CONNECTIVITY AND COVERAGE PRESERVING SLEEP SCHEDULING MECHANISM WITH PREDICTIVE COVERAGE AND MULTIPLE MODE SELECTIONS IN WIRELESS SENSOR NETWORKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Eyuphan Bulut

July, 2007

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. İbrahim Körpeoğlu (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ezhan Karaşan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Özgür Ulusoy

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

CONNECTIVITY AND COVERAGE PRESERVING SLEEP SCHEDULING MECHANISM WITH PREDICTIVE COVERAGE AND MULTIPLE MODE SELECTIONS IN WIRELESS SENSOR NETWORKS

Eyuphan Bulut
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. İbrahim Körpeoğlu
July, 2007

Wireless sensor networks, which consist of a large number of sensor nodes communicating with each other in order to sense the environment, is an emerging field in the area of wireless networking. One of the significant objectives in the design of these wireless networks is the efficiency of energy consumption. Since these networks are densely deployed, sleep scheduling, which puts some sensor nodes into sleep mode and uses only a necessary set of active nodes, is a commonly used technique to extend the network lifetime.

A sleep scheduling algorithm should be distributed, simple, scalable and energy efficient. In this thesis, we investigate the problem of designing a sleep scheduling algorithm which extends the network lifetime while maintaining a user defined coverage and connectivity. We consider three basic units of a sensor node (sensing, processing, and communication) independently and turn a unit on whenever it is necessary. Furthermore, we analysed the expected overlap of a node's sensing area by its neighbors' sensing areas and created a predictive algorithm in the decision of a node's status. By this way, we reduced the number of messages for maintenance procedures. We evaluated our algorithm via simulations and compared the performance of our algorithm with another distributed algorithm which maintains both coverage and connectivity. As a result, we observed that our algorithm provides a significant increase in the network lifetime and overcomes the other algorithm in all cases.

Keywords: Wireless Sensor Networks, Sleep Scheduling, Network Lifetime, Connectivity, Coverage.

ÖZET

KABLOSUZ DUYUCU AĞLARINDA KAPSAMA VE BAĞLANTIYI KORUYAN, TAHMİN EDİLEBİLEN KAPSAMA VE BİR ÇOK DURUM SEÇMELİ UYKU DÜZENLEMESİ

Eyuphan Bulut
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Assist. Prof. Dr. İbrahim Körpeoğlu
Temmuz, 2007

Çevreyi duymak için birbirleriyle haberleşen çok sayıda duyucudan oluşan kablosuz duyucu ağları, kablosuz ağ alanında çığır açan bir teknolojidir. Bu kablosuz ağların tasarımında en önemli amaçlardan biri verimli enerji tüketimidir. Bu ağlarda yüksek sayıda duyucu bulunduğundan, bazı duyucuları uyutan ve gerekli sayıda aktif duyucu kullanan uyku düzenlemesi ağ ömrünü uzatan genel bir tekniktir.

Duyucu ağları için tasarlanmış bir uyku düzenleme algoritması dağıtık, basit, ölçeklenebilir ve enerji kullanımında verimli olmalıdır. Bu tezde kullanıcı tanımlı kapsama alanı ve bağlantı durumunu sürdürerek ağ ömrünü uzatan bu tarz bir dağıtık algoritma tasarımı sorununu inceledik. Bir duyucunun üç ünitesini de ayrı düşündük ve her üniteyi gerekli olduğunda aktif hale getirdik. Ayrıca, bir duyucunun algı alanının komşu duyucuların algı alanları ile beklenen örtüşmesini analiz ettik ve duyucuların durumunu karar vermede tahmin edici bir algoritma yaptık. Bu yöntem sayesinde, algoritmanın işlemleri için gereken mesaj sayısını azalttık. Algoritmamızı simülasyonlarla değerlendirdik ve performansını kapsama ve bağlantıyı sürdürebilen başka bir dağıtık algoritma ile karşılaştırdık. Sonuç olarak, algoritmamızın ağ hayatını önemli miktarda arttırdığını ve diğer algoritmadan daha iyi sonuçlar verdiğini gözlemledik.

Anahtar sözcükler: Kablosuz duyucu ağları, Uyku düzenlemesi, Ağ ömrü, Bağlantı, Kapsama Alanı .

Acknowledgement

Before all, I would like to thank to my supervisor Assist. Prof. Dr. İbrahim Körpeoğlu for his continuous support and invaluable concern throughout this research. Especially, his pleasant and friendly personality and precise guidance made this graduate study more enjoyable.

I also would like to thank to Assoc. Prof. Dr. Ezhan Karaşan and Prof. Dr. Özgür Ulusoy for evaluating my thesis.

I am grateful to my family for standing by me in everything I have done and giving me whatever they can. They have always provided me continuous support, encouragement and their love. I believe that nothing would be possible without the presence of them and the peaceful family environment they provided me during my life.

I am also grateful to my office friends for their supports and feedbacks on my research.

Finally, I would like to thank to TUBITAK for their financial supports during this graduate study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, advances in wireless communications and electronics have enabled the development of low-power and small size sensor devices. A *Wireless Sensor Network* (WSN) consists of a large number of these sensor nodes deployed in a geographic area. Each sensor node has three basic units; sensing unit, processing unit and communication unit. Depending on the type of the sensor it uses, the sensing unit can sense light, temperature, humidity, sound, motion, vibrations, etc. around its location [1].

Sensor networks are utilized in a wide range of applications. Some of the application areas and how they can utilize sensor networks are the following:

- Military: Battlefield surveillance and monitoring to detect enemies, guidance systems of intelligent missiles, detection of attacks by weapons.
- Home: Remote control of home devices, providing comfortable and smart home environment.
- Environment: Forest fire and flood detection, habitat exploration of animals.
- Public: Vehicle tracking, providing security to malls, buildings, monitoring traffic in a city.

- Health: Patient diagnosis and monitoring, automatic heart rate and blood pressure detection with attached sensors on patients.
- Commercial: Monitoring product quality, managing inventory systems.

In a wireless sensor network, each sensor unit senses data and the transmission unit sends the data to a *base station* or *sink* possibly via multihop routing. The base station is assumed to have abundant energy, sufficient memory and complex processing capabilities. The processing unit is responsible for managing the sensor capabilities and dealing with the contents of the sensed data.

Sensor nodes are deployed in an area either deterministically or randomly. For instance, in remote and inhospitable physical environments, the nodes may be thrown from an airplane.

One of the most significant issue regarding the design of sensor networks is energy consumption. The only source of energy for a sensor node to handle sensing, processing and communication operations is mostly a small battery cell. But these batteries can provide limited energy. Once the sensor nodes are deployed into an area, it is not always possible to replace their batteries or recharge them when their energies get exhausted. Therefore, efficient usage of the available energy in sensor nodes is necessary to operate a sensor network for a long time.

Figure 1.1 shows the architecture of a simple sensor node. The sensing, processing and communication units are three basic units of a sensor node. These units get their energy from a power unit which is responsible for distributing the energy generated by a power generator.

Energy consumption in a sensor node occurs in all its three basic units. The most energy consuming operations are data receiving and data sending which are provided by communication unit. Sensing unit energy consumption is usually assumed to be less than these communication unit operations. However, in some studies such as [2], it is assumed that energy dissipated to sense a bit is approximately equal to the energy dissipated to receive a bit. Processing operations also

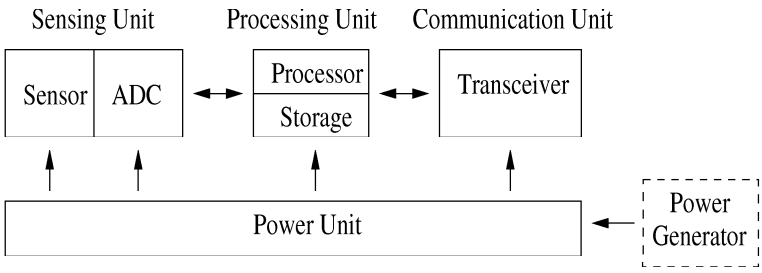

Figure 1.1: Architecture of a simple sensor node

contribute to the energy consumption. But energy consumption due to processing is remarkably low compared to energy consumption due to receiving, sending, and sensing.

Several schemes are proposed for minimizing the total energy consumption in a sensor network in order to increase the network lifetime. In sufficiently dense networks, a common technique is to put some sensor nodes into sleep mode and use only a necessary set of active nodes for sensing and communication. This technique is called *sleep scheduling* or *density control*. Sleep scheduling of nodes in a sensor network controls the number of active sensor nodes and lets the sleeping nodes conserve their energies for future use. A sleep schedule has to provide an even distribution of energy depletion among sensor nodes so that the network can function for a long time. Using only a small subset of nodes as active nodes instead of all of the deployed nodes not only provides energy conservation but also reduces the network traffic, thus decreases packet forwarding delay and avoids packet collisions.

A sleep scheduling algorithm has to decide when and how to determine the active nodes in a sensor network. While deciding which nodes will be operating, the sleep scheduling algorithm should fulfill two requirements: maintenance of *connectivity* and maintenance of *coverage*. A sensor network is connected if every functioning node in the network can reach to the sink via one or multiple hops. Coverage is defined as the area that can be monitored by the active sensor nodes which can reach to the sink. The nodes that cannot reach to the sink but can sense their environment do not contribute to the coverage of a sensor network

because the sink cannot be informed by the sensed data of these nodes. It is usually assumed that a sensor node can monitor all the points within a certain range around itself. This is called the sensing range.

1.1 Motivation

There is already some work done in this area. The studies reported in [3] and [4] give a detailed description and comparison of the most recent energy saving algorithms based on sleep scheduling technique. Considering different application design goals, they have different assumptions about the properties of sensor nodes. Only two of the studies on the topic (OGDC [5] and CCP [6]) can maintain sensing coverage and network connectivity at the same time. In most of the sensor network applications almost full coverage of the sensor field is required and the active sensors are supposed to be connected so that they can send their data to the sink.

Another important issue is that, some of the previous algorithms require quite high amount of message exchange for some maintenance operations. Although high message exchange affects the overall energy consumption dramatically, it is ignored while measuring the energy performance of the algorithms. Obviously, some message exchange is unavoidable, however, the amount of message exhange should be reduced as much as possible for better energy conservation.

Moreover, in most of the algorithms, sleep scheduling scheme is designed independently from the routing scheme. It is assumed that the proposed algorithms can be applied with any routing protocol. However, we think that when sleep scheduling algorithm and the routing protocol are designed together, some redundant operations can be avoided.

As a result, we devised a new energy efficient sleep scheduling algorithm to eliminate the problems mentioned above.

1.2 Contributions

In this thesis, we propose a new distributed sleep scheduling algorithm which maintains both a desired sensing coverage and connectivity. As an original part of this work, we also utilize different sleep modes of sensor nodes. We consider the different units of the sensor nodes (sensing, processing, and communication) independently and turn them on only whenever they are necessary. Besides, we also limit the message exchange and provide a routing scheme using which sensor nodes can communicate with the sink node. These issues were not addressed in most of the previous studies.

The basic features and novelties of our solution are the following:

- It is a distributed solution which is a natural approach to take for sensor networks that require self adaptation.
- Location information of sensor nodes are utilized but in a localized manner. (no global knowledge required)
- Both the sensing coverage and connectivity of the network are maintained.
- Different units of a sensor node are considered separately for turning off and as part of this approach two kinds of functioning modes are defined and utilized.
- A predictive coverage technique is utilized to reduce the messaging overhead. While doing this, coverage maintenance is not violated and same performance is achieved.
- A complete protocol is provided including a routing scheme (based on shortest-path routing) and a sleep scheduling algorithm.
- Maintenance of partial coverage is also considered. This is different than the previous studies which only consider full or no coverage.

1.3 Thesis Structure

In Chapter 2, we start with a background information about wireless sensor networks. Basic concepts related with sensor networks are introduced in this chapter. It also talks about the sleep scheduling problem in sensor networks and includes a detailed discussion of the previous sleep scheduling algorithms in the literature. They are categorized as centralized and distributed algorithms. Chapter 3 gives an analysis of the coverage issue regarding a sensor node and its neighbors. The analysis basically focuses on the problem of *"how many neighbors are sufficient to cover x% of the sensing area of a node?"*. In Chapter 4, we describe our sleep scheduling solution in detail. The steps of our solution and the related algorithms are presented. Chapter 5 presents our simulation model and provides the results of our simulation experiments. In this Chapter, we compare our algorithm with another algorithm in terms of energy efficiency and coverage. Finally, the thesis ends with conclusions and discussions in Chapter 6.

Chapter 2

Background and Related Work

In this chapter, we first introduce some basic concepts that are widely used in sensor network models. After this brief introduction, we talk about sleep scheduling technique and discuss some of the algorithms devised in the literature so far.

Depending on the application, a sensor network must provide some design requirements such as data delivery ratio, quality of service and robustness. Besides, as the common requirements, a sensor network application must satisfy a certain level of coverage and connectivity of sensor nodes. Although these two issues must be considered in the design of a sensor network, there are few sleep scheduling algorithms which satisfy these two requirements at the same time. We will also discuss which algorithms satisfy which requirements.

2.1 Basic Concepts

A sensor network is composed of a large number of sensor nodes which have limited capabilities. Energy, processor speed, and memory size of the sensor nodes are usually very constricted. Each sensor node is responsible for sending its data to a sink node which is located at a specific location in the field. After gathering data from sensor nodes, the sink node can transmit it further to a

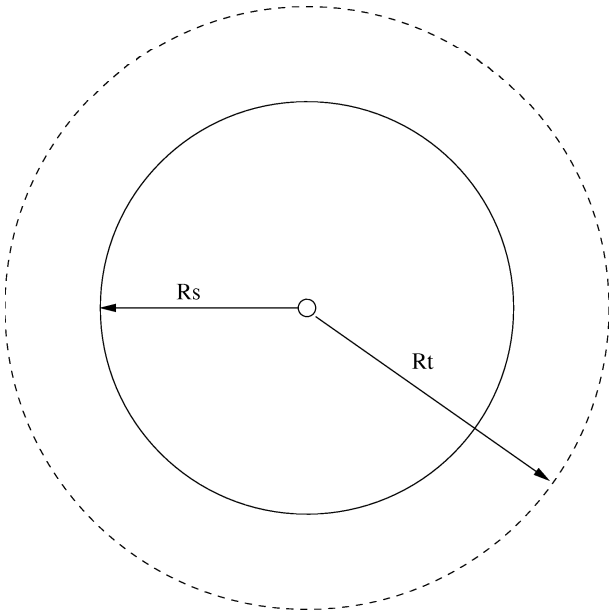

Figure 2.1: Transmission (R_t) and Sensing Ranges (R_s) of a sensor node.

backbone and a server via Internet or a wired communication infrastructure. The sink is assumed to have abundant energy and memory resources and higher computation capabilities than other nodes.

Sensor nodes can sense various metrics about the environment. Temperature, sound and motion are the most common ones.

Each sensor node has two associated ranges: transmission range (R_t) and sensing range (R_s) as shown in Figure 2.1. It is assumed that a sensor node can detect every event happening within a circular area with radius R_s and centered at the location of the sensor node. No event can be sensed outside this area by this sensor node. Similarly, a sensor node can communicate with all other sensor nodes located within the circular region with radius R_t around the sensor node.

Sensor nodes send their data to the sink depending on a data reporting model. These models include event-driven, time-driven, query-driven or a combination of them. In a time-driven model, sensor nodes periodically sense the environment and send their data to the sink. But in an event-driven model, sensor nodes send

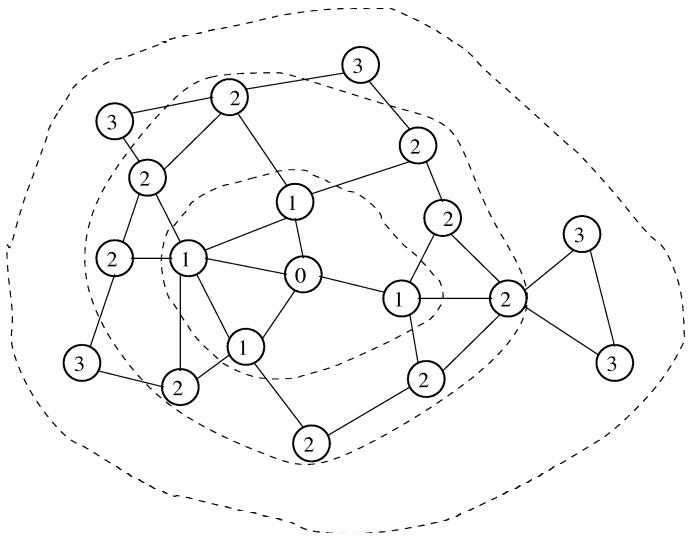

Figure 2.2: Tiers numbers in a sample network. Nodes that can communicate with each other are connected with a line.

data to the sink whenever a specific event occurs. In a query-driven model, nodes send data only when they receive a query from the sink or any other node in the network. In this thesis, we assume that time-driven data reporting model is used.

