5-3-2007

An Energy-Efficient Distributed Algorithm for k-Coverage Problem in Wireless Sensor Networks

Chinh Trung Vu

AN ENERGY-EFFICIENT DISTRIBUTED ALGORITHM FOR *k*-COVERAGE PROBLEM IN WIRELESS SENSOR NETWORKS

by

CHINH TRUNG VU

Under the Direction of Yingshu Li

ABSTRACT

Wireless sensor networks (WSNs) have recently achieved a great deal of attention due to its numerous attractive applications in many different fields. Sensors and WSNs possesses a number of special characteristics that make them very promising in many applications, but also put on them lots of constraints that make issues in sensor network particularly difficult. These issues may include topology control, routing, coverage, security, and data management. In this thesis, we focus our attention on the coverage problem. Firstly, we define the Sensor Energy-efficient Scheduling for *k*-coverage (SESK) problem. We then solve it by proposing a novel, completely localized and distributed scheduling approach, naming Distributed Energy-efficient Scheduling for *k*-coverage (DESK) such that the energy consumption among all the sensors is balanced, and the network lifetime is maximized while still satisfying the *k*-coverage requirement. Finally, in related work section we conduct an extensive survey of the existing work in literature that focuses on with the coverage problem.

INDEX WORDS: Wireless sensor network, Coverage problem, *k*-coverage, Localized and distributed algorithm, Energy-efficiency, Fault-tolerance, Robustness

AN ENERGY-EFFICIENT DISTRIBUTED ALGORITHM FOR *k*-COVERAGE PROBLEM IN WIRELESS SENSOR NETWORKS

by

CHINH TRUNG VU

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2007

AN ENERGY-EFFICIENT DISTRIBUTED ALGORITHM FOR *k*-COVERAGE PROBLEM IN WIRELESS SENSOR NETWORKS

by

CHINH TRUNG VU

Major Professor: Dr. Yingshu Li

Committee: Dr. Sushil Prasad
Dr. Anu Bourgeois

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

April 2007

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Yingshu Li for her expert guidance and introducing me into research.

I would also like to thank committee members, Dr. Sushil K. Prasad and Dr. Anu G. Bourgeois for serving on my committee, for their comments and for taking precious time to review this thesis.

I am thankful to Dr. Rajshekhar Sunderraman - Director of Graduate Studies - for his helps which make my time as a graduate student in this department as smooth as possible.

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. And above all, I am grateful to them for continuously encouraging me in my academic pursuit.

Dedicated to my parents and my late brother.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Base Station	BS
Distributed Energy-efficient Scheduling for *k*-coverage	DESK
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
Quality of Service	QoS
Sensor Energy-efficient Scheduling for *k*-coverage	SESK
Wireless Ad hoc Sensor Network	WSN, WASN

I. INTRODUCTION

Wireless sensor networks (WSNs) have recently attracted much attention of researchers due to its wide ranges of applications. A sensor is a tiny battery-equipped device capable of sensing, communication and computation. The minuscule size, light weight, and portability attributes are special characteristics of a sensor that make WSNs to be the best and/or unique choice in many applications. However, there are also a lot of constraints on sensors such as limit on energy supply, on bandwidth, on computational capability, the uncertainty of sensed data and communicated information, the vulnerability of sensors to environment, etc. that require thorough and prudent researches to overcome. A WSN is a wireless network comprising of a huge amount of static or mobile sensors. Inside a WSN, the sensors autonomously collaborate to sense, collect, process data and transmit those sensed/processed data to some specific data processing centers. 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.

I.1. Constraints of sensors and WSNs.

A sensor network is a wireless network that comprises many sensing devices scattered on an area as guards to record (and eventually 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. Since the recent development in micro-electro-mechanical system (MEMS) technology, wireless communication, and digital electronics has enabled the low-cost, low-power, multifunction, and tiny sensor [AKY02], so that a redundant number of

sensors can be densely deployed to a monitored area to prolong the network lifetime and enhance the surveillance quality. Although a huge number of protocols has been applied to wired and (traditional) wireless network, those protocols cannot directly be employed to sensor network since sensor network possesses some special characteristics and restrictions that distinguish it from other types of networks. Those particular things may include:

- The sensor nodes can only provide communication with very low quality, high latency and variance, limited bandwidth, and high failure-rate. Sensor's transmission range is short and greatly affected by energy. In sensor network, the communications are mainly by broadcasting.
- The most precious resource of a sensor is energy. In some cases, the battery is irreplaceable while all the sensor's operations consume specific amount of energy, therefore energy conservation always is the biggest requirement on designing a sensor network protocol. Also, a sensor consumes much more energy on communication than on computation.
- Huge number of sensors is deployed into hostile environment under tough condition, thus it is very difficult to maintain and manage the network.
- Another sensor's constraint is limitation of computational capability and memory sizes. This limits the types of algorithms and results processing on a sensor.
- Sensing data is tended to be biased under the environment effects such as noise, obstacle, etc.
- Sensor nodes may not have global ID due to the large number of sensors.
- The topology of a WSN changes very frequently due to the movement of sensors or the (temporary or permanent) failure or death of some sensors or the addition of sensors to the network.

- A tiny sensor node tends to fail on operating due to numerous reasons such as depletion of energy, environmental interference [AKY02] and is very vulnerable to the environment (e.g., easy to be physically damaged).

I.2. Protocol design requirements

A WSN being a special kind of network which possesses lots of constraints as being itemized in section I.1, protocols designed for them must satisfy some special requirements to overcome those constraints. The most critical requirements may include:

- **Energy-efficiency**. A sensor is a battery-driven device and in most cases the battery is irreplaceable. However, each operation of a sensor consumes a specific 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 the 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, or may be temporarily or permanently go out of service at any time for various reasons. 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 high failure-rate communication system and unreliable sensing components. Thus, to guarantee the correctness and integrity of sensed data, protocols designed for WSNs must provide high fault-tolerance.

- **Distributed algorithms**. Sensors have very limited computational ability and very small memory size. So 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 huge number of sensors and the topology of a sensor network changes very frequently. Thus, the converge-time of any algorithms for WSN needs to be small enough to keep up with the changes in the networks. For these reasons, in most cases, only distributed algorithms are suitable for WSNs.

I.3. Challenging issues in WSNs and our main focus.

Before a WSN can be brought into real life, many problems need to be carefully resolved. More and more problems are discovered and solved through time. In this sub-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 limitation as mentioned in section I.1, which makes them exceedingly challenging.

- *Algorithm type*: Energy is most critical resource of a sensor since every operation requires certain energy while sensor is battery-driven and battery is not always replaceable. Thus, energy-efficiency should be (and have been) foremost concern of any protocol designed for a WSN. Other limitations of a sensor that require thorough awareness on designing a WSN protocol is sensor's limited memory size, communication and computation capability, thus algorithms for WSNs need to be simple but robust and fault-tolerant. That is reason why decentralized algorithm is always preferable (if it is not the only suitable ones) in WSNs. Some requirements that a "good" protocol aim to are simplicity, energy-efficiency, localized and distributed 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 anytime or any place for various reasons. It follows that the topology of a WSN is so dynamic and unpredictable. For each kind of application, it needs an appropriate topology for it to efficiently function.

- *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, safely and securely route the data through a high-density network is also a big question 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 in multiple-source-multiple-destination basic and the number of sensors in a WSN is in the order of hundreds, thousands or more. Thus, the amount of data collected by a WSN is remarkably huge. How to manage, process and route this data is truly a challenge. Researchers have considered the following sub-problems for this kind of issue: data in-network 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 world and collect information that it is desired and supposed to gather. This problem is thoroughly investigated in this thesis. More specifically, we first propose a special kind of coverage problem named SESK in section II which has high requirements of being energy-efficient and fault-tolerant. We later argue that SESK is an NP-complete problem. We further solve SESK by introducing a heuristic which satisfies all the requirements a sensor network protocol should have (which is shown in section III.3).

The correctness as well as its performance and efficiency are confirmed by providing both theoretical analysis and simulation results. At the end of this thesis, we give an extensive survey of the existing literature on the coverage problem.

- *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 WSN are to guarantee the integrity, confidentiality of the data or to verify the authenticity of entities exchanging the data.
- *Others*: Beside 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), transport-layer protocols (e.g., real-time traffic, reliable transfer).

I.4. WSNs applications

[AKY02] gives a nice survey on applications of WSNs. Here, we briefly itemize some typical and promising applications of WSNs. More detail can be referred from [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, pollution, etc. of soil, marine, atmosphere, etc.
  - To detect a disaster that is about to happen such as forest fire, flood, etc.
  - 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, etc.) of doctor, patience or drug, equipment, etc.
  - To tele-monitor human physiological data (e.g., patience behavior). And the data will be collected and analyze to detect early symptom of a disease, to find new treatment, etc.
- *Home applications:* It can be used to provide smart home in which all the devices can be autonomous or can be controlled from remote.
- *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 to the surface of a planet for scientific research purpose.

II. THE SESK PROBLEM

In this thesis, we will define and solve a very special coverage problem that requires a certain level of fault tolerance and energy balancing. 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 thesis. We further confirm that SESK is an NP-complete problem. The heuristic for this NP-complete problem is given in next section.

Definition 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 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 1. *SESK is a NP-complete problem.*

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

Definition 3. (Network lifetime) *The 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:

$$Max\ m \tag{1}$$

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

Subject to:

$$\bigcup_{j=1}^{m} C_j \subseteq S \tag{3}$$

$$\mathrm{cov}_j \geq k \ for\ all\ j = 1..m \tag{4}$$

$$\sum_{j=1}^{m} e_{j,i} \leq E_i \ for\ all\ i = 1..N \tag{5}$$

Formulation explanations and remarks:

1. Eq. 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. Eq. 2 is an effort to balance the energy consumption among all the sensors.
3. Eq. 3 guarantees that all the set covers are the subsets of the set of all sensors.
4. Eq. 4 assures that the whole region is continuously *k*-covered.
5. Eq. 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.

III. THE DESK ALGORITHM

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.

III.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 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 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.

III.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 V.3.1). In our paper, 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.

III.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 III.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 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 effect 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 sub-section I.3.

- 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 thesis we define c_i as the number of the neighbors n_i who need s_i to be active.

