

SLEEP SCHEDULING FOR ENERGY CONSERVATION IN WIRELESS SENSOR NETWORKS WITH PARTIAL COVERAGE

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Tarık Yardibi

July 2006

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 Karařan (Supervisor)

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. Nail Akar

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.

Asst. Prof. Dr. İbrahim K rpeođlu

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

SLEEP SCHEDULING FOR ENERGY CONSERVATION IN WIRELESS SENSOR NETWORKS WITH PARTIAL COVERAGE

Tarık Yardibi

M.S. in Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Ezhan Karahan

July 2006

Wireless sensor networks, which consist of many sensor devices communicating with each other in order to sense the environment, is an emerging field in the area of wireless networking. The primary objective in these wireless networks is the efficiency of energy consumption. Since these networks consist of a large number of sensors, allowing some of the nodes to sleep intermittently can greatly increase the network lifetime. Furthermore, some applications do not require 100% coverage of the network field and allowing the coverage to drop below 100%, i.e., partial coverage, can further increase the network lifetime.

A sleep scheduling algorithm must be distributed, simple, scalable and energy efficient. In this thesis, the problem of designing such an algorithm which extends network lifetime while maintaining a target level of partial coverage is investigated. An algorithm called Distributed Adaptive Sleep Scheduling Algorithm (DASSA) which does not require location information is proposed. The performance of DASSA is compared with an integer linear programming (ILP) based optimum sleep scheduling algorithm, an oblivious algorithm and with an existing algorithm in the literature. DASSA attains network lifetimes up to 89%

of the optimum solution, and it achieves significantly longer lifetimes compared with the other two algorithms.

Furthermore, the minimum number of sensors that should be deployed in order to satisfy a given partial coverage target with a certain probability while maintaining connectivity is computed and an ILP formulation is presented for finding the minimum number of sensors that should be activated within the set of deployed sensors.

Keywords: Wireless Sensor Networks, Partial Coverage, Sleep Scheduling, Network Lifetime

ÖZET

KABLOSUZ DUYUCU AĞLARINDA ENERJİ KORUNMASI İÇİN KISMİ KAPSAMALI UYKU DÜZENLEMESİ

Tarık Yardibi

Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans

Tez Yöneticisi: Doç. Dr. Ezhan Kardeşan

Temmuz 2006

Birbirleriyle haberleşen çok sayıda duyucudan oluşan kablosuz duyucu ağları, kablosuz ağ alanında çığır açan bir teknolojidir. Bu kablosuz ağlarda en önemli nokta enerji tüketimidir. Bu ağlarda yüksek sayılarda duyucular bulunduğundan, bazı duyucuları aralıklı olarak uyuma moduna sokmak ağ ömrünü büyük oranda arttırabilmektedir. Ayrıca, bir çok uygulama, yüzde yüzlük bir alan kaplaması gerektirmez ve kapsama oranının yüzde yüzün altına düşmesine izin verilerek, ki buna kısmi kapsama denir, ağ ömrünün daha da uzatılması sağlanabilir.

Duyucu ağları için tasarlanmış bir uyku düzenleme algoritması dağıtık, basit, ölçeklenebilir ve enerji kullanımında verimli olmalıdır. Tezimizde, kısmi kapsama alanı ile ağ ömrünü uzatan bu tarz bir dağıtık algoritma tasarımı sorunu incelenmiştir. Önerdiğimiz algoritma fiziksel yer bilgisi gerektirmeyen dağınık ayarlanır uyku düzenleme algoritmasıdır (DASSA). DASSA algoritmasının başarımını doğrusal tamsayı programlamasıyla bulunan en iyi uyku düzenleme algoritması, duyucuların çevrelerinden habersiz oldukları bir algoritma ve literatürde bulunan bir algoritma ile karşılaştırdık. Önerdiğimiz algoritma en iyi çözüme oranla %89'a varan ağ ömrü ve diğer algoritmalara göre de çok daha uzun ağ ömrü elde edebilmektedir.

Ayrıca, belirli bir olasılıkla bir kısmi kapsama hedefini sağlayabilmek ve bağlantılı olabilmek için alana atılması gereken en az duyucu sayısı hesaplanmış ve bu atılan duyucular arasından çalışır durumda ayarlanması gereken en küçük topluluğu bulmak için bir doğrusal tamsayı programlaması sunulmuştur. Bu sonuçlar, kablosuz duyucu ağları için verimli tasarımlar yapmak için kullanılabilir.

Anahtar Kelimeler: Kablosuz Duyucu Ağları, Kısmi Kapsama, Uyku Düzenlemesi, Ağ Ömrü

ACKNOWLEDGMENTS

Before all, I would like to thank to my supervisor Assoc. Prof. Dr. Ezhan Karařan for the continuous support and invaluable concern he has provided me throughout this research. It is his most pleasant personality, understanding and precise guidance which made this graduate study very enjoyable.

I would like to thank to Assoc. Prof. Dr. Nail Akar and Asst. Prof. Dr. İbrahim Körpeođlu for evaluating my thesis.

I am grateful to my family for never stopping to be on my side all the time and always giving me whatever they can. Nothing would be possible without the presence of them and the peaceful family environment they provided me during my life.

Finally, I would like to thank to TÜBİTAK and Nortel Networks Netař for their financial supports during this graduate study.

To Hatice & Cengiz

Yardibi ...

Contents

1	Introduction	1
1.2	Motivation	4
1.3	Contribution	5
1.4	Thesis Outline	6
2	Sleep Scheduling in Wireless Sensor Networks	7
2.1	Basic Concepts	9
2.2	Sleep Scheduling Considering Only Coverage	11
2.3	Sleep Scheduling Considering Only Connectivity	14
2.4	Sleep Scheduling Considering Both Coverage and Connectivity . .	16
2.5	Other Work on Sleep Scheduling	19
2.6	Routing Protocols	21
2.7	Partial Coverage	23
2.8	Definition of Sensor Network Lifetime for Partial Coverage	31
2.9	Our Contribution	32

3	The Minimum Connected Set with Sufficient Coverage	34
3.1	Minimum Set of Nodes Satisfying Only Coverage	35
3.2	Minimum Set of Nodes Satisfying Coverage and Connectivity . . .	39
3.3	Minimum Connected Set of Nodes among N_{min} Nodes Satisfying GoC (ILPMinConCov)	42
4	Centralized Optimum Sleep Scheduling	50
4.1	Network Model	51
4.2	Optimum Scheduling Algorithms	53
4.2.1	Optimum Node Scheduling Without Aggregation (ILPNA)	55
4.2.2	The Centralized Algorithm (CA)	61
4.2.3	Optimum Node Scheduling With Full Aggregation (ILPFA)	63
5	Distributed Adaptive Sleep Scheduling Algorithm (DASSA)	68
5.1	Step I : Neighbor Discovery	71
5.2	Step II : Scheduling Tier 1 Nodes	73
5.2.1	ILP implemented by the Sink (ILPSink)	73
5.2.2	Transmitting the Schedules	76
5.3	Step III : Scheduling Intermediate Nodes	78
5.4	Step IV : Scheduling Far Away Nodes	81
5.5	Step V : Transmitting and Forwarding Data	81

6	Performance Evaluation of Sleep Scheduling Algorithms	85
6.1	No Aggregation	87
6.1.1	Further Analysis of DASSA and OSSA	102
6.1.2	Unequal Transmission and Sensing Ranges	107
6.2	Full Aggregation	112
7	Conclusions	114
	APPENDIX	117
A	Algorithm Parameters	117