Hop count and *tier number* are commonly used terms in multihop sensor network models. Sink is assumed to have tier number 0. The nodes which can reach to the sink with one hop are in tier 1; nodes which can reach to the sink with two hops are in tier 2, and so on. Figure 4.2 shows the tier numbers of nodes in a sample network. Note that in the decision of hop counts, transmission range of sensor nodes is considered. If a node is in the transmission range of another node, these nodes are assumed to be one hop distant from each other.

The nodes having one hop distance from a node are called as *neighhbors* of that node. In some studies, however, two or more hop distant nodes are also considered as neighbors. In this thesis, unless otherwise stated, we use the term neighbors to imply one-hop distant nodes.

2.2 Sleep Scheduling in Wireless Sensor Networks

Sensor networks are usually deployed with high densities (up to 20 nodes/m^3 [7]) to have extended network reliability and lifetime. However, if all the sensor nodes in such as dense deployment scenario operate at the same time, excessive energy consumption will occur. Moreover, packet collisions will also increase due to large number of packets forwarded in the network. In a dense deployment, sensing areas of the sensor nodes may overlap and same data may be sent to sink from different sensor nodes. This creates a redundancy in data transport and high correlation among the adjacent sensor nodes. To avoid these disadvantages, sleep scheduling algorithms are applied in sensor networks.

The sleep scheduling algorithms should be simple, distributed and localized. It should be applicable to many types of applications with minor modifications. Due to the distributed nature of sensor networks, a sleep scheduling algorithm should be as well distributed. Since nodes can reach directly only to the nodes within their transmission range, the sleep scheduling algorithm must be localized and use only local information.

There are many sleep scheduling algorithms proposed in the literature. They differ from each other in terms of their assumptions about several issues including transmission and sensing range ratio, detection model, failure model and location information. Network structure and sensor deployment methods are also other differences of these algorithms.

Since the proposed schemes are designed for different applications, they have different assumptions. Therefore, to make a fair comparison with them, we should carefully consider their assumptions and their approaches to reach the objective.

Sleep scheduling schemes can be categorized using different aspects. One of the most significant aspects that separates them is whether they are centralized or distributed. Sensor nodes have limited energy resources and they are supposed to use most of their energies for sensing and transmitting data rather than control

messages used by the sleep scheduling schemes. The basic difference between centralized and distributed algorithms emerges from this fact. In a centralized scheme, all knowledge about the nodes in the network is known by the sink node. The sinkn node then finds the optimum number of active nodes and decides about the status of the nodes. Furthermore, in each round or periodically the sink node sends control messages to inform the sensor nodes about their status. On the other hand, in distributed schemes every sensor node knows only about its neighbors and tries to decide about its status by this limited information and independently from other nodes.

There are many schemes proposed in both centralized and distributed manner. Below, we will briefly mention about the most significant schemes in both categories.

2.2.1 Centralized Algorithms

Two similar centralized solutions are proposed in [8] and [9]. They assume a dense deployment of nodes and try to achieve an energy efficient design that maintains area coverage. Deployed nodes are divided into disjoint sets, such that each set can independently accomplish monitoring the area. The sets are activated periodically while the nodes in other sets are put into low-energy mode. By this way the area is monitored by only a necessary number of nodes belonging to only one set and energy conservation is achieved. In this solution, the main issue is determining the disjoint sets and increasing their numbers as much as possible. This idea performs well in case the number of nodes deployed is much greater than the optimum number of nodes required to monitor the area.

Another centralized solution approach is some sort of ILP technique using the knowledge (i.e. remaining energy) about deployed nodes for deciding on the active nodes needed. The works [10] and [11] utilize this approach in their proposed solutions. Use of ILP can give optimal solution for minimum energy conservation in the network, however, it may be not be practical to obtain the results quickly and efficiently for large networks.

In [12] and [13], authors propose two different greedy algorithms to determine the active nodes in a round. While [12] selects the nodes which provide maximum benefit in terms of area coverage as active nodes, [13] tries to select first the nodes with maximum remaining energy. Although they are centralized, the resulted network may be far from optimum since they follow greedy approach.

2.2.2 Distributed Algorithms

In [14], *Geographic Adaptive Fidelity* (GAF) algorithm divides a region to be monitored into equal-sized grid cells and tries to leave only one node active in each grid. Each cell of the grid is square shaped with one size being smaller than or equal to $R/\sqrt{5}$, where R is the sensing range of sensor nodes. The sensing range of nodes are assumed to be the same. Nodes switch between *sleeping*, *discovery* and *active* states and the algorithm tries to keep only one active node in each grid.

One of the first distributed sleep scheduling algorithms is PEAS. In PEAS [15] [16], a node decides locally and independently whether it will go into sleep mode or not. For that, a node probes its local environment to check if there are any neighbors active. If it does not sense any active neighbor, the node decides to be the active one in that vicinity. Then it remains active until it dies. But if the node senses an active neighbor from which it can get a reply to its probe message, it decides to go to sleep. It wakes up after some exponentially distributed time interval, and do the same check again. This is a simple and efficient scheme, however, it does not guarantee coverage.

Another pioneering algorithm in distributed sleep scheduling is AFECA [17]. It defines three operating states for a node: sleeping, listening and active. Initially nodes are in sleeping state. After T_s time period, a node switches to listening state. When in listening state, the node has its radio turned on and listens for messages for a time period T_1. If a routing message is received during this time interval, the node participates in routing. If the node decides to send data, it switches to active state. Otherwise, the node returns to sleeping state after

the time period T_1 has elapsed. AFECA algorithm uses the advantage of interchanging activities among nodes in a dense network. This approach increases the lifetime of the network as node density increases.

In SPAN [18], nodes are seperated into two groups: coordinators and non-coordinators. Coordinators are assumed to be the necessary nodes that the traffic of all active nodes are forwarded. Therefore coordinators consume more energy than non-coordinators do. Each node decides about its role (coordinator or non-coordinator) according to the coordinator eligibility rule. If two neighbors of a non-coordinator node cannot reach each other either directly or via one or two coordinators, the node should become a coordinator. At the beginning all nodes are assumed to be non-coordinators. After some random back-off time, each node runs the coordinator eligibility rule. The random back-off is used in order to avoid coordinator announcement contentions. The nodes with higher energy levels are assigned smaller back-off times. In this way, the probability that nodes with higher energy levels become coordinators is increased.

In [19], the authors propose a solution that also focuses on preserving certain coverage. They show a way of finding the overlapping sensing area between a node and its neighbors. In order to find the common sensing area, only a specific angle is needed and only the area of common sector between the sensing areas of nodes is considered. However, the protocol only considers the one-hop neighbors of a node while finding the overlapping sensing area, which may not be enough as we will discuss later.

The common property of the algorithms above is that they do not guarantee the connectivity and initial coverage of the network at the same time. For example, [19] guarantees coverage but not connectivity; [18] guarantees connectivity but not coverage; and [15] do not guarantee either. These algorithms may be useful for some applications, however, in most of the sensor network applications, both coverage and connectivity are very important factors that should not be violated. As far as we know, there are two algorithms published so far that preserve both connectivity and coverage: OGDC [5] and CCP [6].

In [5], Zhang and Hou prove that coverage implies connectivity if the ratio

between the transmission range and sensing range is at least two. Depending on this, they propose Optimal Geographic Density Control (OGDC) algorithm to maximize the number of sleeping sensor nodes with coverage guarantee. A sensor node is active only in the case it minimizes the overlapping area with the existing working sensor nodes and it covers an intersection point of two sensors. A sensor node decides this by using its own location and working sensor nodes' locations.

In [6], Wang et al. propose coverage and connectivity configuration protocol (CCP) which tries to maximize the number of sleeping nodes while maintaining k-coverage and k-connectivity. Here, k-coverage means each point in the monitoring area of the sensor network is sensed by at least k different nodes of the network. The authors prove that k-coverage implies k-connectivity and to decide k-coverage, a node only needs to check whether the intersection points inside its sensing area are k-covered. Similar to OGDC, CCP assumes the transmission range is at least twice the sensing range. But if it is not the case, it combines its algorithm with SPAN so that SPAN can control connectivity. In this case, a node decides to sleep if it satisfies the eligibility rules in both schemes. Otherwise it stays active.

When we consider these algorithms, it seems that we notice that CCP is quite general and effective sleep scheduling scheme, because it preserves both coverage and connectivity at the same time for any ratio between transmission range and sensing range. However, when combined with SPAN, it may cause too much control message overhead in the network. Furthermore, as all other schemes, it does not provide an efficient routing structure or protocol for the active nodes.

Chapter 3

Expected Overlap Analysis

Before going through the details of our algorithm, we want to discuss coverage issue in sensor networks. In most of the algorithms, if there is a check to decide whether a node's coverage area is covered by other nodes in the network, only the effect of one-hop neighbors are considered. However, some nodes which have overlapping sensing areas with the node may not be reached by one-hop from that node. Further its effect may be significant in deciding the result of coverage check.

Consider the example illustrated in Figure 3.1. The nodes B, C, and D are one-hop neighbors of the node A, and the nodes E, F, G, H are other nodes which have common sensing areas with node A. If node A only considers its one-hop neighbors while deciding whether it is sensing area is covered by other nodes (that is, it is coverage eligible or not), it decides to be non-eligible, since the sensing area of node A is not totally covered by one-hop neighbors. However, if other closer nodes to A could be considered, the sensing area of node A is totally covered by other nodes. Hence, we should also consider the effect of other nodes which are closer than $2R_s$ to a sensor node, while applying coverage check on the node.

On the other hand, even a node cannot communicate with another node having an overlapping sensing area, this may not have a critical effect on coverage check.

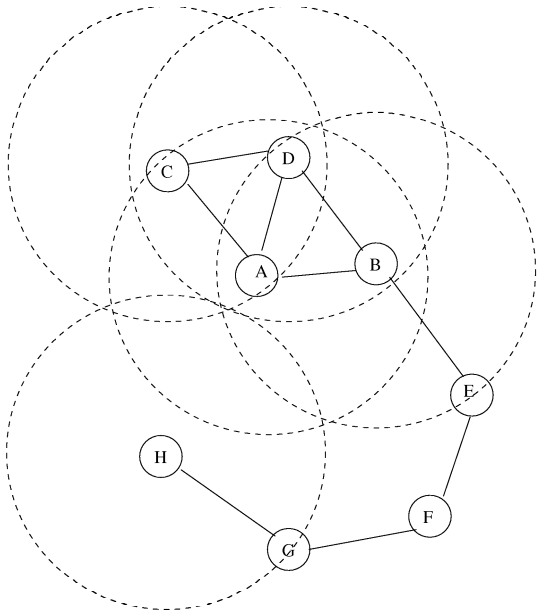

Figure 3.1: Node A's sensing area is totally covered by not only the one-hop nodes of A but also another node H.

Moreover, it may cost too much to communicate with such nodes and maintain their information in a table. In Figure 3.1, for example, even though node H is not far away from node A, node A can communicate with node H over 4 hops. When the hop count increases to reach such nodes, messaging overhead to learn about these nodes increases as well. Hence, there is a tradeoff between good coverage check by the help of these nodes and messaging and storing overhead due to the communication with these nodes.

In addition, most of the algorithms assume a certain ratio between transmission range and sensing range of sensor nodes. Usually their coverage check depends on this ratio. In today's sensor node technology there are different ratios between these two ranges. The possible ratios lie in the interval [1/2, 3]. Therefore, a coverage check algorithm should also consider this ratio while deciding a node's eligibility to go to sleep.

To utilize all nodes having overlapping sensing area with a node and to reduce the amount of messages while maintaining neighborhood tables, we propose a

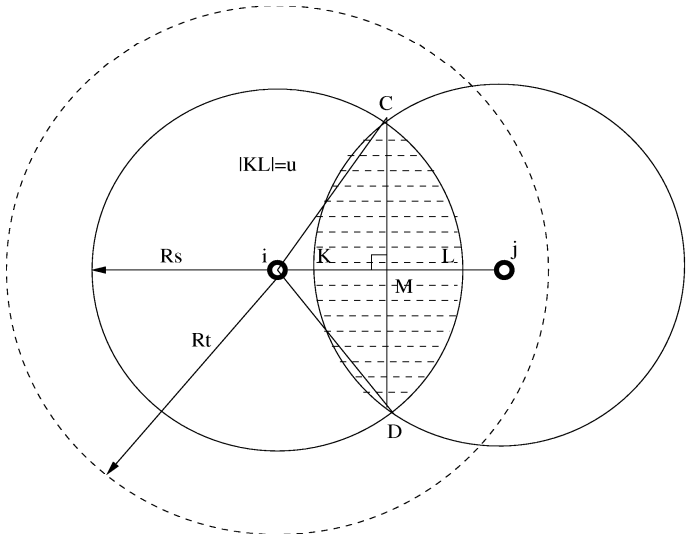

Figure 3.2: Overlap of node i's sensing area by a neighbor j.

predictive coverage check to decide whether a sensor node's sensing area is covered by a certain amount by other nodes in the vicinity.

3.1 Expected Overlap With One Neighbor

Assume we have a node i with only one neighbor in its transmission range and the nodes are identical, i.e. all nodes have the same R_s and R_t. The nodes are also distributed uniformly and randomly over the region. We want to find the expected result of *"how much of a node's sensing area is covered (overlap) by only one neighbor?"* In other words, we are interested in node i and how much of its sensing area is covered by another node.

Let $c(x)$ denote the overlap of the node i's sensing area by another node which is x meter far from the node i; i.e. if j is such a node, $dist(i,j) = x$ meter. We define the overlap as a normalized value which is equal to the ratio of the overlapping sensing area to the whole sensing area of the node of interest.Hence overlap is a real number between 0 and 1.

Let Z_{d_1,d_2} be a random variable that indicates the overlap of the node i's sensing area by a node j, such that $d_1 \leq dist(i,j) \leq d_2$. Possible values z of Z are $0 \leq z \leq 1$.

Then, $E[Z_{d_1,d_2}]$ is the expected overlap of the node i's sensing area by a node j where $d_1 \leq dist(i,j) \leq d_2$.

Let X denote a random variable indicating the distance of the sensor node i to a node in its transmission range. Possible values x of X are $0 \leq x \leq R_t$. The probability density function for X is:

$$f_X(x) = \frac{2\pi x}{\pi R_t^2} = \frac{2x}{R_t^2}$$

Then, the expected overlap (p) of the node i's sensing area by a node which is one-hop away from node i can be denoted as $E[Z_{0,R_t}]$ and is given by:

$$p = E[Z_{0,R_t}] = \int_{x=0}^{x=R_t} c(x) f_X(x) dx$$

To calculate $E[Z_{0,R_t}]$ we need to compute $c(x)$. $f_X(x)$ is given above.

Consider the illustration in Figure 3.2. The shaded area shows the overlapping sensing area of the node i by one of its neighbors, j. Let $dist(i,j)$ be x. Then, $c(x)$ is the ratio of the overlapping sensing area to the whole sensing area of node i.

$$c(x) = \frac{A(\text{CKDL})}{\pi R_s^2}$$

Let u denote the length of |KL| in Figure 3.2. Note that u is a function of x, and vice versa. We can calculate A(CKDL) and x as follows:

$$\begin{aligned}
A(\text{CKDL}) &= 2A(\text{CDL}) \\
&= 2\left(A(\text{ODLC}) - A(\text{ODC})\right) \\
A(\text{ODLC}) &= R_s^2 cos^{-1}\left(\frac{R_s - (u/2)}{R_s}\right) \\
A(\text{ODC}) &= \sqrt{R_s^2 - (R_s - (u/2))^2}(R_s - u/2) \text{ and} \\
x &= 2[R_s - (u/2)] \text{ and } u = 2R_s - x
\end{aligned}$$

Besides, since node j is a neighbor of node i then,

$$\begin{aligned} f_X(x) &= \frac{2x}{R_t^2} \text{ and} \\ f_U(u) &= \frac{2(2R_s - u)}{R_t^2} \end{aligned}$$

When x is in the interval $[0,R_t]$, there is an overlap between the node i's sensing area and the node j's sensing area. In that interval, u takes the values between $2R_s$-R_t and $2R_s$. As a result:

$$\begin{aligned} p &= E[Z_{0,R_t}] \\ &= 4\int_{u=2R_s-R_t}^{u=2R_s} \left(R_s^2 cos^{-1}(1 - \frac{u}{2R_s}) - \sqrt{R_s^2 - (R_s - \frac{u}{2})^2}(R_s - \frac{u}{2}) \right) \frac{2(2R_s - u)}{R_t^2} du \end{aligned}$$

The above equation is valid when $R_t \leq 2R_s$. When $R_t > 2R_s$, the equation becomes:

$$p = E[Z_{0,R_t}] = E[Z_{0,2R_s}](\frac{2R_s}{R_t})^2$$

When $R_t = R_s$, the equation gives the value 0.5865. In other words, if a sensor node has a neighbor, the 58.65% of its sensing area is expected to be covered by that neighbor when $R_t = R_s$. The results of this equation for different R_t/R_s ratios are shown in Figure 3.3. Note that the graph decreases linearly until the ratio 2, then it decreases with a curvic behaviour.