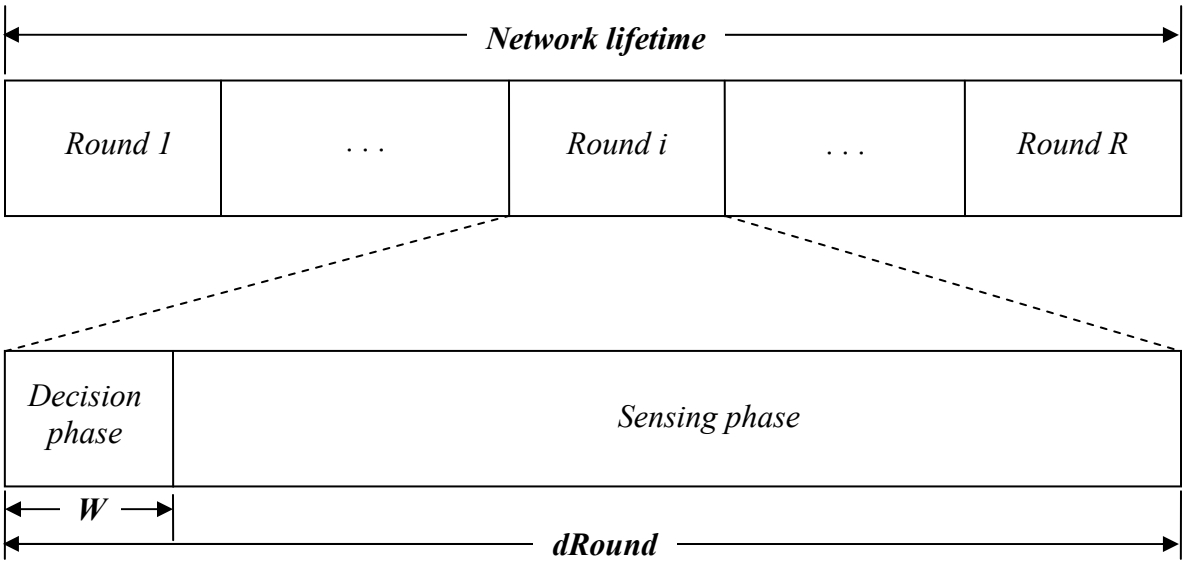

Figure 1. DESK works in rounds

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

$$w_i = \begin{cases} \dfrac{\eta}{n_i^{\alpha} l(e_i, r_i)^{\beta}} \times W + z & \text{if } e_i \geq e_{threshold} \\ W & \text{otherwise} \end{cases} \quad (6)$$

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 IV.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:

$$e_{threshold} \text{ satisfies } l(e_{threshold}, r_i) = W \quad (7)$$

$$\eta = dRound^{\beta} \quad (8)$$

III.4. The algorithm

In this sub-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 subsection.

III.4.1. Pseudo-code

The pseudo-code for DESK is illustrated as in Algorithm 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 IV.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 sub-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.