List of Figures

2.1	Transmission and sensing ranges of a sensor node.	9
2.2	The tier numbers in a network.	10
2.3	The blind point problem when nodes turn off simultaneously. . . .	13
2.4	Sensor field and monitored field in DRS.	25
2.5	Additional nodes required for connectivity in DRS, I.	28
2.6	Additional nodes required for connectivity in DRS, II.	28
2.7	Reporting group of each node in Figure 2.8.	29
2.8	The routing loop problem in DRS.	30
2.9	GoC-L in a sample coverage plot for $\text{GoC} = 0.9$	31
3.1	The network field divided into separate regions.	36
3.2	Coverage area of a point in area A_1, A_2, A_3 or A_4	38
3.3	Coverage area of a point in area B_1, B_2, B_3 or B_4	38
3.4	Analytical and simulation results for a 100m by 100m network. . .	40
3.5	Analytical and simulation results for a 200m by 200m network. . .	40

3.6	The coverage variable $\{v_i\}$	44
3.7	Minimum number of active nodes for a given GoC in a 200m-by- 200m network where $R_s = 50\text{m}$ and $R_t = 50\text{m}$	49
3.8	Minimum number of active nodes for a given GoC in a 300m-by- 300m network where $R_s = 50\text{m}$ and $R_t = 50\text{m}$	49
4.1	Number of packets received and transmitted when there is no ag- gregation applied.	54
4.2	Number of packets received and transmitted when there is full aggregation applied.	54
4.3	Forming the SID	56
4.4	The flow variable f_{ij}^n	58
4.5	Centralized algorithm (CA) for finding the optimum sleep schedule.	64
5.1	Broadcast message format.	71
5.2	The first step of DASSA.	72
5.3	Tier numbers of the network in Figure 5.2.	72
5.4	Broadcast message transmitted by the sink.	76
5.5	The second step of DASSA.	77
5.6	The balanced structure of DASSA.	78
5.7	The third step of DASSA.	79
5.8	Broadcast message transmitted by the tier 1 nodes.	79

5.9	The final step of DASSA.	82
5.10	Flowchart of DASSA.	83
5.11	Operation of DASSA for a sample network.	84
6.1	Network topologies used in the simulations.	88
6.2	Optimum results for Topology 1 and Topology 2.	89
6.3	The optimum p for OSSA.	90
6.4	DASSA, OSSA and the optimum results for $\text{GoC} = 0.9$, Topology 1.	91
6.5	DASSA, OSSA and the optimum results for $\text{GoC} = 0.8$, Topology 1.	91
6.6	DASSA, OSSA and the optimum results for $\text{GoC} = 0.7$, Topology 1.	92
6.7	DASSA, OSSA and the optimum results for $\text{GoC} = 0.9$, Topology 2.	93
6.8	DASSA, OSSA and the optimum results for $\text{GoC} = 0.8$, Topology 2.	93
6.9	DASSA, OSSA and the optimum results for $\text{GoC} = 0.7$, Topology 2.	94
6.10	Performance of DASSA and OSSA with respect to the optimum.	99
6.11	Energy and scheduling plots for $\text{GoC} = 0.9$ for Topology 1.	104
6.12	Energy and scheduling plots for $\text{GoC} = 0.8$ for Topology 1.	105
6.13	Energy and scheduling plots for $\text{GoC} = 0.7$ for Topology 1.	106
6.14	Coverage counts for $\text{GoC} = 0.9$ for Topology 1 in 62 rounds.	108
6.15	Coverage counts for $\text{GoC} = 0.8$ for Topology 1 in 75 rounds.	109
6.16	Coverage counts for $\text{GoC} = 0.7$ for Topology 1 in 93 rounds.	110

6.17 Scheduling overhead of DASSA for Topology 1.	111
---	-----

List of Tables

3.1	Minimum number of nodes required to satisfy GoC for a 200m-by-200m network field.	47
3.2	Minimum number of nodes required to satisfy GoC for a 300m-by-300m network field.	48
6.1	Results of a 100 node network (Topology 1) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	94
6.2	Results of a 100 node network (Topology 2) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	97
6.3	Results of a 100 node network (Topology 3) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	97
6.4	Results of a 100 node network (Topology 4) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	97
6.5	Performance of the algorithms when parameters are same for all topologies.	98
6.6	Results of a 150 node network (Topology 5) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	99

6.7	Results of a 200 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	100
6.8	Results of a 200 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	100
6.9	Results of a 400 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 300m-by-300m field.	101
6.10	Tier sizes for Topology 1 for different R_t	112
6.11	Results of a 100 node network (Topology 1) for $R_t = 60\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	112
6.12	Results of a 100 node network (Topology 1) for $R_t = 75\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	113
6.13	Results of a 100 node network (Topology 1) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field when there is full aggregation.	113
A.1	Parameters of DASSA for Topology 1, 2, 3 and 4.	118
A.2	Parameters of OSSA for Topology 1, 2, 3 and 4.	118
A.3	Parameters of DASSA and OSSA for Topology 5.	118
A.4	Parameters of DASSA and OSSA for a 200 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.	119
A.5	Parameters of DASSA and OSSA for a 400 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 300m-by-300m field.	119
A.6	Parameters of DASSA and OSSA for $R_t = 60\text{m}$, $R_s = 50\text{m}$ and Topology 1.	119

A.7	Parameters of DASSA and OSSA for $R_t = 75\text{m}$, $R_s = 50\text{m}$ and Topology 1.	119
A.8	Parameters of DASSA for Topology 1 with full aggregation. . . .	119

List of Abbreviations

AFECA	Adaptive Fidelity Energy Conserving Algorithm
ASCENT	Adaptive Self Configuring sEnsor Networks Topologies
ASD	Adaptive Scheduling Depth
CA	Centralized Algorithm
CCP	Coverage Configuration Protocol
DASSA	Distributed Adaptive Sleep Scheduling Algorithm
DRS	Data Reporting group Scheduling
EC	Effective Coverage
EDRS	Enhanced Data Reporting group Scheduling
EOD	Energy Optimization Depth
GAF	Geographic Adaptive Fidelity
GoC	Grade of Coverage
GoC-L	Grade of Coverage Lifetime
GP	Grid Parameter
GPS	Global Positioning System
ILP	Integer Linear Programming
ILPNA	Integer Linear Program with No Aggregation
ILPFA	Integer Linear Program with Full Aggregation
ILPMinConCov	Integer Linear Program finding the MINimum CONnected set satisfying the desired COVernage
ILPSink	Integer Linear Program implemented by the SINK
LT	Loss Threshold
MAC	Medium Access Control
MEMS	Micro Electro-Mechanical Systems
NAS	Not Active Scheduling
NRN	Number of Reporting Nodes
NP-hard	Non-deterministic Polynomial-time HARD

NSD	Number of Selected Descendants
NT	Neighbor Threshold
OGDC	Optimal Geographical Density Control
OSSA	Oblivious Sleep Scheduling Algorithm
PEAS	Probing Environment and Adaptive Sleeping
PHY	Physical
SEER	Simple Energy Efficient Routing
SID	Sorted node IDs
S-MAC	Sensor Medium Access Control
SPIN	Sensor Protocols for Information via Negotiation
WSN	Wireless Sensor Network

Chapter 1

Introduction

Recent advances in micro electro-mechanical systems (MEMS)¹ technology have made it possible to equip sensors used for sensing the environment with small but powerful processors and wireless transceivers with moderate ranges. This emerging *Wireless Sensor Networks* (WSN) technology consists of a large number of sensors deployed across a geographic area to monitor the environment by measuring physical parameters such as temperature, motion, sound, etc. [1].

The most common types of sensor networks are :

- Environmental sensor networks to monitor environmental changes in oceans, forests, etc.
- Military sensor networks to monitor battlefields and detect enemy.
- Public sensor networks to provide security to buildings, malls, to monitor traffic in a city.
- Healthcare sensor networks used in biomedical applications to monitor human body.

¹Micro-Electro-Mechanical Systems (MEMS) is the integration of mechanical elements, sensors, actuators, and electronics on a common silicon substrate through microfabrication technology.

