (VSC$^{\gamma}$: γ-virtual-set-cover) *Given real numbers γ where 0<γ and set S of n sensors s_i for i=1..n. The γ-virtual network of S is S^{γ}. A 1-set-cover of S^{γ} is called* γ-virtual-set-cover, *denoted by VSC$^{\gamma}$.*

Definition 6.5. (μ-RSC:μ-real-set-cover) *Given real numbers μ where 0<μ and set S of n sensors s_i for i=1..n. The μ-real-set-cover, denoted by μ-RSC, of a virtual set cover C_j (C_j is not necessarily a VSC$^{\gamma}$) is the set cover where every sensor has sensing range of $\sqrt{\mu}$ times of its sensing range in C_j. Furthermore, a μ-RSC is* feasible *if either of the following conditions is true:*

- *If the network is a fixed-sensing-range network, then every sensor of μ-RSC must have its pre-defined sensing range.*
- *If the network is an adjustable-sensing-range network, then every sensor of μ-RSC must have its sensing range no larger than its pre-defined* maximum *sensing range.*

6.4. The framework for the α-coverage problem

Having all the preliminary concepts, we now introduce our framework. The framework requires three inputs: *i)* the coverage ratio *α*, *ii)* a complete-coverage algorithm $\mathbb{A}$ (sometimes

referred as "*original algorithm*"), and *iii)* the network $\mathcal{S}$ consisting of n sensors $\mathcal{S}=\{s_1,s_2,..,s_n\}$. Our framework transforms an original algorithm as the input to the one that can generate a set of α-set-covers. We sometimes refer the obtained algorithms as "*α-coverage algorithms*".

The framework has four strategies for different cases where sensors can or cannot adjust their sensing ranges (kinds of the networks $\mathcal{S}$) and the original algorithms are designed for fixed or adjustable sensing range WSNs (types of the complete-coverage algorithms $\mathbb{A}$). Thus, in this work we use the terms "framework" and "four strategies" interchangeably.

Next, we introduce some notations and assumptions.

6.4.1. Assumption and notations

We assume the sensing region of a sensor s_i to be a disk centered at s_i. If s_i has a sensing range of R_i, denoted by s_i (R_i), then that disk has a radius of R_i.

For an input algorithm $\mathbb{A}$, conventionally, the result of $\mathbb{A}$ to completely cover a sensor network $\mathcal{S}$ is a set of $\{C_j, t_j\}$ pairs where each C_j is a 1-set-cover and t_j is its working schedule. Each set cover C_j is a set of sensors (with their sensing ranges) that will be turned on to provide complete coverage quality. The schedule t_j of set cover C_j may include the starting point of time and the durations that the set cover C_j will be activated. It is worth emphasizing that an algorithm $\mathbb{A}$ does not always explicitly create a set of all desired set covers are return them as an output. For example, the family of scheduling algorithms that work in round, of which the network timeline is divided into equal-length rounds and the algorithms create a set cover for each round, they are just create one set cover at a time for each round. We made the assumption about of the result of original algorithms just to make our framework easier to explain.

For an adjustable-sensing-range WSN $S=\{s_1,s_2,..,s_n\}$, we assume that each sensor s_i is able to smoothly adjust their sensing ranges under some upper cut-off ranges. This assumption is employed by most works concerning adjustable-sensing-range WSNs such as the ones in [VUC09], [VUT09].

In terms of sensors' sensing ranges and initial energies, our framework has no restriction on the type of WSNs. That is, the WSNs may be heterogeneous or homogeneous, the network may be fixed-sensing-range or adjustable-sensing-range networks. We also has no assumption on the sensors' locations, that is, the sensors may be randomly deployed.

6.4.2. The basic idea of the framework

Before explaining our strategies in detail, it is necessary to emphasize the essences of our framework. Our framework essentially modifies a complete-coverage algorithm to a resulting algorithm for the α-coverage problem. The framework first modifies the original algorithm so that the original algorithm is executed on a virtual network (see Definition 6.3) instead of on a real network. The result of that execution is a set of virtual set covers C_j (because those set covers are of virtual network, i.e., sensors of those set covers have virtual sensing ranges which might be bigger than their pre-defined ranges) and their schedules t_j. The framework then modifies the original algorithm so that the final result is a set of real set covers $\overline{C_j}$, where each set cover $\overline{C_j}$ usually is α-RSC of C_j and $\overline{C_j}$ is feasible set cover (see Definition 6.5).

To clarify our essential idea, we get an example of a simple network in Figure 6.1 for $\alpha=0.25$. The network comprises of 6 sensors s_1,s_2,s_3,s_4,s_5,s_6 locating at O_1, O_2, O_3, O_4, O_5, O_6. The solid rectangle is monitored area. The solid circles are sensing regions in *real* network where sensors' sensing ranges are real, pre-defined ranges. The dashed circles are sensing regions of

virtual network where sensors' sensing ranges are virtual ranges. Since we are considering $\alpha=0.25$, the virtual ranges are twice ($1/\sqrt{0.25} = 2$) of real sensing ranges.

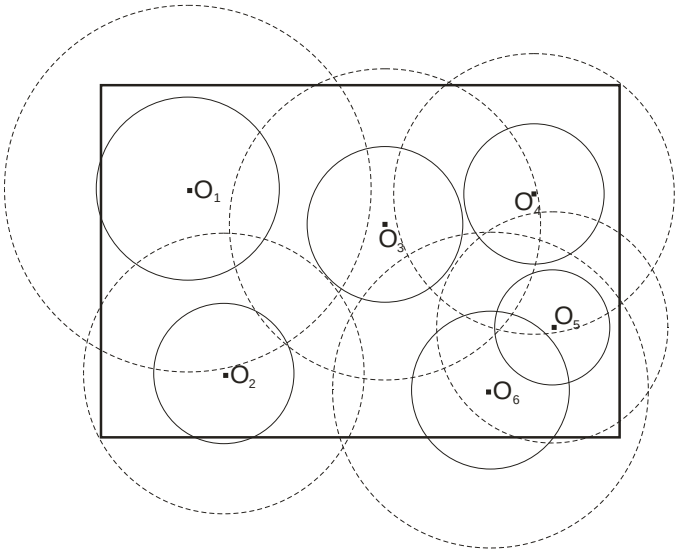


Figure 6.1. Framework example: *virtual* network and *real* network. $\alpha=0.25$

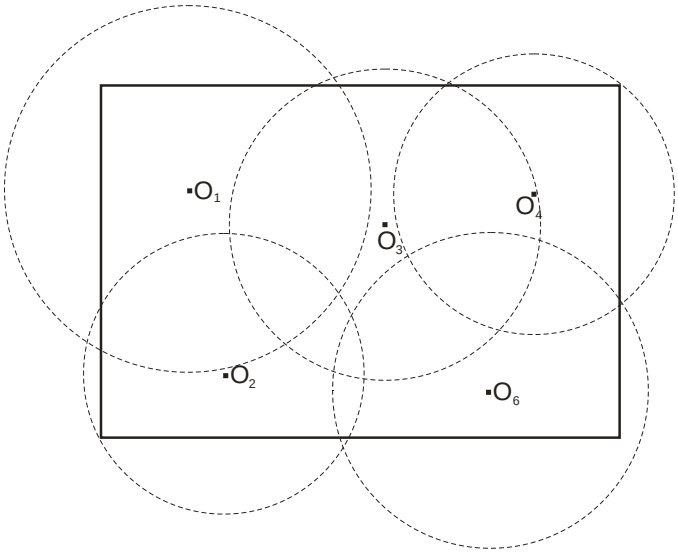


Figure 6.2. A virtual 1-set-cover

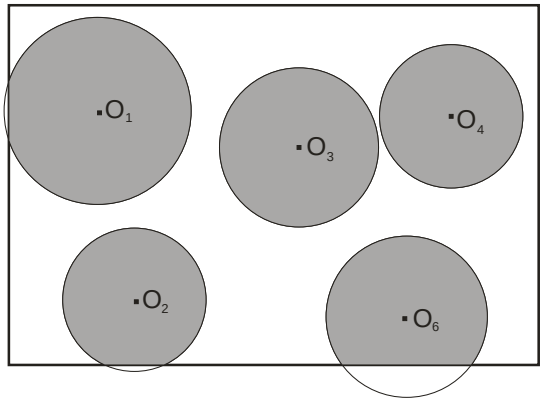


Figure 6.3. Final resulting real set cover

First, our framework has the original algorithms to find a 1-set-cover in virtual network. For example, in Figure 6.2 assume that the virtual 1-set-cover that the original algorithm found is

C_1={ s_1,s_2,s_3,s_4, s_6}. The corresponding *real* set cover $\overline{C_1}$ of C_1 is shown in Figure 6.3. It is proved later that the real set cover can α-cover the area.

Even in Section 6.4.3.1, Section 6.4.3.2, Section 6.4.4.2, our strategies are presented in the form of pseudo-codes (Pseudocode 7, Pseudocode 8, Pseudocode 9, Pseudocode 10), they actually are the general guidelines on how to modify an original algorithm (for complete coverage) to get the desired algorithm for α-coverage. That means when modifying an original algorithm, every step does not have to be exactly the same as shown in pseudo-codes:

- Even in the pseudo-codes, the strategies explicitly create virtual networks and let algorithm $\mathbb{A}$ run on that network, the process to modify the algorithm $\mathbb{A}$ has not always to be the same. For example, instead of creating a separate virtual network, our framework suggest that the original algorithm should be modified in the way that if a line of codes of the algorithm $\mathbb{A}$ has references to sensors' sensing ranges, the strategies just have to replace the original sensing ranges with corresponding virtual sensing ranges.
- Since there exists complete-coverage algorithms that do not generate all set covers at once, their resulting algorithms (for the α-coverage problem) do not have to explicitly create set of all α-set-covers and return them as shown in pseudo-codes. Most of the time, when the algorithm $\mathbb{A}$ generates a set cover of a virtual network, the algorithm $\mathbb{A}$ will be modified in the way that right before a set cover is returned (that set cover is still virtual set cover at that point of time), it will be converted to real set cover.

6.4.3. General strategies

With general strategies, the original algorithm (for complete-coverage) are only allowed to find virtual set covers which completely cover the area (1-set-covers). The final resulting set

covers are usually *α-RSC* of those virtual 1-set-covers. Although the constraint of allowing original algorithm to find only virtual 1-set-cover limits the increasing of network lifetime, it, on other hand, gives numerous fancy characteristics for the final resulting set covers (which are α-set-covers). Those characteristics are analyzed in detail in Section 6.4.3.3. We propose two separate general strategies which are designed for different kind of WSNs: fixed-sensing-range WSNs and adjustable-sensing-range WSNs which we discuss in more detail in next sub-sections.

6.4.3.1. General strategy for fixed-sensing-range WSNs - *Strategy G-1*

Strategy G-1: Assume that a sensor s_i has a fixed sensing range R_i for $i=1,..,n$. The general guidelines of *Strategy G-1* is given in Pseudocode 7.

Pseudocode 7: **Strategy G-1 - General strategy for fixed-sensing-range WSNs**

Require: The network $\mathcal{S}$ is fixed-sensing-range WSN

1: Create α-virtual network $\mathcal{S}^{\alpha}$ of network $\mathcal{S}$ ► *Where a sensor s_i has a sensing range of* $\frac{R_i}{\sqrt{\alpha}}$

2: Execute algorithm $\mathbb{A}$ on the network $\mathcal{S}^{\alpha}$

3: The result is set of l set covers and their schedules: $C=\{C_j, t_j\}_{j=1..l}$ ► *Each set cover C_j is a VSC$^{\alpha}$*

4: ► *Create set covers where sensors use their original (real) ranges*

5: **for** $j=1$ **to** l

6: Create set cover $\overline{C_j}=\{s_i(R_i) \mid s_i \in C_j\}$

7: **end for**

8: **return** $\{\overline{C_j}, t_j\}_{j=1..l}$

This strategy works for both types of original algorithms $\mathbb{A}$:

Type 1: Algorithm $\mathbb{A}$ is designed for fixed-sensing-range WSNs. *Strategy G-1* first runs the original algorithm $\mathbb{A}$ on α-virtual network $\mathcal{S}^{\alpha}$ (where sensor s_i has a sensing range of $\frac{R_i}{\sqrt{\alpha}}$) to get the set of the virtual set covers C_j and their schedules t_j. For a sensor $s_i \in C_j$, its virtual sensing

range in C_j is $\frac{R_i}{\sqrt{\alpha}}$. At line 6 of Pseudocode 7, from virtual set cover C_j, the set cover $\overline{C_j}$ is created of which each sensor s_i has a sensing range of R_i - its real, pre-defined sensing range. It can be seen that each set cover C_j is a VSC^{α} and set cover $\overline{C_j}$ is *α-RSC* of C_j. The final result is the set of set covers $\overline{C_j}$ (where sensors use their real sensing ranges R_i) and their schedule t_j for *j=1..l*.

Type 2: Algorithm $\mathbb{A}$ is designed for adjustable-sensing-range WSNs. *Strategy G-1* runs original algorithm $\mathbb{A}$ on *α*-virtual network S^{α} (where sensor s_i has the *maximum* sensing range of $\frac{R_i}{\sqrt{\alpha}}$) to get the set of $\{C_j, t_j\}$ pairs. Denote the sensing range of a sensor s_i in C_j as R_i^j and thus $R_i^j \le \frac{R_i}{\sqrt{\alpha}}$. In the final resulting set covers $\overline{C_j}$, which is not a *α-RSC* of C_j, the sensor $s_i \in \overline{C_j}$ uses its pre-defined sensing range R_i (i.e., the virtual sensing ranges assigned by the algorithm $\mathbb{A}$ are intentionally ignored) and schedule t_j. Apparently, it is not practical to execute type-2 algorithms on fixed-sensing-range WSNs. We just mention this case here to show the universality of our framework. We highly recommend not to apply *Strategy G-1* for adjustable-sensing-range algorithms on fixed-sensing-range WSNs.

Since *Strategy G-1* forces sensors of the final resulting set covers $\overline{C_j}$ to use their pre-defined, original sensing ranges (without considering the sensors' sensing ranges in the virtual set cover C_j), the following lemmas are directly derived:

Lemma 6.1. *Given a real number α where 0<α<1 and a fixed-sensing-range sensor network, for any original algorithm (complete-coverage algorithm) as input, the α-coverage algorithm of Strategy G-1 generates feasible set covers.*

Lemma 6.2. *The sensing range of a sensor in a final result's set cover ($\overline{C_j}$) is no smaller than $\sqrt{\alpha}$ times of its sensing range in the corresponding virtual set cover (C_j).*

6.4.3.2. General strategy for adjustable-sensing-range WSNs - *Strategy G-2*

Strategy G-2: We assume that each sensor s_i's sensing range has a pre-defined upper bound $MaxR_i$. *Strategy G-2* is given in Pseudocode 8 and works for both types of original algorithms:

Pseudocode 8: **Strategy G-2 - General strategy for adjustable-sensing-range WSNs**

Require: The network S is adjustable-sensing-range WSN
1: Pick an real number β where $\alpha \le \beta$
2: **if** Algorithm $\mathbb{A}$ is designed for fixed-sensing-range WSN **then**
3: Calculate fixed-β-virtual network $S^{\gamma}_{\text{fixed}}$ of network S ► *Where a sensor s_i has a sensing range of* $\frac{MaxR_i}{\sqrt{\beta}}$
4: Execute algorithm $\mathbb{A}$ on the network $S^{\gamma}_{\text{fixed}}$
5: **else**
6: ► *Algorithm $\mathbb{A}$ is designed for adjustable-sensing-range WSN*
7: Calculate β-virtual network S^{β} of network S ► *Where a sensor s_i has the maximum sensing range of* $\frac{MaxR_i}{\sqrt{\beta}}$
8: Execute algorithm $\mathbb{A}$ on the network S^{β}
9: **end if**
10: The result is set of l set covers and their schedules: $C=\{C_j, t_j\}_{j=1..l}$ ► *Each set cover C_j is a VSC$^{\beta}$*
11: ► *Sensors adjust their sensing ranges to $\sqrt{\alpha}$ times of their current ranges*
12: **for** $j=1$ **to** l
13: Create α-*RSC* $\overline{C_j}$ of C_j
14: **end for**
15: **return** $\{C_j, t_j\}_{j=1..l}$

Type 1: The original algorithm $\mathbb{A}$ is designed for fixed-sensing range WSNs. *Strategy G-2* runs the original algorithm $\mathbb{A}$ on fixed-β-virtual network $\mathcal{S}^{\gamma}_{\text{fixed}}$ where $\alpha \leq \beta$. That is, let the algorithms $\mathbb{A}$ execute on the virtual network where a sensor s_i has the fixed sensing range of $\frac{MaxR_i}{\sqrt{\beta}}$. The result of this execution is a set of virtual set covers C_j, each C_j is a VSC^{β}. Line 13 of Pseudocode 8 adjusts the sensing range of a sensor s_i from $\frac{MaxR_i}{\sqrt{\beta}}$ in VSC^{β} C_j to $R_i = \sqrt{\alpha} \times \frac{MaxR_i}{\sqrt{\beta}} = \frac{\sqrt{\alpha}}{\sqrt{\beta}} \times MaxR_i$ in α-RSC $\overline{C_j}$. Since $\alpha \leq \beta$, thus $R_i \leq maxR_i$ which makes the final resulting set covers $\overline{C_j}$ feasible. If we choose $\beta=\alpha$ then this strategy becomes *similar* as *Strategy G-1*.