Algorithm 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 ≤ 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 ≥ 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
```

III.4.2. Description

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 apply 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, this sensor updates its n_i, contribution c_i and recalculate w_i.
- If a sensor receives mASK2SLEEP message from all of its neighbors, then it will send message GOSLEEP to all of its neighbor, 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 removes the neighbor sending that message out of its list L.

III.4.3. Example

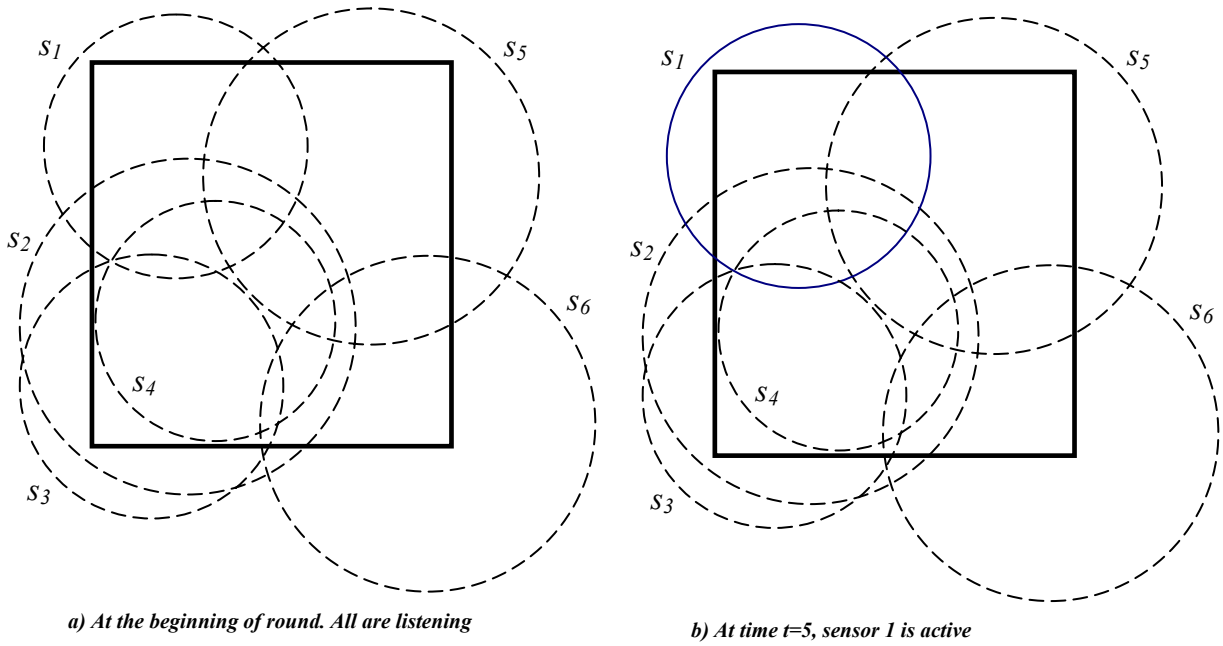

Figure 2. The step by step operations of DESK. Part 1

In Figure 2 and Figure 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 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 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.a).
- At the time t = 45, s_6 and after that, at the time t = 50, s_5 eventually become active (Figure 3.b). After this point, the area is *k*-covered.
- The discovered set cover for this round is $\{s_1, s_2, s_5, s_6\}$.

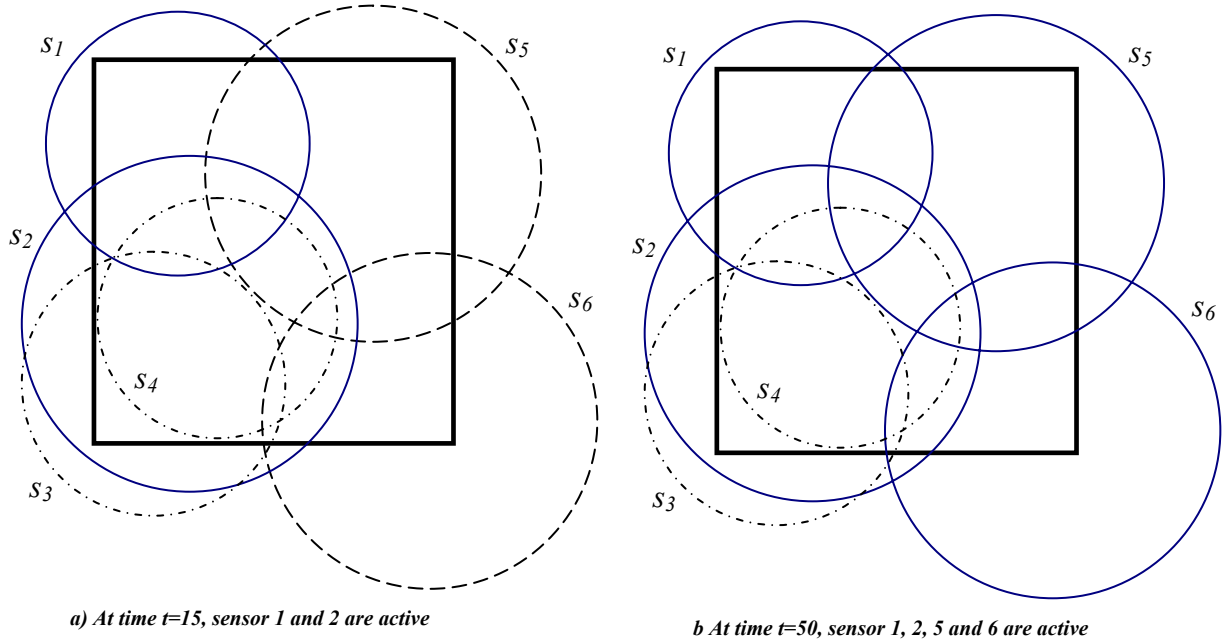

Figure 3. The step by step operations of DESK. Part 2

IV. DESK ANALYSIS

IV.1. Theoretical Analysis

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

Definition 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. *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 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 has as well shown that sleep sensors are *k*-covered. Finally, all the sensors are *k*-covered. According to Theorem 2, the whole monitored area is guaranteed to be *k*-covered. ■

Theorem 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 $O(n\Delta)$. ■

IV.2. Simulation results

In this sub-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 the network lifetime with different number of sensors and different values

of k. We also compare the network lifetimes of the networks with different initial energy of sensors.

IV.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 I.

Table I. 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 I, 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:

$$\text{Linear function: } f(r_i) = \frac{1}{\kappa} \times r_s \tag{9}$$

$$\text{Quadratic function: } f(r_i) = \frac{1}{\kappa} \times r_s^2 \tag{10}$$

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 I 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 I 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:

$$l(e_i, r_i) = \frac{e_i}{\text{Energy consumption in active mode}} \tag{11}$$

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

$$l(e_i, r_i) = \frac{e_i}{9.72 + f(r_i)} \tag{12}$$

IV.2.2. Network configuration

All the parameters used for the simulation are provided in Table II. 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 II, 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 II. 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

IV.2.3. Simulation results

In Figure 4, Figure 5 and Figure 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. 10; and the number of the sensors range from 50 to 200. Figure 7 and Figure 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. 9 and 10, respectively.

Figure 4 shows how many sensors that are active each round. 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.

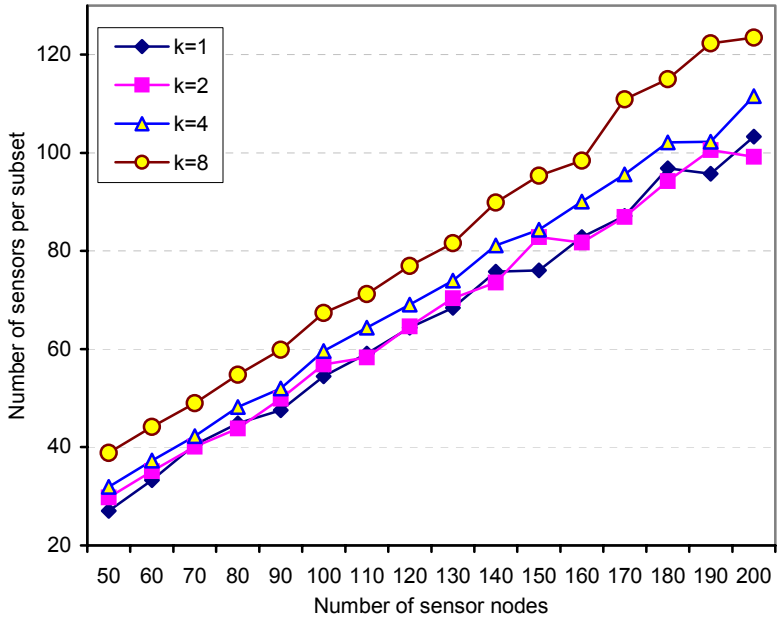

Figure 4. Number of active sensors per subset

Figure 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.

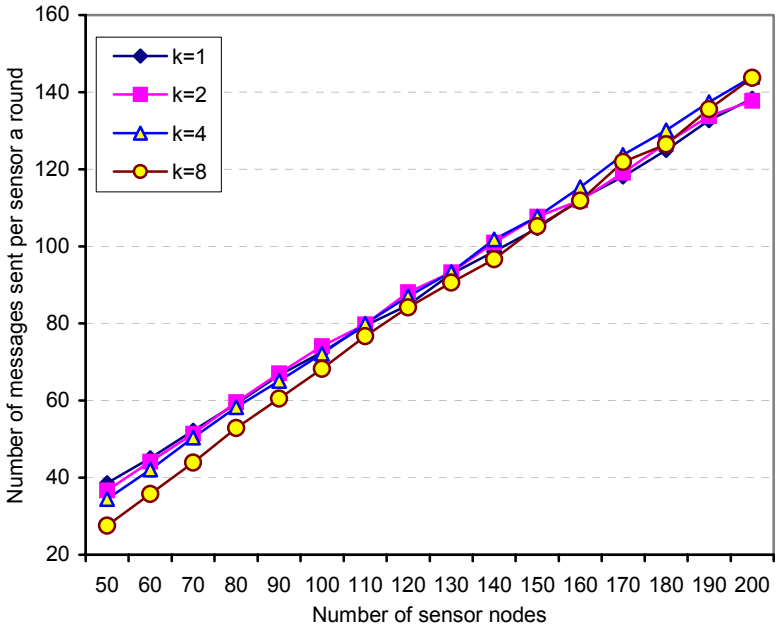

Figure 5. Number of messages sent per sensor each round

In Figure 6, the network lifetime is illustrated. As shown in Figure 6, the 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 the 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.

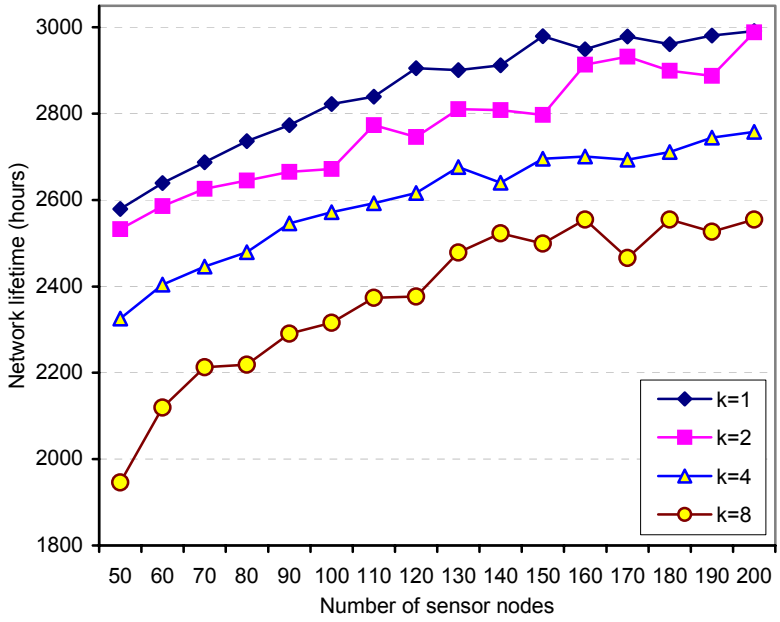

Figure 6. Network lifetime

In Figure 7 (the energy function f is linear) and Figure 8 (the energy function f is quadratic), we conduct measurements of the 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 7 and Figure 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 that 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 7 and Figure 8.

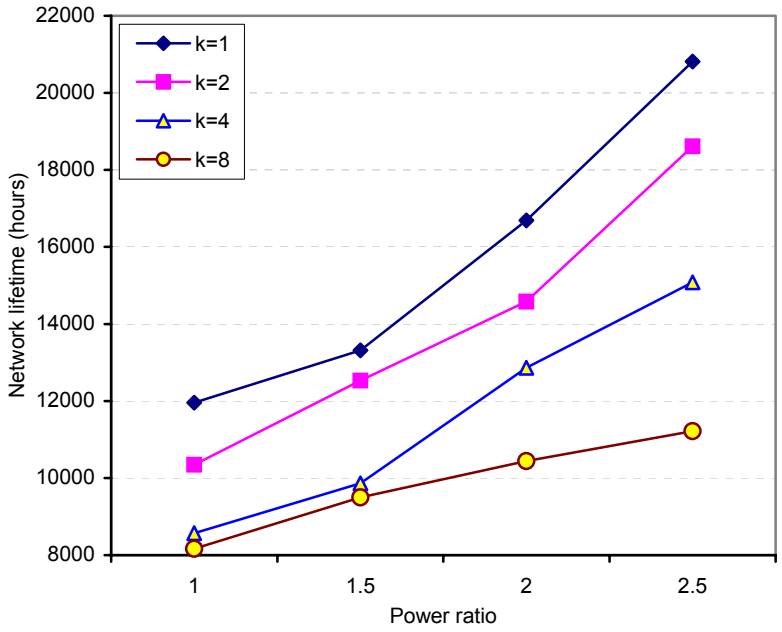

Figure 7. Linear energy model: Network lifetime with different power range

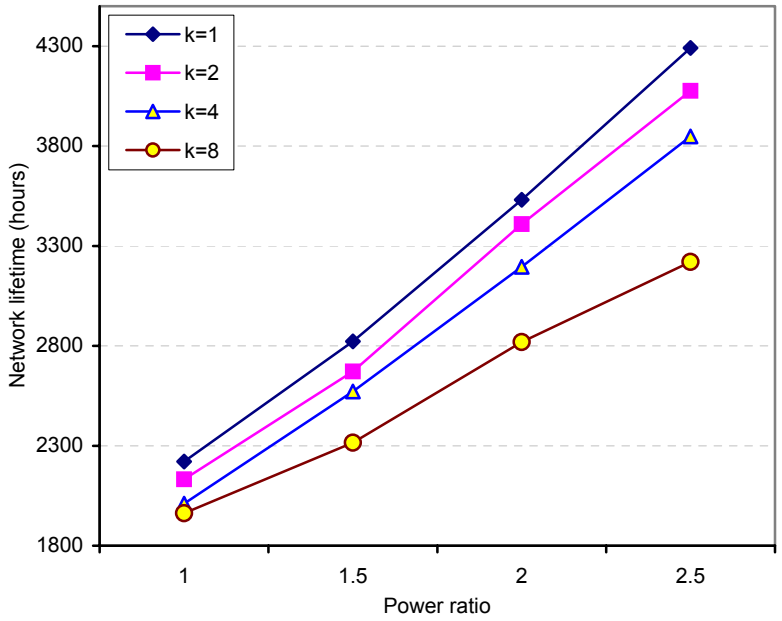

Figure 8. Quadratic energy model: Network lifetime with different power range

V. RELATED WORK

In this section, we conduct an extensive survey on coverage problem in the literature which includes proposals from the very first work introducing this problem until the very recent ones. First of all, we make a classification of those work. For each category (which will be presented in a sub-section), we tabulate a brief comparison of work belonging to that category. We then discuss each work in its category in more detail.

V.1. Classification of coverage algorithms in the literature

V.1.1. Classification metrics

The major objective of coverage problem is to efficiently cover an area or a set of points or objects (targets) under various constrains and limitations of sensors and networks such as energy, computational capability, memory, communication, bandwidth, high failure-rate, etc. while satisfying a number of requirements such as the maximum lifetime and surveillance quality. The fault tolerance issue has also intensively been investigating in literature. In WSN, it is normal to assume that the number of deployed sensors is much larger than the number needed to cover the whole monitored area. The common approach in the literature to solve the coverage problem while prolongs the network lifetime is to partition the set of the sensors into a number of set covers. Each set cover then successively becomes active to accomplish the sensing task while the others go to sleep to save energy. By that way, the lifetime of the network is extended. Various criteria can be used to classify both the problems and solutions into groups:

- The coverage problems for sensor networks can be categorized into three broad types [CAR04] – area coverage (in which, the major objective is to monitor an area), target coverage (where the main objective is to cover a set of targets), and breach coverage (the

goal here is to minimize the numbers of uncovered targets or the ratio of uncovered portion to the whole area).

- The algorithms can be centralized (where the algorithm is executed in a special station and usually requires the global information of the whole network) or decentralized – localized and distributed (where all the sensors simultaneously run the algorithm based upon only local information).
- To enhance network lifetime, most of the work done to date divides the set of sensors into a number of subsets such that every subset, i.e., set cover, can solely accomplish the coverage task; each of them will then be successively activated. In some work, the sensors are organized into disjoint subsets, i.e., subsets that share nothing in common. On the other hand, dividing into the non-disjoint sets (where a sensor can belong to different subsets) is considered in most distributed algorithms.
- Most of the work assumes that the sensors are randomly deployed into the monitored region. For a friendly and accessible environment such as a building, the deterministic deployment method (in which the sensor's position can be determined) can be applied to optimize the coverage level, connectivity and economize energy consumption.
- Only a few works consider the *p*-percentage coverage. That is, cover *p* percent of the whole area as opposed to cover the whole area (*100*-percentage coverage).

V.1.2. Our classification

In this thesis we classify the existing work into following categories:

- Pre-deployment: The algorithms concerned with the following issue (which is 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 methods 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].

V.2. Terminologies, notations and conventions

For the sake of consistency on describing and criticizing work in literature, in this sub-section we list notations and conventions that will be used through this thesis if they are otherwise clearly stated:

- Many problems in WSN field are intrinsically graphic problems, thus the term “sensor”, “node” or “sensor node” are interchangeably used through this text.

- *Set cover*: the subset of the set of all sensors which can solely cover the whole monitored region.
- *Disjoint set cover* (or *disjoint set* for short): the number of set covers who share nothing in common, i.e., each sensor can belong to at most one set cover. *Non-disjoint set cover* (or *non-disjoint set*): A sensor can belong to zero, one or more set covers. In this thesis, we call the approach that give disjoint set covers solution disjoint approach and the similar name for non-disjoint one.
- *Area coverage*: the type of coverage problem of which the major goal is to cover the whole given region.
- *Target or Point coverage*: the type of coverage problem of which the main target is to cover the given set of targets or set of discrete points.
- *n, m,* and *k* is number of sensors, number of targets, and number of set covers, respectively. *S, T* is the set of sensors, targets respectively. s_i (small s) is sensor *i*, and S_j (capital S) is set cover *j*.