A wireless sensor network consists of a large number of nodes either deterministically or randomly deployed, for instance, from an airplane, to monitor the environment. Sensor nodes communicate with each other by multihopping, i.e., by using other sensor nodes in the network as relay nodes. In most applications, all the sensor nodes are required to send their data to a special node called *base station* or *sink* which links the sensor network to the end user, for instance to an airplane passing above the network area. Sink is assumed to have abundant energy resources, complex processing and high range transmission capabilities together with sufficient memory.

One of the most important issues regarding the design of sensor networks is power consumption since these networks consist of a large number of nodes and are usually deployed in hazardous and remote areas where the replacement of batteries is impossible. The most power consuming operation is data transmission and reception. Control messages and sensing and processing operations also contribute to the energy consumption. To use the limited energy sources efficiently, several approaches have been introduced in the literature.

One common approach is *data aggregation*, in which multiple data packets are combined into smaller sized packets by some processing before transmitting so that the transmission and reception power levels decrease. For instance, in a temperature monitoring sensor network, each node can only forward the average of the temperatures it receives from its neighbors rather than forwarding all the temperature values. The sink can then find the average temperature of the area. A widely used method employing aggregation is *clustering*. In this method, nodes are organized into clusters depending on some predefined method, e.g. random clustering. Each cluster has a cluster head which transmits aggregated data collected from the members of its cluster to the sink.

In cases where data of many individual nodes are requested, or where combining the data requires high computational capability, aggregation is not a solution. Consequently, another approach for efficient energy consumption is *sleep scheduling* or *density control* which allows some of the deployed nodes to sleep and conserve energy. Sleep scheduling controls the number of sensors in the operating mode, which are sensing, receiving and transmitting data, such that some user defined constraints are satisfied. For example, in a military sensor network, the user may want the sensor nodes to forward their data to the sink within a certain time limit or the user may want the event detection probability—the probability that an event in the sensor field will be detected correctly—to be larger than some value. The sleep scheduling mechanism has to evenly deplete the energies of the bottleneck nodes, which are typically the nodes closer to the sink, since all the network traffic has to pass through them to reach the sink, so that the network lasts longer. Activating a small subset of nodes rather than all the nodes will not only save energy, but it will also reduce the network traffic, thus avoiding collision of packets and decreasing the delay of reporting data to the sink.

While ensuring only a subset of nodes to be in the operating mode, the sleep scheduling mechanism must fulfill two requirements: *connectivity* and *coverage*. A sensor network is connected if every sensor node in the network can reach every other node and the sink, possibly via multiple hops. Coverage is defined as the area that can be monitored by the active sensors that can reach the sink. Usually, it is assumed that a sensor node can monitor all the points within a certain range, called the sensing range, around it.

The scope of this research is the development of an energy efficient, distributed sleep scheduling algorithm for wireless sensor networks that can easily be implemented in many kinds of such networks without major modifications.

Sleep scheduling in sensor networks is a blossoming area which introduces many benefits to the network in terms of energy efficiency and network traffic density.

1.2 Motivation

Sleep scheduling is a prevailing way of reducing energy consumption in a sensor network. The challenge when some of the nodes do not operate in a sensor network is to ensure the connectivity of the operating nodes and at the same time to provide some minimum coverage while trying to keep the number of active sensors to a minimum. In the lack of global knowledge of the network and location information together with energy scarcity, the problem becomes even harder. A solution which can be applied to general network topologies has to be found and implemented.

Although there are numerous work in this area, there is still need for a protocol which can schedule node operation in an efficient and adaptive manner without requiring location information, global network knowledge and the use of excessive control messages. A location information gathering mechanism such as the global positioning system (GPS) would be very expensive to employ in a typical sensor network consisting of a very large number of sensors [2, 3]. Alternatively, only a certain fraction of nodes may have GPS and other nodes may try to find their locations using the information provided by these nodes. This is also costly due to the message exchange load and the results are not always as precise as desired. Therefore, an algorithm which does not require any location information would be of value.

Another important issue is that in some applications, it might be acceptable to achieve a coverage ratio less than a hundred percent, i.e., partial coverage. This fact can be exploited to extend network lifetime by the use of sleep scheduling. Instead of operating for one week with a hundred percentage coverage, it

might be better to operate for three weeks with a ninety percentage coverage in a temperature monitoring sensor network where the temperature at each individual point in the area is not so crucial. There is little work exploring this tradeoff in the literature. We address the lack of such research and devise an energy efficient node scheduling algorithm called the *Distributed Adaptive Sleep Scheduling Algorithm* (DASSA).

1.3 Contribution

The node scheduling algorithm DASSA is simple to implement, requires no location information, has little message overhead and extends network lifetime by keeping the coverage percentage of the network above a user defined threshold rather than one hundred percent. This approach is new for sleep scheduling algorithms.

DASSA focuses on the nodes closer to the sink since all the network traffic has to pass through these nodes. Also, when all the nodes in the one-hop-neighborhood of the sink die, none of the remaining nodes can reach the sink anymore causing the network to die. Therefore, DASSA carefully schedules the activity of the nodes close to the sink depending on their residual energies and number of neighbors. In DASSA, the sink is responsible for scheduling the activities of the nodes close to itself. Note that the sink has plentiful amount of communication and computational resources. This is a novel approach in scheduling nodes near the sink.

Besides its simplicity, DASSA is fully distributed and minimizes the assumptions about the network. It is assumed that a multihop network structure exists, i.e., every node cannot reach the sink by a single hop and no location information is available. In DASSA, the nodes only have to perform simple computations such as taking the maximum of a certain set of numbers. The algorithm is scalable

due to its simplicity and independency from impractical assumptions. This is an important issue to be addressed since sensor networks consist of a very large number of sensors.

DASSA can reach up to 90% of the optimum results when tuned properly. In the thesis, DASSA is compared with a second algorithm called OSSA and an existing algorithm in the literature. DASSA outperforms the algorithm in the literature and DASSA outperforms OSSA in all the cases considered. As the desired coverage level decreases, DASSA achieves very close results to the optimum. DASSA is a very flexible algorithm and with the proper tuning of its parameters, it can achieve very high performance for many network sizes.

1.4 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 presents an overview of existing sleep scheduling algorithms in the literature. In Chapter 3, the minimum number of sensors that have to be deployed in order to provide a certain level of coverage with high probability is found and the minimum number of connected nodes among these sensors which provide sufficient coverage is calculated using an integer linear programming approach. The chapter concludes with some numerical results. Chapter 4 includes our centralized linear programming approach to the sleep scheduling problem for finding the optimum sleep schedules depending on the desired coverage level. Next, our proposed distributed algorithm DASSA is described in Chapter 5. Comprehensive simulation results of DASSA, a second algorithm OSSA and an existing work together with the optimum results are provided in Chapter 6. Finally, the possibilities for future work in the area together with the conclusion of the thesis is given in Chapter 7.

Chapter 2

Sleep Scheduling in Wireless Sensor Networks

The major design objective for wireless sensor network applications is to minimize the energy consumption in order to maximize network lifetime. Among various approaches for efficient use of energy including clustering [4, 5] and data aggregation [6, 7], sleep scheduling is the most commonly used one.

To improve a sensor network's reliability and extend its longevity, sensor networks are deployed with high densities (up to 20 nodes/m³ [8]). However, if all sensor nodes in such a dense deployment scenario operate at the same time, energy will be consumed excessively. Also, packet collisions will increase as a result of the large number of packets being forwarded in the network. In addition, most of the data forwarded in the network will be redundant since when node density is high, sensing regions of the nodes will overlap and the data of adjacent sensor nodes will be highly correlated. In summary, sleep scheduling reduces both energy consumption and network traffic by avoiding the transmission of redundant data.

A practical sleep scheduling algorithm should both choose the minimum number of active nodes and satisfy user defined constraints. The non-sleeping nodes must be chosen so that they are connected to the sink and they provide some minimum coverage of the network field. User defined constraints may vary depending on the application type. For instance, the user may want the network to be connected and provide some minimum coverage for as long as possible or the user may want the network to be connected and provide full coverage of the network field while ensuring some minimum delay in gathering data.