Type 2: The original algorithm $\mathbb{A}$ is designed for adjustable-sensing-range WSNs. *Strategy G-2* runs the original algorithm $\mathbb{A}$ on β-virtual network $\mathcal{S}^{\beta}$, $\alpha \leq \beta$ (where sensor s_i has the *maximum* sensing range of $\frac{MaxR_i}{\sqrt{\beta}}$) to decide sensors' sensing ranges and their schedules, i.e., a set of $\{C_j, t_j\}$ pairs. Let R_i^j be the sensing range that $\mathbb{A}$ decides for sensor s_i in a virtual set cover C_j, clearly $R_i^j \leq \frac{MaxR_i}{\sqrt{\beta}}$. At line 13 of Pseudocode 8, the sensing range of sensor s_i is adjusted to $\sqrt{\alpha}R_i^j$ in the final resulting set cover $\overline{C_j}$. Because $\alpha \leq \beta$, $\sqrt{\alpha}R_i^j \leq \sqrt{\alpha} \times \frac{MaxR_i}{\sqrt{\beta}} \leq MaxR_i$. Thus, the set cover $\overline{C_j}$ is a feasible set cover.

Based on the explanation above, the following lemmas hold:

Lemma 6.3. *Given a real number α where 0<α<1 and an adjustable-sensing-range sensor network, for any original algorithms (complete-coverage algorithms), Strategy G-2 generates feasible set covers whose sensors' sensing ranges are no larger than their pre-defined upper bounds.*

Lemma 6.4. *The sensing range of a sensor of a final result's set cover ($\overline{C_j}$) is exactly $\sqrt{\alpha}$ times of its sensing range in the corresponding virtual set cover (C_j).*

6.4.3.3. Analysis

Before proving the correctness of the proposed two general strategies, we first state a supporting lemma:

Lemma 6.5. *Given a real number α where 0<α<1, an infinite monitored area, and a sensor network of which sensors' sensing regions are disks, assume the network can completely cover the monitored area. If the sensing ranges of all the active sensors shrink down with the ratio $\sqrt{\alpha}$ then the network with the new sensing ranges assignment can α-cover the monitored area.*

Proof: Denote *M* as the monitored area. Let *A* be the region that the original network can cover, i.e., *A* is union region of sensing regions of all active sensors. Clearly $M \subseteq A$. Let $\overline{A}$ be the region that the network after sensors' sensing ranges are shrunk down by ratio of $\sqrt{\alpha}$. Let $\delta = \sqrt{\alpha}$, then from *δ-compression* theorem (Theorem 6.5 of Section 6.6) we have: $\left\|\overline{A}\right\| \geq \delta^2 \|A\| = \alpha \|A\| \geq \alpha \|M\|$ which proves the correctness of this lemma ■

Theorem 6.1. (Proof of correctness) *Given a real number α where 0<α<1 and an infinite monitored area. Our proposed two general strategies work correctly.*

Proof: To prove the correctness of the two general strategies, we need to prove following two conditions:

a) The final resulting set covers are feasible: According to Lemma 6.1 and Lemma 6.3, the resulting set covers of α-coverage algorithms of both strategies are feasible.

b) The final resulting set covers can α-cover the area: According to Lemma 6.2 and Lemma 6.4, the sensing ranges of sensors of final (real) set covers are *no smaller* than $\sqrt{\alpha}$ times of those of the corresponding virtual set covers. Since the virtual set covers can completely cover the area, then according to Lemma 6.5, the final set covers can α-cover the area. Notice that this proof is for infinite monitored area. The simulation (Section 6.5) shows that it strongly holds for finite monitored area as well.

The resulting set covers of α-coverage algorithms of the both strategies are feasible and can α-cover the monitored area, which proves the correctness of the both strategies. ■

For the partial coverage problem, it is always desirable to evaluate how uniformly the network α-covers the area. A good metric for such evaluation is *Sensing Void Distance* of which the definition is adapted from [LIU05] and [WUA08] as follows:

Definition 6.6. (Sensing Void Distance - SVD) *Sensing Void Distance is the* maximum *distance from a point that is not covered by any active sensors to the* nearest *point that is covered by an active sensor.*

Based on the definition of SVD, the upper bound of SVD of any resulting α-coverage algorithms of the two general strategies is given in following theorem:

Theorem 6.2. *Let R_{max} be the maximum sensing range a sensor may have. That is, for fixed-sensing-range WSNs, R_{max} is the largest sensing range of all the sensors and for adjustable-*

sensing-range WSNs, R_{max} is the largest sensing range of maximum sensing ranges of all the sensors. For any original algorithms, the SVDs of α-coverage algorithms created by two general strategies are bounded by the following values:

- $SVD \le \frac{1-\sqrt{\alpha}}{\sqrt{\alpha}} R_{\max}$ *for Strategy G-1.*
- $SVD \le \frac{1-\sqrt{\alpha}}{\sqrt{\beta}} R_{\max}$ *for Strategy G-2.*

Proof: Assume that *P* is the uncovered point that the distance from it to the nearest covered point is SVD. Since the point *P* is covered by the virtual network (*α*-virtual network of *Strategy G-1*, *β*-virtual network or fixed-*β*-virtual network of *Strategy G-2*), there exists a sensor *s* such that *P* is covered by s(R_{vir}) where R_{vir} is the virtual sensing range of *s* in the virtual network. In Figure 6.4, the inner solid circle is the *s*'s sensing region when *s* operates in the final resulting network (the network for *α*-coverage), i.e., *s* works with its real sensing range R_{real}. The outer dashed circle (whose radius is R_{vir}) is the sensing region of *s* in the virtual network. According to our two general strategies, we have $R_{real} \ge \sqrt{\alpha}\, R_{vir}$.

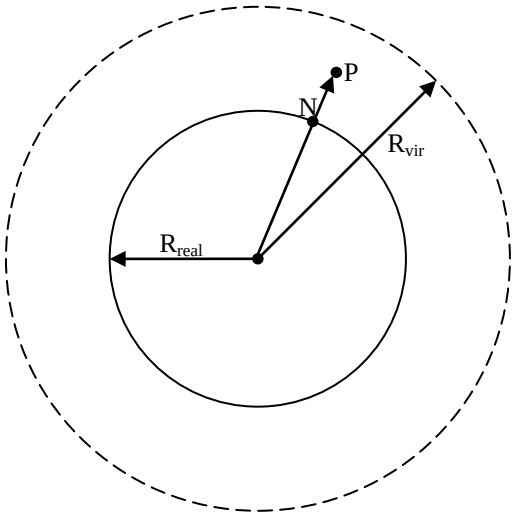


Figure 6.4. Sensing Void Distance

Denote $|sP|$ as the Euclidean distance from sensor s to point P, then $|sP| \leq Rvir$. Assume the line sP intersects with the inner circle at point N, clearly $|sN| = R_{real}$.

Because the way we choose point P, we have: $SVD \leq |NP| = |sP| - |sN| = |sP| - R_{real} \leq R_{vir} - R_{real} \leq R_{vir} - \sqrt{\alpha}\ R_{vir} = R_{vir}\ (1-\sqrt{\alpha}\ )$.

For *Strategy G-1*, the original algorithms work with α-virtual-network, thus $R_{vir} \leq \frac{R_{\max}}{\sqrt{\alpha}}$.

Therefore $SVD \leq \frac{1-\sqrt{\alpha}}{\sqrt{\alpha}} R_{\max}$.

For *Strategy G-2*, the original algorithms work with β-virtual-network, thus $R_{vir} \leq \frac{R_{\max}}{\sqrt{\beta}}$.

Therefore $SVD \leq \frac{1-\sqrt{\alpha}}{\sqrt{\beta}} R_{\max}$. ■

For complete coverage ($\alpha=1$) we have SVD=0. Clearly, our SVD bound is more meaningful than the bound given in [WUA08] which is $r_{tmax}+r_{smax}-r_{smin}$ (this bound is the same for all values of α) where r_{tmax}, r_{smax}, r_{smin} are maximum communication range, maximum sensing range, minimum sensing range, respectively. For *Strategy G-2*, the larger the value of β, the smaller the value of SVD. Thus, the user may use the large value of β to get a better coverage uniformity. However, if β too large, virtual network might not be able to completely cover the monitored area since the sensors' sensing ranges of virtual network are too small.

Theorem 6.3. *The time complexity of any resulting α-coverage algorithm of two general strategies is the same as that of the original algorithm.*

Proof: The proposed general strategies essentially adjust the source code of an original algorithm in the way the resulting algorithm can α-cover the area.

To achieve that, the general strategies do two things:

1) Let the original algorithm work with virtual network where every sensor *s* has its virtual sensing range. In this step, the framework (which also includes two extended strategies) only considers any reference to sensors' sensing range and merely changes a sensor's real sensing range to a virtual sensing range. Thus, our framework does not change the time complexities of the original algorithms since it merely adds a simple calculation operation (which is a division) to the original algorithms.

2) Convert the virtual set covers to real set covers. Essentially, before returning the results (i.e., set covers with schedules), the two general strategies simply adjust piece of codes in an original algorithm such that instead of returning the virtual sensing ranges, it returns real sensing ranges which are usually $\sqrt{\alpha}$ times of the virtual sensing ranges. That also means that the set of *real* set covers is 1-1 corresponding with the set of *virtual* set covers.

Thus, the time complexity of the resulting algorithm of two proposed general strategies is the same as that of original algorithm. ■

6.4.4. Extended strategies

6.4.4.1. The basic idea of the extended strategies

Usually, a complete-coverage algorithm is designed to find out set of 1-set-covers. The algorithm keeps generating set covers until some explicit or implicit *stopping criteria* are satisfied. Among those *stopping criteria*, the most important stopping criteria are *coverage stopping criteria* which force the algorithm to stop when some lower bounds of coverage quality are violated. In case of complete-coverage algorithms, a *coverage stopping criterion* is usually

the condition that the network cannot completely cover the area even with the maximum support of all the network's resources. For example, for fixed sensing range WSNs, a *coverage stopping criterion* may be the condition that even turning on all the *live* sensors, the network still cannot completely cover the monitored area. In the case of adjustable sensing range WSNs, a *coverage stopping criterion* may be the condition that even turning on all the *live* sensors and those sensors' sensing ranges are adjusted to the maximum, the network still cannot cover the whole monitored area. We use the term *non-coverage stopping criteria* to refer to all the stopping criteria other than *coverage stopping criteria*. An example of a *non-coverage stopping criterion* for a coverage algorithm that also maintains connectivity (of set covers) is the state when the network is disconnected, i.e., none of *live* sensors could communicate with each other. A coverage algorithm would have several *coverage stopping criteria* and might have some *non-coverage stopping criteria*. Usually, the algorithm stops as soon as any of them is satisfied.

As a matter of fact, most of complete-coverage algorithms can easily be modified such that they can be executed without any *coverage stopping criterion*, for example, they might keep generating set covers until there are no live sensors left in the network. We name *"exhaustible algorithms"* for such family of the complete-coverage algorithms since they can be *extended* (i.e., modified) to exhaustively utilize the network resources. Example of algorithms in *exhaustible algorithms* family are the ones in [VUC06], [VUT07], [VUC09], [VUT09]. Since most of coverage algorithms can be freed from *coverage stopping criteria*, the family of *exhaustible algorithms* is a large subfamily of complete coverage algorithms.

One drawback of general strategies is that they only generate *α-RSC*s of VSC^{α} (*Strategy G-1*) or VSC^{β} (*Strategy G-2*) where VSC^{α} or VSC^{β} is 1-set-covers of virtual networks. However, it is highly possible that even a virtual set cover C_j of virtual network could not 1-cover the area

(i.e., C_j is neither a VSC^{α} nor a VSC^{β}), its corresponding real set cover $\overline{C_j}$ (i.e., $\overline{C_j}$ is *α-RSC* of C_j) can still α-cover the area. The simulations in Section 6.5 can help confirm this claim.

Network lifetime can be extended if the drawback of general strategies is overcome. For family of *exhaustible algorithms*, that can easily be obtained if the original algorithms are modified in the way that the *coverage stopping criteria* is removed from them. That is main idea of how we extend two general strategies *Strategy G-1* and *Strategy G-2* which we discuss in more detail next

6.4.4.2. Two extended strategies description

In this section, we describe extended strategies *Strategy E-1* and *Strategy E-2* which are extensions of general strategies *Strategy G-1* (Section 6.4.3.1) and *Strategy G-2* (Section 6.4.3.2), respectively. Consequently, *Strategy E-1* is for fixed-sensing-ranges WSNs, for family of *exhaustible algorithms* and is given in Pseudocode 9. In Pseudocode 9, we assume that a sensor s_i has a fixed sensing range R_i for $i=1,..,n$.

Pseudocode 9: Strategy E-1 - Extended strategy (of G-1) for fixed-sensing-range WSNs and family of *exhaustible algorithms*

Require: The network $\mathcal{S}$ is fixed-sensing-range WSN and $\mathbb{A}$ is an *exhaustible algorithm*

1: Create α-virtual network $\mathcal{S}^{\alpha}$ of network $\mathcal{S}$ ► *Where a sensor s_i has a sensing range of* $\frac{R_i}{\sqrt{\alpha}}$

2: Execute algorithm $\mathbb{A}$ on the network $\mathcal{S}^{\alpha}$ *without* any *coverage stopping criteron*

3: The result is set of l set covers and their schedules: $\mathcal{C}=\{C_j, t_j\}_{j=1..l}$ ► *A set cover C_j is not always a VSC^{α}*

4: ► *Create set covers where sensors use their original (real) ranges*

5: **for** $j=1$ **to** l

6: Create set cover $\overline{C_j}=\{s_i(R_i) \mid s_i \in C_j\}$

7: **end for**

8: **return** $\{(\overline{C_j}, t_j) \mid \overline{C_j}$ can α-cover the area $\}$

Strategy E-2 is for adjustable-sensing-range WSNs, for family of *exhaustible algorithms* and is given in Pseudocode 10. Also, in Pseudocode 10 we assume that each sensor s_i's sensing range has a pre-defined upper bound $maxR_i$. Similar to *Strategy G-1* and *Strategy G-2*, *Strategy E-1* and *Strategy E-2* also work for both types of original algorithms. We also highly recommend not to apply *Strategy E-1* for adjustable-sensing-range algorithms on fixed-sensing-range WSNs.