- *k-coverage*: The problem considering to cover the whole area or set of targets satisfying that each point in the area or each target is covered by at least *k* sensors at the same time – in this sense, *k* is the coverage level or coverage degree unless otherwise explicitly stated.
- *Base Station (BS)*: The very special data processing center which is assumed to have unlimited computation and memory capacity. Thus, the centralized algorithm is usually executed at BS. BS can also have unlimited bandwidth to the outside world, and it usually works as a gateway for a sensor network to communicate with outside world.
- r_c, r_s: Sensor's communication range, sensing range, respectively.

V.3. Pre-deployment

For applications that permit to manually deploy the sensors, 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 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). Table III shows a comparison of work that will be discussed in this sub-section:

Table III. Work in literature that considers pre-deployment 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; take boundary effect into account.
[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).
[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.

V.3.1. Coverage/connectivity conditions

In [WAN06], Wan and Yi investigate the issue on how the probability that a WSN can *k*-cover an area varies depending upon the sensor' sensing radius and the number of the sensors. This work considers two kinds of deployments which are uniformly random deployment and Poisson point process deployment. Especially, this work first time takes into account the boundary effect which is very technically challenging to handle. Most of existing work concerning the similar issue uses the toroidal metric to avoid the boundary effect because the coverage level of border part tends to be smaller than the center part under the Euclidean metric. The main result of the theoretical analysis of this work is the probability for an area is *(k+1)*-covered. However, the resulted formulas are not cited here due to their length and intricacy.

Similarly to [WAN06], [KUM04] also considers the *k*-coverage problem for an area. However, the question it answers is the number of the sensors needed to provide *k*-coverage for an area under three kinds of deployments: *a)* $\sqrt{n} \times \sqrt{n}$ grid – where each of the *n* grid point hosts a sensor, *b)* random uniform – where all the sensors have the same probability to be placed at any location and *c)* Poisson point process with the rate *n* under the assumption that each sensor is active with probability *p* and is dependent from the others. The authors make use the concept of *slowly growing* function $\phi(np)$ which is a monotonically increasing to infinity function (hence the name "growing") and is *o(loglog(np))* (hence "slowly") for their theoretical results. Those results are summarized as follows:

- If there exists a function $\phi(np)$ that satisfies a condition (whose formula is not cited here) then all the points in the region are *almost always k*-covered (meaning the probability that they are *k*-covered reaches to 1 when *n* reaches to infinity). And if there exists another

function $\phi(np)$ that satisfies another condition, then there may exist holes (the portion that is not covered) in the network.

- For different kinds of deployments, the conditions are different but the same results (the coverage level of the network) are concluded.

Although the results are really practical, the authors omit a clear guidance on how to find *slowly growing* function $\phi(np)$ or how to verify if such function exists or not.

V.3.2. Deployment schemes

Used the *path exposure* (see section V.5.2) as the criterion to evaluate the goodness of a deployment scheme, the focus of [CLO02] is to determine the number of sensors needed for a "good" deployment. The path exposure is a metric to estimate the likelihood that an object (target) to be detected when it traverses through a sensor network. The concept of this kind of metric is to be better investigated in section V.5.2. In this work, the total energy (of signal) that sensor s_i can measure when a target is at the position u is formulated as:

$$E_i(u) = \frac{K}{\|u - s_i\|^k} + N_i$$

where K is the energy emitted at the target, k is decay coefficient (typically from 2.0 to 5.0), $\|u - s_i\|$ is distance from the sensor s_i to the target and N_i is noise energy at s_i. The probability $D_v(u)$ that the total $\sum_{i=1}^{n} E_i(u)$ for all the sensors of the network greater than a threshold η is the probability that the target to be detected by the network when it is at the

location u: $D_v(u) = \text{Prob}\left[\sum_{i=1}^{n} E_i(u) > \eta\right]$. However, because of the presence of noise, the probability of consensus false target detection (false alarm) is $F_v = \text{Prob}\left[\sum_{i=1}^{n} N_i > \eta\right]$.

The exposure of a path can be estimated by the probability that an object is detected anywhere along the path, i.e., by $D_v(u)$. The authors divide the monitored area into a fine grid, and then assign each grid segment (an edge) a weight equal to the total of $D_v(u)$ for all the points within that segment. To find the least exposure path, the authors utilize Dijkstra algorithm to find the path having minimum weight. Obviously, the precise of the solution highly depends on how fine the grid is. This is also the method used in [MEQ01] (section V.5.2) to find the minimal exposure path.

The objective of the paper is to deploy the sensor such that the exposure of any path through the sensor network is minimized. The solution is to deploy sensors one at a time, and each time a new sensor is deployed, the least exposure path is re-calculated. This method can reduce the number of deployed sensors; it however consumes relatively a lot of time for exposure calculation and introduces communication overhead while every newly deployed sensor has to report its location.

In [ZOU03], Zou and Chakrabarty propose virtual force algorithm (VFA) and target location query. The idea is to use the virtual force to find the "good" location for a sensor. Virtual force is similar to the force that two magnetized objects exert on each other – the force is attractive if both objects are positive or negative, and is repulsive if they are different. With the virtual force, several parameters can be taken into account on calculating the force imposing on each sensor such as the obstacles, the neighbors (which exert the repulsive force on the sensor

since the sensor tends to keep away from them for the better coverage and smaller number of necessary sensors) and preferential areas such as an area with low radio interference (which exerts the attractive force on the sensor since those areas can better support the coverage task of the sensor). Each kind of force is mathematically formulated, so it is easy to calculate the total force (consisting of the magnitude and the orientation) exerting on a sensor. Other parameters can as well be added with their own formulas. The process to find the proper position for a sensor is as follows. Firstly, the sensor network is randomly deployed. The virtual force for each sensor is then calculated based on sensors' locations, and each sensor moves to new position by that force. This process of calculation the force and moving the sensor is looped for several times. Obviously, the bigger the number of loops, the better the coverage quality the resulted network can achieve but the longer the time for executing the algorithm.

[POD04] uses the similar idea of the force as in artificial potential field to calculate the position for each sensor with the objective of maximizing the coverage and each sensor has at least K communication neighbors. To attain both the coverage and the number of the neighbors requirements, two kinds of forces are introduced: F_{cover} - the force that neighboring sensors repel each other to increase the coverage area (by decreasing coverage overlapping) and F_{degree} – neighboring nodes attract each others to satisfy the constraint of at least K neighbors for each sensor. The total force exerting on a sensor is the summation of net force (i.e., F_{cover}+ F_{degree}) of all of its neighbors. Initially, each node has more than K neighbors. Each node then calculates the above summation force and keeps moving under the exertion of that force until it has only K neighbors left. This work bases on the assumption of uniform and isotropic-kind of sensor network – i.e., all the sensors have the same sensing and communication ranges and they are as well isotropic in both sensing and communicating – which is not always the case in practice and

sometime is hard to achieve. Besides, the algorithm is for mobile sensor network rather than the traditional wireless sensor network of which sensors are not self-movable. Moreover, the authors have no discussion on the boundary part of the monitored area where not every node always has more than *K* neighbors if the network is randomly deployed.