Besides, the sleep scheduling algorithm must be simple, distributed and localized. It must be applicable to many kinds of networks with minor modifications. Due to the distributed nature of sensor networks, it must be a distributed approach and it should only use local information since each node has a limited transmission range. It is also desirable not to require any location information since it is very costly for a sensor network. Although sleep scheduling is not a new approach to extending network lifetime, there is almost no work satisfying all these requirements simultaneously.

The chapter continues with introducing the basic concepts that are widely used in sensor network models. After this brief introduction, we discuss the algorithms devised in the literature. The common requirements of most sensor network applications are to provide some level of coverage and to provide connectivity of the sensor nodes. These requirements should always be considered together when designing a sensor network. However, many studies in the literature consider these two issues separately. Accordingly, first, sleep scheduling algorithms which consider only coverage or only connectivity are presented and then algorithms which consider coverage together with connectivity are discussed. Then, a different class of sleep scheduling algorithms which try to maintain a certain number of active sensors at every round without considering coverage is presented. A brief discussion of routing protocols which can be used in conjunction with the sleep

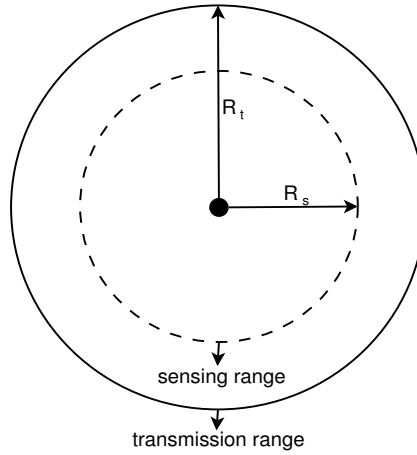


Figure 2.1: Transmission and sensing ranges of a sensor node. R_t is the transmission range and R_s is the sensing range.

scheduling algorithms is provided next. Finally, studies which exploit partial coverage and maintain a coverage level less than one hundred percent for the sake of longer network lifetime are presented.

2.1 Basic Concepts

A sensor network consists of a number of nodes, which are limited in terms of energy, memory and processor speed, and a sink which is located at the origin point $(0, 0)$ of the field and has higher computation capability than other nodes and abundant energy and memory resources. All sensor nodes are responsible for sending their data to the sink. After collecting all data, the sink is responsible for the rest of the process. For example, it might transmit data to a data gathering point in a distant location using satellite communications.

Sensor nodes sense the environment by using sensors such as ultrasonic sensors, temperature sensors, infrared sensors and etc. Each sensor node is associated with a transmission and a sensing range, R_t and R_s , as shown in Figure 2.1. It is assumed that a sensor node can detect every event occurring within a distance less than or equal to R_s with probability 1 and cannot sense any event

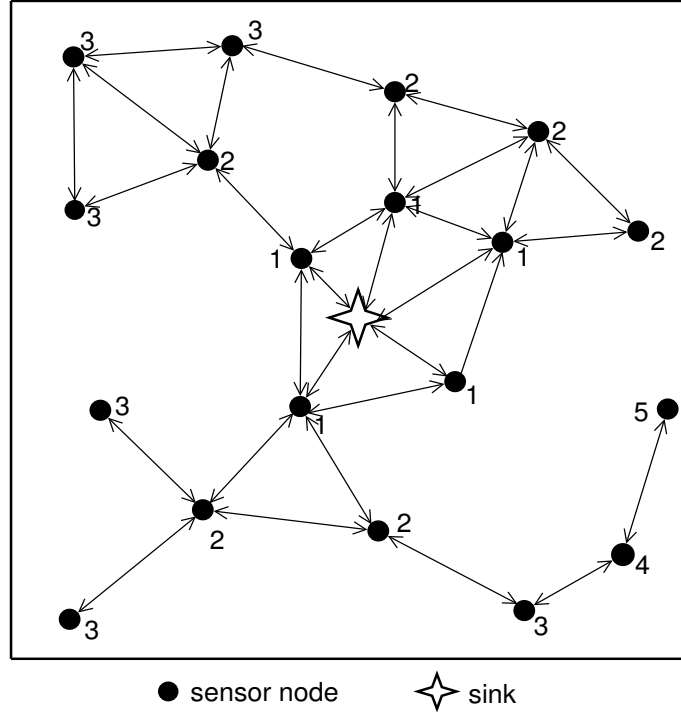


Figure 2.2: Tier numbers in a sample network. Nodes at the edges of a two sided arrow are in the transmission ranges of each other.

outside this distance. Similarly, two nodes can communicate with each other if the distance between them is less than or equal to R_t .

Hop count and *tier number* are commonly used terms in the literature and throughout this thesis. We use the terms hop count and tier number interchangeably. Nodes which can reach the sink in at least one hop are in tier 1, nodes which can reach the sink in at least two hops are in tier 2 and so on. Figure 2.2 shows the tier numbers of nodes in a sample network. Note that, hop counts are determined by the transmission ranges of the sensor nodes, i.e., if a node is in the transmission range of another node, then these two nodes have one hop distance to each other.

Sensor nodes send their data to the sink depending on a data reporting model which can be time driven, event driven, query driven or a combination of these models. In the time driven model, sensor nodes periodically sense the environment and send their data towards the sink, whereas in the event driven model,

sensor nodes are only responsible for sending their data when a specific event occurs. In query driven data reporting, nodes send data only when they receive a query from the sink or another node in the network. It is assumed that a time driven model is used for data reporting in this thesis.

After this brief introduction, we now proceed with the related work in the subject.

2.2 Sleep Scheduling Considering Only Coverage

In a general sense, coverage can be defined as the area in the network field which can be sensed by the sensor nodes, i.e., an area for which all the points in the area are in the sensing range of at least one active sensor node. The coverage problem addresses on finding the minimum set of sensors which can cover the same area as the deployed sensors, thus avoiding redundant data transmissions in the network. The sensing region of a node is generally assumed to be a disk with radius equal to the sensing range of the sensor node. Other sensing models can also be assumed [9].

In [10], two types of sensor nodes with different costs and sensing ranges and a grid based network structure are assumed. One node type has a larger sensing range than the other which is on the other hand cheaper. They find the minimum cost placement of sensor nodes while ensuring that all grid points are covered adequately with a linear programming approach. In addition, the problem of determining the grid points to locate the sensor nodes such that the grid positions of targets can be uniquely identified from the subset of sensors that detect the targets is analyzed in the paper. Similarly, in [11], a linear programming approach is used to determine the minimum number of sensors

which can cover a certain area. After a certain number of nodes are deployed to a field according to a uniform distribution, the minimum set of sensor nodes to cover the same area as the original network is found. Additionally, locating several disjoint sets of sensor nodes which can cover the area is discussed. This way, each subset can be scheduled to be operational during a different time slice for providing a balanced operation which will increase the utilization of resources. Related to this approach, in [12], a heuristic which finds mutually exclusive sets of sensor nodes where each set entirely covers the network field is proposed. The algorithm ensures that only one set is active at a time. Using only a subset of nodes at each time saves energy while maintaining the coverage.