3.2 Expected Overlap With Multiple Neighbors

We have calculated the expected overlap of a node's sensing area by one of its neighbors. If the number of neighbors increases, it may not be easy to derive a mathematical formulation as in one neighbor case. However, by using a probabilistic approach we can find out the overlap of multiple neighbors.

Assume we have a sensor node of interest located at point O. Let X denote a random variable indicating the distance of the sensor node to a point in its

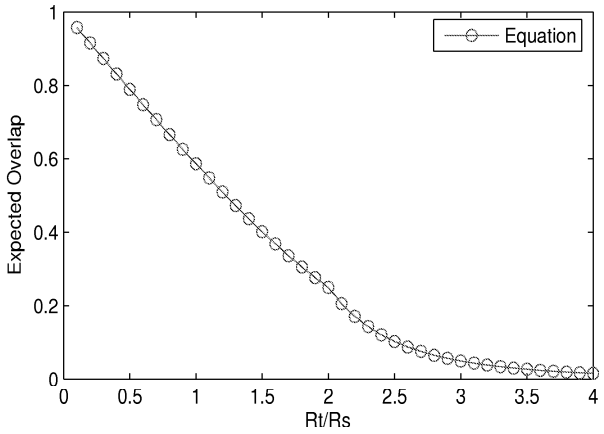

Figure 3.3: Expected overlap of a node's sensing area with different R_t/R_s ratios.

sensing range (R_s). Possible values x of X are $0 \leq x \leq R_s$. The probability density function for X is:

$$f_X(x) = \frac{2\pi x}{\pi R_s^2} = \frac{2x}{R_s^2}$$

Assume that the probability of a point P that is inside the sensing area of the node and that is x m away from the node is covered by a neighbor of the sensor node is $p(x)$. It is obvious that this probability is not same for all points and it depends on the distance x of the point to the sensor node. When there are n neighbors of the node, we can formulate the probability that a point is not covered by any of these n neighbors as $(1 - p(x))^n$. Then the probability that a point is covered by any of these neighbors becomes $1 - (1 - p(x))^n$. As a result, if we can find $p(x)$ for each point and integrate it over the sensing area of the node, we can find out the expected overlap in multiple neighbors case.

Consider the Figure 3.4. We have the sensor node located at point O. We want to find $p(x)$ of point P. For point P to be covered by a neighbor of the sensor node there should be a neighbor inside the shaded region. In other words, a neighbor which is not more than R_s distant from point P and which is inside the communication range of the node should exist. Therefore, for point P, $p(x)$ is equal to the ratio of shaded area in Figure 3.4 to whole communication area of

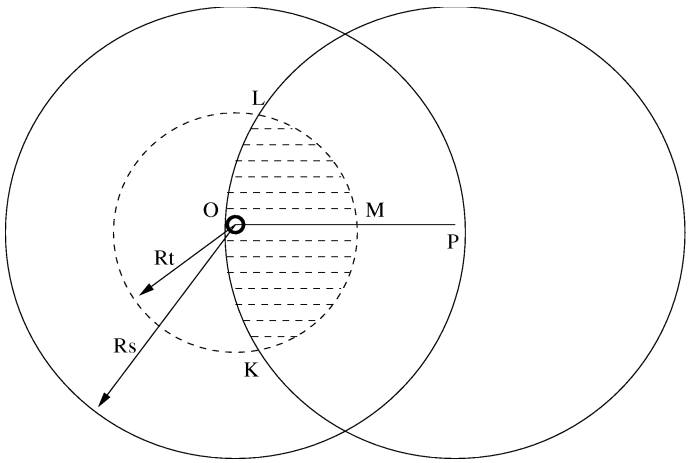

Figure 3.4: Probability that a point P inside the sensing area is covered by a neighbor of the node is proportional to the shaded area.

the sensor node, i.e., πR_t^2.

To calculate the area of the shaded region, we first put our model into an x-y coordinate plane as it is shown in Figure 3.5. Since the shaded region is enclosed by two circle equations, we find their equations depending on y value, take their difference and integrate it with respect to y. Lower limit for y is 0, whereas upper limit for y is h. An important issue in this calculation is, in some cases we cannot calculate the required region as it is told. Because, in some cases such as in Figure 3.6, the integral of the difference of the circle equations can not give the total area of the required region. In these cases, we first find the complementary region and subtract it from the whole communication area. For instance, we first find $A(\text{NKTL})$ and then subtract it from πR_t^2. By this way we can calculate the correct value of the area.

These two different cases seperate from each other in the border where the height of the required region becomes R_t. Figure 3.7 shows this case. The points which have longer distance to the center (i.e. sensor node) than this point use the first approach, whereas the points which have closer distance to the center use the second approach.

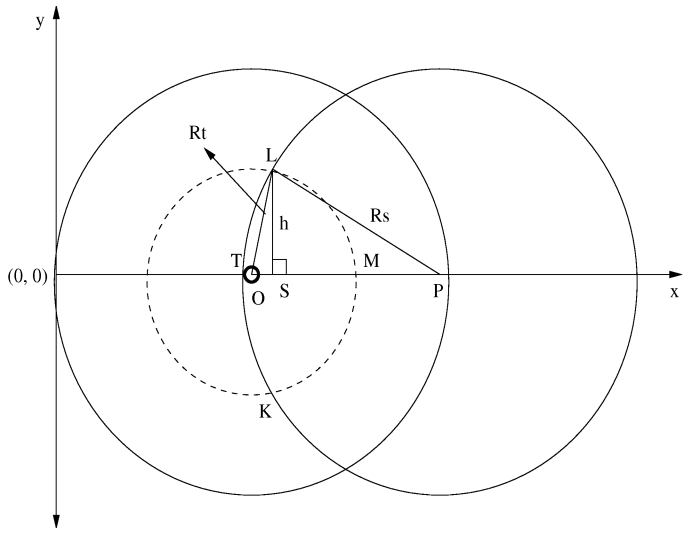

Figure 3.5: Height of the region projects on the right of center O.

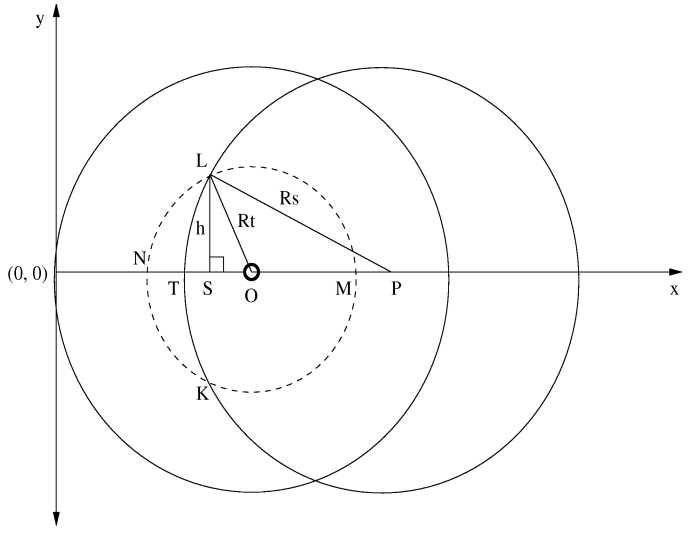

Figure 3.6: Height of the region projects on the left of center O.

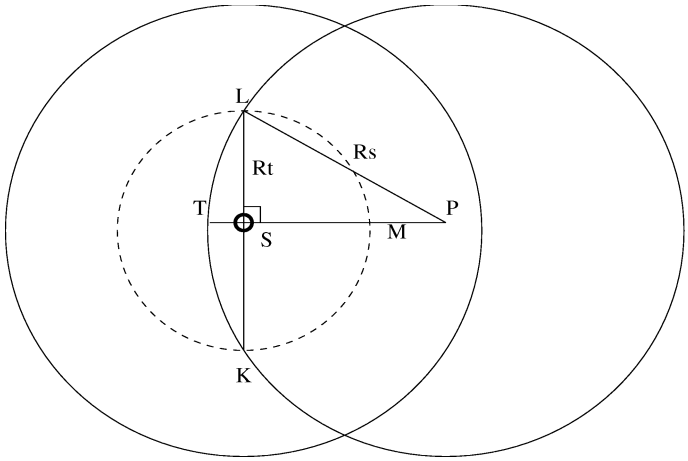

Figure 3.7: Height of the region projects on center O.

In Figure 3.5, let x denote the distance between the sensor node and the point (i.e. x=|OP|). Note that, x is not the $x-$axis value anymore for this analysis. Let $|\text{OS}| = b$; then $|\text{PS}| = x - b$, and:

$$\begin{aligned}
b^2 + h^2 &= R_t^2 \\
h^2 + (x-b)^2 &= R_s^2 \\
h^2 + x^2 - 2xb + b^2 &= R_s^2 \\
R_t^2 + x^2 - 2xb &= R_s^2 \\
b &= \frac{R_t^2 - R_s^2 + x^2}{2x} \\
h &= \sqrt{R_s^2 - b^2} \\
A(\text{TKML}) &= 2\int_{y=0}^{y=h} \left(\sqrt{R_t^2 - y^2} + \sqrt{R_s^2 - y^2} - x \right) dy \\
p_1(x) &= \frac{A(\text{TKML})}{\pi R_t^2}
\end{aligned}$$

And in Figure 3.6, let $|\text{OS}| = b$ again. Then $|\text{PS}| = x + b$, and in a similar way:

$$\begin{aligned}
b &= \frac{R_s^2 - R_t^2 + x^2}{2x} \\
h &= \sqrt{R_s^2 - b^2} \\
A(\text{TKML}) &= \pi R_t^2 - A(\text{TKNL})
\end{aligned}$$

$$
\begin{aligned}
A(\text{TKNL}) &= 2\int_{y=0}^{y=h}\left(\sqrt{R_t^2-y^2}-\sqrt{R_s^2-y^2}+x\right)dy \\
p_2(x) &= 1-\frac{A(\text{TKNL})}{\pi R_t^2}
\end{aligned}
$$

The border value of x that separates these cases can be calculated as:

$$
x_{border} = \sqrt{R_s^2-R_t^2}
$$

As a result when $R_t < R_s$, the expected value of the probability ($E[p(X)]$) that a point inside the sensing area is covered by a neighbor is:

$$
\begin{aligned}
p = E[p(X)] &= \int_{x=0}^{x=R_s} p(x)f_X(x)dx \\
p = E[p(X)] &= \int_{x=0}^{x=R_s-R_t}(R_t/R_s)^2 f_X(x)dx + \\
&\quad \int_{x=R_s-R_t}^{x=x_{border}} p_2(x)f_X(x)dx + \\
&\quad \int_{x=x_{border}}^{x=R_s} p_1(x)f_X(x)dx \text{ where,} \\
f_X(x) &= \frac{2x}{R_s^2}dx
\end{aligned}
$$

Note that, p is used for the general notation of the expected overlap of a node's sensing area by one of its neighbors. Then, $p = E[Z_{0,R_t}]$, which is computed in the previous section. When $R_t > R_s$ we can find p in a similar way. But there are two different calculations in this case. The border situation is illustrated in Figure 3.8, $R_t = R_s\sqrt{2}$.

When $R_s \leq R_t \leq R_s\sqrt{2}$:

$$
\begin{aligned}
p &= \int_{x=0}^{x=R_t-R_s}(R_s/R_t)^2 f_X(x)dx + \\
&\quad \int_{x=R_t-R_s}^{x=x_{border}} p_2(x)f_X(x)dx + \\
&\quad \int_{x=x_{border}}^{x=R_s} p_1(x)f_X(x)dx
\end{aligned}
$$

When $R_s\sqrt{2} \leq R_t \leq 2R_s$:

$$
\begin{aligned}
p &= \int_{x=0}^{x=R_t-R_s}(R_s/R_t)^2 f_X(x)dx + \\
&\quad \int_{x=R_t-R_s}^{x=R_s} p_2(x)f_X(x)dx
\end{aligned}
$$

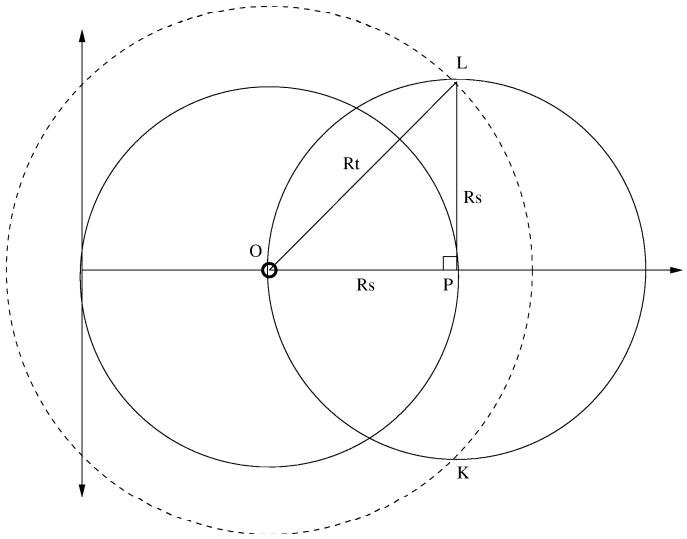

Figure 3.8: Border case when $R_t > R_s$.

Now, let p_{n,d_1,d_2} denote the expected overlap of a node i's sensing area by n nodes, where these nodes have distance from the node in the interval $[d_1, d_2]$. And, let p_n denote the expected overlap by n one-hop neighbors. Then, p_n=$p_{n,0,R_t}$. When n=1, it is equal to p.

Assume there are n nodes in interval [0,R_t] after deployment. Then,

$$p_n = p_{n,0,R_t} = \int_{x=0}^{x=R_s} (1 - (1 - p(x))^n) f_X(x) dx$$

To validate the correctness of this probabilistic approach, we have also obtained the overlap in multiple neighbors case from a simulation. We created random multiple neighbors to a node within its transmission range and calculated the overlap of the node's sensing area by its neighbors. Besides for each multiple neighbor count, the simulation is run 1000 times and the result is obtained as the average of them. The Figures 3.9, 3.10 and 3.11 show the comparison of formula and simulation results in three different R_t/R_s values. According to the results, our probabilistic approach gives same values with the simulation.

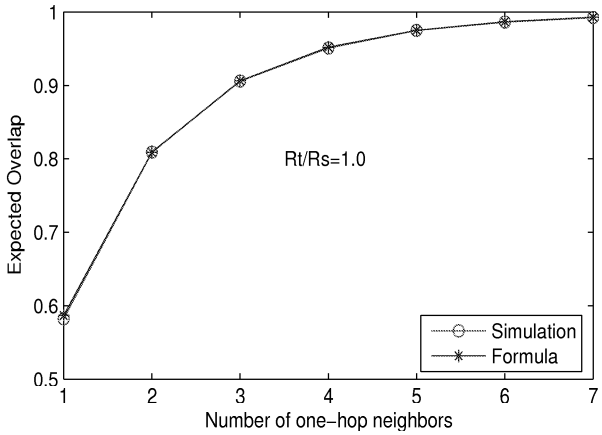

Figure 3.9: Expected overlap values from simulation and formula when R_t/R_s=1

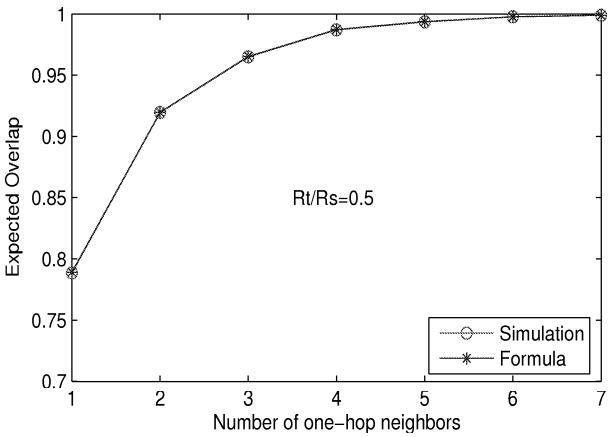

Figure 3.10: Expected overlap values from simulation and formula when R_t/R_s=0.5

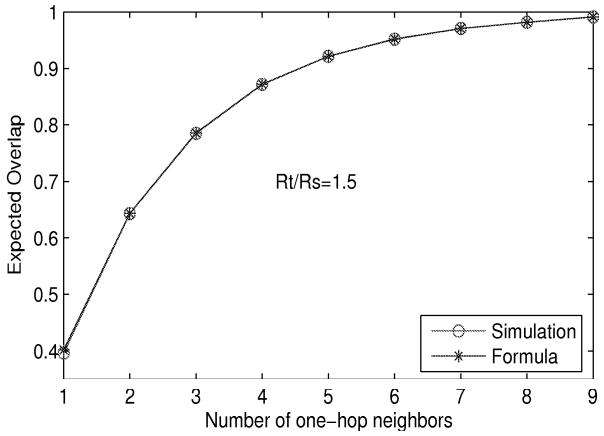

Figure 3.11: Expected overlap values from simulation and formula when R_t/R_s=1.5

3.3 Expected Overlap With Multiple Hop Counts

A sensor node may not have only one hop neighbors but also multiple hop neighbors which have overlapping sensing area with the sensor node. Especially, when the R_t/R_s value gets smaller, multiple hop neighbors create a significant overlap on the node's sensing area. To see the effect of multiple hop neighbors we need to calculate expected overlap as in one-hop neighbor case. Again, the overlap is expressed with a value between 0 and 1.