Pseudocode 10: **Strategy E-2 - Extended strategy (of G-2) for adjustable-sensing-range WSNs and family of *exhaustible algorithms***

Require: The network S is adjustable-sensing-range WSN and $\mathbb{A}$ is an *exhaustible algorithm*

1: Pick a real number β where $\alpha \leq \beta$

2: **if** Algorithm $\mathbb{A}$ is designed for fixed-sensing-range WSN **then**

3: Calculate fixed-β-virtual network $S^{\gamma}_{\text{fixed}}$ of network S ► *Where a sensor s_i has a sensing range of* $\frac{MaxR_i}{\sqrt{\beta}}$

4: Execute algorithm $\mathbb{A}$ on the network $S^{\gamma}_{\text{fixed}}$ *without* any *coverage stopping criterion*

5: **else**

6: ► *Algorithm $\mathbb{A}$ is designed for adjustable-sensing-range WSN*

7: Calculate β-virtual network S^{β} of network S ► *Where a sensor s_i has the maximum sensing range of* $\frac{MaxR_i}{\sqrt{\beta}}$

8: Execute algorithm $\mathbb{A}$ on the network S^{β} *without* any *coverage stopping criterion*

9: **end if**

10: The result is set of l set covers and their schedules: $C=\{C_j, t_j\}_{j=1..l}$ ► *A set cover C_j is not always a VSC$^{\beta}$*

11: ► *Sensors adjust their sensing ranges to $\sqrt{\alpha}$ times of their current ranges*

12: **for** $j=1$ **to** l

13: Create α-*RSC* $\overline{C_j}$ of C_j

14: **end for**

15: **return** $\{(\overline{C_j}, t_j) \mid \overline{C_j}$ can α-cover the area $\}$

Let $\mathbb{A}$ be an algorithm of family of *exhaustible algorithms*. With two extended strategies, the original algorithm $\mathbb{A}$ first generates set of 1-set-covers (which are *VSC*$^{\alpha}$ for *Strategy E-1* and

VSC$^{\beta}$ for *Strategy E-2*) for the virtual network until it cannot find any more such set covers. So far, the extended strategies work exactly the same as their corresponding general strategies. To easily explain various phenomena of extended strategies in simulation results (Section 6.5.1), we term the *point of time* that no more 1-set-cover can be found in virtual network as *premature death* since the algorithms are supposed to stop here and the network are supposed to die here as being done by general strategies. However, instead of stopping here (because of *coverage stopping criteria*), the extended strategies modify the original algorithm $\mathbb{A}$ in the way that the algorithm $\mathbb{A}$ ignores all *coverage stopping criteria*, and thus algorithm $\mathbb{A}$ keeps generating set covers (which are now no longer 1-set-covers) until any *non-coverage stopping criterion* is satisfied or there is no *live* sensors left. The final set covers of the resulting algorithms of two extended strategies are only the ones that α-cover the area.

Again, the two pseudo-codes are essentially the guidelines to specify algorithms modification. Notice that even in the *Strategy E-1* and *Strategy E-2*, we clearly state that the resulting α-coverage algorithms return only real set covers that could α-cover the monitored area (i.e., α-set covers), as already mentioned in Section 6.4.2, algorithms modification process does not always has to be exactly step-by-step as shown in the pseudo-codes. Thus the resulting algorithms of two extended strategies do not always have to verify the coverage quality for each set cover they generate since this verification is not trivial. In fact, the resulting α-coverage algorithms keep creating real set covers without knowing which set cover is the first one that cannot α-cover the area. Usually, the point of time that such first cover is founded is considered as the point of time the network dies, thus the resulting algorithms of extended strategies might not always know lifetime of the network it creates. This, however, is not important since the foremost objective of this work is to extend network lifetime.

Since the returning set covers of resulting algorithms of extended strategies are only α-set-covers, the correctness of two extended strategies is a consequence of the correctness of the two general strategies. Therefore, the following theorem holds:

Theorem 6.4. (Proof of correctness) *Given a real number α where $0<\alpha<1$. Our proposed two extended strategies work correctly.*

Also, for some types of algorithms such as algorithms that work in rounds, the time complexity of resulting algorithms is guaranteed to be the same as that of the original algorithms. But the extended strategies cannot guarantee the same property for all algorithms of *exhaustible algorithms* family. Notice that before *premature death*, the two extended strategies work just the same as their two corresponding general strategies, thus the set covers generated by resulting algorithms of two extended strategies should have all characteristics of those of general strategies. For example, the set covers that are generated before *premature death*, would have bounded SVD as derived by Theorem 6.2. Those set covers also possess various other nice characteristics of set covers of resulting algorithms of general strategies as discussed later in simulation section.

6.4.5. Advantages of our framework

Followings are the main advantages of our four strategies:

1) The framework is very universal in the sense that it can convert almost any original algorithm for complete coverage into a one for α coverage. The universality of our framework originates from its simplicity as the conversion process involves only the adjusting (simple multiplication or division) of the sensors' sensing ranges by some well defined ratio.

2) For each of two general strategies, the SVD is bounded by a constant and that bound is the same for all original algorithms, i.e., the SVD bound does not depend on the characteristics of the original algorithms.

3) Except the coverage quality, the resulting α-coverage algorithms preserve all the characteristics of the original algorithms.

4) To the best of our knowledge, up to date, applying our framework on a distributed and parallel complete-coverage algorithm is the only way to design a distributed and parallel algorithm for the α-coverage problem in the literature.

5) To provide complete or partial coverage at different points of times (e.g., rainy and dried seasons in forest fire detection application), a network can run a single algorithm instead of running different algorithms for different points of times. Just tuning a single parameter α suffices.

6) The existing algorithms cannot provide different coverage qualities at different sub-regions without relying on some sorts of clustering methods. With our framework, this can easily be achieved by letting sensors at critical sub-regions use larger α than the others.

6.5. Simulation

6.5.1. Simulation settings

In this section, we evaluate the framework through extensive simulations. For energy consumption, we employ the quadratic energy model as in Section 3.3.2.1. That is, when a sensor is in active mode, the amount of energy consumption for 1 unit of time is proportional to square of the sensor's sensing range. Please refer to Eq. 4.1 (Section 4.6) for the formal formula.

Table 6.1. Partial-coverage framework simulation setting

Network size	*800m × 400m*
Energy range	*100 → 120 mJoules*
κ	*8000*
Maximum sensing range	100^m

The simulation configuration is given in the Table 6.1. To evaluate the efficiency of our strategies, we employ two fully distributed and parallel algorithms: DESK (CHAPTER 3) which is designed for fixed-sensing-range WSNs and SPIDT (CHAPTER 4) which is designed for adjustable-sensing-range WSNs. Both DESK and SPIDT belong to *exhaustible algorithms* family. We conduct our simulations for six combinations: DESK+Strategy G-1 (i.e., using DESK as an input for *Strategy G-1*), DESK+Strategy G-2, DESK+Strategy E-1, DESK+Strategy E-2, SPIDT+Strategy G-2, SPIDT+Strategy E-2. For short, from now on we denote them by DESK+ G-1, DESK+ G-2, DESK+ E-1, DESK+ E-2, SPIDT+ G-2, SPIDT+ E-2, respectively.

For each comparison metric, we compare the results of each combination with that of the original algorithm. The comparison data is the average results of 50 times running the simulation on 50 completely random networks. Since both DESK and SPIDT are parallel algorithms, it is not fair and not necessary to compare our work with any existing non-parallel algorithm.

6.5.2. Simulation results

We have conducted the simulations for various values of α, however, due to the page limitation we only show the plots of simulation results in their full size for $\alpha = 0.5$. We also explain in detail only for results where $\alpha = 0.5$. The plots of simulation results for $\alpha = 0.3$, $\alpha = 0.7$, $\alpha = 0.9$, $\alpha = 0.95$ are given in reduced resolution. For most of the plots, the value of β is set to *1*.

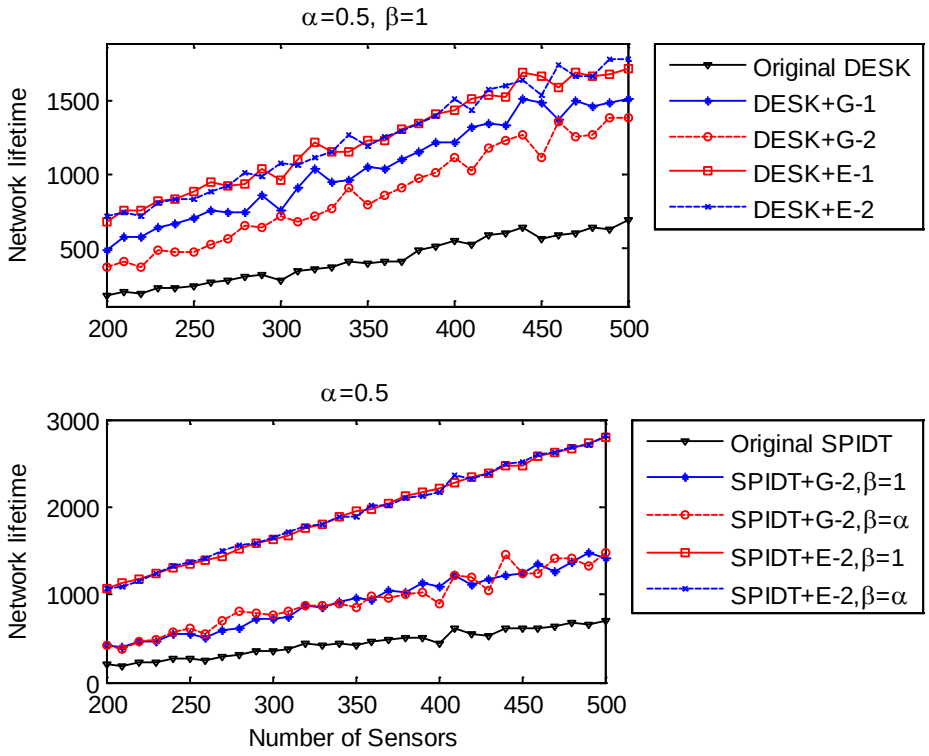


Figure 6.5. Network lifetime for $\alpha=0.5$, $\beta=1$.

In Figure 6.5, we evaluate network lifetime when the coverage quality is decreased to 50% ($\alpha=0.5$). For resulting α-coverage algorithms of two general strategies, network lifetime is the duration until the virtual network cannot completely cover the area. For resulting α-coverage algorithms of two extended strategies, network lifetime is the interval until the α-coverage algorithms cannot find any more α-set-covers. Understanding the lifetime this way and due to the quadratic energy model employed in simulations implementation, network lifetime of any α-coverage algorithm is theoretically *at least* $\frac{1}{\alpha}$ times of that of the corresponding original algorithm. This fact can also be witnessed in Figure 6.5 and in Figure 6.6. Also, in Figure 6.5, we measure network lifetime for SPIDT+G-2 and SPIDT+E-2 with different values of β. It appears that β only has effect on network lifetime of resulting algorithms of SPIDT+G-2 and the

extended strategies eliminate the dissimilarity that the two general strategies make on their resulting α-coverage algorithms.

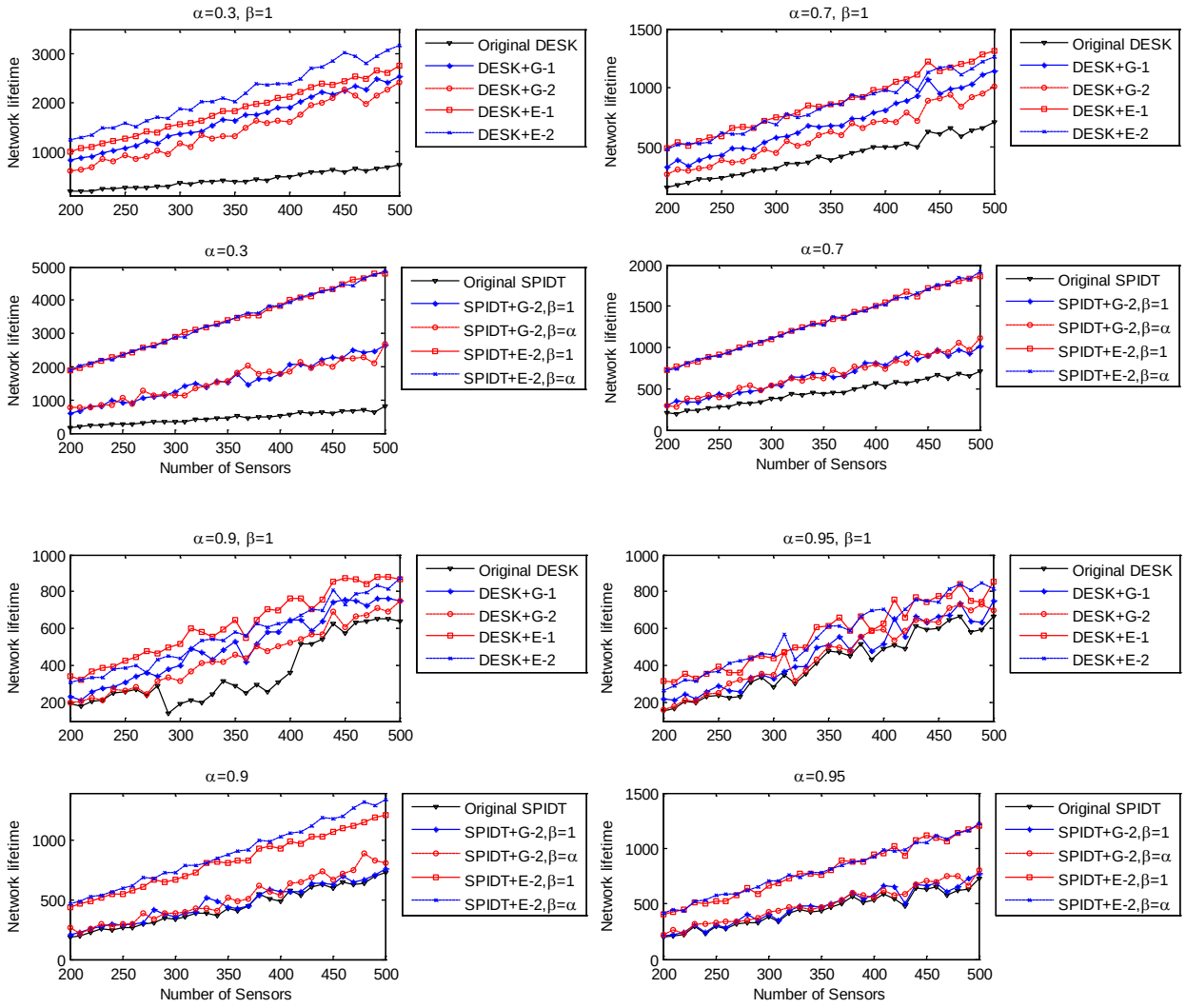


Figure 6.6. Network lifetime for $\alpha=0.3$, $\alpha=0.7$, $\alpha=0.9$, $\alpha=0.95$, and $\beta=1$

In the Figure 6.6, we measure network lifetime ratio for $\alpha=0.3$, $\alpha=0.7$, $\alpha=0.9$, and $\alpha=0.95$. Due to the extension of ignoring all *coverage stopping criteria* of the two extended strategies, network lifetimes of the algorithms of two extended strategies are significantly longer than that of two corresponding general strategies.