In [13], it is determined whether the network area is k – covered, in the sense that every point in the area is covered by at least k sensors. A sensor is k perimeter covered if all points in the perimeter of its sensing area is covered by at least k sensors other than itself. It is proven geometrically that, if each sensor node is k perimeter covered, the whole network is k covered. Also, the model can be extended to non-disk coverage models as long as the sensing regions can be precisely defined. Using a geometrical approach in parallel to [13], [14] proposes a backoff based node scheduling scheme in which nodes which are redundant in terms of coverage are turned off. The idea is that if the whole sensing area of a node can be covered by the its neighbors, then the node can be turned off. However, if all the nodes decide to turn off simultaneously, then blind points, which are areas not covered by any active node, will occur. For example, in Figure 2.3, if nodes 1 and 5, whose sensing areas are totally covered by their neighbors, decide to turn off at the same time, a blind point will occur. Therefore, a random backoff procedure is presented in which nodes that decide to turn off broadcast a message after a random backoff time. A node hearing this broadcast message checks whether its coverage area is still covered by its awake neighbors and acts accordingly. This way, blind points are avoided. It is claimed

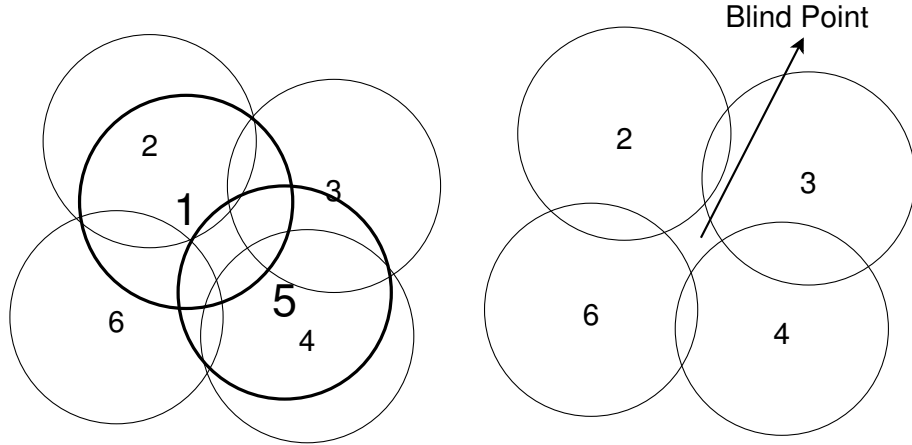


Figure 2.3: The blind point problem when nodes turn off simultaneously.

that, if a node can determine the angle of its neighbors with respect to the location of itself, then the node scheduling algorithm may be used without location information. However, the results will be worse in the sense that more nodes will be awake than necessary and assuming precise knowledge of angle information is not practical. Also, for this algorithm to work properly, communication ranges of nodes should be equal to twice the sensing range in which case full coverage guarantees connectivity. Otherwise, the set of nodes selected to be active cannot be guaranteed to be connected to the sink. [15] extends the work in [14] by decreasing the number of active sensors even further and therefore increasing the network lifetime.

A different coverage definition is introduced in [16] where Voronoi diagram and Delaunay triangulation techniques are used to compute the *maximal breach path* and *maximum support path*. Maximal breach path is a path in the sensor field such that the distance from any point on this path to the closest sensor is maximized. This gives the worst case coverage. The maximum support path is the path in the sensor field such that the distance from any point on this path to the closest sensor is minimized. This corresponds to the best case coverage. If new sensors can be deployed such that the weight, which depends on the distance of sensor nodes from each point on the path, of the maximal breach path is decreased, then worst case coverage will be improved. The same argument is

valid for the maximum support path. It is also claimed that worst case coverage and best case coverage have an asymptotic behavior and increasing the number of randomly deployed sensors beyond a value does not make any significant improvements.

Note that, unless otherwise stated, all the above references assume the disk model for coverage and they are centralized. All of them require global knowledge about the network, i.e., the number of nodes, the precise locations of nodes and the neighbors of the nodes. Also, the algorithms used for the solutions are generally complex and require long run times. Nevertheless, the analysis described here are usually done before deploying the network to estimate its behavior. For example by the work in [11], one can have an idea about how many nodes will be redundant in the average. None of the above coverage maintenance studies consider connectivity. They assume that the set of nodes which provides the required coverage somehow can reach each other.

2.3 Sleep Scheduling Considering Only Connectivity

A network is connected if all sensor nodes can reach the sink, which also means that every node can reach each other, possibly by multiple hops. In this section, we present sleep scheduling algorithms which only consider connectivity regardless of coverage.

In [17], a *Geographic Adaptive Fidelity* (GAF) algorithm is introduced which requires location information of the nodes. The algorithm divides the network area into virtual grids. Nodes compute the grid which they belong to from their location information. Grids are arranged so that any node in a grid can reach all the nodes in the adjacent grids. This puts a bound on the grid size (less than

$R_t/\sqrt{5}$) and this bound is independent from the node density. Nodes switch between *sleeping*, *discovery* and *active* states, with the requirement that one node in each grid stays awake in order to ensure connectivity. In the discovery state, nodes exchange discovery messages to find other nodes within the same grid.

Similar to GAF, Span [18] aims to increase system lifetime by sleeping the redundant nodes without making a substantial effect in connectivity. Span adaptively elects coordinator nodes which stay awake continuously while other nodes remain in sleep mode and periodically check whether they should become a coordinator. Obviously, the coordinator nodes spend much more energy than other nodes. So, the coordinator role is rotated as time goes on. Nodes become coordinators if two of its neighbors cannot reach each other directly or by one or two coordinators. Coordinators refrain from their role if their neighbors do not require them or if a certain time has elapsed. Nodes with higher energies become coordinators with higher probabilities and this provides better utilization of resources. Span uses geographic routing in which coordinators forward data to the coordinator which is closest to the data gathering point. Geographic routing uses location information for finding the shortest path. Although it is mentioned that Span can also work without location information, it is not clear whether a routing protocol not using location information can still function with only the coordinator nodes Span chooses.

In ASCENT [19], nodes can be in one of the four states; *test*, *passive*, *active* or *sleep*. ASCENT tries to keep the number of active neighbors of each node above a user specified *neighbor threshold* (NT) value and to keep the data loss level below a user specified *loss threshold* (LT). LT is measured at the sink from the number and quality of the received data. In the test state, nodes test the environment for satisfying the NT and LT requirements and change their states accordingly. They move into the active state, or into the passive state. In the passive state,

nodes only listen to the environment and if they decide that there is no need for them to be active, they finally move to the sleep state in which no energy is consumed. A node in the active state forwards data until it dies. ASCENT does not consider the residual energies of the nodes while determining the states and this makes the algorithm unbalanced in terms of energy consumption at each round. ASCENT does not use location information.

In AFECA [20], each node keeps a count of the number of its neighbors and has a randomized sleep time proportional to this number. This results in more nodes sleeping as the density of the network increases. The parameters of the algorithm are chosen as to increase the probability of connectivity. AFECA does not require location information.

2.4 Sleep Scheduling Considering Both Coverage and Connectivity

We will now turn our focus to the work considering both connectivity and coverage at the same time. Note that finding the minimum set of sensors which cover the entire deployment area and are connected is an NP-hard problem [21].

A theoretical analysis of the connectivity with coverage problem is given in [22] in which a grid based network is considered where each node can fail probabilistically. A sufficient and necessary condition for connectivity with full coverage of the deployed area is given as a function of the sensing range, which is taken to be equal to the transmission range, number of nodes and probability of failure. Additionally, it is shown that connectivity does not imply coverage. The work presented in this paper assumes fixed locations for the nodes and the randomness is due to node failure. This can be extended to the situation where the randomness is due to node deployment as in [23].

There are a number of distributed algorithms for providing connectivity and coverage at the same time. PEAS [24] is a distributed, probing based sleep scheduling algorithm. Nodes wake up and probe their local neighborhood to find out whether a working node exists within a certain probing range. If there is no working node in its probing range, a node will start to operate, otherwise it will sleep. Thus, the distance between two operating nodes is at most the probing range. The algorithm requires nodes to adjust their power levels in order to check the neighborhood only in the probing range which is not very practical. If power level adjustment is not available, the probing range should be chosen equal to the transmission range or location information would be required. The algorithm does not guarantee complete coverage and is shown to form an asymptotically connected network if the transmission and probing ranges satisfy some requirements. To sum, the algorithm guarantees neither coverage nor connectivity.