Consider the Figure 3.12. The expected overlap by a two-hop neighbor (p'') can be denoted as $p_{R_t,2R_t}$ and it can be calculated as follows:

$$\begin{aligned} p_{0,R_t} &= \int_{x=0}^{x=R_t} \frac{A}{\pi Rt^2} f_X(x)dx \\ p_{0,2R_t} &= \int_{x=0}^{x=2R_t} \frac{A+B}{\pi(2Rt)^2} f_X(x)dx \\ p'' = p_{R_t,2R_t} &= \int_{x=R_t}^{x=2R_t} \frac{B}{\pi(2Rt)^2 - \pi {R_t}^2} f_X(x)dx \\ &= \frac{4p_{0,2R_t} - p_{0,R_t}}{3} \end{aligned}$$

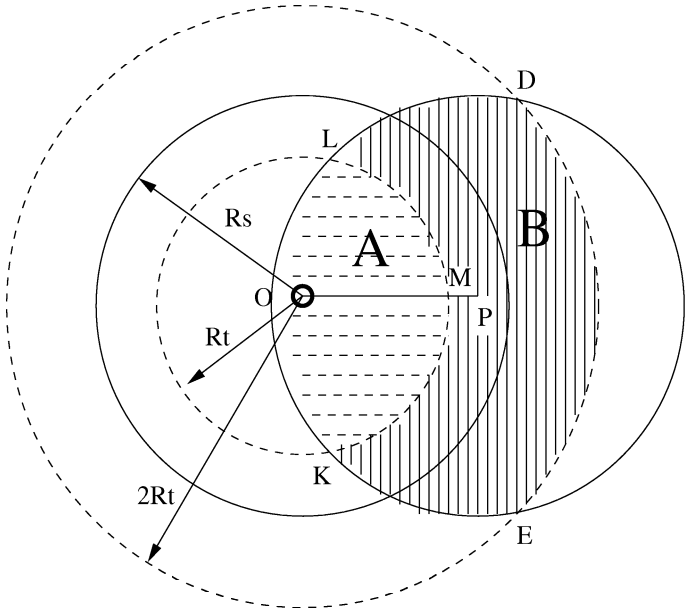

Figure 3.12: Expected overlap of a two-hop neighbor

For example, when $R_t = R_s$:

$$\begin{aligned} p_{0,R_t} &= 0.5865 \\ p_{0,2R_t} &= 0.25 \\ p'' = p_{R_t,2R_t} &= (4(0.25) - 0.5865)/3 = 0.1378 \end{aligned}$$

Furthermore, if there are n two-hop neighbors of a node, using the same probabilistic approach we can formulate the expected overlap by these two-hop nodes as:

$$\begin{aligned} p_{n,R_t,2R_t} &= \int_{x=0}^{x=R_s} (1-(1-p(x))^n) f_X(x) dx \text{ where,} \\ p(x) &= \frac{B}{3\pi R_t^2} \end{aligned}$$

Let p_{n_1,n_2} denote the expected overlap by n_1 one-hop and n_2 two-hop neighbors. Then, to find the expected value of p_{n_1,n_2} we again use the same probabilistic approach and combine the expected overlap of each hop. For instance, if there are n_1 one-hop neighbors and n_2 two-hop neighbors, the expected overlap by one-hop

	0	1	2	3	4	5	6	7	8
0	0	0.1378	0.2544	0.3532	0.4372	0.5086	0.5696	0.6218	0.6665
1	0.5865	0.6502	0.7035	0.7482	0.7856	0.8171	0.8435	0.8658	0.8847
2	0.8088	0.8401	0.8661	0.8877	0.9057	0.9206	0.9331	0.9435	0.9522
3	0.906	0.922	0.9352	0.9461	0.9551	0.9626	0.9688	0.9739	0.9782
4	0.952	0.9604	0.9673	0.9729	0.9776	0.9815	0.9846	0.9872	0.9894
5	0.9748	0.9793	0.983	0.986	0.9885	0.9905	0.9922	0.9935	0.9947
6	0.9865	0.9889	0.9909	0.9926	0.9939	0.995	0.9959	0.9966	0.9972
7	0.9927	0.994	0.9951	0.996	0.9967	0.9973	0.9978	0.9982	0.9985
8	0.996	0.9967	0.9973	0.9978	0.9982	0.9985	0.9988	0.999	0.9992

Table 3.1: Expected Overlap Table of a node when R_s=R_t=100. Rows indicate number of one-hop neighbors, whereas columns indicate number of two-hop neighbors.

and two-hop neighbors together can be calculated as:

$$p_{n_1,n_2} = \int_{x=0}^{x=R_s} \left(1-\left(1-p(x,1)\right)^{n_1}\left(1-p(x,2)\right)^{n_2}\right) f_X(x)dx$$

Here $p(x,1)$ and $p(x,2)$ are:

$$\begin{aligned} p(x,1) &= \frac{A}{\pi R_t^2} \\ p(x,2) &= \frac{B}{3\pi R_t^2} \end{aligned}$$

Consequently, when a node knows the number of one-hop and two-hop neighbors, it can find the expected overlap of its sensing area by these nodes. As an example, when $R_s = R_t$, we calculated the expected overlap for different n_1 and n_2 values. Table 3.1 shows expected overlap ratios of a node for all cases when $0 \leq n_1 \leq 8$ and $0 \leq n_2 \leq 8$. Rows indicate values for n_1, whereas columns indicate values for n_2.

Furthermore, we summarize the meanings of all symbols used in our expected overlap analysis in Table 3.2.

Symbol	Meaning
$c(x)$	Overlap of a node's sensing area by a neighbor that is x meter away from the node
Z_{d_1,d_2}	Random variable indicating the overlap of a node's sensing area by another node which has a distance to the node in the interval $[d_1,d_2]$
$E[Z_{0,R_t}]$ or p	Expected overlap of a node's sensing area by a one-hop neighbor
X (in Section 3.1)	Random variable indicating the distance of a sensor node to another node in its transmission range
X (in Section 3.2)	Random variable indicating the distance of a sensor node to a point in its sensing range
$f_X(x)$	The probability density function of X
$p(x)$	The probability of a point P that is inside the sensing area of the node and that is x m away from the node is covered by a neighbor of the sensor node
p_{n,d_1,d_2}	Expected overlap of a node's sensing area by n nodes, where these nodes have distance from the node in the interval $[d_1, d_2]$
p_n	Expected overlap of a node's sensing area by n one-hop neighbors
$p_{R_t,2R_t}$ or p''	Expected overlap of a node's sensing area by a two-hop neighbor
$p_{n,R_t,2R_t}$ or p''_n	Expected overlap of a node's sensing area by n two-hop neighbors
p_{n_1,n_2}	Expected overlap of a node's sensing area by n_1 one-hop and n_2 two-hop neighbors

Table 3.2: All symbols used in the analysis

Chapter 4

Our Solution

Our goal with this thesis is to design a distributed sleep scheduling algorithm which preserves both the coverage and connectivity of the functioning nodes and reduces the number of messages used for maintenance procedures. In this chapter, we first describe our network model and give our assumptions. After that, we give the details of each step in our algorithm.

4.1 Network Model

We assume that sensor nodes are uniformally and randomly deployed in a square area. There are N nodes including the sink node and the sink node is located at the center of the area, at origin point (0,0). The sink node is assumed to have abundant energy, enough memory resources and higher computation capability than other sensor nodes.

The sensor nodes in the network are assumed to be identical, and have constant transmission range (R_t) and sensing range (R_s). Besides, we assume that all nodes know their locations. This may be achieved by GPS devices or some GPS-less localization algorithms [20, 21, 22, 23, 24]. Besides, the location information of each node may be hard-coded inside the device. Moreover, all nodes

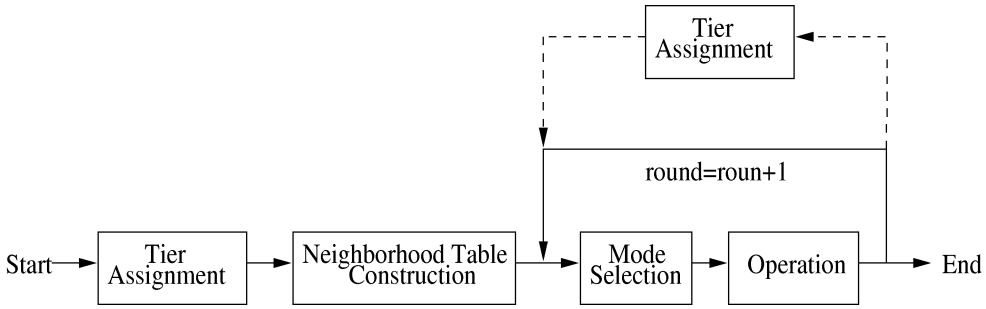

Figure 4.1: Running order of the phases during the network lifetime.

are assumed to be time synchronized with each other and they are not mobile, i.e., the network is static.

4.2 Our Algorithm

Our algorithm is a distributed and localized algorithm such that each sensor node gives its decisions based on only local information (i.e neighbors' location).

The algorithm consists of four phases; tier assignment, neighborhood table construction, mode selection, and operation phases. While the first two phases are run only at the beginning, the later two phases are run in each round. But in some conditions tier assignment process may be run throughout the network lifetime. Figure 4.1 shows the phases during the lifetime of the sensor network.

In tier assignment phase, nodes are assigned tier numbers starting with tier number 0 at the sink node. In neighborhood table construction phase, each node constructs a table of its one, two and three-hop neighbors. In mode selection phase, each node determines its mode that it will stay during the rest of the round and in operation phase each node fulfills its responsibilities depending on the mode it selected. After this brief introduction to phases, we give detailed information about each phase in the next sections.

4.3 Tier Assignment Phase

After sensor nodes are deployed, the sink node initiates the process of assigning tier numbers to all nodes in the network. It creates a *TierAssignment* message containing its ID with tier number zero and broadcasts it. Then, each node receiving a *TierAssignment* message creates its own *TierAssignment* message by incrementing tier number by one and putting its own ID. As the next step, the node broadcasts this new message to all of its neighbors. If a node receives multiple *TierAssignment* messages, it only considers the message which has lower tier number than the its current tier number. Furthermore, among the *TierAssignment* messages having same tier number, the one coming from the closest node is considered. This can be determined by measuring the received signal strength value.

Figure 4.2 shows the result of tier assignment process in a sample network. The sink node is assigned the tier number 0; the nodes which can reach to the sink with one hop are assigned the tier number 1; nodes which can reach to sink with two hops are assigned the tier number 2, and so on.

In this phase, our aim is to create a tree-like network structure rooted at the sink to be used in routing of the packets. The active nodes in the network sense the environment and forward their data together with their descendants' data towards the sink. In forwarding data towards the sink, nodes use their tier number with respect to the sink as a reference and send their data to one of their neighbors with smaller tier number. For example a node with tier number 3 sends its data to a parent node with tier number 2. By this way, shortest path routing in terms of hop counts is achieved and the possible routing loops are avoided. Moreover, data aggragation is used while forwarding the data packets.

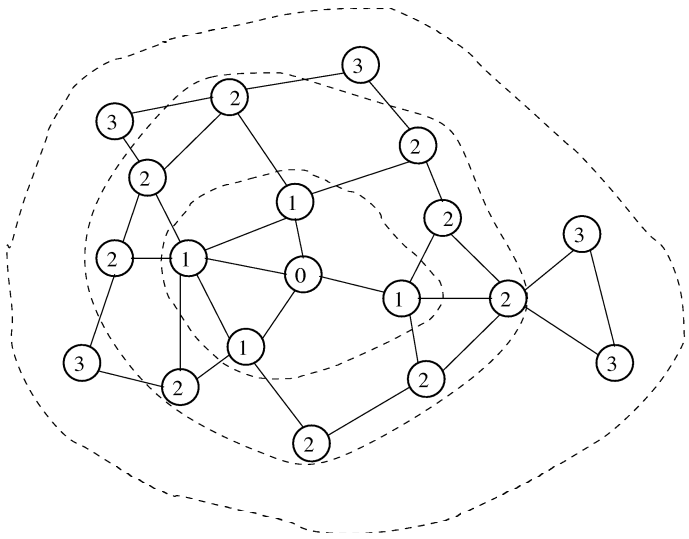

Figure 4.2: Tiers are assigned to all nodes.

4.4 Neighborhood Table Construction

Since sensor nodes need information about their local neighborhood during the phases in rounds, they start a discovery phase for detecting the sensor nodes in the vicinity and constructing a neighborhood table. In order to give a good decision to sleep or not, each sensor node needs to find out all other sensor nodes which have overlapping sensing areas with itself. In this thesis, we consider the neighbors up to three hops and within $2R_s$ distance. This is because in a dense sensor network, even if the R_t/R_s value is quite small (i.e. 0.5), the neighbors which are more than three-hop away cover either minor or no part of the node's sensing area as shown in Chapter 3.

At the beginning of the Neighborhood Table Construction phase, each sensor node broadcasts a *Discovery* message which contains the ID and location of itself and a TTL value. Since each node searches for up to three hops, the TTL is set to 3 initially.

Each node receiving this *Discovery* message records the ID and location value

in the message into its *Neighborhood Table*, unless the node originating the message is further than $2R_s$. Moreover, the receiver node decreases the TTL by 1 and if the TTL is still bigger than zero, the node forwards the message to other nodes within its transmission range. If a node receives multiple *Discovery* messages with the same ID (same source), a similar procedure as in Tier Assignment phase is applied. The messages with higher TTL value have priority over the messages with lower TTL value. Furthermore, among the *Discovery* messages having the same TTL value, the message coming from the closest node is considered. Measuring the signal strength value for the received message can help in determining closeness.

For example, for the sample network illustrated in Figure 4.4, the Neighborhood Table of node 1 is constructed as in Table 4.4. Nodes which are one, two and three hops away from node 1 and which are within $2R_s$ distance are recorded into the Neighborhood Table of node 1. Note that, even though node 17 is three hops away from node 1, it is not recorded into Neighborhood Table of node 1 because it is not within $2R_s$ distance from node 1. Since at the beginning all nodes are assumed to be active, the status of each node in Neighborhood Table is assigned to be active initially.

4.5 Mode Selection Phase

Since our algorithm has a distributed approach, each node in the network should be able to decide which mode it will be during the round, using only local information. In our algorithm, each node can be in one of the following modes:

- ON-DUTY: All three units (sensing, communication, and processing units) of the sensor node are turned on.
- TR-ON-DUTY: Only the transmission and processing units of the sensor node are turned on. Sensing unit is turned off. The node does not sense the environment.

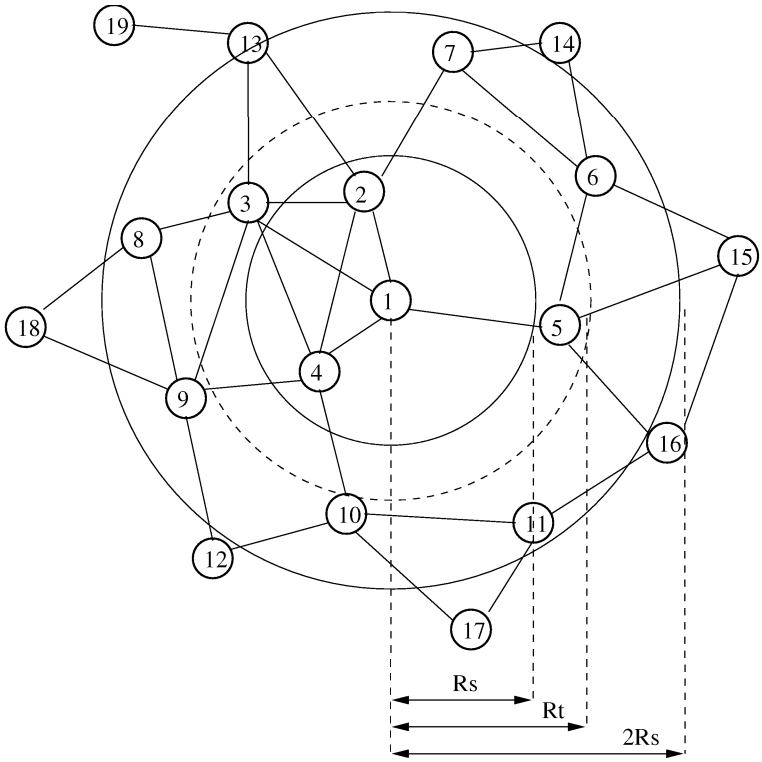

Figure 4.3: A sample network

Neighborhood Table for node 1		
1-hop		
ID	Location	Status
2	Loc_2	Active
3	Loc_3	Active
4	Loc_4	Active
5	Loc_5	Active
2-hops		
ID	Location	Status
7	Loc_7	Active
13	Loc_{13}	Active
8	Loc_8	Active
9	Loc_9	Active
10	Loc_{10}	Active
6	Loc_{11}	Active
3-hops		
ID	Location	Status
11	Loc_6	Active

Table 4.1: Neighborhood Table of node 1 in the sample network shown in Figure 4.4.