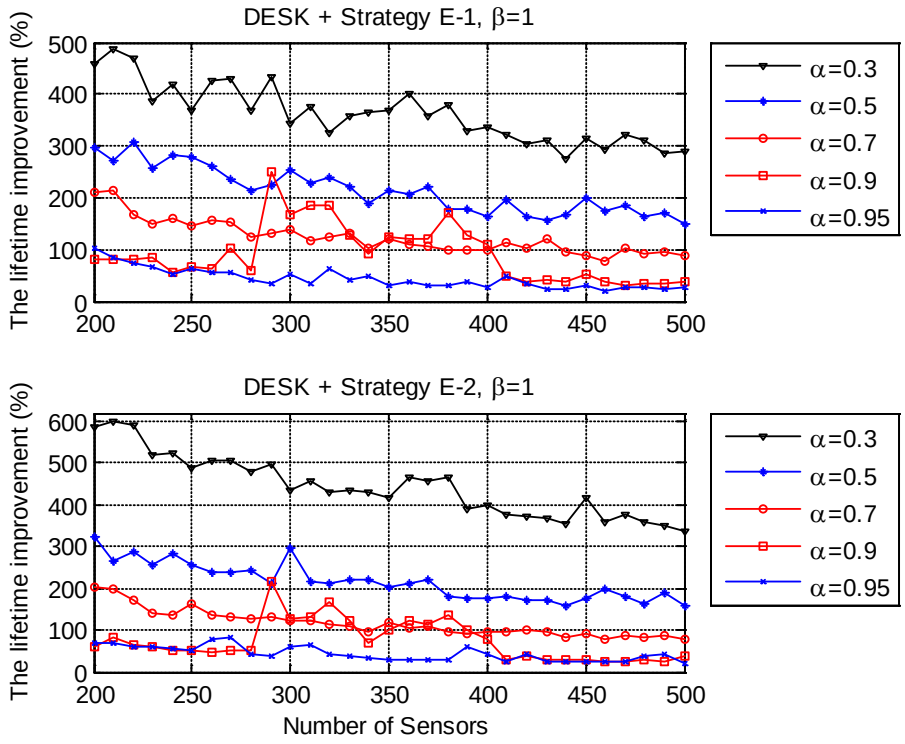


Figure 6.7. Network lifetime improvement for DESK+ E-1 and DESK+ E-2, $\beta=1$

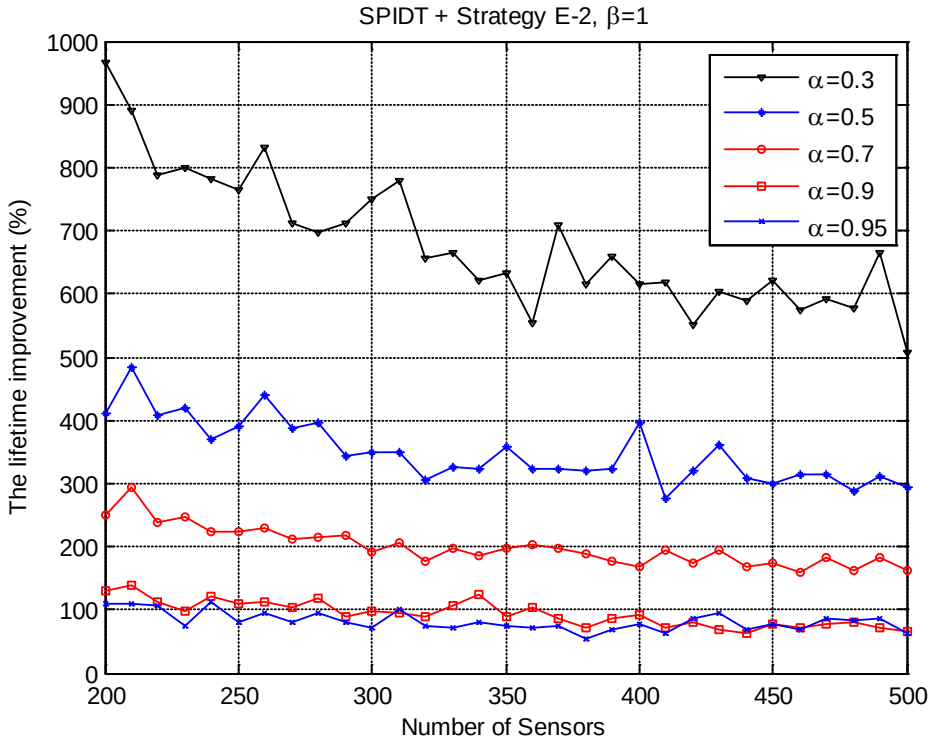


Figure 6.8. Network lifetime improvement for SPIDT+E-2, $\beta=1$

To illustrate that, in Figure 6.7 and Figure 6.8, we measure the *lifetime improvement* for extended strategies with different values of α when we trade the coverage quality for network lifetime. By definition, *lifetime improvement* is the percentage of *extra* lifetime that the resulting α-coverage algorithms of *strategy E-1* and *E-2* gain (in compared with the original algorithms) when they decrease the coverage quality to α-portion. It can be seen that the *lifetime improvement* decreases as the number of sensors increases. This is because the more the number of the sensors, the higher the density of the network. Remember that both DESK and SPIDT are originally designed to find 1-set-covers, even they are modified to ignore all *coverage stopping criteria*, when the virtual network cannot completely cover the area, both DESK and SPIDT still try to find virtual set cover that could cover *as much* portion of the area as they can. This causes coverage redundancy for the final real set covers. Thus, after the virtual network cannot completely cover the area (i.e., after *premature death*), the higher the density, the more redundancy of the real set covers. That clarifies the decreasing of the *lifetime improvement*.

Figure 6.9 and Figure 6.10 show two factors of coverage quality. Figure 6.9 shows *percentage of covered area* - the percentage of the area which is covered over the whole area. Figure 6.10 shows the *area coverage level*. If we define the *coverage level* of each point in the monitored area as the number of active sensors covering that point, then the *area coverage level* is the *average* of *coverage level* of all points in the monitored area. In the figures, most of the curves for both *percentage of covered area* and *area coverage level* are nearly horizontal lines, which means those measurements of most combinations are roughly constant. Especially, *percentage of covered area* and *area coverage level* of all combinations of G-1 and G-2 are almost constant regardless the size (i.e., the number of sensors) of the network which shows the stability of general strategies for both original algorithms (DESK and SPIDT).

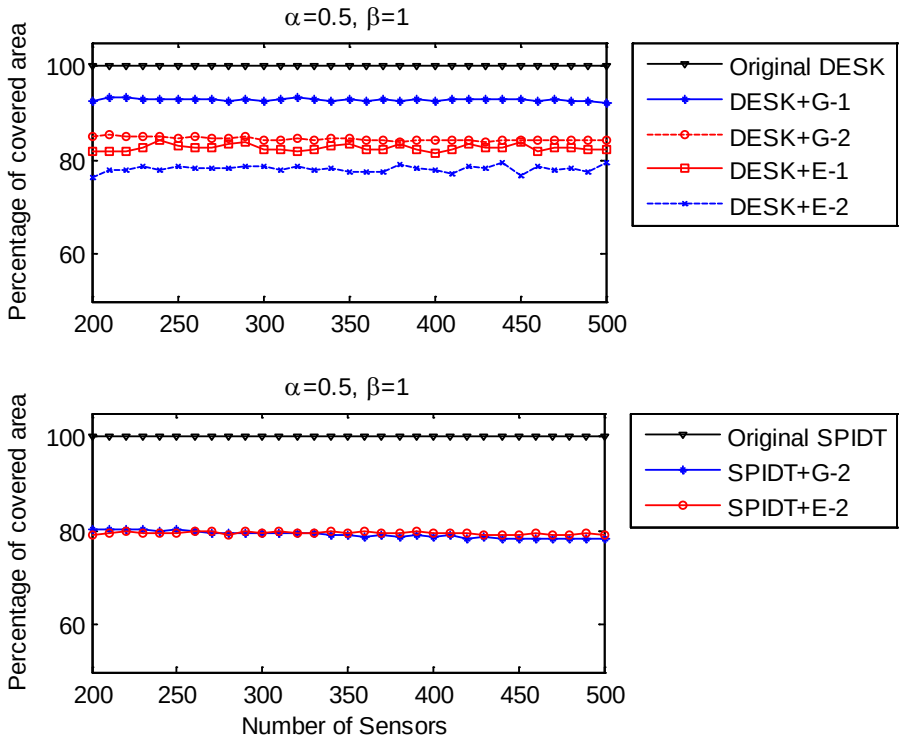


Figure 6.9. Percentage of covered area for $\alpha=0.5, \beta=1$

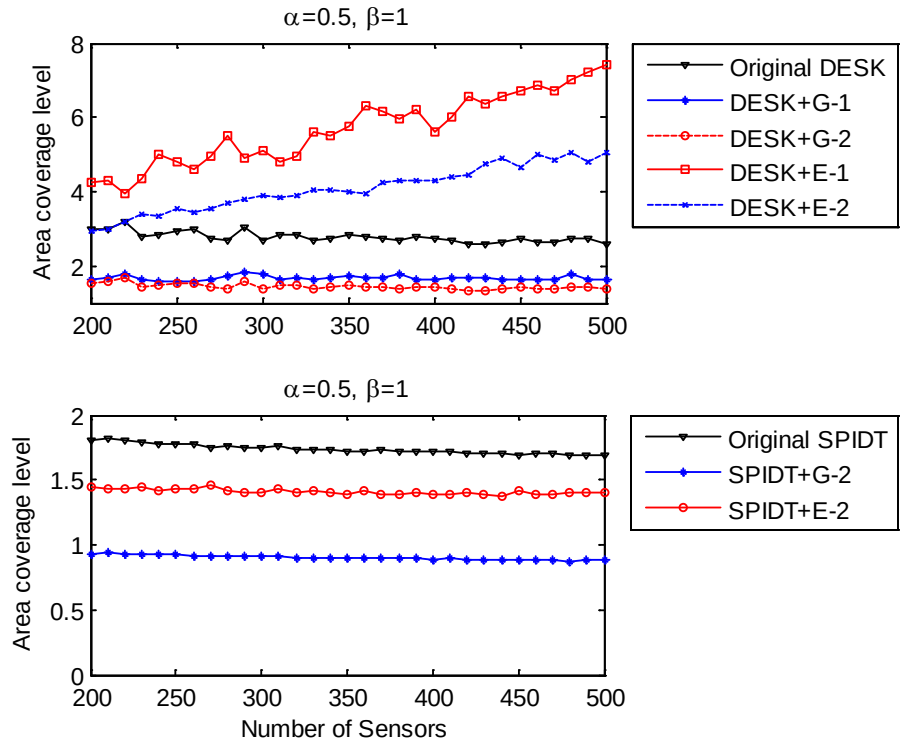


Figure 6.10. Area coverage level for $\alpha=0.5, \beta=1$

Even though the resulting algorithms of two general strategies make some redundancy on the *percentage of covered area* (Figure 6.9), they greatly reduce the redundancy of the *area coverage level* (Figure 6.10). Taking the *area coverage level* into consideration, the redundancy of *percentage of covered area* of general strategies is acceptable since the *area coverage level* alone has done the job of reducing the coverage quality by 50% ($\alpha=0.5=50\%$). That redundancy is originated from the dense nature of a WSN and the fact that the coverage level in the central region is usually very high.

After the original algorithm cannot find any more 1-set-cover for virtual network, i.e., after *premature death*, the effort to generate a virtual set covers that cover as much portion of the monitored area as possible causes such redundancy for virtual set covers which in turn cause redundancy for the final real set covers. That is why the two extended strategies make more redundancy than their two corresponding general strategies as it can be observed in Figure 6.10. Especially, the *area coverage level* of DESK+E-1 and DESK+E-2 is much larger than that of the original DESK. This is another indication which illustrates the *"exhaustible"* characteristics of DESK and SPIDT after the extended strategies remove *coverage stopping criteria* from them.

Figure 6.11 and Figure 6.12 show the *percentage of covered area* and *area coverage level* for different values of α.

Notice that the minimum value of *y*-axis of each plot in Figure 6.9 (for $\alpha=0.5$) and in Figure 6.11 (for $\alpha=0.3$, $\alpha=0.7$, $\alpha=0.9$, $\alpha=0.95$) is $\alpha*100$ which is the minimum coverage ratio required by the corresponding simulation. It can be observed that *percentage of covered area* of each of combination is always higher than the required minimum ratio which confirms that Theorem 6.1 also strongly holds for finite monitored area.

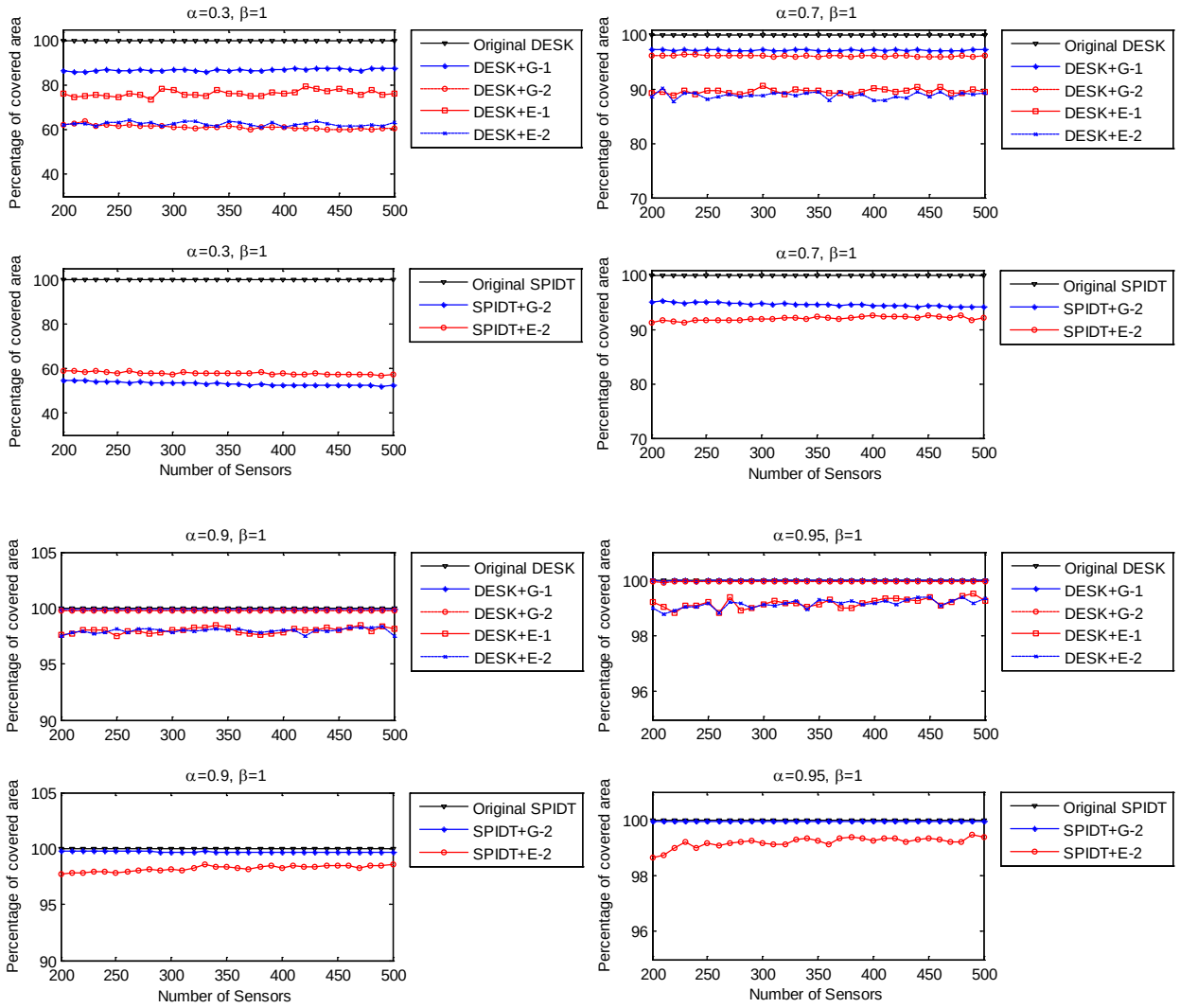


Figure 6.11. Percentage of covered area for *α=0.3*, *α=0.7*, *α=0.9*, *α=0.95*, and *β=1*

Due to the rectangle shape of monitored area and circle shape of sensors' sensing regions, any set cover that completely cover the monitored area have to *redundantly* cover the area, in other words, they actually cover more than 100% of the area. As a result, even we try to reduce the coverage quality of that set cover, the resulting set cover also *redundantly* α-cover the area. That is the reason why in Figure 6.9 and Figure 6.11, the percentages of covered area of α-coverage algorithms are usually somewhat higher than the value of α.