In CCP [25] and OGDC [26], it is proven that if the radio range is at least twice the sensing range, complete coverage of a network field guarantees connectivity. CCP schedules nodes to sleep depending on the coverage degrees of the intersection points on a node's sensing disk with its neighbors sensing disks. It is proven that if all intersection points between any sensor node and any other sensor node and all intersection points between any sensor node and the field boundaries are k -covered, then the field is k -covered. They propose an eligibility algorithm which uses location information to determine whether a node can sleep when the transmission range is larger than twice the sensing range. A node decides to sleep if every location within its coverage is already k -covered by other active nodes. Randomized backoff times are used to avoid shutting down nodes at the same time. If the communication range is smaller than twice the sensing range, the eligibility algorithm is combined with SPAN [18] to form a connected covering set. CCP depends on location information.

In OGDC [26], it is proven that in order to cover a large region with the minimum overlap of the selected sensors, at least one pair of disks must overlap and the crossing points of these disks must be covered where disk refers to a circular sensing area. First, the optimum location of a third disk to cover the intersection points of two other disks with minimum overlapping sensing areas is found. Then, OGDC tries to schedule nodes which are close to these optimum locations to be operating. Nodes exchange messages containing their location and their computation of the direction at which a working node should be located at. The algorithm runs in rounds and at the beginning of each round, a set of one or more starting nodes are randomly selected as working nodes. Nodes use a backoff procedure to avoid collisions and simultaneous turning offs. A node decides to operate if it is the closest node to the optimum location to cover the intersection points of two working nodes. The backoff delay is directly related to the distance of a node from this optimum location; as the node gets closer, the delay decreases. If the node is in the optimum location, then the delay is zero, so that the node does not sleep. In contrary, a node decides to sleep if its neighbors cover its sensing area. Like CCP, OGDC requires location knowledge and cannot be adapted to operate without this knowledge.

Also, [21] addresses the problem of connected coverage. A centralized approach is prepared for finding a subset of nodes which ensure both coverage and connectivity. The algorithm yields a solution which is within $O(\log n)$ factor of the optimum solution where n is the number of nodes in the network. Additionally, a distributed version of the centralized algorithm is proposed. Both the centralized and the distributed algorithm require location information. Also, the distributed algorithm requires complex operations and reliable broadcast of messages to all the nodes within $2r$ hops of each node, where r is defined as the maximum communication distance between any two sensors whose sensing regions intersect. Additionally, the value of r has to be calculated, which requires a complex probabilistic analysis. Thus, some r values that can be used in place

of the exact r value are proposed. The distributed algorithm is heuristic based and does not guarantee the $O(\log n)$ factor.

The references mentioned here are the most popular ones in the literature regarding both connectivity and coverage. Further information about protocols in this context can be found in a survey about energy efficient protocols [27].

2.5 Other Work on Sleep Scheduling

In addition to the references described in the previous sections, there are also sleep scheduling algorithms in which the aim is to keep the total number of active nodes in a network at a certain value by using feedback messages from the sink.

In [28], an algorithm which tries to maintain the total number of active sensor nodes at a certain value throughout the network lifetime is proposed. Each sensor node independently decides whether to sleep or to be active using the Gur game strategy [29]. Each node has a finite, discrete time $2N$ state automaton consisting of N negative and N positive consecutive states. Sensors change state depending on a probability value sent from the sink. This probability depends on the total number of active sensors at a given time and the desired number of sensors. It is demonstrated that $N = 3$ provides good results. All sensors, including the sleeping ones, listen for the information from the sink. This way, the total number of sensors in the network converges to a constant value in the case where nodes do not die. In the more realistic case where nodes die, the algorithm does not converge to the desired value most of the time and makes big fluctuations. It is assumed that the network has a star topology, i.e., all the nodes can reach the sink in a single hop. Also, due to the nature of the Gur automata, the number of active sensors is limited to the half of the total number of sensors, i.e., the total number of active sensors can be held at a maximum of half of the number of total sensors by this method.

Similar to [28], [30] uses feedback from the sink. [30] provides an improvement to the work in [28] by using the mean expected lifetime of sensor nodes assuming exponentially distributed lifetimes. In addition, a new algorithm for keeping the number of active sensors at a given value is proposed. In this algorithm, each node has a certain number of states, 3 states in the paper, and transmits data according to its current state. A probability is assigned to each state, and sensors transmit or do not transmit data depending on this probability. Sensors make state transitions depending on the information they gather from the sink. It is assumed that a random access communication protocol will be used which uses acknowledgement packets. Therefore, after receiving a packet from a node, the sink will send a packet depending on the number of active nodes, added to the acknowledgement signal. The node makes a state transition depending on this information; if the number of active nodes is less than the desired value, it will transit to a state where it will make a transmission with higher probability. With this acknowledgement scheme, sensors modify their states individually. It is shown that this scheme outperforms the algorithm in [28]. This algorithm also assumes a star topology for the network. The probabilities at each state affect the algorithm performance and the initial probabilities affect the convergence time to the desired total number. [31] extends the work in [30] by making a theoretical analysis in which the effect of state probabilities on the mean and variance of the total number of nodes which are active, named as QoS, are analyzed. The choice of these probabilities creates a tradeoff between the diversity, which is the equality of participation among sensors, and the variance of the QoS. Finally in [32], an algorithm in which the participation among nodes is more balanced and which does not use knowledge of the total number of available sensors is proposed.

[30, 31, 32] are limited to communication protocols where acknowledgements are used and together with [28] they assume that each node can reach the sink in a single hop. It is not clear whether the algorithms may be used in a multihop

scenario and how the sink will send feedback to each node in such a case. Also, [28, 30, 31, 32] do not address the importance of individual sensor nodes in terms of coverage. Actually, they do not take coverage into account and only try to maintain the total number of sensors at a certain value. The performance of these algorithms gets worse as nodes die since neither of them considers the residual energies of the nodes. Also, they suffer from fluctuations, which in some cases make large differences in the total number of active nodes and the desired number of active nodes.

In contrast to references described previously in this section, [33, 34] tries to keep the total number of active sensors at a certain value by considering the effects of the MAC and PHY layers. The same problems mentioned in the previous paragraph remain in these works.

2.6 Routing Protocols

There are many routing protocols proposed for sensor networks in the literature. Since the routing protocol used is not in the direct scope of this research and after deciding the set of nodes which are connected and provide sufficient coverage, many routing algorithms can be an alternative, we refer the reader to [35, 36]. These references include a detailed description and comparison of many routing protocols.

We consider SEER [37] separately from other protocols since it uses a similar routing structure with ours. SEER is an *event-driven* (nodes transmit data when they sense an event), *source initiated* (nodes do not flood an interest) routing protocol. The algorithm starts with the *network setup and neighbor discovery* phase. In this step, every node discovers its hop count with respect to the sink and its neighbors and their remaining energy levels by broadcast messages. First, the sink broadcasts a message and initiates this step. Then each node broadcasts

a message containing its hop count and remaining energy. At the next step, which is the *transmitting new data* step, nodes forward data to their neighbor with the smallest hop count and highest remaining energy. Before forwarding a message, the remaining energy level entry for the neighbor is reduced by a certain amount. Next, in the *forwarding data* phase of the algorithm, nodes receiving a message update the energy entries in their list for the transmitting neighbor. These energy values may not always be accurate since a node can be used by more than one neighbor in forwarding data. Therefore, nodes broadcast energy update messages in the *energy update* step, when their energies fall below a certain threshold.

This process is repeated from the beginning, i.e., from the network setup and neighbor discovery phase, periodically to take into account node failures and to maintain correct energy values, i.e., the *network maintenance* step of the algorithm. SEER does not require location information and complex operations at sensor nodes. SEER is shown to outperform the two most popular protocols SPIN [38] and Directed Diffusion [39].