- DEEP-SLEEP: All units of the sensor node are turned off.

The necessary information for a sensor node to decide its new mode is obtained by the construction of a neighborhood table as it is described in the previous section. Mode selection phase consists of three parts; backoff delay computation, coverage eligibility check and connectivity check. Each node must check not only whether its sensing area is covered by the nodes in its neighborhood table as much as a user defined threshold value but also check whether the connectivity is violated in the case of its mode change. However, due to the independent and distributed processing of sensor nodes, some closer nodes may want to decide its mode at the same time. This causes message contention and provides unhealthy results for eligibility checks. Therefore, we assign randomized backoff delays for each node to resolve contention.

Figure 4.4 shows the overall mechanism in mode selection phase. At the beginning of each round, each node assigns a backoff delay time and waits until this time expires. In the waiting period, its mode is TR-ON-DUTY because sensing time has not started yet and the other nodes can only need communication with this node. When the backoff timer expires, the node first applies our coverage check algorithm. If it coverage check cannot be passed, the node decides to be in ON-DUTY mode, otherwise the node applies a second algorithm on itself: connectivity check. In case it passes the connectivity check, the node decides to be in DEEP-SLEEP mode, otherwise it goes into TR-ON-DUTY mode. Furthermore, when sensing time comes, each node fulfills the requirements of its new mode.

4.5.1 Backoff Delay Computation

At the beginning of each round, all sensor nodes decide their modes that they will stay throughout the remaining time of the current round. Since the main objective is energy conservation, we need to put as much nodes into sleep mode as possible and try to choose the ones which have higher remaining energies as active nodes. If the nodes attempt to determine their modes at the same time, message contention occurs, thus no reasonable results may occur. We resolve the

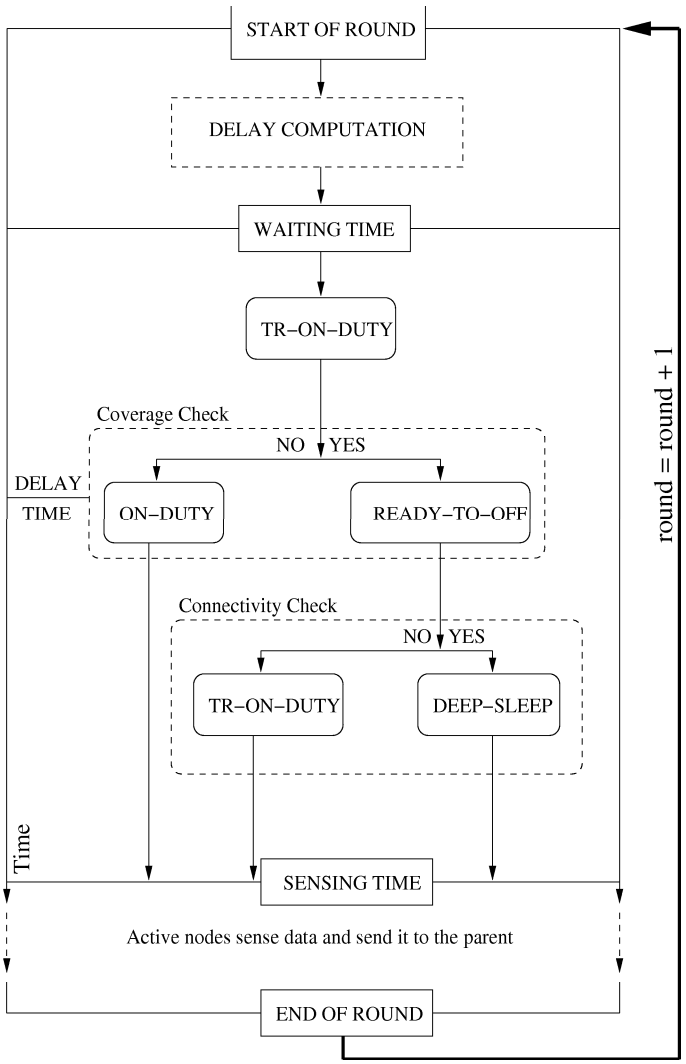

Figure 4.4: The overall mechanism to decide which mode a sensor node will be in each round.

contention problem by using a backoff delay mechanism similar to the one that SPAN [18] applies. Each node chooses a backoff value and delays the decision of its mode as that amount of time. When the exact time for mode selection comes, it decides its mode according to the states of the nodes in its neighborhood table at that moment.

In the light of our energy conservation objective, we should highlight the effect of the remaining energy levels of nodes. The nodes with lower energies should have priorities for energy conservation. In other words, if a chance exists for them to close some or all of their units, they could be permitted to go to sleep earlier than the nodes having higher remaining energy. Let $E_r(i)$ denote the remaining energy at the battery of node i and $E_t(i)$ denote the total energy available at the beginning. The nodes having smaller value of $E_r(i)/E_t(i)$ can have higher priorities for shutting down their functions, because they suffer from energy lack more than the others do.

Furthermore, the nodes which have more neighbors in their transmission range should have longer backoff time than the nodes having less neighbors so that they will be more likely to stay active. Because as long as these nodes are active, they contribute to the coverage of more neighbor nodes and provide a connectivity between them. In other words, when these nodes stay active, more nodes benefit from their coverage while coverage eligibility check is performed. Let $N(i)$ be the neighbor count of node i and N_{max} be the maximum of $N(i)$'s in a network. The nodes having higher value of $N(i)/N_{max}$ shuld have lower priorities for turning on their units due to their effect on coverage and connectivity.

If we combine these issues, we propose the following as our random backoff delay equation:

$$backoff_{delay} = \left(\alpha(\frac{E_r(i)}{E_t(i)}) + \beta(\frac{N(i)}{N_{max}}) + R \right) \times T \tag{4.1}$$

Here R is a uniform random value in the interval [0,1-α-β], T is the size of random backoff time choices, and α, β are weights of energy and coverage parameters.

Although this delay mechanism produces different delays, it is possible that some nodes may have same or very close backoff delays. Because of these cases, some nodes may decide their status at the same time, thus some blind points that are covered by no node may occur. To prevent such kind of these cases, we force each node to wait a short period of time T_w after deciding its status. This time interval should be enough to receive possible status messages from a neighbor. If a message is received from a neighbor in T_w time period, the node recalculates its off-duty eligibility. Otherwise, it continues in standard procedure and changes its status to what it has decided. Besides it sends a status message to its neighbors. Choosing bigger T values can also decrease the contentions.

In this thesis, we use the round time as the period of the execution of our algorithm. We assume that in each round, each node sends its data to the sink node multiple times. Furthermore, we assume that each round in the network lifetime is large enough so that we can have a large backoff time selection window. By this way, each node can select a backoff value such that the difference between its backoff value and the nearest backoff value of other nodes provide enough time to do all required operations for determining its status.

Another issue is that, starting from the beginning of each round, each node can receive messages from its neighbors. Even backoff time of the node has not expired yet, the node can receive some messages from other neighbors and it replies them accordingly. For instance, a *StatusMessage* from a one-hop neighbor can be received and Neighborhood Table of the node is updated. But, the states of two-hop and three-hop nodes are updated when they are needed. We will talk about this in the next section.

Furthermore, there is always a possibility to lose messages due to collisions. To avoid these situations, a general approach of ACKs and retransmissions can be utilized. In this thesis, we do not address this issue and ignore the message loss due to collisions.

4.5.2 Coverage Eligibility Rule

As soon as the delay time expires for a node, it runs a coverage eligibility check algorithm with its current neighborhood information. Knowing the locations of nodes in its neighborhood table, each node checks whether its circular shape sensing area with radius R_s is covered by its neighbors with a ratio greater than a threshold value k or not. Here k is a user defined parameter of the algorithm ($0 \leq k \leq 1$).

Deciding how much of the sensing area of a node is covered by the neighbors is not a simple problem. Up to now, there are various approaches proposed in different studies. The most remarkable ones are the following:

- The first aproach is to build a grid over the sensing area of the node and check whether each grid cell is also covered by any of the neighbors. (Figure 4.5) We simply calculate the distance between the center of a grid cell and its neighbor. If it is smaller than R_s, the grid cell is also covered by that neighbor. Thus, to find the overlap of neighbors over a sensor node's sensing area, we simply take the ratio of the number of grid cells covered by neighbors to all cells in the node's sensing area. This approach is basically straightforward and is quite accurate. However, it has quite large time complexity.

- Another approach is the one used in [19]. The only required decision criteria to find out the coverage is the angle between the node and the intersection points of the node's sensing area with its neighbors' sensing area. If all angles extracted from each neighbor sums up to 2π, then a full coverage by neighbors is assumed. Therefore, a node can decide this by only dealing with the predefined angles.

- In [5], a simple way of dealing with coverage decision is proposed. It proves that if all intersection points of all other nodes' sensing areas inside the node's sensing area are covered, the area is also covered. In this method, we need to find the defined intersection points.

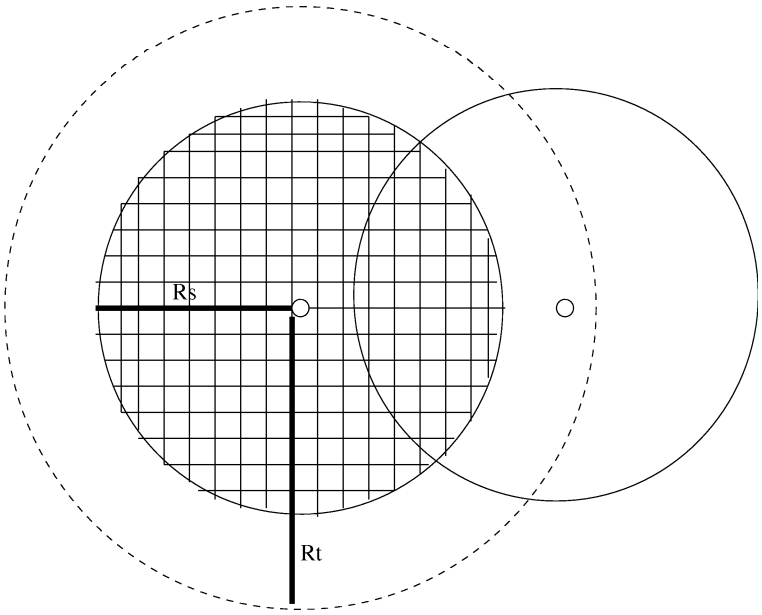

Figure 4.5: The sensing area of a node is divided into grid cells for coverage area calculation.

- In [25], it is proved that if the perimeter of sensing areas of sensors are k-covered then the area is k-covered. Here k is a positive user defined number and k-covered means the specified area is monitored by k different nodes.

Among the above methods, only the first one can give us the overlap ratio by neighbors. The others can only decide whether there is a full coverage or not. Therefore, in our solution we use this method which can simply be called as *GridCoverageCheck* method. On the other hand, this method's time complexity must be reduced. As a solution, we propose to do a predictive and step by step checking while deciding how much of a node's sensing area is covered by other nodes.

To achieve a good coverage check, a node should know all of other nodes which have common sensing areas with itself. However the number of nodes within $2R_s$ distance may be remarkably larger and this may require creating lots of control messages to have an updated knowledge of these nodes' status. Here, we propose to utilize the expected overlap tables which are derived in the Chapter 3.

An expected overlap table in a node contains the expected values of overlaps of a node's sensing area by n_1 one-hop and n_2 two-hop active neighbors of the node. The table contains the expected overlap value for different values of n_1 and n_2. In the table a cell (n_1,n_2) shows the expected overlap when there are n_1 one-hop and n_2 two-hop active neighbors of the sensor node of interest.

Algorithm 1 shows the pseudo-code of our coverage algorithm. Since nodes know their R_s and R_t values, they can compute the *Expected Overlap Table*. According to the desired overlap, which is a user defined parameter and which indicates the minimum required overlap of other nodes to pass coverage check, the node decides on the set of nodes in its Neighborhood Table that will be considered in the coverage check. Here, if the required set is only one-hop nodes, no extra message is needed because one-hop neighbor information is always updated via *StatusUpdate* messages. However if the set also includes other nodes, the node should learn their current status from its neighbors. By this algorithm, we not only lessen the load of control messages which update the status of nodes in Neighborhood Table but also make a good coverage check of a node depending on its R_s and R_t values.

When the node decides to use the nodes in two and three hops in coverage check, it needs to update the status information of these nodes. The node creates a *StatusQuery* message and broadcasts it with a TTL value. If the node wants to learn the updated status of two-hop neighbors, TTL is set to 1, whereas if the node wants to learn the status of three-hop neighbors, TTL is set to 2. If a node receives a *StatusQuery* message, it decreases its TTL by 1. If it is equal to 0, it replies back with a *StatusReply* message which contains the updated status of one-hop neighbors in its NT. Otherwise, the *StatusQuery* message is forwarded to other nodes.

Our algorithm becomes simpler when R_t/R_s value exceeds 2. Since in this case only one-hop neighbors will have overlapping area with the node's sensing area, the algorithm defines the *coverSet* to be the set of one-hop neighbors only.

Algorithm 1 CoverageCheck(n_1,*desiredCoverageRatio*)

```
if ExpectedOverlapTable(n1)≥ desiredCoverageRatio then
  coverSet = { nodes in n1 }
else
  Broadcast StatusQuery message with TTL 1
  Wait Tw time for receiving StatusReply messages
  n2 = Number of two-hop nodes in ON-DUTY mode.
  if ExpectedOverlapTable(n1,n2)≥ desiredCoverageRatio then
    coverSet = {nodes in n1 and n2}
  else
    Broadcast StatusQuery message with TTL 2
    Wait Tw time for receiving StatusReply messages
    coverSet = {nodes in Neighborhood Table that are active}
  end if
end if
coveredArea = 0;
desiredCover = desiredCoverageRatio × πR_s^2
for each node i in coverSet do
  coveredArea = coveredArea ∪ SensingArea(i);
  if coveredArea ≥ desiredCover then
    return TRUE;
  end if
end for
return FALSE;
```

4.5.3 Connectivity Eligibility Rule

After a sensor node passes coverage check, as the next part of the mode selection phase it runs connectivity check process. Even the sensing area of a sensor node is covered by its neighbors with a ratio more than a threshold, the node may be vital for the connectivity of sensors. The active nodes in a sensor network must be connected to be able to send their data to the sink node.

For connectivity check of a node, we follow a different procedure than earlier approaches. A node is connectivity eligible, i.e. can be turned off without harming connectivity, if and only if its one hop neighbors can send their data to sink over a path that does not contain this node. To figure out whether each of the neighbors can achieve this, we use the tier number concept. After the Tier Assignment phase, all nodes have tier numbers. Further each node assigns the node which sent the tier assignment message to itself, as its parent node. By this way, we have a tree structure for routing which enables shortest path to the sink node. This also decreases possible energy consumption that may occur due to long paths for routing. When a node wants to learn whether the routes of their one hop neighbors' to sink node include itself, it is sufficient to check only the nodes that know this node as its parent node.

If a node passes the coverage check, it changes its mode to *READY-TO-OFF*. Then it broadcasts its new mode to the neighbors in its transmission range via *StatusUpdate* messages. Receiving this message, each neighbor evaluates its current condition and sends a reply message to questioning node according to the following conditions.

1. If the parent node of neighbor node is not the questioning node, it sends a *Connectivity-Ok* message, because it does not need the questioning node for sending its data to the sink.

2. If the current parent node of neighbor node is the questioning node, it looks for another possible parent node among its neighbors. The node i sends a *TierQuery* message asking their tier numbers to its one-hop neighbors.

According to the *TierReply* messages which contain the tier number of neighbors, it creates a set of Possible Parent (PP) nodes which is defined as follows:

$$PP(i) = \{j \in N(i) \mid tier(j) = tier(i) - 1\} \tag{4.2}$$

- If PP(i) set is not empty, node i selects one of them as its new parent node and sends a *Connectivity-Ok* message to the questioning node.
- On the other hand, if $PP(i)$ set is empty, node i sends back a *Connectivity-Not-Ok* message, because there is no way for node i to send its data to the sink node without using the questioning node and without possibly making the path longer.