With the objective of minimizing the number of sensors needed to provide both coverage and connectivity for a region, [KAR03] proposes deployment schemes for three cases: *a)* for infinite convex 2-dimentional region, *b)* for finite 2-dimentional region and *c)* for a set of targets. However, it only considers networks of which sensing radius and communication radius are equal and they are also identical for all the sensors. The proposed deployment schemes highly depend on this assumption which is not reasonably practical. The basic "piece" for all the proposed schemes is the *r-strip* (*r* is sensing/communication range) – as being shown in Figure 9 – of which sensors are placed side by side and the distance between two adjacent sensors are *r*. For the infinite 2-dimentional region (case *a*), infinite number of *r-strips* are placed such that they are horizontally parallel to each others (and parallel to x-axis) and $(\frac{\sqrt{3}}{2}+1)r$ apart. To attain coverage, the two adjacent *r-strips* are placed *r/2* vertically apart from each other. To achieve the connectivity, an addition vertical strip is added along the y-axis. The similar scheme is employed for finite region (case *b*) except the strip for connectivity may not be vertical; that strip is placed in the angle such that it intersects all the horizontal *r-strips* and the intersection points need to be inside the monitored region. For the point coverage case (case *c*), the minimum spanning tree (MST) reaching all those points is firstly constructed and the sensors are then deployed along the edges of that tree.

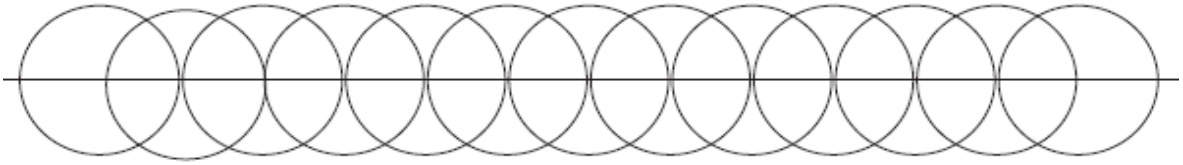

Figure 9. An *r-strip* [KAR03]

As in [KAR03], [WAH05] also proposes a sensor placement scheme. The striking point of this work is that it can deal with an arbitrary shape of the monitored area as opposed to the assumption of an open and/or rectangular area as most of the existing work has to depend on. It also allows sensors to have any ratio between communication range and sensing range. The objective is to properly place the sensors such that both connectivity and coverage are attained and the number of deployed sensors is minimized. The idea is to partition the arbitrary-shaped region into a number of sub-regions such that each sub-region is a polygon. The problem becomes deploying sensor for each sub-region. In the fields of pre-deployment, the following fact is well and widely known: three sensors have the sensing range of r_s can cover the maximum continuous area if they locate at the vertices of an equilateral triangle whose edge's length is $\sqrt{3}r_s$ (see Figure 10). If ratio between communication range and sensing range is $\sqrt{3}$, both the connectivity and coverage requirements are satisfied if sensors are places at those vertices. By that reason, the algorithm tries to optimize the number of sensors needed to fit into the sub-region for several cases in which the relation between communication ranges r_c and sensing ranges r_s are: $r_s > r_c; r_s = r_c; r_s < r_c < \sqrt{3}r_s$ *and* $r_c > \sqrt{3}r_s$. The proposed algorithm works only with uniform sensor network. Besides, the solution is definitely not optimum since there may exist unnecessary coverage overlapping between neighboring sub-regions (beside these between sensors within the same sub-region). Also, it is not easy to place the sensors exactly at the

positions figured out by the algorithm. At last, it is really difficult to extend the algorithm such that it can function with the region having some curved portions on the border (e.g., a cone).

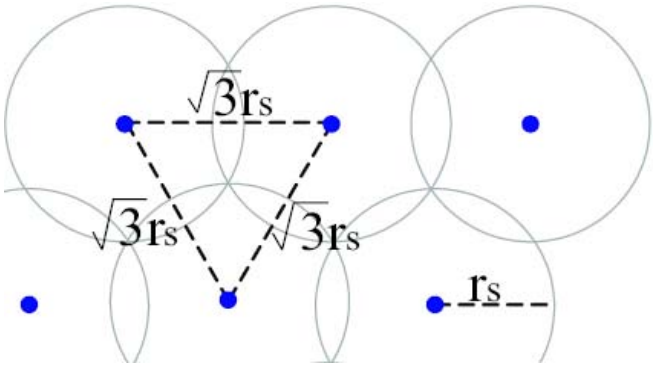

Figure 10. Position of sensors to minimize coverage overlapping [WAH05]

V.4. Sensors scheduling to achieve coverage/connectivity.

With a centralized approach, the algorithm usually runs at a special and powerful center (usually a base station) 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.

V.4.1. 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 also through out 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. Table IV itemizes work that will be considered in this sub-section

Table IV. 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 the network lifetime for smoothly adjustable sensing range sensor network under energy constraint.
[BER04]	YES	Non-Disjoint	Area/ Target	Considering partial *q*-coverage/ Taking communication cost into account.

A. Centralized algorithms that result in disjoint set covers

[SLI01] is one of the first work dealing with the coverage problem. Thus, the proposed algorithm is relatively straightforward and inefficient. The paper formulates a decision problem named "SET K-COVER", which is used by several subsequent papers, as follows:

INSTANCE: Collection C of subsets of a set A, a positive integer K.

QUESTION: Does C contain K disjoint covers such that each cover contains all the elements of A.

The authors prove that this problem is NP-Complete by transforming from the minimum cover problem. Thus, a heuristic for "SET K-COVER" is provided of which the objective is to maximize K. Firstly, the heuristic greedily selects a critical element (the most sparsely covered element). And among all the sensors being able to cover this element, it then chooses the sensor having biggest objective function value to add to current disjoint set of sensor. The objective function measures the number of critical elements that chosen sensors.

Besides, the authors suggest a method to covert from area coverage into target coverage as being illustrated in Figure 11. The paper first defines the concept of field which is a set of points that are covered by the same set of sensors. For example, the sensors in Figure 11 partition the monitored area (the dashed-line rectangle) into eight fields. By considering each field as a target, the area coverage problem is easily and accurately transformed into the target problem. In our own opinion, this is big contribution since it eliminates the border of two broad groups of the coverage problem: Area coverage and Target coverage.

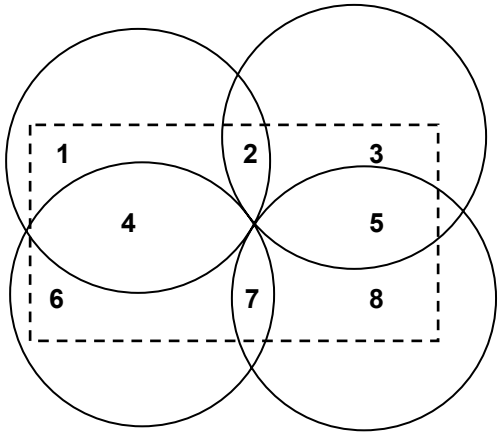

Figure 11. Transformation from area coverage to targets coverage [SLI01]

In [ABR04], the coverage problem is as well abstracted into a problem named "the SET K-COVER problem". Although having the same name as the problem in [SLI01], the objective of the problem in this work is different from that in [SLI01] and also different from *k*-coverage problem. In this problem, a set of areas and another set of sensors are given. A variable *k* is also given as a user-defined parameter. Additionally, each sensor can cover some areas. The objective is to efficiently partition the set of sensors into *k* set covers such that the number of times the areas is covered is maximized. In reality, this objective does not make much sense in most cases. This paper proposes two centralized heuristics as being discussed next:

Randomized algorithm

This very attractive algorithm is remarkably simple. It just randomly picks a sensor *i* and puts it into set cover *j*, where both *i* and *j* is a random number in the range *1..n* and *1..k*, respectively.

Despite the simplicity of the algorithm, the algorithm still guarantees the high coverage quality. The authors theoretically prove that an area is covered within $1-\frac{1}{e}$ of the maximum number of times possible. That is, the total number of times the algorithm's resulting set covers

being able to cover an area is equal to or greater than $1-\frac{1}{e}$ that of optimum solution. Moreover, the probability for the least covered area is covered within *ln(n*) of the maximum number of times possible is very high. Definitely, the time complexity of this algorithm is really low - *O(n)*.

Centralized greedy algorithm

Consider set of sensors $\{s_i\}$ where $i=1..n$ and set of areas $\{A_j\}$ where $j=1..m$. Each sensor s_i can cover a number of areas, which is denoted as $|s_i|$. In the centralized algorithm, each sensor maintains a table of size k x $|s_i|$ whose all the entries are initially assigned 1. At a given step of the algorithm, each area is assigned a *weight* of $(1-\frac{1}{k})^{y_v-1}$ where y_v is the number of sensors who can cover the area v but have not been assigned into any set cover. The pseudo-code for this algorithm can be shown as follows:

Algorithm 2: Centralized Greedy Algorithm

1: Initialize C={S_1:=0, …, S_k:=0 } // *set of set covers*
2: **For** i=1 **to** n
3: Find $j = \max \sum_{v:v\in s_i \wedge v\notin \cup_{s_j\in S_j} s_i} (1-\frac{1}{k})^{y_v-1}$
4: $S_j := S_j \cup s_i$ /* Assign sensor s_i to set cover S_j */

At each step, the algorithm selects a sensor having biggest summation of the weights of uncovered areas that it can covers to the set cover. The idea behind this algorithm is to greedily choose a sensor which covers the largest possible uncovered areas to add to the set covers. The weight of each area is a parameter measuring how likely that area will be covered in the future iteration. This algorithm guarantees the performance ratio of $1-\frac{1}{e}$ in compared to the optimal

solution. The time complexity of this centralized algorithm is $O(n) = nk|\ s_{max}\ |$ where $|s_{max}|$ is the biggest number of areas that a sensor can cover.