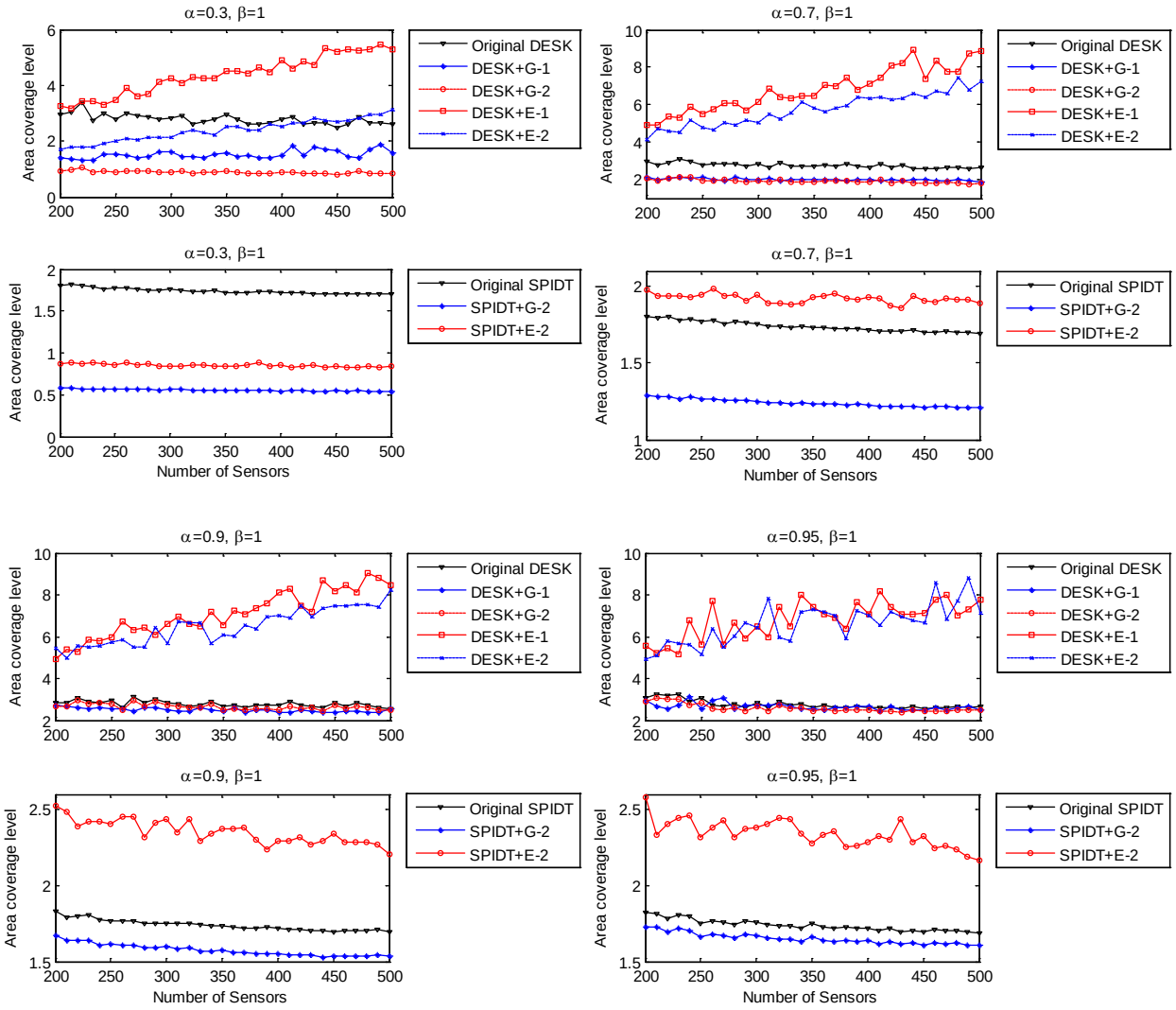


Figure 6.12. Area coverage level for *α=0.3*, *α=0.7*, *α=0.9*, *α=0.95*, and *β=1*

As it can be observed that the pattern of simulation results for *area coverage level* for *α=0.3*, *α=0.7*, *α=0.9*, *α=0.95* is similar to that for *α=0.5*. Since SPIDT are able to adjust sensors' sensing ranges to minimize coverage redundancy, for the same size of network, the coverage level of *α*-coverage algorithm of SPIDT is quite smaller than that of DESK – a fixed-sensing-range algorithm. It can also be witnessed that for different values of *α*, the *area coverage level* are not much different. This shows that our framework performs somewhat independently on *α*.

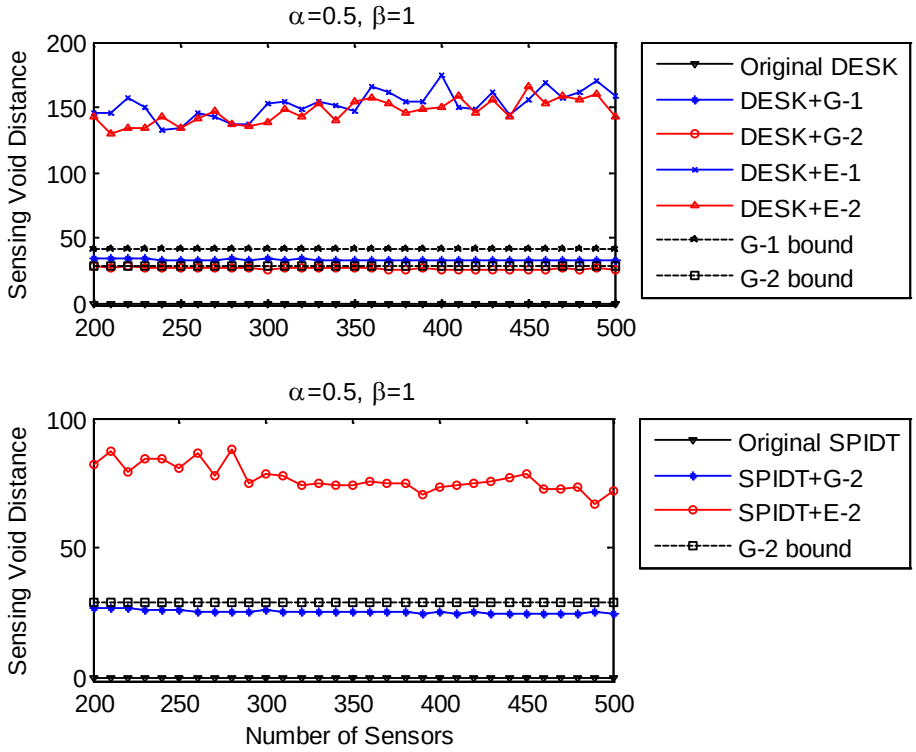


Figure 6.13. Sensing Void Distance for $\alpha=0.5$, $\beta=1$

In Figure 6.13, the SVD of resulting algorithm of each combination is measured. We also show the SVD's theoretical upper bounds derived by Theorem 6.2 for both general strategies. As it can be witnessed in the figure, the SVD of each α-coverage algorithm of two general strategies is really close to the corresponding upper bound which confirms that the SVD bounds derived in Theorem 6.2 are very tight. As expected, the SVD for Original DESK or Original SPIDT is 0. For two general strategies, the SVDs of α-coverage algorithms are approximately constant which is another indication to show the stability and advantage of our two general strategies. We can claim that for a particular original complete-coverage algorithm, the resulting α-coverage algorithm of the proposed general strategies can guarantee almost the same degree of coverage uniformity the area for all sizes of networks.

Notice that the constant upper bound of SVD given in Theorem 6.2 no longer holds for resulting α-coverage algorithms of extended strategies due to the fact that the virtual set covers do not always completely cover the whole area, especially after *premature death*, thus the SVDs of corresponding real set covers are no longer bounded by constants. This explains why the curves for SVDs of both extended strategies are far above those of corresponding general strategies and even above the upper bounds of two general strategies.

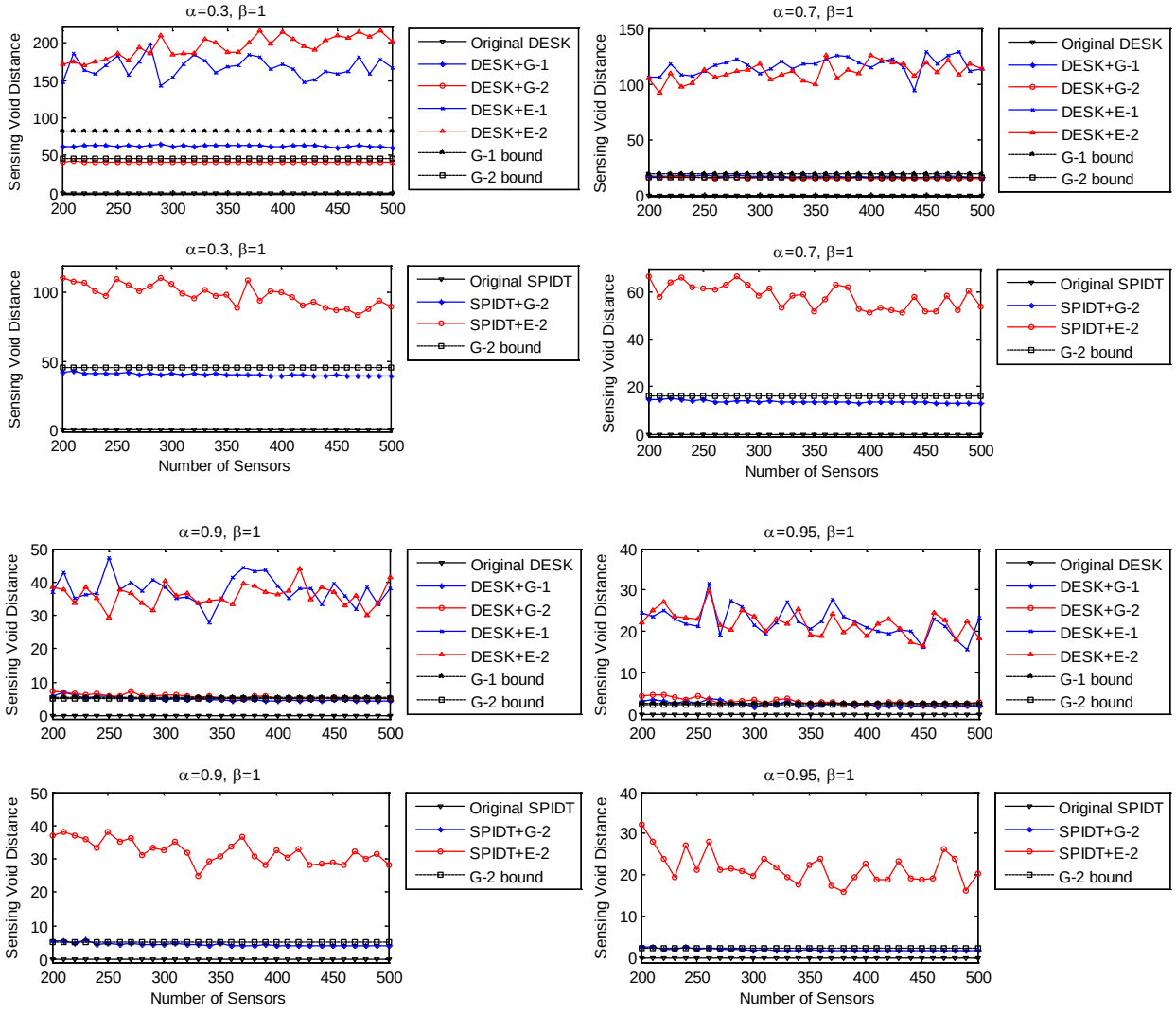


Figure 6.14. Sensing Void Distance for α=0.3, α=0.7, α=0.9, α=0.95, and β=1

In Figure 6.14, the *Sensing Void Distance* for different values of α is presented.

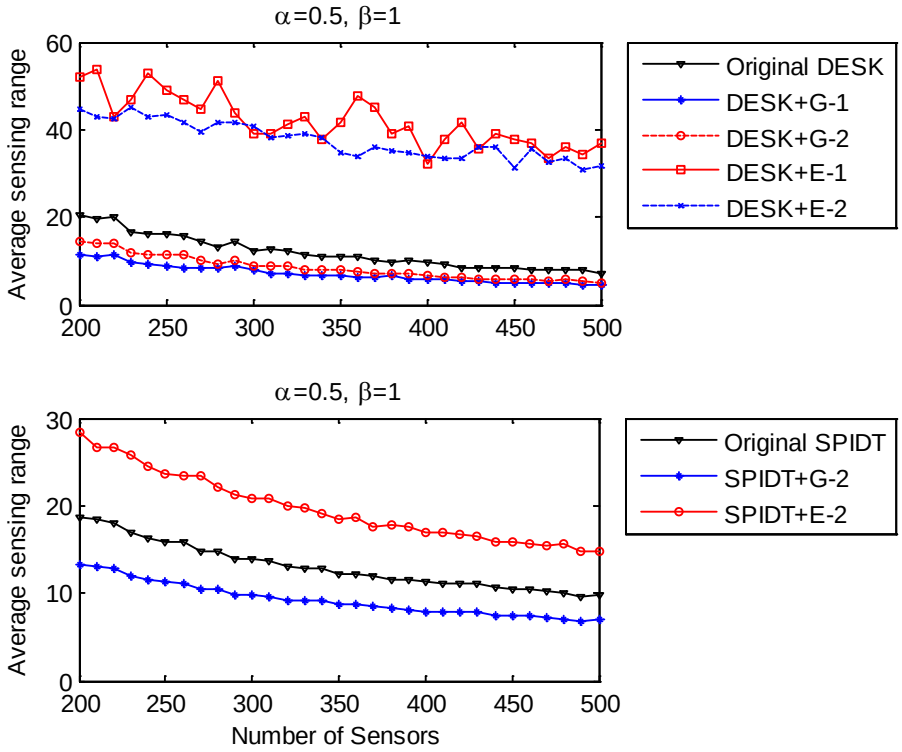


Figure 6.15. Average sensing range for *α=0.5, β=1*

Figure 6.15 shows the average sensors' sensing range. The average sensing range is calculated as the average sensing range of all live sensors (i.e., sensors that have not consumed all of their energies) with the convention that the sleep sensors have sensing range of 0. Theoretically, for *Strategy G-2*, *β=1* and *Strategy G-1*, the average sensing range of an *α*-coverage algorithm is roughly $\sqrt{\alpha}$ times of that of the corresponding original algorithm. This theoretical fact can be confirmed by Figure 6.15. Again, the effort of extended strategies to discover the best set cover which could cover as large portion of the area as possible cause the unnecessarily large average sensing range after the *premature death*. In other word, the *"exhaustible"* property of resulting algorithms of extended strategies make both their *network lifetime* and *average sensing range* longer than that of their corresponding general strategies

even though these two parameters usually vary in opposite directions (i.e., if *network lifetime* increases, then usually *average sensing range* decreases and vice versa).

From all of the above simulation results, it can be concluded that our two general strategies performs stably and independently on the size of the network. Even the two extended strategies can greatly extend network lifetime in compared with two general strategies, they also cause lots of redundancies. Due to the "*exhaustible"* characteristic of their resulting α-coverage algorithms, they always try to utilize all the network resources to obtain as high coverage quality as possible. Especially, after *premature death*, i.e., after the virtual network cannot be able to completely cover the monitored area when there is no virtual 1-set-cover can be found, the nature of DESK and SPIDT find 1-set-cover has them to find set covers that have the best coverage quality at that point of time. This effort causes redundancies in each set cover they generate which explains the higher values of extended strategies (in compared with their corresponding general ones) in all simulating measurements we conduct. Nonetheless, ultimate objective to maximize network lifetime also greatly achieved.

6.6. δ-compression theorem

We dedicate this section to prove an important theorem which helps to prove the correctness of our framework. We first introduce some common notations and supporting definitions.

6.6.1. Notations

Notation *C(O,R)* means a circle *C* has radius *R* and is centered at the point *O*. The region enclosed by the circle *C* is denoted *D(C)* (*D* stands for disk). We use notation $\|A\|$ to denote the area of region *A*. For a line XY , $|XY|$ is its length.

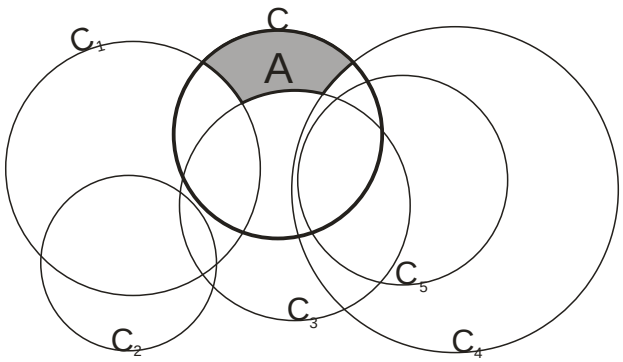
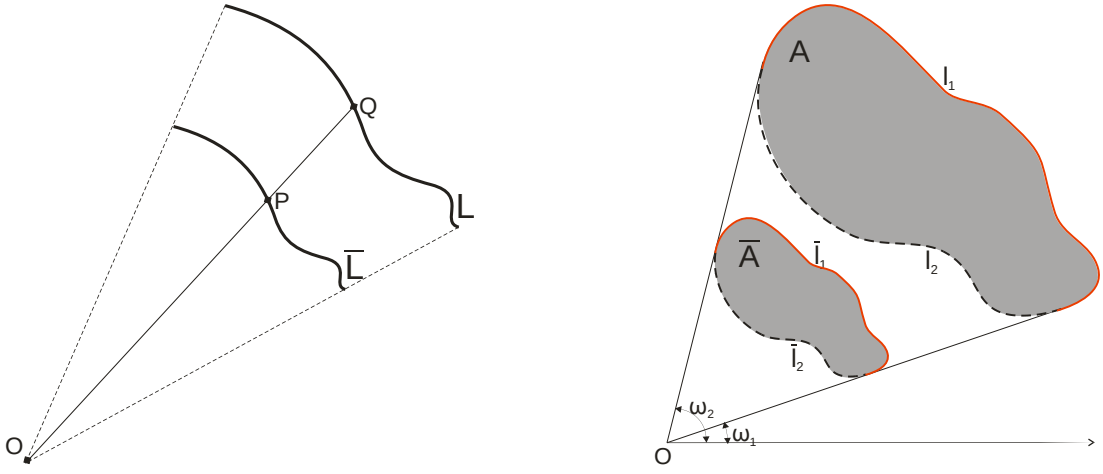


Figure 6.16. An example of private region and neighbors

For 2 two-dimensional regions $\mathcal{A}$ and $\mathcal{B}$, we write $\mathcal{A} \subseteq \mathcal{B}$ if for every point $P \in \mathcal{A}$, we also have $P \in \mathcal{B}$. In other words, $\mathcal{A}$ is fully covered by $\mathcal{B}$. We write $\mathcal{A} \not\subset \mathcal{B}$ if there exists a point $P \in \mathcal{A}$, but $P \notin \mathcal{B}$. For example, as shown in Figure 6.16, we have $D(C_5) \subseteq D(C_4)$, $D(C_5) \not\subset D(C)$.