As the next step, after waiting sufficient time for the operations above, the questioning node checks whether it has received a *Connectivity-Not-Ok* message or not. If it is the case, it changes its mode to TR-ON-DUTY mode in which only transmission and processing units of the sensor node are functioning, i.e. sensing unit is turned off. Otherwise, it passes the connectivity check as well and decides to be in DEEP-SLEEP mode in the rest of the current round. Moreover, if the node decides to be in DEEP-SLEEP mode, it informs each one-hop neighbor about its mode via *StatusUpdate* messages so that its neighbors can update their Neighborhood Tables.

Figure 4.6 shows the status of a network before and after a connectivity check on a sample node, E. The children of node E can connect another parent node so that node E decides to be in sleep mode.

In each round, this connectivity eligibility algorithm is executed as usual. However, if a node notices that it will probably die (lost its energy) in the next round, it informs its situation to one hop neighbors via *Bye* messages and ask them to make a local recovery about their tiers. As a consequence of these messages, each child of this node tries to connect itself to one of the other nodes in its transmission range following the process of PP set creation (sending a *TierQuery* message and receiving *TierReply* messages). In the selection of a new parent node, the one with minimum tier number and closest distance is considered first.

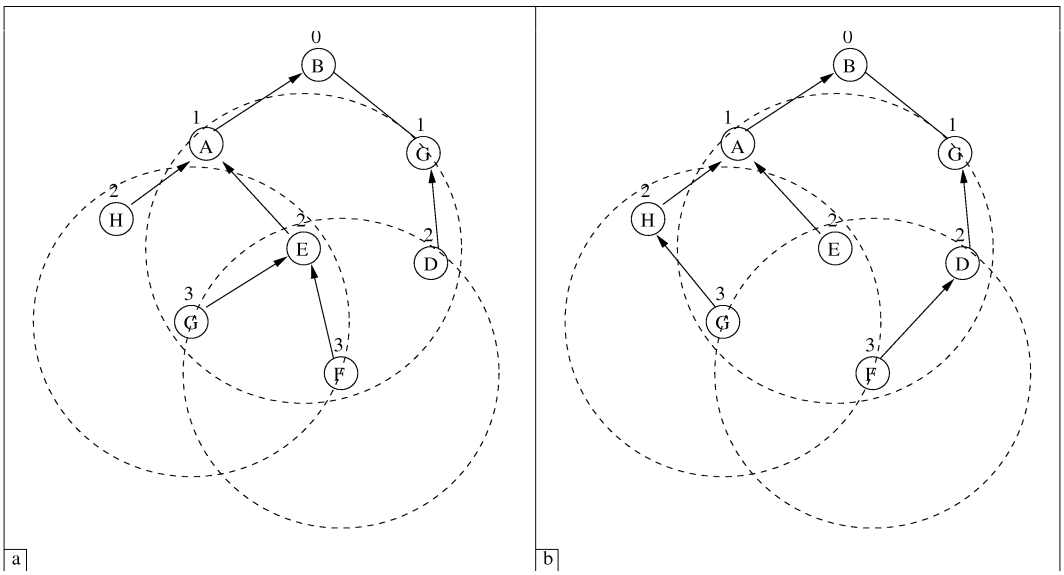

Figure 4.6: The status of the network before (a) and after (b) connectivity check on node E. The neighbors of node E can connect to another parent. Therefore it changes its mode to DEEP-SLEEP mode.

If the tier number of a child changes, then it also wants its children to update their tier numbers according to itself. This is achieved by *TierUpdate* messages.

Figure 4.7 shows two different cases of a dying node situation. In Figure 4.7a, a sample network is shown. Node A detects that it will probably die due to loss of energy in the next round. Therefore, it sends a *Bye* message to its neighbors. In Figure 4.7b, the chidren of node A try to connect other parent nodes. Node B can find a PP node, D, but nodes C and D cannot find a PP node. Then, node C chooses node E (same tier with node E) as its parent node and sends a *TierUpdate* message to its children. Finally, node C becomes a PP node for node P and node P selects node C as it is new parent node.

However, in some cases some of the children of the dead node may not find new parents. In these cases, these children send a *GlobalTierAssignment* message indicating this condition to the sink node and a new global tier assignment process is started by the sink node.

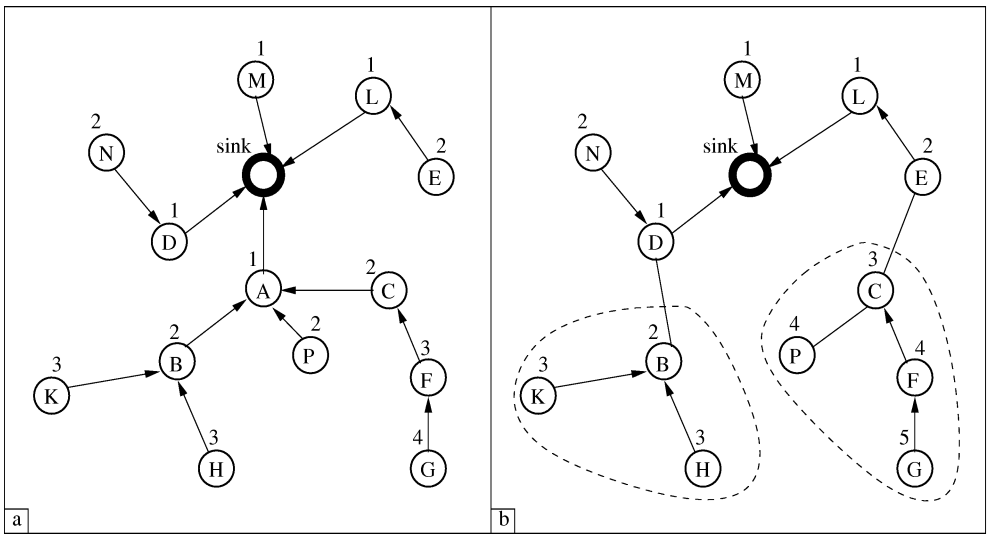

Figure 4.7: (a) The node A informs its neighbors about its death in the next round. (b) Tier numbers of children do not change (left enclosed region). No *TierUpdate* message is required. Tier numbers of children change (right enclosed region). Other children in the levels below are informed, and they update their tier numbers accordingly.

4.6 Operation Phase

After mode selection phase, each sensor node decides its prospective status during the rest of the round. According to the mode it selects, each node fulfills its duty until the beginning of the next round. Each node is in one of the following states:

- ON-DUTY: All units of the sensor node are turned on. Node senses data from the environment within its sensing range and sends it to the parent node. Further it relays the traffic of other nodes if it is the parent node of them.
- TR-ON-DUTY: Only the transmission and processing units of the sensor node are turned on while sensing unit is turned off. The node only relays traffic of other nodes.
- DEEP-SLEEP: All units of the sensor node are turned off. Only a synchronized timer works within the sensor node. When the time of the next round

comes, the node wakes up and decides its new mode according the updated information of neighboring nodes.

Note that, in this thesis, we assume that nodes use data aggregation while forwarding data towards to the sink node.

Chapter 5

Performance Evaluation of Our Algorithm

In this chapter we report the results of our simulation experiments that we performed to evaluate our algorithm. For the simulations, we designed a visual simulator and implemented it using the Java platform.

We have done two kinds of simulations: 1) coverage performance tests; and 2) system lifetime tests. While the tests in the first category are done to see the performance of coverage eligibility rule in our algorithm, the tests in the second category are performed to see how long a network's lifetime is extended with our solution.

5.1 Coverage Performance Tests

In this part of simulations we wanted to see how effective our algorithm is in putting the nodes into sleep mode, i.e. what percent of nodes can be put into sleep mode without harming the network's expected functionality, which is covering the deployment area to sense and monitor and maintaining the connectivity of active nodes.

To test the performance of our coverage eligibility rule, we used a sensor network model similar to the one used in CCP [6]. The model includes 100 randomly deployed static nodes, and the sensing and communication range of each node is set to 10 m. The region to be monitored is a 50 m by 50 m region. The coordinates of 100 nodes are determined in a random manner at each run of the simulation experiments. To calculate the sensing coverage area of a node, we used a common approach. We divide the region into 1x1 m small grid cells and if the center of a grid cell is covered by a node, the whole grid cell is assumed to be covered by that node. Besides, the results in the graphs are obtained as the average of five different runs.

Figure 5.1 and Figure 5.2 show the working node count for different number of deployed node counts. As it is seen in both graphs, our algorithm needs less number of nodes than CCP-SPAN algorithm and nearly the same number of nodes as CCP-SPAN-2Hop algorithm. However, the number of working nodes needed in our algorithm include both ON-DUTY and TR-ON-DUTY nodes. Figure 5.3 shows the number of TR-ON-DUTY and ON-DUTY node count for the case in Figure 5.1. As it is stated before, the nodes in TR-ON-DUTY mode turn off their sensing units hence their energy consumption is less than the nodes in ON-DUTY mode in which all three units are turned on.

Figure 5.4 shows a sample network with 100 nodes and Figure 5.5 shows the resulting network after our algorithm is applied on this network. The nodes in ON-DUTY mode are shown with a black circle and the nodes in TR-ON-DUTY mode are shown with a white circle.

We have also made simulations to see the effect of R_t/R_s ratio on the number working nodes. This time, we have decreased the value of R_s to 6.25 m to see the difference more clearly when R_t/R_s ratio is bigger. The number of deployed nodes is 800 when the R_t/R_s ratio is 0.5, and 200 in all other ratios.

Figure 5.6 shows the the number of working node count for different R_t/R_s ratios. Our algorithm needs less number of nodes than CCP-SPAN algorithm for all ratios. On the other hand the difference in the number of working nodes needed by our algorithm and CCP-SPAN-2Hop algorithm is very small for all

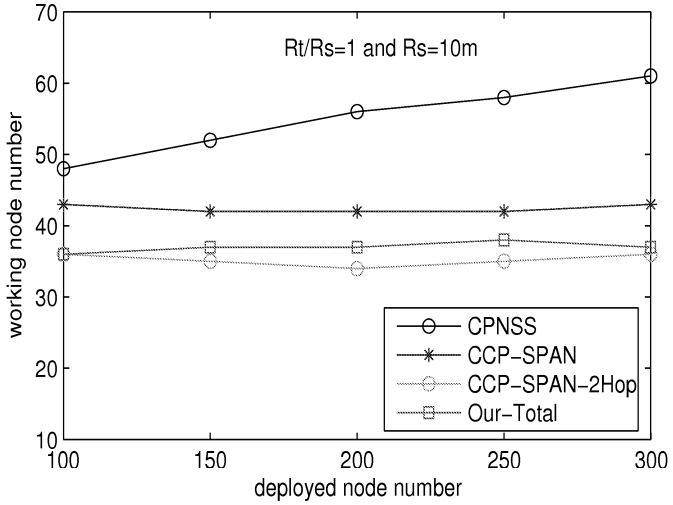

Figure 5.1: Working node count vs. Deployed node Count. R_s=10m and R_t=10m.

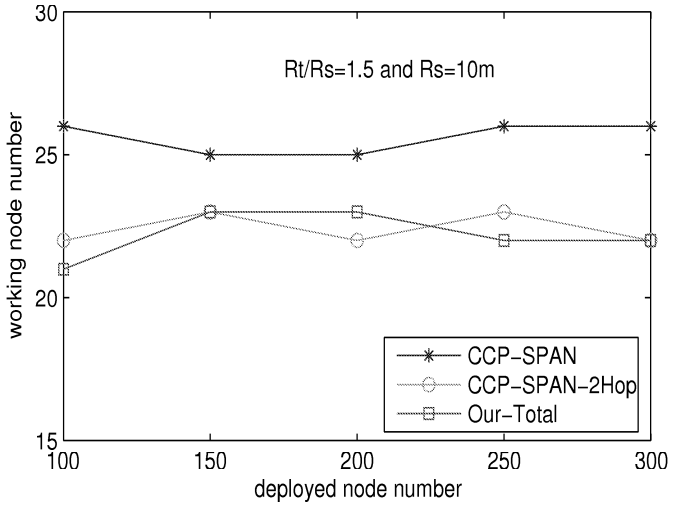

Figure 5.2: Working node count vs. Deployed node Count. R_s=10m and R_t=15m.

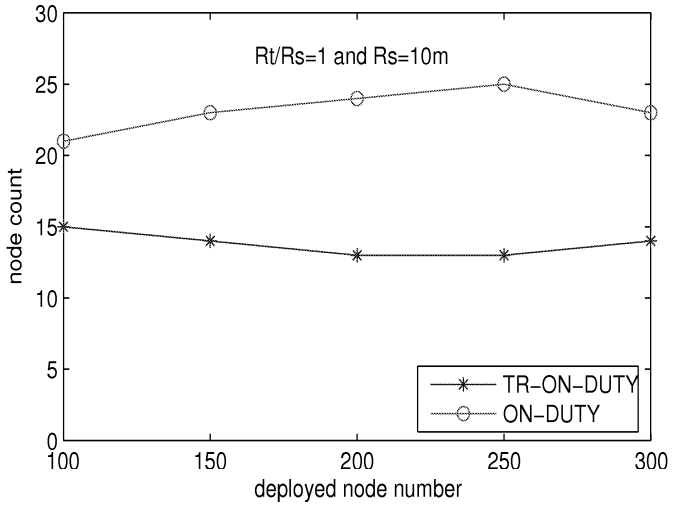

Figure 5.3: TR-ON-DUTY and ON-DUTY node count vs. Deployed node Count. R_s=10m and R_t=10m.

ratios; our algorithm needs a little less number of nodes. When the R_t/R_s ratio is 0.5 the difference is the biggest. This is due to our predictive coverage check algorithm which gives better results for small R_t/R_s ratios.

While in CCP algorithm nodes always update the status of their one-hop neighbors via *HELLO* messages, in CCP-2Hop algorithm, nodes update the status of their two-hop neighbors. When they are combined with SPAN algorithm, each node needs to update their two-hop neighbors because SPAN needs them for the coordinator announcement rule. However, in our algorithm, at first we only know the status of one-hop neighbors and whenever it is needed, we update the status of two and three-hop neighbors.

We wanted to compare the overhead due to these kinds of messages used by our algorithm and CCP-SPAN and CCP-SPAN-2Hop algorithm. Figure 5.7 shows the total number of neighbors whose information is used by nodes to perform their coverage check. Our algorithm uses slightly more nodes than CCP-SPAN algorithm; however, it uses remarkably less number of nodes than CCP-SPAN-2Hop algorithm. We see that our algorithm can achieve the same number of working nodes as CCP-SPAN-2Hop algorithm while using less number of neighbors

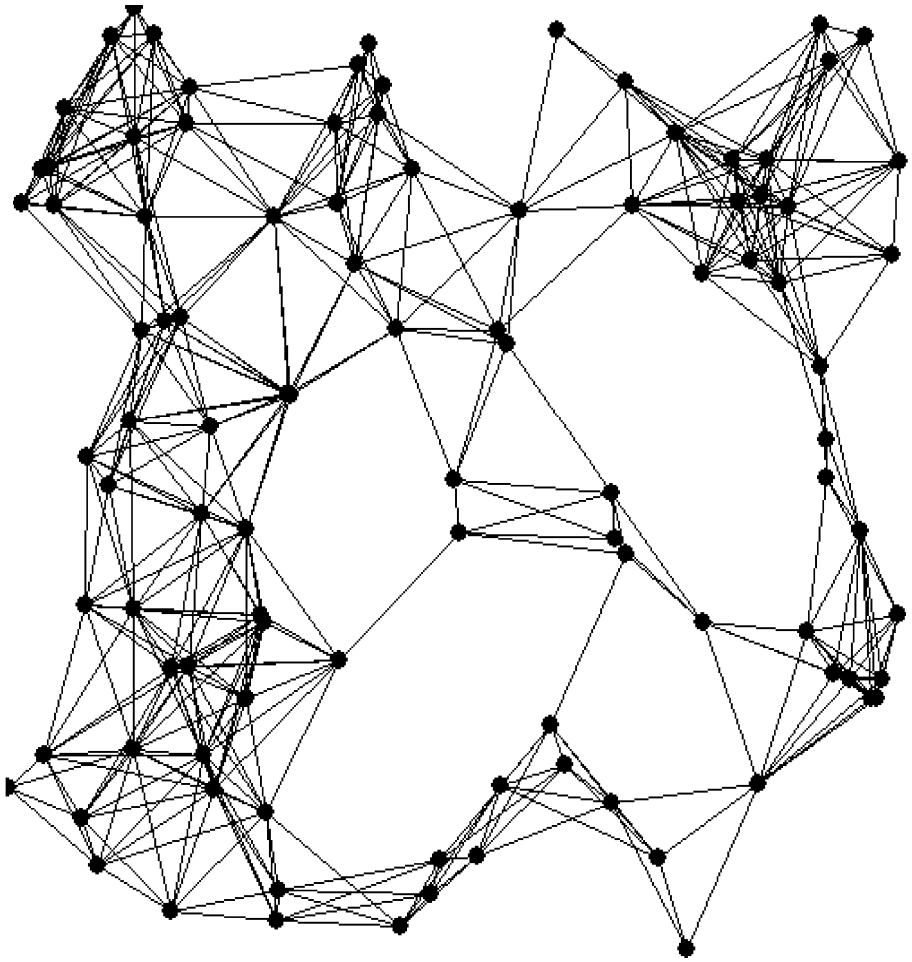

Figure 5.4: A sample network with 100 randomly deployed nodes. Nodes in transmission range of each other are connected with a link. The size of the area is 50 m by 50 m and $R_s = 10$ m.