The objective of the problem that [CHE05] aims to is to divide the set of sensors into a number of disjoint set covers such that the cardinality of each set is not over a number W, which is the bandwidth of the network, and the possibility for any target *t* not being covered by any sensor in the whole lifetime of the network is minimized. The paper's contributions include the mathematical ILP-based (Integer Linear Programming) formulation for above problem, the proof of its NP-Complete property and suggestion of two heuristics. The first heuristic is the widely-used RELAXATION method. In ILP, some variables are required to get the integer values which make ILP to be NP-Complete problem. So the first step of this method is to relax the ILP into LP problem. Any available LP algorithm is then used to solve that LP problem; and finally, greedily convert the LP result back to ILP result. This method's advantages are the easiness to use and the capability of flexibly applying to numerous problems. However, its drawbacks are inefficiency and very slow running time. By using the state-of-the-art LP algorithm [YYE91], the running time of this first heuristic is $(n(n+m)/W)^3$. Because of that reason, Cheng et al. propose another heuristic named MINBREACH. Of which, instead of using LP to find the solution satisfying the huge number of constraints presenting in the problem formulation, this heuristic combines altogether the constraints into one constraint (in the form of a big-sized matrix) and solves that combined formulation. Thus, the running time significantly improves to $O(n^2m(n+m))$.

[GAO06] utilizes the *k*-NC rule proposed by Huang & Tseng [HUT05] to verify the condition for an area to be *k*-covered. By that rule, each sensor in the network only needs to check its perimeter (the border line of its sensing region) to determine the coverage level of the whole monitored area. If all the sensors' perimeters are *k*-covered; i.e. all the points in their

perimeter are inside the sensing ranges of at least k other active sensors; then the whole area is k-covered. That is the reason why the authors firstly introduce a parameter named *Perimeter-Coverage-Level* (*PCL*) for each sensor. PCL of a sensor is the number of sensing neighbors that cover any point on that sensor's perimeter. The algorithm then greedily chooses sensors into set cover by their PCL. That is, the algorithm greedily select sensor with the biggest PCL to add into set cover until all the sensors belonging to current set cover can provide k-coverage for the each sensor's perimeter. However, because of greedy method, there may exist some redundant sensors in newly-created set cover. Thus, one more step to reduce the set cover size is carried out by executing a procedure named *PruneGreedySelection* right after a new set cover is created to prune any possibly redundant sensors. The running time of proposed algorithm is $O(n^2dlogd)$ where d is the degree of the network (maximum number of neighbors that a sensor may have). It is proven in this work that the number of set covers the algorithm being able to produce is $\left\lfloor \frac{K}{k} \right\rfloor$ where K (big K) is the coverage level that the whole sensor network can provide (when all the sensors turn on concurrently). At last, the authors present a formula concerning the density of sensors close to border line of the monitored area for the algorithm to produce more set covers (by balancing the sensors density between the center region and border-line region).

Besides solving the k-coverage as [GAO06], [ZHO04] also considers connectivity problem. This paper simultaneously attacks this combined problem by introducing the concept of *K-Benefit path*. As a definition, the *K-benefit* is the ratio between the numbers of "new" *valid sub-elements* (a valid sub-elements is an area covered by the same set of sensors and is inside the considered region). A sub-element is said to be "new" if it is currently covered by less than K sensors. The centralized algorithm in this paper constructs the set cover S_j by greedily adding a

sensor from the set of sensor $S^* \subseteq S$ which has the maximum K-benefit. The set S^* here is the set of sensors that can communicate with at least one sensor in set cover S_j (so the connectivity is maintained). The authors also prove that the size of the set cover is at most $2r(\log Kn)|OPT|$ where r is the maximum communication distance, in terms of number of hops, between any two sensors whose sensing regions overlap.

B. Centralized algorithms that result in non-disjoint set covers

It can be observed that non-disjoint set covers can provide better lifetime as compared to disjoint set covers. Let us take an example to illustrate this observation. Figure 12 shows a topology taken from [CAT05]. Assume that each sensor can be continuously active for 1 unit of time. For disjoint approach, the optimal solution could have at most two set covers, thus the network life time is 2. With the non-disjoint approach, we can divide sensors into four set covers: $S_1=\{s_1, s_2\}$, $S_2=\{s_2, s_3\}$, $S_3=\{s_1,s_3\}$ *and* $S_4=\{s_4\}$. S_4 activates for 1 unit of time, and the others activate for 0.5 unit of time. Totally, the network lifetime is 2.5, which is 25% better than the disjoint solution. In this sub-section, we will examine several typical recent algorithms that produce the non-disjoint set covers.

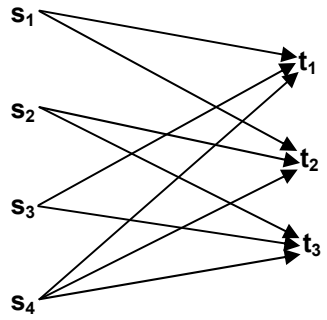

Figure 12. A sample topology [CAT05]

In, [CAT05] the authors abstract the objective of maximizing the network lifetime under Maximum Set Cover (MSC) problem. MSC is proven to be NP-Complete by a polynomial transformation from a well-known NP-complete problem – 3-SAT.

The MSC problem is formulated as ILP formulation with the objective of maximizing the summation lifetime of all the set covers. To better illustrate the non-disjoint property of solution of MSC, we cite that formulation as follows:

Maximize $time_1 + ... + time_k$. *//Maximize the network lifetime*

Subject to $\sum_{j=1}^{k} x_{ij} time_j \leq 1$ for all sensor s_i. *//Sensors' lifetime constraint*

$\sum_{i \in C_l} x_{ij} \geq 1$ for all target t_l and j=1,..,k. *//Coverage constraint*

Where $x_{ij} = 0,1$ ($x_{ij}=1$ iif $s_i \in S_j$).

In which $C_l = \{ i \mid$ sensor s_i cover target $t_l\}$ and $time_j$ is time that set cover S_j will be active

As can easily be seen from formulation, no relation between set cover S_j ($j=1,..,k$) is specified. That allows them to share some sensors in common.

Based on that definition, the authors propose two heuristics:

Using LP Relaxation

Once again, LP relaxation is used to solve ILP problem. Similar to [CHE05], this heuristic transforms the ILP into LP by relaxing equalities requiring integer values for variables into the inequalities which allow those variables to get non-integer values. Using any existing LP algorithm to solve the relaxed problem, and then greedily convert LP result back to ILP result.

Centralized greedy algorithm

Similar to [SLI01], this work uses the idea of critical element. In this paper, the critical targets can be understood as the most sparsely covered targets, both in terms of the number of sensors being able to cover those targets and their residual energy. At each step, the algorithm chooses a critical target and selects the sensor having the biggest contribution to cover that critical target. The contribution of a sensor can be defined as the number of targets that it is able to cover (i.e., within it sensing region) and its residual energy. However, the authors do not mathematically model the concept of sensor's contribution and critical target.

[CAW05] may be considered as the generalization of the work in [CAT05] just mentioned above. In this work, the authors consider a special type of sensors which can discretely adjust its sensing range. Each sensor can work at the fixed number of sensing ranges, hence the term "discretely" as opposed to "smoothly" adjustable sensing range sensor investigated in [DHA06] which will be discussed shortly. The authors abstract the objective of maximizing the network lifetime under the Adjustable Range Set Covers (AR-SC) problem. AR-SC is also NP-Complete problem since MSC ([CAT05] above) is its special case.

Three algorithms are proposed in this paper. Two of them are similar to two heuristics discussed in [CAT05]. In ILP-based approach, although the formulation is adapted to meet the sensors' adjustable sensing ranges assumption, the solution is still similar to that of [CAT05]. In the centralized greedy approach, the contribution is mathematically defined in terms of energy consumption (corresponding with sensing range) and the number of uncovered targets that sensor can cover. However, the main idea of the algorithm is fairly similar.

As another extension of [CAT05], [THAI05] in fact solves the problem of minimum breach with bandwidth constraint mentioned in [CHE05] using a non-disjoint approach. The

authors as well consider the maximum network lifetime issue under bandwidth constraint which is generalization of the work in [CAT05]. The authors formulate both problems as ILP problems and use the RELAXATION method as being used by [CAT05] and [CHE05] to solve. However, the algorithm can produce non-disjoint set, thus the result for the "Minimum Coverage Breach Problem" is better in compared with [CHE05].

[DHA06] is the first work dealing with a special type of sensors - sensors with smoothly adjustable sensing range. This type of sensors can flexibly adjust its sensing range to arbitrary distance within a certain max value. The authors formulate the maximum lifetime problem under a LP formulation. However, instead of traditionally using a LP algorithm to solve, this paper applies an approximate but very fast method (we call it "method" instead of "algorithm" since it can be used to solve problems other than packing LP) named Garg-Könemann [GAR98] to find a number of set covers. Following is the frame of the Garg-Könemann (GK) method:

Algorithm 3: Garg-Könemann (GK) method