Finally, a different aspect of the routing problem is presented in [40]. This work formulates the maximum lifetime routing problem as a multi commodity flow integer linear program and proposes heuristic algorithms. [41] extends the analysis of maximum lifetime routing problem to the latency domain. Basically, they propose two linear programming approaches; the first one tries to minimize the latency while minimizing energy consumption is the secondary objective and the second one tries to minimize the energy consumption while minimizing the latency is the secondary objective. It is shown that the second choice yields comparable results in terms of lifetime and achieves a lower latency. However, the first choice balances the energy consumption among nodes far better than the second choice. Note that, this algorithm is centralized and requires global network information.

2.7 Partial Coverage

In some applications, covering 100% of the sensor field continuously may not be very critical. Instead, network lifetime can be prolonged if we keep the coverage level below 100%. For example, in a temperature or humidity monitoring sensor network, it may be sufficient to cover 90% of the total sensor field in order to increase network lifetime. This is named as partial coverage. Also, in many scenarios, the sensed data is highly correlated. Thus, covering 90% of the network may be sufficient to obtain a global knowledge about the sensor field.

This concept is fairly new in sleep scheduling area and is not extensively studied in the literature. [42] makes the definition of partial coverage and studies the connected coverage problem with a given coverage guarantee. The theoretical bounds for the number of active nodes to satisfy certain coverage while being connected with a constraint on the transmission range are presented. Also a heuristic algorithm is provided to show that the network lifetime increases as the desired coverage percentage decreases.

pCover [43] proposes a distributed algorithm for the partial coverage problem. According to the algorithm, all the nodes are in one of the four states: *probing*, *sleep*, *awake* and *readyoff*. Every node calculates the percentage of its sensing area that is covered by its awake neighbors, which is called the *local coverage*. A node in the probing state turns on if its local coverage is lower than a threshold, called the *on threshold*, whereas a node in the readyoff state turns off if its local coverage becomes higher than a threshold, called the *off threshold*. By varying the on and off thresholds, various coverage levels are obtained in the network. pCover assumes all the nodes are aware of their locations and their neighbors locations within a distance of $2R_s$, where R_s is the sensing range of a node. It is also assumed in the paper that the communication range is larger than at least twice the sensing range. The second assumption makes it easier to

provide connectivity of the operating nodes, since the set of nodes selected for sufficient coverage will be connected with high probability when $R_t \geq 2R_s$. For the energy consumption model, it is assumed that a node can function for 1000 minutes regardless of the number of its descendants which is also an impractical assumption.

Finally, [44] finds an upper bound of lifetime when only a portion of the network field is to be covered assuming that the deployed nodes form a homogenous Poisson point process. The network lifetime is defined as the time until the coverage ratio drops below the desired coverage. It is assumed that the transmission range is larger than twice the sensing range.

A distributed and more practical algorithm which does not assume location information and takes $R_t = R_s$ is introduced in [45, 46] which exploits partial coverage in a different objective than ours as to provide full coverage of the network within some delay, such that the coverage percentage at each round is lower than 100%. In other words, the algorithm tries to find disjoint sets of sensors at each round of operation such that the coverage at each round is above a user defined value and all of the sensors are able to report their data within a certain delay. An important point is that this set of sensors has to be connected. In order to ensure connectivity, the disjoint property may be violated. Note that, in our work, we try to maximize the number of rounds which we can provide a coverage level above a user defined threshold, whereas [46] tries to cover the entire area within a certain number of rounds and at each round it tries to keep the coverage level above a user defined threshold.

[46] contains and extends the work in [45] and proposes an implementable algorithm. From now on, we refer to the work in [46] as *Data Reporting group Scheduling* DRS.

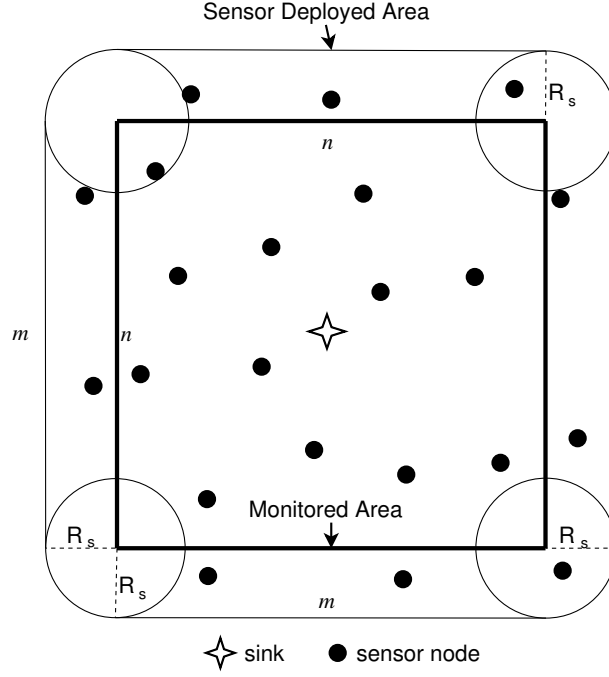


Figure 2.4: Sensor field and monitored field in DRS. R_s is the sensing range of a sensor node.

As mentioned before, DRS tries to keep the coverage above a certain user defined coverage level which we call *Grade of Coverage* (GoC) throughout the thesis, where $0 \leq \text{GoC} \leq 1$ and GoC equals 1 corresponds to 100% coverage of the field, GoC equals 0.8 corresponds to 80% coverage of the field and so on. It is assumed in DRS that the sensors are deployed over a larger area then the monitored area which has a distance equal to the sensing range from the deployed area as shown in Figure 2.4.

Assuming that the sensor nodes are uniformly distributed in the sensor deployed area, the minimum number of sensors k , that should be deployed in order to cover a certain percentage of the monitored area can be found by a probabilistic analysis. Due to the structure assumed for the monitored and deployed areas, the probability that a point is not covered by a sensor node is given as,

$$P = \frac{F - S}{F} \quad (2.1)$$

where S is a circular disk centered at the point and has radius equal to the sensing range of a node, i.e., $S = \pi R_s^2$ and F is the deployed area, i.e.,

$F = n^2 + 4nR_s + \pi R_s^2$. This result comes from the fact that a point is not covered by a sensor node if the sensor node is at a distance greater than the sensing range of a node.

The probability that a point is not covered by k sensors is equal to P^k . Therefore, the probability that a point is covered by any of the k sensors is $1 - P^k$ which gives the overall coverage of the area.

$$\text{GoC} = 1 - P^k \quad (2.2)$$

The number of sensors k required to provide GoC can be found from Eq. (2.2) and Eq. (2.1) in terms of the field parameters shown in Figure 2.4.

$$k = \left\lceil \frac{\log(1 - \text{GoC})}{\log\left(\frac{n^2 + 4nR_s}{n^2 + 4nR_s + \pi R_s^2}\right)} \right\rceil \quad (2.3)$$

After this analysis, a randomization technique for selecting k reporting sensors in each round is proposed in DRS. A *data reporting group* (RS_{s_i}) is a set of sensors which are selected to operate in a given round. There are $\delta = \left\lfloor \frac{|N|-1}{k} \right\rfloor$ data reporting groups, where N is the total number of sensor nodes including the sink. Each node decides to operate at only one data reporting group with a probability of $1/\delta$ so that the expected value of the number of nodes at each group is k .

Reporting cycle is the periodicity of a node to report its data. Sensors belonging to a reporting group send their sensed data only in the round corresponding to the group and wait until the next cycle. This way, all the area will be covered within the cycle.

Nodes at each data reporting group have to be connected in order to send their data to the sink. The estimate number of additional sensor nodes, named as $\hat{k}$, which will be required for connectivity, is computed in a probabilistic manner. As we mentioned before, normally sensors are only active in the round

corresponding to their reporting groups. However, additional sensors from other reporting groups have to be active in order to provide connectivity to the sensors in the current reporting group. In Figures 2.5 and 2.6, there exists two data reporting groups: square nodes and circle nodes. In Figure 2.5, square nodes are transmitting and in Figure 2.6 circle nodes are transmitting. We can observe that, some of the circle nodes are active when square nodes are transmitting in order to provide connectivity for square nodes and vice versa. Note that the sink is located at $(0, 0)$ location.