We write $\mathcal{A}$-$\mathcal{B}$ to denote a (difference) region which consists of all point $P \in \mathcal{A}$ but $P \not\subset \mathcal{B}$.

6.6.2. Supporting definitions



(a) *(O,δ)-compression* for a curve (b) *(O,δ)-compression* of a region

Figure 6.17. *(O,δ)*-compression

Definition 6.7. (Neighbors) *In the plane, given a circle C and l other circles C_1,C_2, …,C_l., a circle C_i (i=1…l) is said to be neighbor of C if C and C_i overlap.*

For example, in Figure 6.16, C_1,C_3,C_4,C_5 are all the neighbors of *C* while C_2 is not.

Definition 6.8. (Private region) *The region of disk D(C) that is not covered by any of its neighbors is called* private region.

For example, the shaded region *A* in Figure 6.16 is private region of *D(C)*. Mathematically, $A = D(C) - \bigcup_{i=1}^{5} D(C_i)$.

Definition 6.9. ((O,δ)-compression) *Given a real number δ where 0 < δ < 1, a point O, a curve L, and a convex region A in the plane, we define:*

- *(O,δ)-compression of curve L is the curve $\bar{L}$ consisting of all point P such that $\frac{|OP|}{|OQ|} = \delta$ for every point $Q \in L$.*
- *(O,δ)-compression of area A is the area $\bar{A}$ consisting of all point P such that $\frac{|OP|}{|OQ|} = \delta$ for every point $Q \in A$.*

As in Figure 6.17.a, $\bar{L}$ is *(O,δ)-compression* of *L*. In Figure 6.17.b, $\bar{A}$ is *(O,δ)-compression* of *A*. The following theorem could easily be proved:

Lemma 6.6. *Given a real number δ where 0 < δ < 1, a point O and two convex regions A and $\bar{A}$ in the plane. Regions A and $\bar{A}$ are enclosed by curves L, $\bar{L}$, respectively. $\bar{A}$ is (O,δ)-compression of A if and only if $\bar{L}$ is (O,δ)-compression of L.*

Lemma 6.7. *If $\bar{\mathcal{A}}$ is (O,δ)-compression of $\mathcal{A}$, then $\|\bar{\mathcal{A}}\| = \delta^2\|\mathcal{A}\|$ where $\|\bar{\mathcal{A}}\|$ and $\|\mathcal{A}\|$ are the areas of $\mathcal{A}$ and $\bar{\mathcal{A}}$, respectively.*

Proof: We have two cases:

Case 1: Point *O* lies outside region $\mathcal{A}$, i.e., $O \notin \mathcal{A}$ .We partition the curves enclosed regions $\mathcal{A}$ to two curves l_1,l_2 as in Figure 6.17.b. Similarly, we partition the curves enclosed regions $\bar{\mathcal{A}}$ into two curves $\bar{l}_1, \bar{l}_2$. Clearly, $\bar{l}_1, \bar{l}_2$ are *(O,δ)-compression* of l_1,l_2, respectively.

If we consider regions under the polar coordinate, then from [ERI09] we have:

$$\|\bar{\mathcal{A}}\| = \frac{1}{2}\int_{\omega_1}^{\omega_2}\left(\bar{l}_1^{\,2} - \bar{l}_2^{\,2}\right)d\omega = \frac{1}{2}\int_{\omega_1}^{\omega_2}\left((\delta l_1)^2 - (\delta l_2)^2\right)d\omega = \frac{1}{2}\delta^2\int_{\omega_1}^{\omega_2}\left(l_1^2 - l_2^2\right)d\omega = \delta^2\|\mathcal{A}\|$$

Case 2: Point *O* lies inside region $\mathcal{A}$, i.e., $O \in \mathcal{A}$. Let *l* and $\bar{l}$ be curves enclosing regions $\mathcal{A}$ and $\bar{\mathcal{A}}$, respectively. We have:

$$\|\bar{\mathcal{A}}\| = \frac{1}{2}\int_0^{2\pi}\left(\bar{l}\right)^2 d\omega = \frac{1}{2}\int_0^{2\pi}\left(\delta l\right)^2 d\omega = \frac{1}{2}\delta^2\int_0^{2\pi} l^2 d\omega = \delta^2\|\mathcal{A}\|$$

For all the cases, we always have $\|\bar{\mathcal{A}}\| = \delta^2\|\mathcal{A}\|$ ■

6.6.3. The δ-compression theorem

Theorem 6.5. (δ-compression theorem) *In the plane, given n circles $C_i(O,R_i)$ for i=1..n and disks $D(C_i)$, i=1..n may overlap. If we "shrink" all n circles by radius ratio of δ where 0<δ<1, we will have n new circles $C_i^*(O_i, R_i^*)$ where $R_i^* = \delta R_i$ for i=1..n. If we denote $\mathcal{A} = \bigcup_{i=1}^{n} D(C_i)$ and $\mathcal{A}^* = \bigcup_{i=1}^{n} D(C_i^*)$, then $\|\bar{\mathcal{A}}\| \geq \delta^2\|\mathcal{A}\|$.*

Proof: We prove this theorem by induction on the number of the disks:

Basic step: n = 1: Trivial

Inductive hypothesis: Assume the theorem holds for $k = n - 1$ for some $n \geq 2$.

Inductive step: Prove for $k = n$. Among n circles, assume circle $C(O,R)$ is the one has the *smallest* radius. Denote other $n-1$ circles $C_i(O_i,R_i)$, $i=1..n-1$. Thus, we have $R \leq R_i$, $i=1..n-1$.

Let:

$$\mathcal{A}_{n-1} = \bigcup_{i=1}^{n-1} D(C_i) \tag{6.2}$$

$$\mathcal{A} = \left[\bigcup_{i=1}^{n-1} D(C_i)\right] \bigcup D(C) = \mathcal{A}_{n-1} \bigcup D(C) \tag{6.3}$$

Intuitively, $\mathcal{A}$ is the union regions of n disks $D(C)$ and $D(C_i)$, $i=1..n$-1 while $\mathcal{A}_{n-1}$ is union region of $n-1$ disks (not including $D(C)$).

After n circles shrink with ratio of δ we have n new circles $C_i^*(O_i, R_i^*)$, $i=1..n-1$ and $C^*(O,R^*)$ where $R_i^* = \delta R_i$ and $R^* = \delta R$.

Let:

$$\mathcal{A}_{n-1}^* = \bigcup_{i=1}^{n-1} D(C_i^*) \tag{6.4}$$

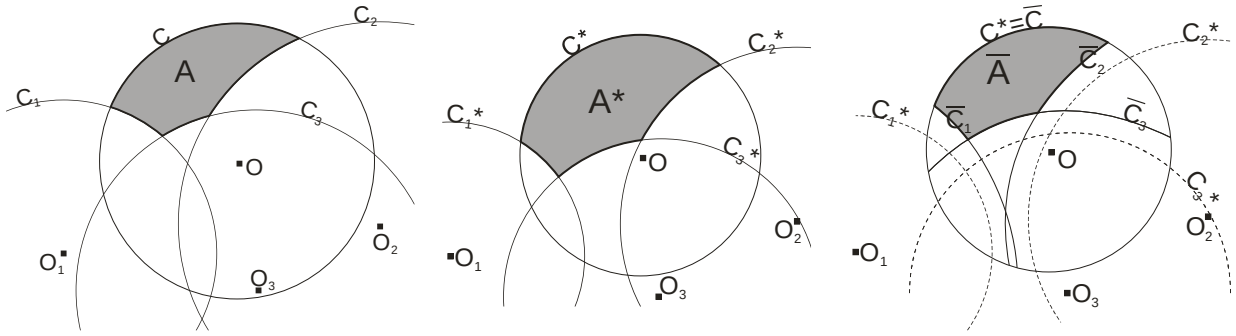
$$\mathcal{A}^* = \left[\bigcup_{i=1}^{n-1} D(C_i^*)\right] \bigcup D(C^*) = \mathcal{A}_{n-1}^* \bigcup D(C^*) \tag{6.5}$$

Intuitively, $\mathcal{A}^*$ is the union regions of n disks $D(C^*)$ and $D(C_i^*)$, $i=1..n$-1 while $\mathcal{A}_{n-1}^*$ is union region of $n-1$ disks (not including $D(C^*)$).

By the conventions in Eq. 6.2, 6.3, 6.4, and 6.5, the inductive hypothesis can be rewritten as $\left\|\mathcal{A}^*_{n-1}\right\| \geq \delta^2 \left\|\mathcal{A}_{n-1}\right\|$ and we need to prove $\left\|\overline{\mathcal{A}}\right\| \geq \delta^2 \left\|\mathcal{A}\right\|$. There are two cases:

Case 1. If before shrinking, the disk *D(C)* is completely covered by all of its neighbors, i.e., $D(C) \subseteq \bigcup_{i=1}^{n-1} D(C_i) = \mathcal{A}_{n-1} \Rightarrow \mathcal{A} = \mathcal{A}_{n-1} \bigcup D(C) = \mathcal{A}_{n-1}$. Then we are done because $\left\|\mathcal{A}^*\right\| \geq \left\|\mathcal{A}^*_{n-1}\right\| \geq \delta^2 \left\|\mathcal{A}_{n-1}\right\| = \delta^2 \left\|\mathcal{A}\right\|$.

Case 2. Otherwise, the disk *D(C)* is not fully covered by all of its neighbors



(a) The *private region* before shrinking

(b) The *private region* after shrinking

(c) Compressed *private region*

Figure 6.18. Private regions

Without loss of generality, assume that before shrinking *C* intersects with *l* other circles $C_i(O_i,R_i)$ for *i=1..l*. For example, in Figure 6.18a we have *l = 3*. By assumption, we have $R \leq R_i$ for *i=1..l*.

Let A be the *private region* of disk D(C). Then the region can be covered by all *n* disks is $\mathcal{A} = \mathcal{A}_{n-1} + A$. Hence $\left\|\mathcal{A}\right\| = \left\|\mathcal{A}_{n-1}\right\| + \left\|A\right\|$.

After all the disks shrink as in Figure 6.18b, let A^* be the new *private region* of disk *D(C*)*. After shrinking, the region that could be covered by all the *n* disks is $\mathcal{A}^* = \mathcal{A}^*_{n-1} + A^*$. Hence $\|\mathcal{A}^*\| = \|\mathcal{A}^*_{n-1}\| + \|A^*\|$.

We need to prove $\|\mathcal{A}^*\| \geq \delta^2 \|\mathcal{A}\| \Leftrightarrow \|\mathcal{A}^*_{n-1}\| + \|A^*\| \geq \delta^2 (\|\mathcal{A}_{n-1}\| + \|A\|)$. From *inductive hypotheses*, we have $\|\mathcal{A}^*_{n-1}\| \geq \delta^2 \|\mathcal{A}_{n-1}\|$. So, we only need to prove $\|A^*\| \geq \delta^2 \|A\|$ to complete our proof.

In Figure 6.18c, we introduce a concept of *compressed private region*. We create *(O,δ)-compression* $\overline{C_i}$ of circles C_i (original circle before shrinking shown in Figure 6.18a) for *i=1..l*. where *O* is the center of circle *C*, C^*. It is necessary to emphasize that $C^* \equiv \overline{C}$, i.e., C^* and $\overline{C}$ are the same circle. We only consider the border portions of $\overline{C_i}$ which are inside the disk $D(\overline{C})$. From now on, we also use notation $\overline{C_i}$ to denote those portions. The new *private region*, which is denoted by $\overline{A}$, created by $D(\overline{C})$ and curves $\overline{C_i}$ is called *compressed private region*.

Each curve $\overline{C_i}$ partitions the disk *D(C*)* into two sub-regions (halves), we name $\overline{A_i}$ for the sub-region that contains the *compressed private region* $\overline{A}$. That is, $\overline{A_i} = D(C^*) - D(\overline{C_i})$. We always have $\overline{A} \subset \overline{A_i} \subseteq D(C^*)$.

Similarly, we define $A_i^* = D(C^*) - D(C_i^*)$. If $D(C_i^*)$ overlaps *D(C*)*, circle C_i^* also partitions the disk *D(C*)* into two halves, A_i^* is the half that contains the *private region* A^*. Otherwise, if $D(C_i^*)$ does not overlap *D(C*)*, then $A_i^* = D(C^*)$. Based on those conventions and notations, we have:

$$\overline{A} = \bigcap_{i=1}^{l} \overline{A_i} \tag{6.6}$$

and

$$A^* = \bigcap_{i=1}^{l} A_i^* \tag{6.7}$$

It is easy to see that $\overline{A}$ is *(O,δ)-compression* of *A*, by Lemma 6.7, we have $\left\|\overline{A}\right\| = \delta^2 \|A\|$. Thus, instead of proving $\left\|A^*\right\| \geq \delta^2 \|A\|$, we are going to prove $\left\|A^*\right\| \geq \left\|\overline{A}\right\|$.

We claim that for *i = 1..l,* $\overline{A_i} \subseteq A_i^*$. In other words, $\overline{A_i}$ is completely inside A_i^*. This and Eq. 6.6 and Eq. 6.7 lead to the consequence that $\overline{A} \subseteq A^*$, i.e., $\overline{A}$ is completely inside A^*. Hence, we have $\left\|\overline{A}\right\| \leq \left\|A^*\right\|$. Thus, if the claim is proved, our theorem is consequently proved.

Now, we prove our claim. It is easy to see that if $D(C_i)$ overlaps $D(C)$ then $\overline{C_i} \neq \varnothing$. Thus for *i = 1..l,* we have $\overline{C_i} \neq \varnothing$. Based on that fact and since C^* has the *smallest* radius among all disks, for a particular *i, 1≤ i≤ l,* there exist only two sub-cases:

<u>*Case 2.1*</u>: C_i^* does not intersect $D(C^*)$. Then, $\overline{A_i} \subseteq D(C^*) = A_i^*$. Hence $\overline{A_i} \subseteq A_i^*$.

<u>*Case 2.2*</u>: C_i^* does intersect $D(C^*)$. We will prove that $\overline{A_i} \subseteq A_i^*$ i.e., in Figure 6.19 the shaded region ($\overline{A_i}$) is completely inside the thick-bordered region (A_i^*). To prove this, we only have to prove $\overline{C_i}$ is completely inside A_i^*.