```
1: Initialize the same weight for all the sensors
2: While (true)
3:     Call f-approximation function to find a set cover
4:     Update the weight for each sensor
5:     If (the stop condition is true) break;
```

In this combined algorithm, each sensor is assigned a new property, namely weight. The weights of all sensors are equally initialized. In this paper, the typical greedy algorithm is used as *f-approximation* function with the responsibility of finding set covers. After discovering a set cover, the weights are updated in order to help the greedy algorithm find the "better" set cover at next Garg-Könemann's iteration. Let us describe how the greedy algorithm (*f-approximation*) works: the greedy algorithm selects a sensor with the biggest contribution (here, contribution of a sensor is defined as the ratio between the number of uncovered targets that it is able to cover and

its sensing range) along with the smallest weight to add to set cover. Theoretically, the combined algorithm can discover as many as 2^n set covers. However, to improve the performance, this paper stops when it discovers n set covers.

V.4.2. 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 V provides a brief list of work that (not all, however) will be considered in this sub-section along with some of their characteristics.

Table V. 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.
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.

A. 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 this 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 13). 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.

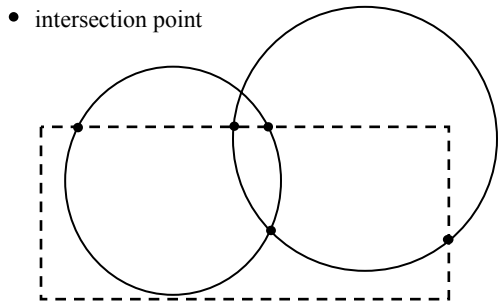

Figure 13. Intersection points example

B. 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 which 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 V.4.1 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" (refer to section V.3.2 - Figure 10 for an example). 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 14. When a node receives that message, it adds this neighbor to its neighbors list and checks if its sensing region is covered by sensors in its neighbors 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.*

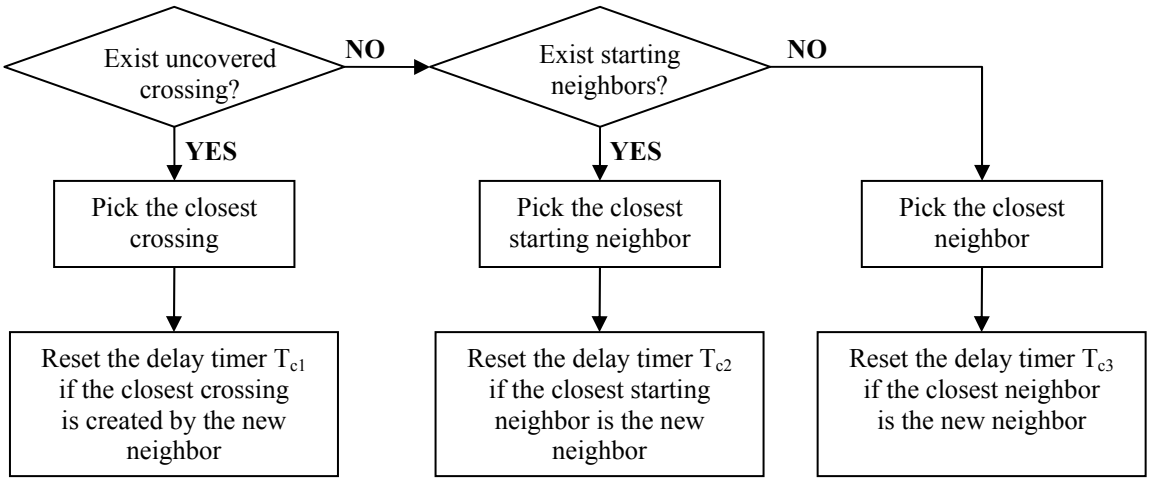

Figure 14. The procedure when a node receives a power-on message [ZHA03]

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 15 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.

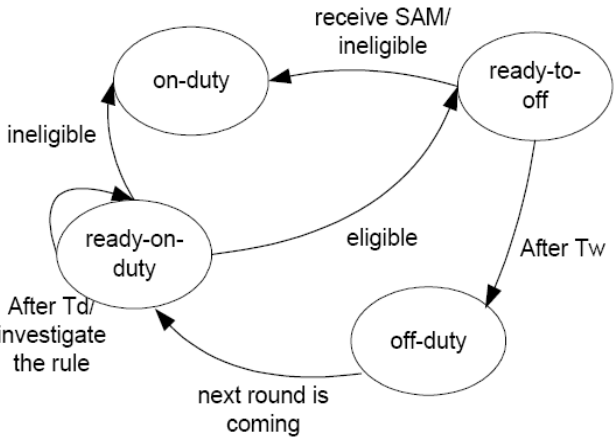

Figure 15. The status transition graph [TIA02]

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.

V.4.3. 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 16.a illustrates such arches. The angles corresponding with those arches are shown in the Figure 16.b. From

Figure 16.b, the coverage level of sensor s_i perimeter can easily be calculated by traversing the range from 0 to 2π.

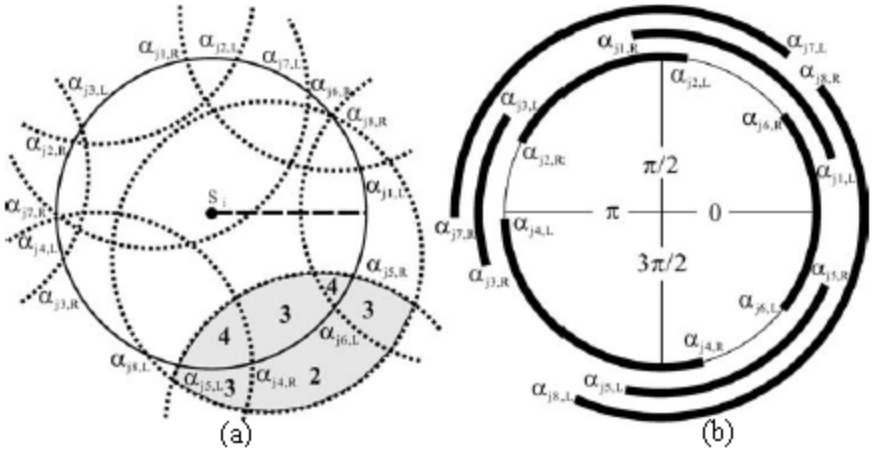

Figure 16. Transform to perimeter coverage [HUT05]

Surprisingly, this statement can as well be applied for sensors with irregular-shaped sensing regions. Figure 17 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.

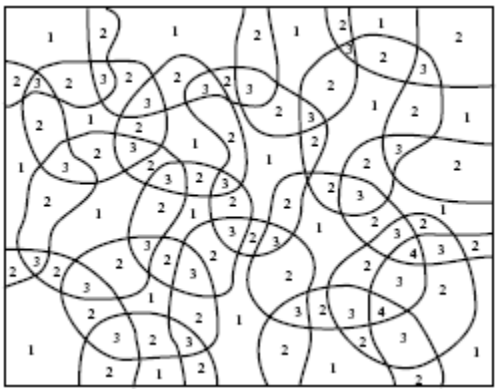

Figure 17. Irregular sensing regions [HUT05]

Beside two centralized algorithms discussed in section V.4.1, [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.

V.5. 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. By that reason, this

section is divided into two sub-sections corresponding with those two metrics. Work considering evaluating a sensor network are listed in Table VI:

Table VI. 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].

V.5.1. Maximal breach/ support paths

Maximal breach path (or *worst-case coverage*) shows the goodness of a sensor network in terms of coverage quality in worst case, i.e., in the case that an object is least likely to be detected. Or in other word, it measures the vulnerability of the network. As an informal definition, maximal breach path is the path that an object *p* moves from source *s* to destination *t* through the sensor network such that the distance from object *p* to *the closest sensor* is *maximized*. With the *maximal support path* (or *best-case coverage*), on the other hand, that

distance is desired to be *minimized*; that is, it measures the efficiency of the network. It can be seen that maximal breach/support path corresponds with the worst/best case coverage of the sensor network, respectively, i.e., the worst/best coverage quality on detecting a moving object that the network is able to provide. In literature, best-case coverage and maximal support path are equivalent. So are worst-case coverage and maximal breach path. Usually, there exist more than one maximal breach or support paths through a network. In addition to being as metrics for evaluating coverage quality, the constructions of those paths have several real applications such as intruder detection, or moving object protection. It is worth noting that although in [CHE05] and [THAI05], the issue of breach is also discussed (refer to section V.4.1 above for more detail), however, the goal of those papers is to create set covers that minimize the probability of undetected object in the monitored field, not to consider the quality of a deployed sensor network.

[MEK01] is the very first work addressing the maximal breach/support path ([MEG05] is journal version of [MEK01]). In this work, the *Voronoi diagram* and its dual *Delaunay triangulation* structures are utilized as base structures for finding the maximal breach path and maximal support path, respectively. Given a set of points in two-dimensional plane (corresponding to the set of sensor nodes S), *Voronoi region* (or *Voronoi tessellation*) of a sensor s_i is set of points that are closer to s_i than any other sensors in set S. For each sensor s_i, Voronoi region is a convex polygon whose vertices are called *Voronoi vertices*, edges are called *Voronoi edges*. The *Voronoi diagram* of set S is the union of Voronoi regions of all the sensors in S. So if connecting each sensor node with neighbors who share common Voronoi region's boundary, a dual structure of Voronoi diagram is created. This dual structure is *Delaunay triangulation*. If no three points are on the same line and no four sensors are co-circular, then each Voronoi vertex

has exactly three edges incident on it and thus Delaunay triangulation exists and is unique. The property of maximal distance to a sensor of Voronoi diagram resembles the objective of maximal breach path. While Delaunay triangulation can be used to find the closest neighbor by considering the shortest edge of triangulation. These properties could be directly applied to find the maximal breach/support path. It is now clear that maximal breach path must go along edges of Voronoi diagram while edges of maximal support path must lie on edges of Delaunay triangulation. The algorithms for finding those paths first assign a weight to each edge: with Voronoi diagram, weight of an edge is the distance from it to the closest sensor in S; with Delaunay triangulation, weight of an edge is its length. The algorithms then search by binary-search for the largest weight (namely *breach-weight, support-weight* for maximal breach/support path, respectively) such that edges having weight equal or bigger than that weight (breach-weight or support-weight) can still make a path, i.e., can continuously connect the source *s* and destination *t*. The algorithms use Breadth-First-Search (BFS) to check the connectivity of edges qualifying for weight filtering (i.e., equal or bigger than breach-weight or support-weight). Those algorithm are centralized ones with worst-case complexity of $O(n^2 logn)$. Although the results look optimal, this work lacks of theoretical analysis; or in other word, the algorithms are likely to base on intuition other than solid theoretical support. [LIW03] (below) provides the compensation for that insufficiency and shows that this theoretical deficiency actually leads to faulty.

[LIW03] can be thought as the compensation and extension of [MEK01] discussed above. The definitions and theoretical analysis are systematically provided. However, this work only considers best-coverage-path and some of its extensions, which is intrinsically maximal

support path as in [MEK01]. To facilitate the proposed algorithms, the authors first define two kinds of graphs:

- Constrained *relative neighborhood graph* over set of nodes V, denoted by *RNG(V)*: RNG(V) has an edge (u,v) only if there is no other node in $B(u,\|uv\|) \cap B(v,\|uv\|)$ (this intersection region is called *lune*) of which $B(u,\|uv\|)$ is disk centered at u and has radius of distance from *u* to *v*.
- Constrained *Gabriel graph* over set of nodes V, denoted by *GG(V)*, GG(V) has edge *uv* only if $\|uv\| \le 1$ and the closed disk with diameter *uv* (i.e., disk centered at central point of edge *uv* with radius of $\frac{\|uv\|}{2}$) does not contain any other node.

Based on those structure graphs, the authors propose a decentralized best-coverage-path algorithm and two of its extensions. The common step in all three algorithms is to create the edge from source *s* and destination *t* to their closest sensors; and these edges will be part of discovered paths.

- ***Algorithm 1:*** *Best-Coverage-Path*: The objective is to find the path that minimizes the maximal distance to the closest sensor, which essentially is maximal-support-path. Instead of searching the path along Delaunay triangulation edges as in [MEK01], this work searches on much more smaller graph which is RNG. Each node *u* bases on its 1-hop neighbors list $N_1(u)$ to construct its local RNG graph. For each neighbor *v* of this local RNG, the edge *uv* is assigned a weight of $\frac{1}{2}\|uv\|$. The next step is to use a distributed variant of Bellman-Ford algorithm to discover the shortest path. The weight β of the path is the weight of the edge

(belonging to that path) with maximum weight. The time and communication complexity are $O(n^2 \log n)$ and $O(n \log n)$ bits, respectively.

- ***Algorithm 2:*** *Energy-Conserving-Best-Coverage-Path*: The objective is to find the path that the maximal distance to the closest sensor and energy consumed by that path are both minimized, i.e., best-coverage-path with minimum energy consumption. First, *algorithm 1* is executed to find the value of the shortest-path weight β. Each node then constructs its local GG graph and prunes all the edges having weight larger than β. Each of the remaining edges is assigned a weight which is proportional to the total energy consumed by that edge. Then the distributed short-test path algorithm is applied. The path with minimum total weight will be the final solution. The time and communication complexity are both $O(n \log n)$.

- ***Algorithm 3:*** *Small-Travelling-Best-Coverage-Path*: The objective is to find the best-coverage-path that the maximal distance to the closest sensor is minimized and the total length of the path is not more than 5/2 of the shortest one. Firstly, *algorithm 1* is run to find the value of the shortest-path weight β. Each node then constructs its local Delaunay graph (*LDEL*) and prunes all the edges having weight larger than β. Each of the remaining edges is assigned a weight of its length. Then the distributed short-test path algorithm is applied. The path with minimum total weight will be the solution. The authors confirm that the ratio between the length of best-coverage-path constructed by this method and that of the shortest best-coverage-path is no more than $\frac{4\sqrt{3}\pi}{9}$ (which is smaller than $\frac{5}{2}$). This is the trade-off between quality of solution and communication-complexity. The shortest path can be found

by using Unit Disk Graph (UDG), the communication complexity is however larger than that of this method which is *O(n logn)*.

The authors also provide correctness proof for their proposed algorithms and a proof for maximal-support-path algorithm in [MEK01].

Also considering worst/best-case coverage issues; [HUR05], instead of finding the maximal breach/support paths as above work, solves the problem of determining the maximum/minimum sensing range of a uniform sensor network such that there are totally uncovered maximal breach paths or completely covered maximal support paths. Those radii are called worst-coverage radius (denote r_{worst}) and best-coverage radius (denote r_{best}), respectively. Let *U(r)* be the region of union of coverage disks with radius of *r* centered at sensors, then *U(r)* should be the union of several connected regions for any fixed value of *r*. Let $\overline{U(r)}$ be the complement of *U(r)*. Similarly, $\overline{U(r)}$ as well contains some connected regions. If both source *s* and destination *t* are in the same connected region of *U(r)*, then there will exist a fully covered maximal support path. The smallest *r* for this to satisfy is called best-coverage radius r_{best}. Similarly, the biggest *r* such that *s* and *t* are in the different connected regions of $\overline{U(r)}$ is called worst-coverage radius r_{worst}. It follows that the problem of determining radius r_{best} and r_{worst} is translated into the problem of connectivity of *U(r)* and $\overline{U(r)}$, respectively. Let *G(U(r))* be the connectivity graph whose vertices are the sensor nodes in *U(r)*. *G(U(r))* has an edge *uv* if the coverage disks of *u* and *v* overlap. The problem of best-coverage radius now becomes determining the radius *r* such that vertices corresponding with disks containing source and destination are connected in *G(U(r))*. By trial and error method on a sequence of radius r_i such that $r_i = \alpha \times r_{i-1}$, the α-approximation of optimal r_{best} is easily be found in $log_2 R$ time where