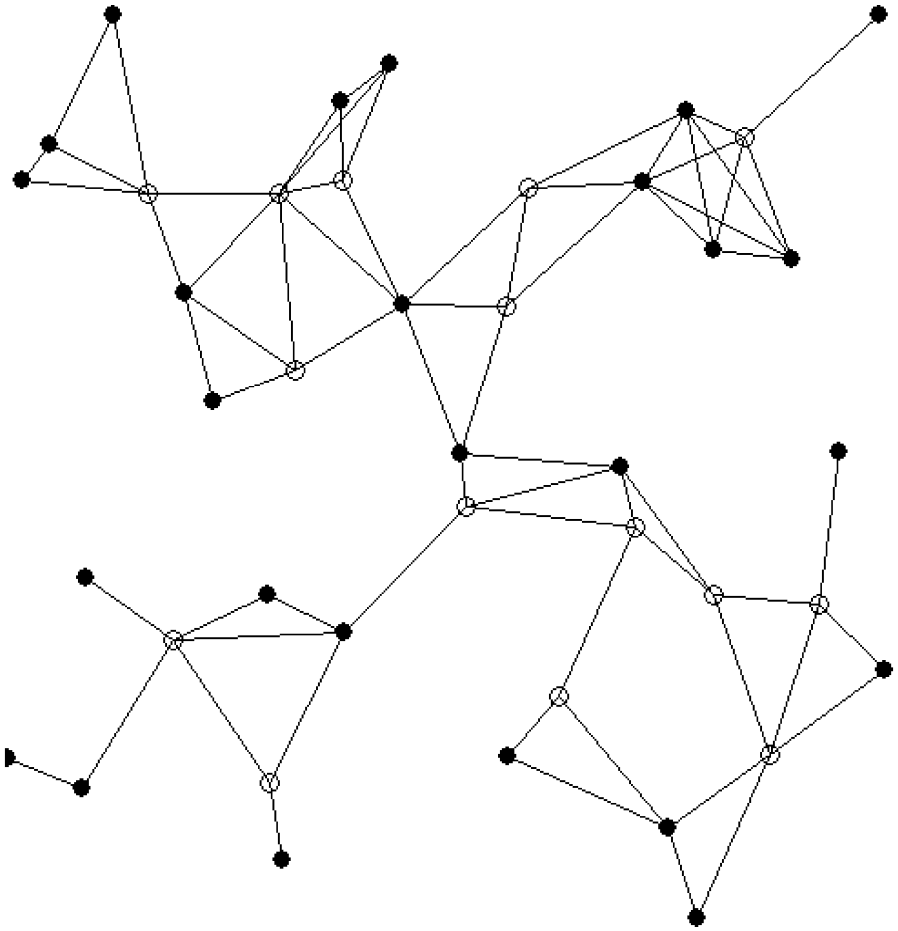

Figure 5.5: The resulting network after our algorithm is applied on the sample network in Figure 5.4. The nodes in ON-DUTY mode are shown with a black circle and the nodes in TR-ON-DUTY mode are shown with a white circle.

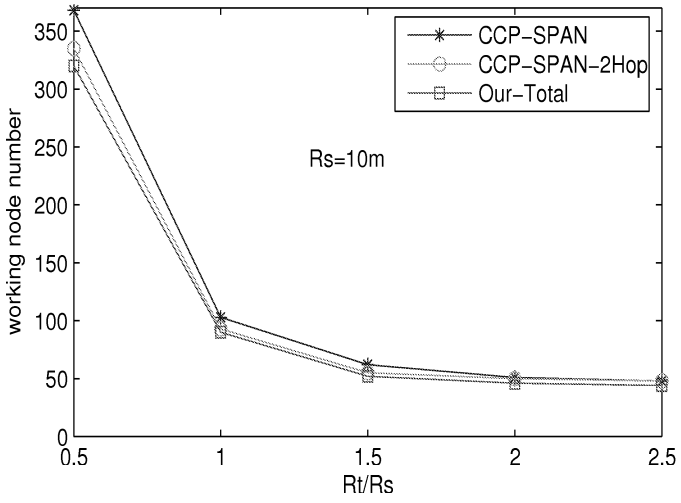

Figure 5.6: Working node count vs. R_t/R_s ratio.

in coverage check. This figure only shows the total number of neighbors used in coverage check, but if we consider the messaging cost, our algorithm needs less messaging than both CCP-SPAN and CCP-SPAN-2Hop algorithm due to two-hop neighbor updates required by SPAN algorithm.

5.2 System Lifetime Tests

In this part of simulations, we evaluated the energy efficiency and sensing coverage performance of our algorithm.

In most of the early studies, while analysing the energy consumption of wireless sensor networks, only the energy spent in communication unit is considered. However, sensor nodes consist of three different basic units that can consume energy: sensing, processing and communication units. Since the energy consumed while processing data is very small compared to the energy consumed in other units, for most applications, we can assume that the energy consumption in processing unit does not have a significant effect on the lifetime of the sensor

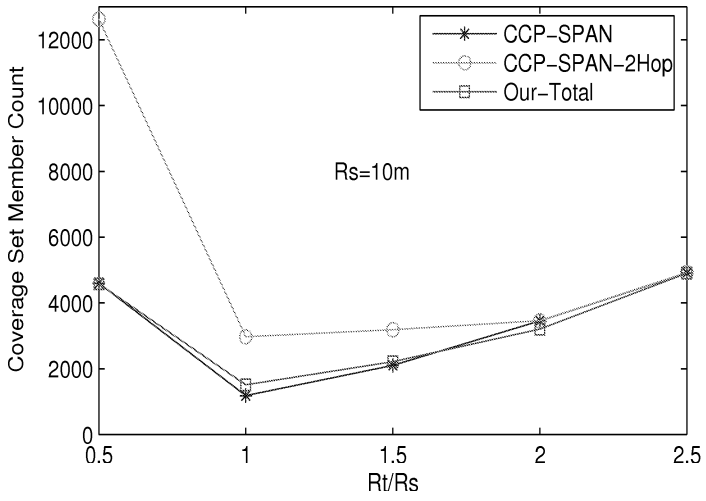

Figure 5.7: Total number of neighbors used while performing the coverage check rule in a network.

network. That's why this energy consumption is generally ignored in simulations. However, in some simulations authors also ignore the energy consumption in sensing unit. But, according to [2] and [26], the energy dissipated for sensing one bit of information is approximately equal to the energy dissipated in receiving a bit. Therefore, in our simulations we consider the energy consumptions in both sensing and communication units.

In the communication unit, we use the following energy consumption model for a node i in a network that applies a tree-based routing scheme. (Tiers create a tree-based routing structure). This is the model used in [27]. We also assume the sensor nodes in the network perform data aggregation while forwarding their data.

$$E_{i,Communication} = E_{Receiving} * n_i + E_{Sending} \tag{5.1}$$

This indicates that a node i spends energy while receiving data packets from n_i neighbors (children in the routing tree) and while sending the aggregated data packet to the next node (the parent). The constants $E_{Receiving}$ and $E_{Sending}$ depend on the communication technology. Some studies assume these constants to be equal [27], and some studies consider the $E_{Sending}$ constant to be slightly

larger than the $E_{Receiving}$ constant [28, 29]. We assume a ratio of 2/2.5 for $E_{Receiving}/E_{Sending}$.

Besides, we also consider the energy consumption in sensing unit. As it is indicated in [2] and [26], energy consumption while receiving a bit is approximately equal to the energy consumption while sensing a bit worth of information. Therefore, we assume energy consumption ratio while sensing, receiving and sending data as 2:2:2.5, respectively. Moreover, if the size of a data message is q times the size of a *StatusUpdate* message, we assume that q is 20 in all simulations. However, we also made a simulation experiment showing the effect of the q constant.

For energy and coverage performance experiments, we set the sensing range to 10 m. Besides, to see the effect of range ratio, we performe three different simulations for ratios 0.5, 1.0 and 1.5. The number of deployed nodes are 80, 150 and 300, respectively. The region is a square of 50 m x 50 m. The base station is located at the center of the region. Initially, each node is assumed to have 100 units of energy. In each round of communication, each node senses the environment, packetizes the information, and sends it towards the sink. The system is simulated until the coverage becomes very low.

Figures 5.8, 5.9, 5.10 show the coverage percentage, number of still alive nodes, and total remaining energy in nodes respectively. The range ratio is set to 0.5, that is, $R_t = 5$ m and $R_s = 10$ m. Here, coverage percentage is calculated considering only the coverages of nodes which can reach to the sink node, i.e. which can send data to the sink node. This is different than some coverage percentage definitions which consider the coverage of all sensors without looking their reachibility to the sink node.

Figures 5.11, 5.12, 5.13 show the same metrics when the R_t/R_s ratio is 1.

Finally, Figures 5.14, 5.15, 5.16 show the same metrics when R_t/R_s ratio is 1.5. Here, we notice that the performance of CCP-SPAN-2Hop algorithm gets closer to our algorithm. This is due to the fact that as R_t/R_s ratio increases, messaging cost of CCP-SPAN-2Hop algorithm decreases. We also observed this fact in Figure 5.7.

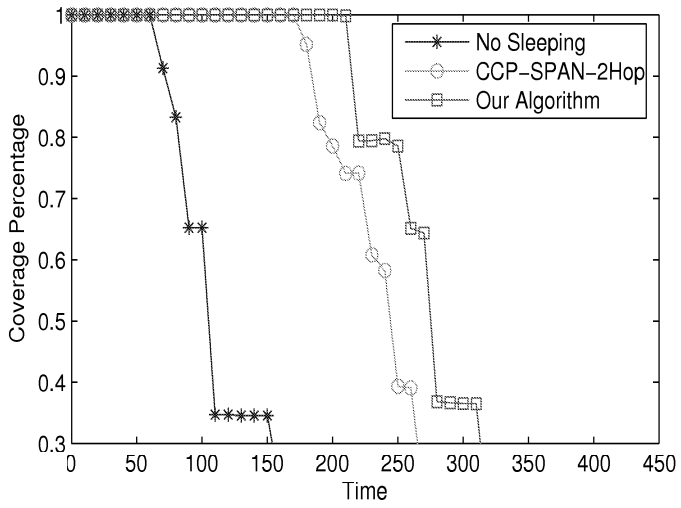

Figure 5.8: Coverage percentage in a sample run when $R_t/R_s = 0.5$.

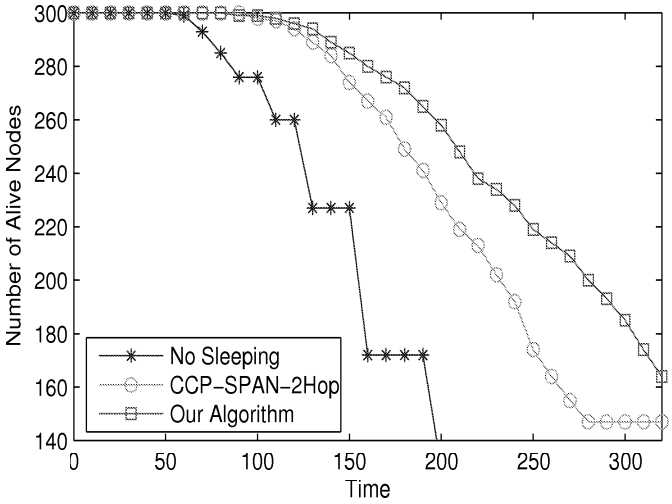

Figure 5.9: Number of still alive nodes in a sample run when $R_t/R_s = 0.5$.

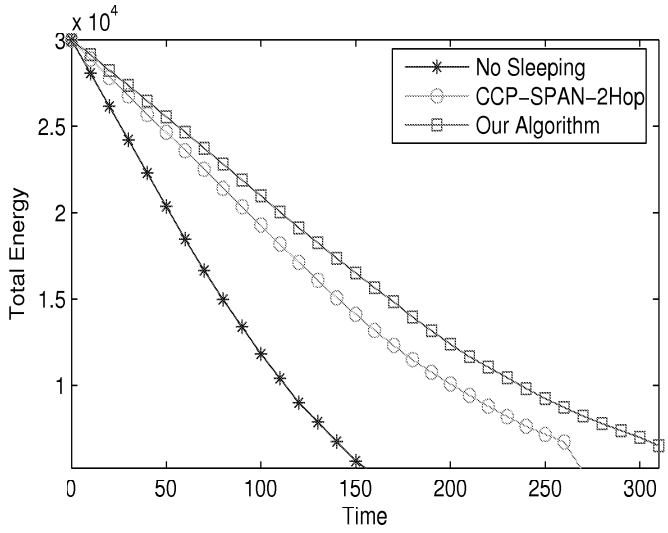

Figure 5.10: Total energy in nodes in a sample run when $R_t/R_s = 0.5$.

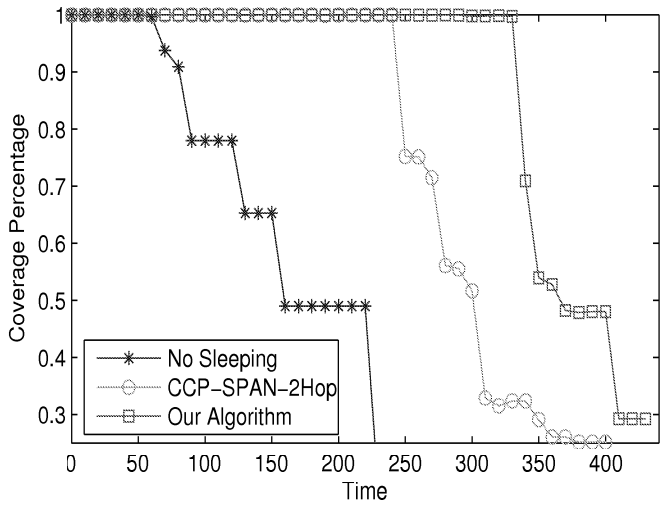

Figure 5.11: Coverage percentage in a sample run when $R_t/R_s = 1$.

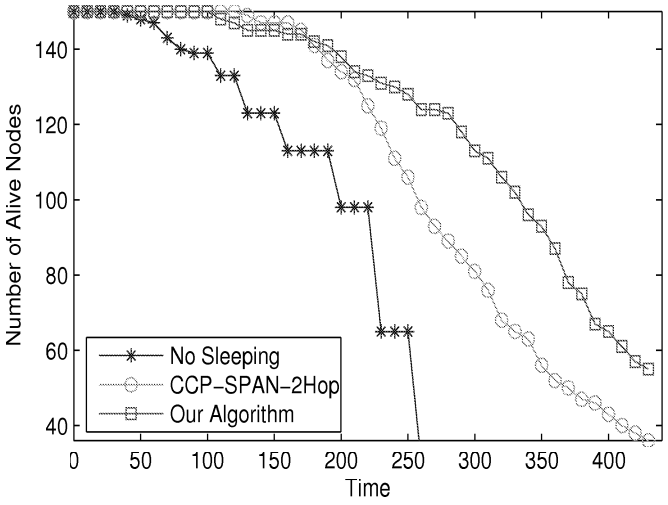

Figure 5.12: Number of still alive nodes in a sample run when $R_t/R_s = 1$.

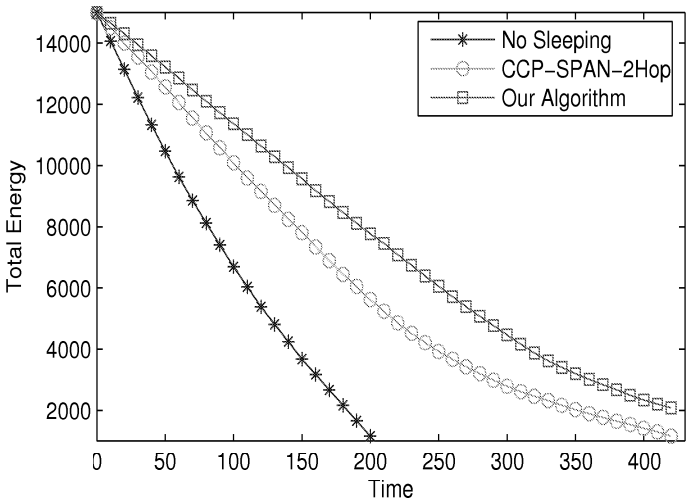

Figure 5.13: Total energy in nodes in a sample run when $R_t/R_s = 1$.