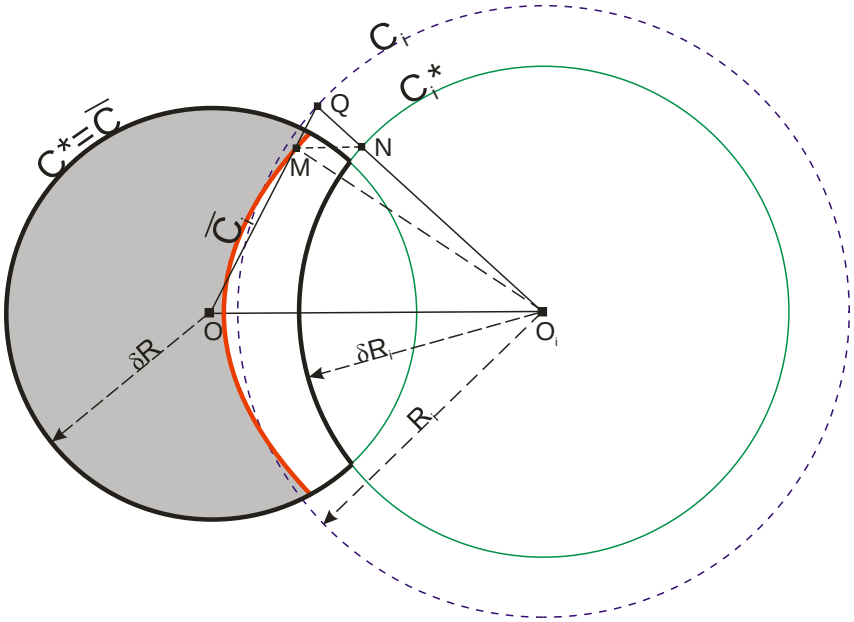


Figure 6.19. Claim proof

Consider an arbitrary point $M \in \overline{C_i}$, we will prove that $M \in A_i^*$. Assume that line OM intersects C_i at Q and line O_iQ intersect C_i^* at N. We denote $\angle MNO_i$ as the angle made by two rays NM and NO_i. We have $\frac{|OM|}{|OQ|} = \delta = \frac{|O_iN|}{|O_iQ|}$. Thus line MN is parallel with line OOi. Consequently:

$$\angle MNO_i + \angle NO_iO = \pi \tag{6.8}$$

Since $\overline{C_i}$ is inside $D(C^*)$ and $M \in \overline{C_i}$, thus $M \in D(C^*) \Rightarrow |OM| \leq \delta R$. Hence $|OQ| = \frac{|OM|}{\delta} \leq R \leq R_i = |O_iQ|$. Consider triangle QOO_i: since $OQ \leq O_iQ$, $\angle QO_iO \leq \angle QOO_i \Rightarrow$ $\angle QO_iO \leq \pi/2$ (because $\angle QO_iO + \angle QOO_i + \angle OQO_i = \pi$). That and Eq. 6.8 mean $\angle MNO_i \geq \pi/2$. Since $\angle MNO_i + \angle NMO_i + \angle NO_iM = \pi \Rightarrow \angle NMO_i < \pi/2 \leq \angle MNO_i \Rightarrow |O_iM| > |O_iN| = \delta R$.

$|O_iM| > \delta R$ means that the point M lies outside $D(C_i^*)$, i.e., $M \notin D(C_i^*)$. Also $M \in D(C^*) \Rightarrow M \in [D(C^*) - D(C_i^*)] = A_i^*$. Since M is an arbitrary point in $\overline{C_i}$, $\overline{C_i} \subseteq A_i^* \Rightarrow \overline{A_i} \subseteq A_i^*$.

For all the cases, we have $\overline{A_i} \subseteq A_i^*$ which proves our claim and consequently completes our proof. ■

CHAPTER 7.

CONCLUSION AND FUTURE WORK

7.1. Conclusion

In this dissertation, we concentrate on area coverage with the optimization goal of maximizing network lifetime by designing various localized, distributed and parallel, energy-efficient solutions.

In this dissertation, we tackle various generalizations and variants of the basic coverage problem. Each problem is considered in separate chapter with the proposed solutions (including algorithms, a scheme, and a framework). Along with the solutions, the theoretical analysis and extensive simulations prove the efficiency of proposed solutions. At the very end of each chapter, we also discuss various improvements and extensions of proposed solutions.

Initially, in CHAPTER 3, we consider the *k*-coverage issue in wireless sensor networks which is formulated as SESK problem. We then propose a completely distributed and parallel algorithm named DESK to solve that NP-complete problem. We as well provide analysis and simulations to support DESK's correctness and efficiency. An improvement to make DESK location-free is also given.

Next, we propose three distributed and parallel coverage algorithms for adjustable-sensing-range WSNs. Two of them also guarantee the connectivity at all cases. A solution to allow the third one (which is the most efficient one) to provide connectivity without relying on the assumption of sensor's communication range at least twice of its sensing range is mentioned at the end of the CHAPTER 4.

Regarding to emergency alarming applications that explicitly require low event notification delay and that require the network to be able to detect multiple types of simple events, we deal with “composite event detection” problem in CHAPTER 5. Since no currently existing methods are efficient for such applications, we propose a novel scheme for detecting composite events and delivering the warnings in timely manner. As being shown, the simple scheme can amazingly reduce the energy wasted on unnecessary communication and thus significantly improves network lifetime. The scheme can also minimize the notification delay. Based on that scheme, we suggest an energy-efficient, cluster-based algorithm that solves the joint-issue including fault tolerance, topology, routing control. At the end of CHAPTER 5, we explain how to extend the proposed cluster-based algorithm to a full network consisting of many clusters.

Concerning with the partial coverage problem, we propose a framework consisting of four strategies. The framework utilizes existing works for the complete-coverage problem and converts them to algorithms for the partial coverage problem. Two general strategies can not only greatly increase network lifetime but also guarantee numerous fancy properties such as constant-bounded SVD, unchanged algorithms time complexity. For the other two extended strategies, although their resulting algorithms cannot guarantee a constant degree of coverage uniformity, they even further increase network lifetime in compared with resulting algorithms of two corresponding general strategies.

A brief survey about coverage algorithms in literature is also carried out in CHAPTER 2. We first classify those works into groups, and give a brief comparison of work in each group.

Finally, we outline some interesting issue that we will consider in our future work which are discussed in more detail next.

7.2. Future work

In this dissertation, we highly focus on area coverage problem and we are interesting only in energy-efficient, localized, distributed and parallel algorithms. There are still some existing issues that can be tackled to improve the works in this text. In addition, there are still numerous variants of coverage problem in general, area coverage problem in specific. Different issues (coverage, connectivity, event detection), different requirements (energy efficiency, fault-tolerance, *k*-coverage, α-coverage, distributed algorithms) and network constraints (fixed/adjustable sensor's sensing range network, heterogeneous/homogeneous network, directional/omni-directional sensing network, non-isotropic/isotropic sensors network, unlimited/limited network bandwidth) might need to be taken into account for each variant. In the future, we focus on the following work:

Work 1. In CHAPTER 5 and [LIC09], on discussion about the extension the cluster-based algorithm to a large, full network, we assume that the network is partitioned into a (fixed) number of clusters and that we have enough gateways such that there is at least one for each (fixed) cluster. If we relax that assumption, the work becomes much more challenging. We need to dynamically partition the area into clusters relatively to the positions of gateways. Our partitioning method might base on Voronoi diagram so each gateway will be at the center of a Voronoi cell. We then may combine one or some cells into one cluster depending on the size of neighboring cells. The sensing and routing jobs are also challenges for the clusters that are too big.

Work 2. In CHAPTER 6, we have proved a *δ-compression theorem* with the assumption that the sensing region of a sensor is a disk. We conjecture that the theorem still holds even we relax that assumption, that is, the sensing region of a sensor is an arbitrary convex shape. If our

conjecture is true, we can extend the work in CHAPTER 6 to WSNs where sensors have arbitrary shape of sensing region.

Work 3. Concerning strategies for partial coverage discussed in CHAPTER 6, it can be seen that for *Strategy G-2* and *Strategy E-2*, the results may be different for different values of β. However, we still have no recommendation of the "good" value of β. That value of β should depend on the characteristics of original algorithms. In future work, we may conduct more simulations to characterize the pattern for such values of β.

Work 4. In most of the work, an implicit assumption about the omni-directional sensing ability of sensors is assumed. That is, all the sensors could be able to sense equally at all directions. This assumption is not always the case in practice. For example, for visualized sensor network where sensors are cameras, each sensor only has some physically predefined Field-Of-View (FOV) and can only sense (i.e., take pictures) at some directions. Clearly, the basic coverage problem is a special case of the coverage problem for this kind of WSNs for the case FOV=360^0.

BIBLIOGRAPHY

[ABR04] *Z. Abrams, A. Goel, and S. Plotkin*, "Set k-cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks", Proc. of Third International Symposium on Information processing in sensor networks, pp. 424 - 432, Berkeley, California, USA, 2004.

[AKY02] *F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci*, "Wireless sensor networks: a survey", Computer Networks, 38:393-422, 2002.

[BAI05] *H. Bai, X. Chen, Y. Ho, and X. Guan*, "Percentage Coverage Configuration in Wireless Sensor Networks", Parallel and Distributed Processing and Applications, Volume 3758/2005, p:780-791 Springer Berlin / Heidelberg, 2005.

[BAI06] *X. Bai, S. Kuma, D. Xua, Z. Yun, and T.H. La*, "Deploying wireless sensors to achieve both coverage and connectivity", MobiHoc'06, May 22–25, 2006, Florence, Italy.

[BEJ08] *Y. Bejerano*, "Simple and Efficient *k*-Coverage Verification without Location Information", pages: 291-295, INFOCOM 2008, 13-18 April 2008, Phoenix, AZ.

[BER04] *P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky*, "Power efficient monitoring schedules in sensor network", in IEEE Wireless communication and Networking conference, 2004.

[BRE05] *J.L. Bredin, E.D. Demaine, M.T. Hajiaghayi, and D. Rus*, "Deploying sensor networks with guaranteed capacity and fault tolerance", Proc. of the 6th ACM international symposium on Mobile ad hoc networking and computing, pp: 309 – 319, 2005.

[BUL05] *N. Bulusu, S. Jha*, "Wireless Sensor Networks", © 2005 Artech House, Inc.

[CAD05] *M. Cardei and D.-Z. Du*, "Improving Wireless Sensor Network Lifetime through Power Aware Organization", ACM Wireless Networks, Vol. 11, No. 3, pp. 333-340, May 2005.

[CAR04] *M. Cardei and J. Wu*, "Energy-Efficient Coverage Problems in Wireless Ad Hoc Sensor Networks", Journal of Computer Communications on Sensor Networks, 2004.

[CAR06] *M. Cardei, J. Wu, M. Lu*, "Improving network lifetime using sensors with adjustable sensing ranges", International Journal of Sensor Networks, 2006.

[CAR06] *M. Cardei and I. Cardei*, "Energy-Efficient Connected-Coverage in Wireless Ad Hoc Sensor Networks", Manuscript, 2006.

[CAT05] *M. Cardei, M. Thai, Y. Li and W. Wu*, "Energy-Efficient Target Coverage in Wireless Sensor Networks", IEEE INFOCOM 2005, March 2005, Miami, USA.

[CAW05] *M. Cardei, J. Wu, N. Lu and M.O. Pervaiz,* "Maximum Network Lifetime with Adjustable Range", IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob'05), Aug. 2005.

[CAY06] *E. Cayirci, H. Tezcan, Y. Dogan, V. Coskun,* "Wireless sensor networks for underwater survelliance systems",Volume 4, Issue 4, July 2006, Pages 431-446, Ad Hoc Networks, 2006, Elsevier.

[CHA02] *K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho,* "Grid coverage for surveillance and target address in distributed sensor networks", IEEE Trans. Comput. 51(12):1448–1453 (Dec. 2002).

[CHE02] *B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris,*"Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", pp. 481-494, Vol. 8, No. 5, Springer Netherlands, September, 2002.

[CHE05] *M.X. Cheng, L. Ruan, and W. Wu,* "Achieving Minimum Coverage Breach under Bandwidth Constraints in Wireless Sensor Networks", Proc. of the 24th conference of the IEEE Communications Society (INFOCOM), 2005.

[CHI99] *C.F. Chiasserini and R.R. Rao,* "Pulsed battery discharge in communication devices", Proc of the 5th annual ACM/IEEE on Mobile computing and networking, Seattle, Washington, United States, pp.88-95, 1999.

[CLO02] *T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, K.K. Saluja* "Sensor deployment strategy for target" WSNA'02, September 28, 2002, Atlanta, Georgia, USA.

[CLO03] *T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, K.K. Saluja* "Sensor Deployment Strategy for Detection of Targets Traversing a Region" Vol. 8, No. 4 , pp. 453-461, Mobile Networks and Applications, Springer, August, 2003.

[DHA06] *A. Dhawan, C. T. Vu, A. Zelikovsky, Y. Li and S. K. Prasad,* "Maximum Lifetime of Sensor Networks with Adjustable Sensing Range", SAWN 2006, Las Vegas, USA.

[DHI02] *S. S. Dhilon, K. Chakrabarty, and S. S. Iyengar,* "Sensor placement for grid coverage under imprecise detections", Proc. 5th Int. Conf. Information Fusion (FUSION'02), Annapolis, MD, July 2002, pp. 1–10.

[ELS02] *J. Elson and K. Romer,* "Wireless Sensor Networks: A New Regime for Time Synchronization". ACM Mobile Computing and Communication Review (MC2R), Vol.6 No.4, pp.59-61. October, 2002.

[ERI09] *Weisstein, Eric W.* "Area." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/Area.html.

[GAL06] *A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic*, "Localized Sensor Area Coverage with Low Communication Overhead", Pervasive Computing and Communications, PerCom 2006, 2006.

[GAO06] *S. Gao, C. T. Vu, and Y. Li*, "Sensor Scheduling for *k*-Coverage in Wireless Sensor Networks", 2nd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2006), Hong Kong, China, December 13-15, 2006.

[GAO08] S. *Gao, X. Wang, and Y. Li*, "p-Percent Coverage Schedule in Wireless Sensor Networks", 17th International Conference on Computer Communications and Networks (ICCCN 2008), St. Thomas, Virgin Islands, August 3-7, 2008.

[GAR98] *N. Garg and J. Könemann*, "Faster and simpler algorithms for multicommodity flows and other fractional packing problems", Proc. of 39th Annual Symposium on the Foundations of Computer Science, pp 300-309, 1998.

[GUI01] *L.J. Guibas, J. Hershberger, S. Suri, and L. Zhang*, "Kinetic connectivity for unit disks", Discrete Computing Geom. 25 591–610, 2001.

[GUP03] *H. Gupta, S. Das, and Q. Gu*, "Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution", MobiHoc '03, Annapolis, Maryland, USA, June 1-3, 2003.

[HAL88] *P.Hall*, "Introduction to the Theory of Coverage Processes", Wiley, New York, 1988.

[HEF07] *M. Hefeeda*, "Forest Fire Modeling and Early Detection using Wireless Sensor Networks", Technical Report TR 2007-08, School of Computing Science, Simon Fraser University, August 2007.

[HEO02] *N. Heo and P. K. Varshney*, "A distributed self-spreading algorithm for mobile wireless sensor networks", Proc. IEEE Wireless Communications and Networking Conf. (WCNC'03), New Orleans, LA, March 2003, pp. 1597–1602.

[HOM02] *A. Howard, M. J. Matari, and G. S. Sukhatme*, "An incremental self-deployment algorithm for mobile sensor networks", Autonomous Robots special issue on intelligent embedded systems 13(2):113–126 (2002).

[HOW02] *A. Howard, M. Mataric, and G. Sukhatme*, "Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem", Proc. 6th Int. Symp. Distributed Autonomous Robotic Systems (DARS'02), Fukuoka, Japan, June 2002, pp. 299–308.

[HUA07] *S.C.H. Huang, P-J. Wan, C. T. Vu, Y. Li, and F. Yao*, "Nearly Constant Approximation for Data Aggregation Scheduling in Wireless Sensor Networks", IEEE INFOCOM 2007, Anchorage, Alaska, USA, May 6-12, 2007.

[HUL05] *C.F. Huang, L.C. Lo, Y.C. Tseng, W.T. Chen*, "Decentralized energy-conserving and coverage-preserving protocols for wireless sensor networks", pp 640- 643, Vol. 1, Circuits and Systems, ISCAS 2005, May 2005.

[HUR05] *H. Huang, AW. Richa, and M. Segal*, "Dynamic Coverage in Ad-Hoc Sensor Networks", pp 9-17, Vol. 10, No. 1-2, Mobile Networks and Applications, February, 2005, Springer.

[HUT05] *C.F. Huang and Y. Tseng*, "The coverage Problem in a Wireless Sensor Network", Mobile Networks and Applications 10, 519–528, 2005.

[IEEE4] *M. Naeve*, "IEEE 802.15.4 MAC Overview", https://mentor.ieee.org/802.15/file/04/15-04-0218-01-004a-ieee802-15-4-mac-overview.ppt.
- *IEEE Computer Society*, http://standards.ieee.org/getieee802/download/802.15.4a-2007.pdf.
- *Akiba*, http://freaklabs.org/index.php/Articles/Zigbee/IEEE-802.15.4-in-the-Context-of-Zigbee-Part-2-MAC-Layer.html.
- *S. C. Ergen*, "ZigBee/IEEE 802.15.4 Summary", http://www.sinemergen.com/zigbee.pdf.