$R=r_{max}/r_{min}$. To update r_{best} so as to maintain the connectivity of $G(U(r))$ while network topology changes, the kinetic data structure and the algorithm in [GUI01] is utilized. This guarantees $(1+\varepsilon)$-approximation of r_{best} for any $\varepsilon > 0$, update cost is $O(\log^3 n)$ and $O(\log n)$ query cost (for fixed ε). Another special data structure is used to maintain the connectivity region in $\overline{U(r)}$. However, before applying that data structure, the sensor's coverage region is transformed from 2-norm (which is a circle) to infinity-norm (which is square). The reason for this transformation is that the algorithm needs to maintain only 4 points for to keep track of sensing region of each sensor. The proposed algorithm, by maintaining more four points, the cost for updating the change in topology (because of sensors' leaving or joining the network) is limited to updating up to four edges. As consequent, the algorithm for worst coverage radius guarantees $(\sqrt{2}+\varepsilon)$-approximation of r_{worst} for any $\varepsilon > 0$, update cost is $O(\log^2 n)$ and $O(1)$ query cost (for fixed ε). At last, a centralized $O(n \log n)$ run-time algorithm to find the maximal support path is proposed which bases on r_{worst}.

V.5.2. Exposure

Although maximal breach/support path can be used to measure the coverage quality of a network, they however merely base on the distance from an object to sensors as criterion to evaluate the quality. This metric is too simple for such evaluation. Another criterion can be used to evaluate coverage quality is *exposure* which takes into account the intensity of the sensing signal and the cooperation of sensors in the network on accomplishing coverage task. Exposure path is similar to maximal breach path except parameters it considers. Because of more involved metric, constructing a exposure path is much more difficult than building a maximal breach path.

[MEQ01] is the very first work coping with this kind of problem and is the first to introduce the concept of *exposure* of a sensor network, more specific, its objective is to calculate a *minimal exposure path*. Before giving a formula for exposure, the authors first mathematically formulate the concept of *sensor field intensity* (or observability [LIW03]). Generally, *intensity* is the strength of signal an object radiates that a sensor can measure. The intensity that a sensor *s* "senses" the object *p* is $\frac{\lambda}{[d(s,p)]^K}$ where λ , *K* are constant and *d(s,p)* is Euclidean distance from *s* to *p*. Sensor field intensity is the intensity of signal strength from point *p* in field *F* that sensors can (co-operatively) measure. Two kinds of intensity are mentioned in this paper: *Closest sensor intensity* (denote $I_C(F,p)$) – the intensity of the sensor closest to the object; and *All-sensor field intensity* (denote $I_A(F,p)$) – the total intensity of all active sensors who can sense the object. Based on field intensity, the exposure of an object moving along the path *p(t)* in field *F* during interval (t_1,t_2) is formulated as following integral:

$$E(p(t),t_1,t_2) = \int_{t_1}^{t_2} I(F,p(t))\left|\frac{dp(t)}{dt}\right|dt$$

The objective of this paper is to find a path having the smallest value of exposure. A simple case, of which a network consists of only a single sensor, is considered as an example of a solution for exposure problem. In general case, however, finding the solution is much more difficult. The most challenging issue in computing the exposure of a path *p(t)* comes from its (the path's) continuous nature. To overcome this challenge, the continuous field is transformed to discrete one by two steps:

1. The field (here is the whole sensor network) is first partitioned into $l \times l$ squares where *l* is some integer number.

2. Each square is then further divided into $m \times m$ smaller squares, and minimal exposure path is restricted along edges or diagonals of those squares. See Figure 18 for an example of this transformation.

By that transformation, it is easy to calculate the integral for exposure since any path *p(t)* is the union of finite numbers of straight lines. After above transformation, the algorithm then utilizes Dijkstra's Single-Source-Shortest-Path algorithm to find the path with minimal exposure. Clearly, the precise of solution highly depends on the fineness of the transformation, i.e., the value of *m* and *l*.

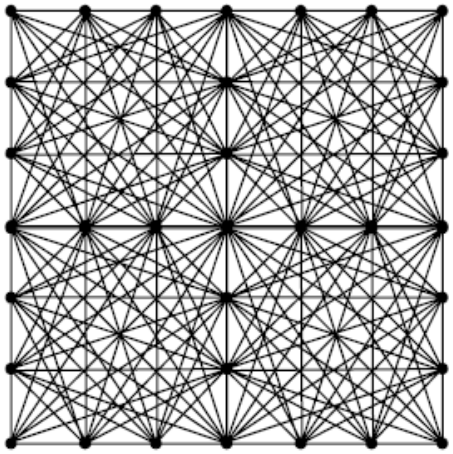

Figure 18. Third-order grid with *l=2, m=3* [MEQ01]

As mentioned above, it is difficult to calculate the exposure path for an arbitrary trajectory. Thus, the very first task that most of the work concerning exposure path needs to accomplish is to partition the space into parts such that the exposure can more easily be computed. In [MES01], the partition scheme is Voronoi diagram. The advantage of this scheme is that each sensor can calculate exposure based on local information (however, with a small error). The node s_i closest to the source *s* starts the path finding process by broadcasting *path_request* message to all the neighbors who share common Voronoi's edges with it, namely

"Voronoi neighbors". Node s_i also calculates and stores to profile exposure for each of its Voronoi's edges. Since it is difficult to calculate the exposure for a whole edge, exposure is computed for discrete sample points along that edge. Node s_i then sends *edge_update* to all the Voronoi neighbors. Each time a node s_j receives this request, it computes exposure for the common edge with sender of *edge_update* message if there is no exposure for that edge in its profile. It then combines its own profile's exposure with the one attached in *edge_update* message to get the minimum exposure for each discrete point along that common edge. If nothing changes (meaning the message's exposures are greater than s_j profile's exposures), s_j sends back *abort_update* message (without continuing searching for the path). Otherwise, it does all steps exactly as node s_i did (e.g., broadcasting *edge_update* to Voronoi neighbors, calculating exposures for Voronoi edges) in order to expand the path finding process. Also, there is a global user-specified variable λ which plays as exposure threshold, i.e., a node sends *edge_update* message to a neighbor only if there is a point in their common edge having exposure smaller than λ. When the path reaches destination *t*, the value of λ will be updated to be the minimum of the current value of λ and the exposure at *t*. This new value of λ is scattered back through the network. A new and better exposure path is discovered by the same mechanism as just discussed with the new lower exposure threshold λ. The path is then traced back from *t* to *s* (the authors intentionally omit this step in the paper). There are several drawbacks in this work. For example, each sensor may have to maintain a huge exposure profile. Similarly, the *edge_update* message may be reasonably large since it contains the exposure for all the points in an edge. Besides, the authors do not provide any theoretical analysis to support the idea behind using Voronoi diagram as a partitioning method. Also, no correctness proof which guarantees that the proposed

algorithm is indeed able to find a path limited by the last threshold λ (the one derived after a path reaches to destination *t*).

As opposed to [LIW03], [XUH05] solves the worst coverage problem, i.e., finds the worst-coverage-path. The frame of localized algorithm represented in this paper is similar to FindBestCoverage algorithm proposed in [LIW03]. However, the observability [LIW03] is used to evaluate the path instead of distance as in [LIW03] and [MEG05]. Note that with that method to evaluate a path, the problem is solved in this work may be placed in *Exposure* section rather than maximal breach/support path. Because of worst coverage problem, a variant Voronoi diagram named *aberrant Voronoi diagram* is used. Aberrant Voronoi diagram is the intersection between Voronoi diagram and monitored area. It is proven in this paper that there exists a path with minimal observability on the aberrant Voronoi graph. The other difference from work in [LIW03] is the introduction of Vertex Selection Algorithm (VSA) which is used to find vertices v_s, v_t for source *s* and destination *t*, respectively, such that the observability of final constructed path is not smaller than that at *s* or *t*. To find the v_s for source *s*, VSA starts at node u_s closest to *s*. u_s locally constructs its aberrant Voronoi region. The vertex of this local region which is closest to *s* will be chosen as v_s. The time and message complexity of this algorithm is *O(d logd)* and *O(d)*, respectively, where *d* is the maximum number of neighbor nodes that a sensor may have.

VI. CONCLUSION AND FUTURE WORK

In this work, we consider the *k*-coverage issue in wireless sensor networks which is formulated as SESK problem. We further propose a completely distributed and localized algorithm named DESK to solve that NP-complete problem. We as well provide analysis and simulation to support the correctness and efficiency of the proposed algorithm. Nonetheless, we leave the following issues as our future work:

- We may take network bandwidth constraint into consideration.
- We extend out work to deal with adjustable sensing range. By giving sensors the ability to flexibly adjust its sensing range; they can more energy-efficiently cover the area.
- We also plan to enhance our algorithm to monitor three-dimensional areas.
- In this thesis, we assume that the communication range it is at least twice the sensing range. Therefore, the *k*-coverage also guarantees *k*-connectivity [WAN03]. We do not consider the connectivity for the general case of which a WSN having an arbitrary ratio between sensing range and communication range. It would be our consideration in the future.
- In this thesis, we implicitly assume (as most other work in literature do) that sensor ‘s sensing region is isotropic and is non-attenuate, that is, a sensor can sense the same signal for every point within its sensing range and the same in every direction; which is not really practical. For our next work, we may take the sensor’s non-isotropic and attenuate attribute on sensing ability into consideration.

- For our future work, we would also take the *relaxation effect* of battery into account in the simulation section. We assure that would give better results.

In this thesis, we also conduct an extensive survey about coverage algorithms in literature. We first classify those work into groups, and give a brief comparison of work in each group and then discuss them in more details. We further point out advantages as well as drawbacks of each work which is being discussed. For papers (even are in the different sections) that relate, e.g., employ the same rule, we also give reference to each other.

VII. 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.

[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.

[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.

[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 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.

[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.

[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.

[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.

[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.

[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.

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

[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.

[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.

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

[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.

[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.

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

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

[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.

[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.

[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.

[WAL06] *G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz and J. Lees* "Deploying a Wireless Sensor Network on an Active Volcano", IEEE Internet Computing, 2006.

[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.

[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.

[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.

[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.

[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.

[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.

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