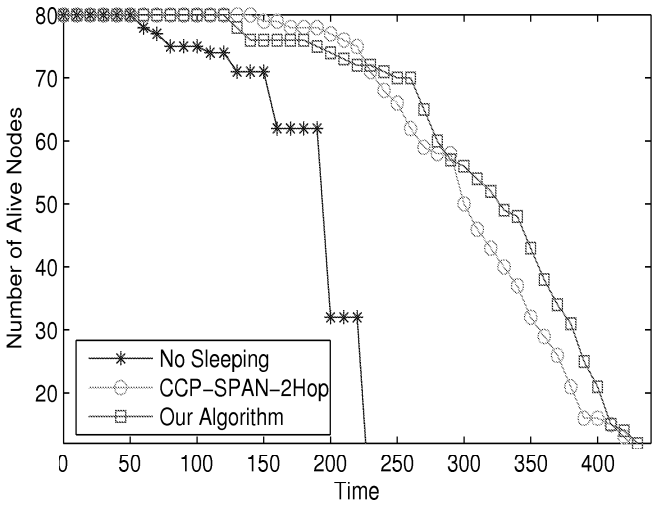

Figure 5.14: Coverage percentage in a sample run when $R_t/R_s = 1.5$.

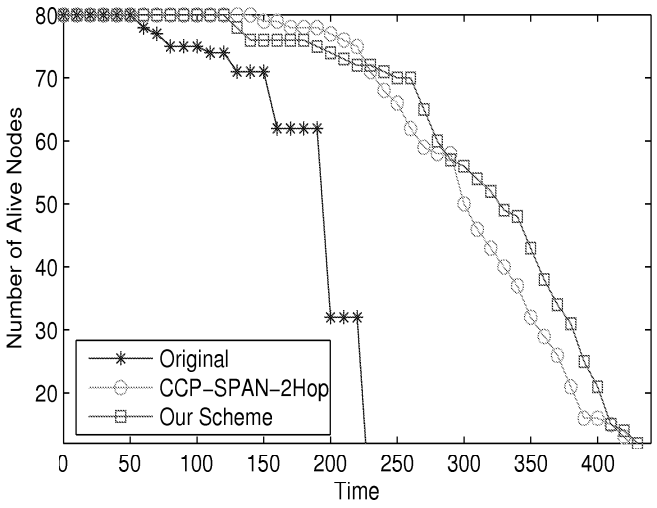

Figure 5.15: Number of still alive nodes in a sample run when $R_t/R_s = 1.5$.

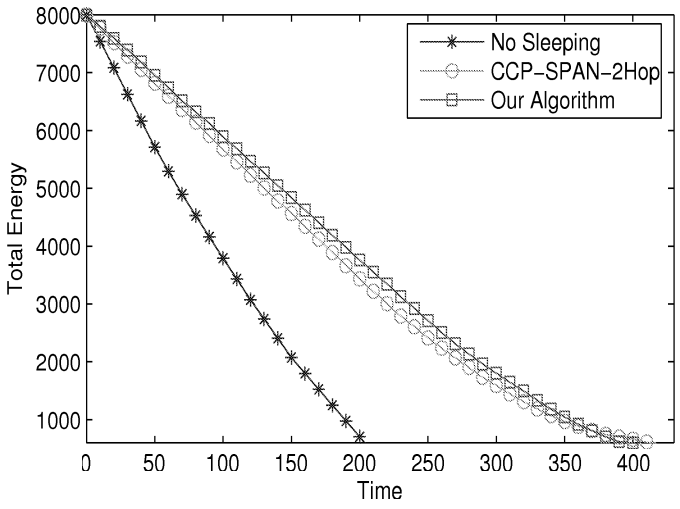

Figure 5.16: Total energy in nodes in a sample run when $R_t/R_s = 1.5$.

We also made a simulation showing the effect of the q constant (i.e. the ratio of the size of a data message to the size of a control message). Figures 5.17, 5.18 and 5.19 show the remaining energy of nodes in a sample network when CCP-SPAN-2Hop and our algorithm are applied, and when q is equal to 10, 20 and 50, respectively. Note that, when q increases, the energy consumption in CCP-SPAN-2Hop and our algorithm get closer to each other. Our algorithm reduces the amount of messages for maintenance and reduces the effect of these messages in energy consumption. However, if the size of these messages gets smaller with respect to data messages, then performance of our algorithm and CCP algorithm get closer.

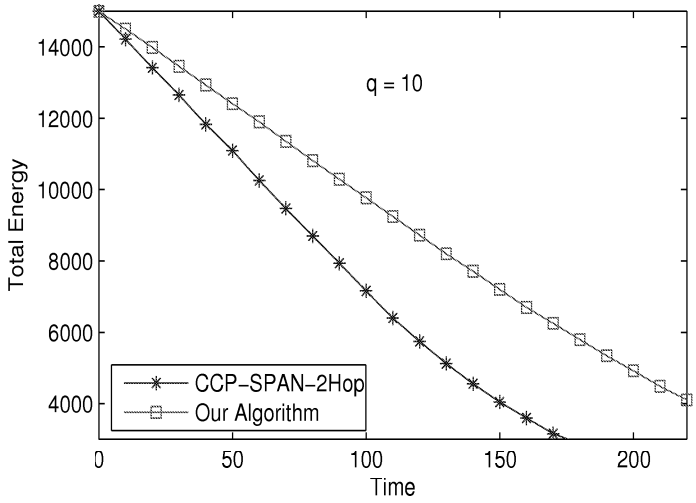

Figure 5.17: Total energy in nodes in a sample run when $q = 10$.

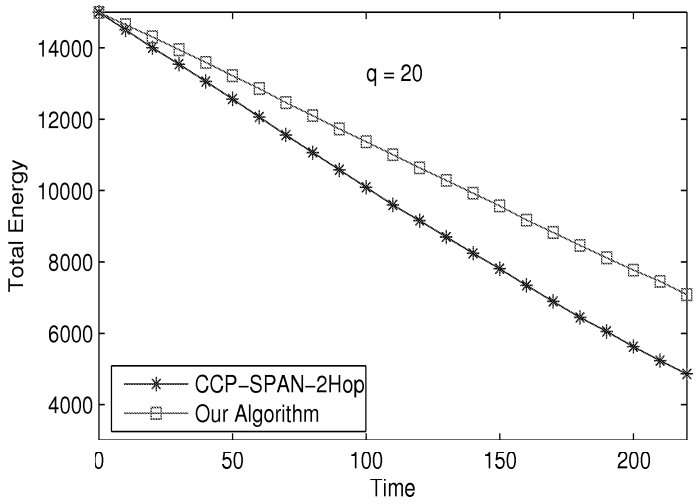

Figure 5.18: Total energy in nodes in a sample run when $q = 20$.

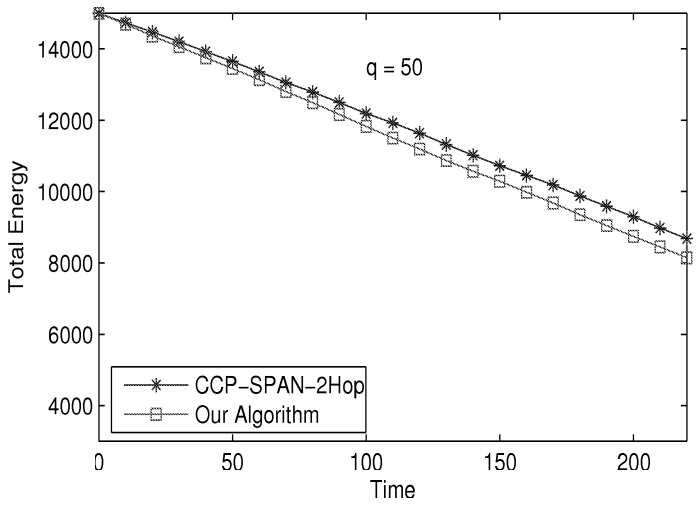

Figure 5.19: Total energy in nodes in a sample run when $q = 50$.

Chapter 6

Conclusion

In this thesis, we investigated the sleep scheduling problem for energy conservation in wireless sensor networks. Sleep scheduling aims keeping only a necessary subset of sensor nodes active, which can satisfy some user defined requirements such as certain coverage of the region. Besides covering the region adequately, a set of active nodes should also be well connected so that they can send their data to the sink.

We first did an analysis on the coverage of a node's sensing area, and found out that a sensor node can find the expected coverage ratio of its sensing areas by considering its one-hop and two-hop neighbors. The coverage ratio depends on the number of neighbors (density) and the transmission/sensing range ratio.

As the main contribution of the thesis, we provide a sleep scheduling solution that inludes a connectivity and coverage preserving algorithm with predictive coverage and multiple mode selections. In our solution, a sensor node can be in one of the three different modes by considering its basic units separately. In ON-DUTY mode, the sensor node has all its basic units (sensing unit, communication unit and processing unit) turned on. In TR-ON-DUTY mode, the sensor node has the sensing unit turned off, and communication and processing units turned on. In SLEEP mode, the sensor node has all the three basic units turned off. This is different than previous studies which only consider a sensor node as either

active or sleeping.

Our algorithm is also flexible in terms of the range ratio (tranmission range over sensing range). In other words, we do not assume a certain range ratio for the transmission range and sensing range. This is also different from most of the previous studies which provide a solution for only a certain tranmission and sensing range ratio (R_t/R_s). Our solution preserves a desired coverege and connectivity independent of this range ratio.

Furthermore, our algorithm also reduces the number of control messages that update the information about the states of the sensor nodes in the network. We use the update messages only whenever they are required.

We have performed simulation experiments and compared our algorithm with the only coverage and connectivity maintaining and distributed sleep scheduling algorithm called CCP. We have seen that in most cases our algorithm outperforms the CCP algorithm. Additionally, when the range ratio decreses, our algorithm provides better reults.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[2] Mohamed Younis, Moustafa Youssef, and Khaled Arisha. Energy-aware routing in cluster-based sensor networks. In *IEEE/ACM MASCOTS*, October 2002.

[3] Mihaela Cardei and Jie Wu. Energy-efficient coverage problems in wireless ad hoc sensor networks. *Computer Communications, special issue on Sensor Networks*, 2005.

[4] Lan Wang and Yang Xiao. A survey of energy-efficient scheduling mechanisms in sensor networks. *Mobile Network Applications*, 11(5):723–740, 2006.

[5] H. Zhang and J. C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Wireless Ad Hoc and Sensor Networks: An International Journal*, 1(1-2):89–123, 2005.

[6] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *SenSys'03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 28–39, New York, NY, USA, 2003. ACM Press.

[7] Eugene Shih, Seong-Hwan Cho, Nathan Ickes, Rex Min, Amit Sinha, Alice Wang, and Anantha Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *MobiCom'01:*

Proceedings of the 7th annual international conference on Mobile computing and networking, pages 272–287, New York, NY, USA, 2001. ACM Press.

[8] X. Cheng M. Min X. Jia D. Li D. Z. Du M. Cardei, D. MacCallum. Wireless sensor networks with energy efficient organization. *Journal of Interconnection Networks*, 3-4(3), 2002.

[9] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, pages 472–476, 2001.

[10] M. Potkonjak S. Megerian. Low power 0/1 coverage and scheduling techniques in sensor networks. In *UCLA Technical Report*, 2003.

[11] T. Yardibi. Sleep scheduling for energy conservation in wireless sensor networks with partial coverage. In *M.S.Thesis in Bilkent University EE Department*, 2006.

[12] Himanshu Gupta, Samir R. Das, and Quinyi Gu. Connected sensor cover: self-organization of sensor networks for efficient query execution. In *MobiHoc'03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing*, pages 189–200, New York, NY, USA, 2003. ACM Press.

[13] E. Bulut and I. Korpeoglu. Dssp: A dynamic sleep scheduling protocol for prolonging the lifetime of wireless sensor networks. In *International Workshop on Heterogeneous Wireless Networks (HWISE),in Proceedings of the IEEE 21st International Conference on Advanced Information Networking and Applications (AINA), IEEE Computer Society*, 2007.

[14] Ya Xu, John S. Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Mobile Computing and Networking*, pages 70–84, 2001.

[15] Fan Ye, Gary Zhong, Jesse Cheng, Songwu Lu, and Lixia Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. In *ICDCS'03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 28, Washington, DC, USA, 2003. IEEE Computer Society.

[16] S. Lu F. Ye and L. Zhang. Gradient broadcast: A robust, long-lived large sensor network. In *http://irl.cs.ucla.edu/papers/grab-tech-report.ps*, 2001.

[17] Ya Xu, John Heidemann, and Deborah Estrin. Adaptive energy-conserving routing for multihop ad hoc networks. Research Report 527, USC/Information Sciences Institute, October 2000.

[18] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8(5):481–494, 2002.

[19] Di Tian and Nicolas D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *WSNA'02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 32–41, New York, NY, USA, 2002. ACM Press.

[20] Saikat Ray, Wei Lai, and Ioannis Ch. Paschalidis. Statistical location detection with sensor networks. *IEEE/ACM Trans. Netw.*, 14(SI):2670–2683, 2006.

[21] Nirupama Bulusu, John Heidemann, and Deborah Estrin. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.

[22] Dragos Niculescu and B. R. Badrinath. Ad hoc positioning system (aps) using aoa. In *INFOCOM*, 2003.

[23] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *MobiCom'01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 166–179, New York, NY, USA, 2001. ACM Press.

[24] Yu-Chee Tseng, Sheng-Po Kuo, Hung-Wei Lee, and Chi-Fu Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. *Computer Journal*, 47(4):448–460, July 2004.

[25] Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor network. In *WSNA'03: Proceedings of the 2nd ACM international*

conference on Wireless sensor networks and applications, pages 115–121, New York, NY, USA, 2003. ACM Press.

[26] Yufu Jia, Tianlin Dong, and Jian Shi. Analysis on energy cost for wireless sensor networks. In *ICESS'05: Proceedings of the Second International Conference on Embedded Software and Systems (ICESS'05)*, pages 144–151, Washington, DC, USA, 2005. IEEE Computer Society.

[27] Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient Communication Protocols for Wireless Microsensor Networks. In *International Conference on System Sciences*, Maui, HI, January 2000.

[28] O. Kasten. Energy consumption, http://www.inf.ethz.ch/kasten/research /bathtub/energy/consumption.html.

[29] E. Jung and N. Vaidya. An energy efficient mac protocol for wireless lans. In *INFOCOM*, 2002.

Appendix A

Message Types Used By Our Protocol

In our sleep scheduling algorithm, different kinds of messages are used to maintain the algorithm. The first 4 bits of a message shows the type of the message. When a node receives a message, firstly it reads the first 4 bits of the message and reacts according to the type of the message. The messages used in our algorithm and the parts other than the message type field stored in messages are as follows:

- *TierAssignment Message:* It is used in Tier Assignment phase and consists of the ID and the tier number of the source node.
- *Discovery Message:* In Neighborhood Table Construction phase, each node creates this message. It includes the ID and location of the source node and a TTL value.
- *StatusUpdate Message:* Nodes inform their status change via this message. The message consists of the ID and status of the source node. There is no TTL value, because the message is sent only to one-hop neighbors.
- *StausQuery Message:* This message is used when a node needs to ask the current status of two or three-hop neighbors. The message contains the ID

of the source node and a TTL value which indicates two or three-hop neighbors. For two-hop neighbors TTL is set to 1 and for three hop neighbors TTL is set to 2. Because, for instance, the updated status of a two-hop neighbor can be found in the *Neighborhood Table* of a one-hop neighbor.

- *StatusReply Message:* When a node receives a *StatusQuery* message, it replies back to the source node with this message. The message contains the ID and status of the one-hop neighbors of the node and the destination ID .

- *TierQuery Message:* Nodes ask current tier numbers of their neighbors via this message. The message only contains the ID of the source node.

- *TierReply Message:* When a node receives a *TierQuery* message from a neighbor node, it replies back with this message which contains its ID and tier number.

- *Bye Message:* When a node notices that it will lose its energy in the next round, it sends a *Bye* message to its one-hop neighbors so that they look for another parent if it is necessary. The message contains only the ID of the source node.

- *TierUpdate Message:* If a node changes its tier number throughout the lifetime of the network, it informs its one-hop neighbors via this message. The message contains the ID and tier number of the source node. Children nodes, which send their data to sink via this node, then update their tier numbers accordingly and they also send a new *TierUpdate* message.

- *GlobalTierAssignment Message:* When a node cannot find a new parent node after the death of its parent, it sends this message to the sink node to start a global tier assignment process.

- *Connectivity-Ok* Message: If a node can connect to another parent node, it broadcasts this message. The message contains only the ID of the node.

- *Connectivity-Not-Ok* Message: If a node cannot connect to another parent node, it broadcasts this message. The message contains only the ID of the node.