[ILY05] *M. Ilyas and I. Mahgoub*, "Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems", CRC Press LLC, 2005.

[INT00] *C. Intanagonwiwat, R. Govindan and D. Estrin*, "Directed diffusion: a scalable and robust communication paradigm for sensor networks". ACM MobiCom, 2000, pp.56-67.

[ITL09] *Intel Corp.*, Heterogeneous Networks with Intel XScale, http://www.intel.com/

[JAN05] *D. Janakiram, A.V.U.P. Kumar and A.M. Reddy V*, "Component Oriented Middleware for Distributed Collaboration Event Detection in Wireless Sensor Networks", MPAC05, Nov 28-Dec 2, 2005 Grenoble, France.

[JIA04] *J. Jiang and W. Dou*, "A Coverage-Preserving Density Control Algorithm for Wireless Sensor", In Proc. of 3rd International Conference, ADHOC-NOW 2004, Vancouver, Canada, July 22-24, 2004.

[KAR03] *K. Kar and S. Banerjee*, "Node Placement for Connected Coverage in Sensor Networks:, Proc. of WiOpt 2003: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2003.

[KHU94] *S. Khuller, B. Raghavachari and N. Young*, "Balancing minimum spaning and shortest path trees", Algorithmica, 120(4):305-321, 1994.

[KUG04] *J. Kumagai*, "Life of birds [wireless sensor network for bird study]", Vol. 41, Issue: 4, pp 42- 49, Spectrum IEEE, April 2004.

[KUM04] *S. Kumar, T.H. Lai and J. Balogh*, "On *k*-coverage in a Mostly Sleeping Sensor Network", in Proc of the 10th international Conference on Mobile computing and networking, Philadelphia, PA, USA, Pages 144 - 158, 2004.

[LIC09] *Y. Li, C. Ai, C. T. Vu, Y. Pan, R. Beyah*, "Delay Bounded and Energy Efficient Composite Event Monitoring in Heterogeneous Wireless Sensor Networks", submitted to IEEE Transactions on Parallel and Distributed Systems (TPDS).

[LIE02] *J. Liebeherr, M. Nahas, W. Si*, "Application-Layer Multicasting With Delaunay Triangulation Overlays", IEEE Journal On Selected Areas In Communications, Vol.20, No. 8, Oct. 2002.

[LIR04] *Q. Li and D. Rus*, "Global Clock Synchronization in Sensor Networks". In Proc. of IEEE Infocom 2004, Hong Kong, China, March, 2004.

[LIU05] *Y. Liu, W. Liang*, "Approximate Coverage in Wireless Sensor Networks", The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05), pp. 68-75, 2005.

[LIW03] *X. Li, P. Wan, and O. Frieder*, "Coverage in Wireless Ad Hoc Sensor Networks", IEEE Transactions on Computers, Vol. 52, No. 6, Jun. 2003.

[MAL00] *N. Malpani, J.L. Welch, N. Vaidya*, "Leader election algorithms for mobile ad hoc networks", Workshop on Discrete Algothrithms and Methods for MOBILE Computing and Communications, Pages 96 - 103, 2000.

[MEG05] *S. Meguerdichain. F. Koushanfar, M. Potkonjak, and M. Srivastava*, "Worst and Best-case Coverage in Sensor Networks", pp. 84-92, IEEE Transactions on Mobile Computing, No.1, Jan-Feb 2005.

[MEK01] *S. Meguerdichain. F. Koushanfar, M. Potkonjak, and M. Srivastava*, "Coverage problems in wireless ad-hoc sensor networks", Proc. IEEE INFOCOM, '01, pp. 1380-7, 2001.

[MEQ01] *S. Meguerdichain. F. Koushanfar, G. Qu, and M. Potonjak*, "Exposure in wireless ad-hoc sensor network", ACM SIGMOBILE 7/01, Rome, Italy, pp.139-50, 2001.

[MES01] *S. Meguerdichian, S. Slijepcevic, V. Karayan, M. Potkonjak*, "Localized algorithm in Wireless Ad-Hoc networks: Location Discovery and Sensor Exposure", MobiHoc, Long Beach, CA, USA, 2001.

[MEH03] *D.P. Mehta, M.A. Lopez, L. Lin*, "Optimal Coverage Paths in Ad-hoc Sensor Networks", IEEE International Conference on Communications, Vol.1, pp. 507-11, May, 2003.

[POD04] *S. Poduri and G.S. Sukhatme*, "Constrained coverage for mobile sensor networks", Robotics and Automation, 2004. Proc. of ICRA'04. Vol. 1, pp. 165- 171, 26 April-1 May 2004.

[POT00] G. J. Pottie and W. J. Kaiser, Wireless integrated network sensors, Communications of the ACM, 43(5):51-58, 2000.

[RAG02] *V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava,* "Energyaware wireless microsensor networks", IEEE Signal Processing Magazine, March 2002.

[ROM04] *K. Römer and F. Mattern*. "Event-based systems for detecting real-world states with sensor networks: a critical analysis". DEST Workshop on Signal Processing in Sensor Networks at ISSNIP, December 2004, pp.389-395.

[SIB77] *R. Sibson,* "Locally equiangular triangulations", Comput. J., vol. 21, pp.243245, 1977.

[SLI01] *S. Slijepcevic and M. Potkonjak,* "Power efficient organization of wireless sensor networks", IEEE International Conference on Communications ICC, 2001.

[SOH07] *K. Sohraby, D. Minoli, T. Znati*, "Wireless sensor networks: technology, protocols, and applications", 2007 by John Wiley & Sons, Inc.

[SON07] *B. Son, Y-S Her, J-G Kim,* "A Design and Implementation of Forest-Fires Surveillance System based on Wireless Sensor Networks for South Korea Mountains", IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.9B, pp. 124-130, September 2006.

[STO01] *I. Stojmenovic and X. Lin,* "Power-Aware Routing in Ad Hoc Wireless Networks", IEEE transactions on parallel and distributed systems, 2001.

[SZE04] *R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Main-waring, D. Estrin,* "Habitat Monitoring with Sensor Networks", Communications of the ACM 47(6):34-40, 2004.

[THAI05] *M. T. Thai, Y. Li, F. Wang, and D-Z. Du*, "Minimum Coverage Breach and Maximum Network Lifetime in Wireless Sensor Networks", Submitted to IEEE Transactions on Wireless Communications, 2005.

[TIA02] *D. Tian and N.D. Georganas*, "A Coverage-Preserving Node Scheduling Scheme for Large Wireless Sensor Network". In Proc. of 1st ACM Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, USA, 2002.

[TINYOS] TinyOS, http://www.tinyos.net/.

[UPA03] *S. Upadhyayula, V. Annamalai, and S. K. S. Gupta*. "A Low- Latency and Energy-Efficient Algorithm for Convergecast in Wireless Sensor Networks". GLOBECOM, 2003, 22(1):3525– 3530.

[VAS03] *S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, D. Towsley,* "Leader election algorithms for wireless ad hoc networks", DARPA Information Survivability Conference and Exposition, pages 261- 272, 22-24 April 2003.

[VUC06] *C. T. Vu, S. Gao, W. P. Deshmukh, and Y. Li*, "Distributed Energy-Efficient Scheduling Approach for *k*-Coverage in Wireless Sensor Networks", Military Communications Conference 2006 (MILCOM 2006), Washington, DC, October 23-25, 2006.

[VUC07] *C. T. Vu, R. A. Beyah, and Y. Li*, "Composite Event Detection in Wireless Sensor Networks", 26th IEEE International Performance Computing and Communications Conference (IPCCC 2007), New Orleans, Louisiana, USA, April 11-13, 2007.

[VUT07] *Chinh Vu*, Master's Thesis, "An Energy-Efficient Distributed Algorithm for k-Coverage Problem in Wireless Sensor Networks", Georgia State University, April 09, 2007, http://etd.gsu.edu/theses/available/etd-04112007-165119/.

[VUC09] *C. T. Vu and Y. Li*, "Delaunay-triangulation based complete coverage in wireless sensor networks", IQ2S2009 in conjunction with PERCOM 2009, Galveston, TX, March 9-13, 2009.

[VUT09] *C. T. Vu, Z. Cai, and Y. Li*, "Distributed Energy-Efficient algorithms to maximize network lifetime for coverage problem in adjustable sensing radii Wireless Sensor Networks", submitted to Discrete Mathematics, Algorithms and Applications (DMAA).

[VUV09] *C. T. Vu, G. Chen, Y. Zhao, and Y. Li*, "A universal framework for α-coverage problem in Wireless Sensor Network", submitted to The 17th IEEE International Conference on Network Protocols (ICNP 2009).

[VUY09] *C. T. Vu, G. Chen, Y. Zhao, and Y. Li*, "An extended framework for α-coverage problem in Wireless Sensor Network", manuscript.

[WAC03] *G. Wang, G. Cao, and T. LaPorta*, "A bidding protocol for deploying mobile sensors", Proc. 11th IEEE Int. Conf. Network Protocols (ICNP'03), Atlanta, GA, Nov. 2003, pp. 80–91.

[WAH05] *Y.C. Wang, C.C. Hu, and Y.C. Tseng* , "Efficient Deployment Algorithms for Ensuring Coverage and Connectivity of Wireless Sensor Networks", Proc. of the First International Conference on Wireless Internet – WICON'05, 2005.

[WAN03] *X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill*, "Integrated coverage and connectivity configuration in wireless sensor networks", SenSys'03, November 5–7, 2003, Los Angeles, California, USA.

[WAN04] *G. Wang, G. Cao, and T. LaPorta*, "Movement-assisted sensor deployment", Proc. IEEE InfoCom (InfoCom'04), Hong Kong, March 2004, pp. 80–91.

[WAN06] *P.J. Wan and C.W. Yi*, "Coverage by randomly deployed wireless sensor networks", IEEE Transaction on Information Theory, Vol. 25, No. 6, June 2006.

[WAN07] *J. Wang, S. Medidi*, "Energy Efficient Coverage with Variable Sensing Radii in Wireless Sensor Networks", Wireless and Mobile Computing, Networking and Communications, 2007 - WiMOB 2007, White Plains, NY, USA.

[WER06] *G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees and M.Welsh*, "Deploying a Wireless Sensor Network on an Active Volcano", In press. Sensor Nets issue of IEEE Internet Computing, 2006.

[WUA08] Y. Wu, C. Ai, S. Gao, and Y. Li, "p-Percent Coverage in Wireless Sensor Networks", WASA 2008, Dallas, TX, October 26-28, 2008.

[WUL99] *J. Wu and H. Li*, "On calculating connected dominating sets for effcient routing in ad hoc wireless networks", Proc. of DialM99, pp. 714, 1999.

[WUY04] *J. Wu and S. Yang*, "Coverage and Connectivity in Sensor Networks with Adjustable Ranges", International Workshop on Mobile and Wireless Networking (MWN), Aug. 2004.

[XBOW09] *CrossBow Technology*, http://www.xbow.com/.

[XUH05] *H. Xu, L. Huang, Y. Wan, and K. Lu*, "Localized Algorithm for Coverage in Wireless Sensor Networks", Proc. of Sixth Int'l Conf. on Parallel and Distributed Computing, Application and Technology (PDCAT'05), pp750-754, 2005.

[YAN03] *T.Yan, T. He, and J. A. Stankovic*, "Differentiated surveillance for sensor networks", in ACM Int'l Conf. on Embedded Networked Sensor System (SenSys), pp 51-62, 2003.

[YAN06] *S. Yang, F. Dai, M. Cardei, J. Wu, F. Patterson*, "On Connected Multiple Point Coverage in Wireless Sensor Networks", International Journal of Wireless Information Networks, 2006 – Springer.

[YUN05] *L. Yu, N. Wang, X. Meng*, "Real-time forest fire detection with wireless sensor networks", Wireless Communications, Networking and Mobile Computing, 2005, proceedings, pp1214- 1217, Vol. 2 23-26 Sept. 2005.

[YYE91] *Y.Ye*, "An O(n^3L) potential reduction algorithm for linear programming", Mathematical Programming, pp 239-258, Vol. 50, No. 1-3, March, 1991, Springer Berlin / Heidelberg.

[ZHA03] *H. Zhang and J. C. Hou*, "Maintaining Sensing Coverage and Connectivity in Large Sensor Networks", Technical Report UIUC, UIUCDCS-R- 2003-2351, 2003.

[ZHA04] *H. Zhang and J. Hou*, "On deriving the upper bound of α-lifetime for large sensor networks", Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing, 2004.

[ZHA05] *H. Zhang and J. Hou,* "Maximizing α-Lifetime for Wireless Sensor Networks", in 3rd International Workshop on Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks (SenMetrics 2005), San Diego, CA, USA, July 21 2005.

[ZHE05] *R. Zheng, G. He, and X. Liu,* "Location-free Coverage Maintenance in Wireless Sensor Networks", Technical Report Number UH-CS-05-15, July 21, 2005.

[ZHO04] *Z. Zhou, S. Das, and H. Gupta*, "Connected *k*-coverage Problem in Sensor Networks", Proc. of the 13th International Conference on Computer Communications and Networks (ICCCN), 2004.

[ZOC03] *Y. Zou and K. Chakrabarty*, "Uncertainty-aware sensor deployment algorithms for surveillance applications", Proc. IEEE Global Communications Conf. (GLOBECOM'03), Dec. 2003.

[ZOU03] *Y. Zou and K. Chakrabarty*, "Sensor deployment and target localization based on virtual forces", INFOCOM'03, 2003.

APPENDIX A.

RELATED PUBLICATIONS

A1. Refereed journal articles

[1] **Chinh T. Vu**, Zhipeng Cai, and Yingshu Li, *Distributed Energy-Efficient algorithms for coverage problem in adjustable sensing radii Wireless Sensor Networks*, to appear in Discrete Mathematics, Algorithms and Applications (DMAA).

[2] **Chinh T. Vu**, Guantao Chen, Yi Zhao, and Yingshu Li, *An extended framework for α-coverage problem in Wireless Sensor Network*, manuscript.

[3] Yingshu Li, Chunyu Ai, **Chinh T. Vu**, Yi Pan, Raheem Beyah, *Delay Bounded and Energy Efficient Composite Event Monitoring in Heterogeneous Wireless Sensor Networks*, submitted to IEEE Transactions on Parallel and Distributed Systems (TPDS).

A2. Refereed conference articles

[4] **Chinh T. Vu**, Guantao Chen, Yi Zhao, and Yingshu Li, *A universal framework for α -coverage problem in Wireless Sensor Network*, submitted to The 17th IEEE International Conference on Network Protocols (ICNP 2009).

[5] **Chinh T. Vu** and Yingshu Li, *Delaunay-triangulation based complete coverage in wireless sensor networks*, IQ2S2009 in conjunction with PERCOM 2009, Galveston, TX, March 9-13, 2009.

[6] Scott C.H. Huang, Peng-Jun Wan, **Chinh T. Vu**, Yingshu Li, and Frances Yao, *Nearly Constant Approximation for Data Aggregation Scheduling in Wireless Sensor Networks*, IEEE INFOCOM 2007, Anchorage, Alaska, USA, May 6-12, 2007.

[7] **Chinh T. Vu**, Raheem A. Beyah, and Yingshu Li, *Composite Event Detection in Wireless Sensor Networks*, 26th IEEE International Performance Computing and Communications Conference (IPCCC 2007), New Orleans, Louisiana, USA, April 11-13, 2007.

[8] Shan Gao, **Chinh T. Vu**, and Yingshu Li, *Sensor Scheduling for k-Coverage in Wireless Sensor Networks*, 2nd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2006), Hong Kong, China, December 13-15, 2006.

[9] **Chinh T. Vu**, Shan Gao, Wiwek P. Deshmukh, and Yingshu Li, *Distributed Energy-Efficient Scheduling Approach for k-Coverage in Wireless Sensor Networks*, 25th Military Communications Conference 2006 (MILCOM 2006), Washington DC, USA, October 23-25, 2006.

[10] Akshaye Dhawan, **Chinh T. Vu**, Alex Zelikovsky, Yingshu Li, and Sushil K. Prasad,*Maximum Lifetime of Sensor Networks with Adjustable Sensing Range*, 2nd ACIS International Workshop on Self-assembling Wireless Networks (SAWN 2006), Las Vegas, Nevada, USA, June 19-20, 2006.

APPENDIX B.

DEFINITIONS AND THEOREMS

B1. List of definitions

B2. List of theorems