The paper also proposes a distributed algorithm to connect the sensors in each data reporting group. Each sensor node finds an upstream sensor as a next hop node to reach the data gathering point which belongs to either the same reporting group with the sensor or has the shortest hop distance to a sensor with the same reporting group or to the sink. In this algorithm, nodes use setup messages which contain the information about the node which is closest to the same reporting group or the sink.

The algorithm is initiated with a setup message from the sink. The setup message is denoted by S and is given as $S = \{RS_{s_j}, (c_1, O_1), (c_1, O_1), \dots, (c_l, O_l)\}$, where $1 \leq l \leq \delta$, s_j is the sender, O_l is the origin that resets c_l , which is the hop counter from O_l to the receiver of this message, and RS_{s_j} is the reporting group of node j . c_l is reset to 1 or -1 depending on RS_{s_j} of the receiver s_j , and it decreases or increases by one until it is reset. Also, each sensor keeps a record which includes the current best candidate to reach the closest neighbor with all reporting sequences. This record is denoted as R_{s_i} , which is modified when S messages are received, and is given as $R_{s_i} = \{(c_1, O_1, s_1), (c_2, O_2, s_2), \dots, (c_l, O_l, s_j)\}$ where s_j is the best candidate to reach the reporting sequence l and c_l is the hop distance to O_l . For example, if a node belongs to data reporting group 2, it will forwards its data to sensor node s_2 . Setup messages S are based on R_{s_i} and are broadcasted by each node. The detailed description of the algorithm is quite

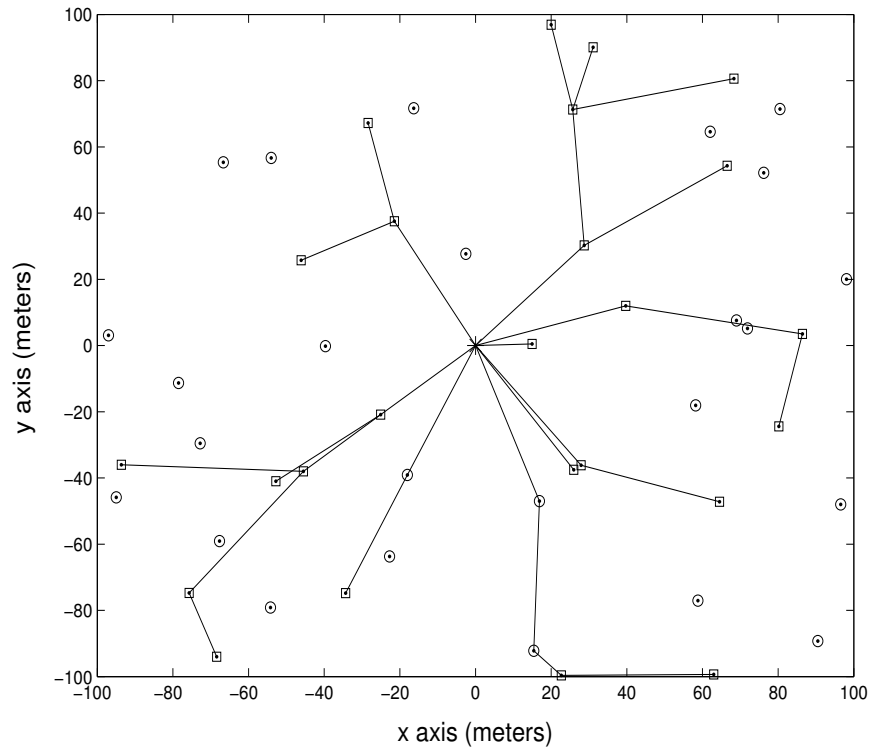


Figure 2.5: Additional nodes required for connectivity in DRS I. Square nodes are transmitting. Circles are only active if they are required for connectivity.

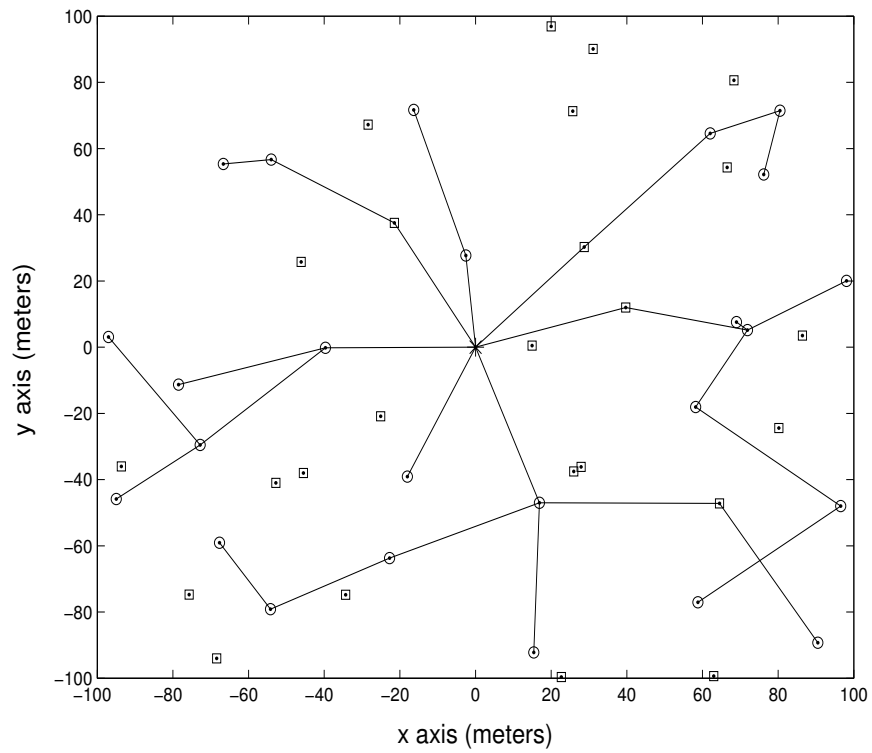


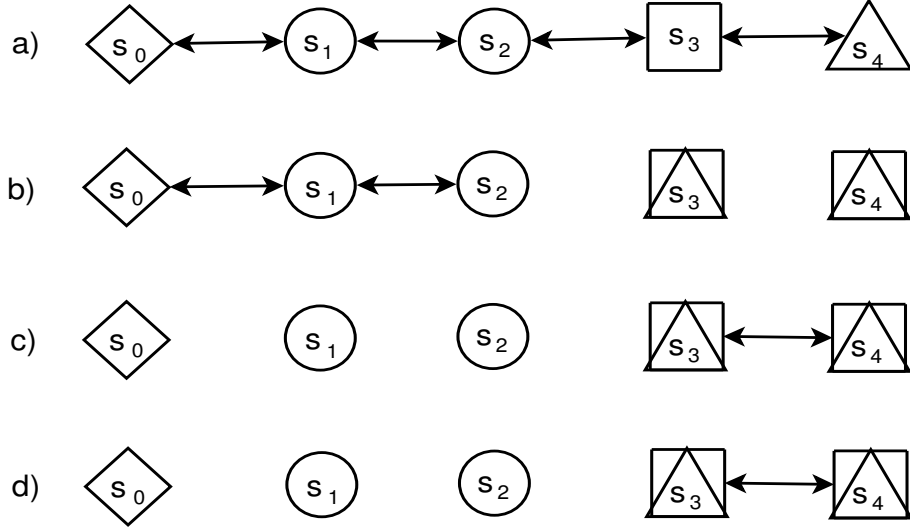
Figure 2.6: Additional nodes required for connectivity in DRS II. Circle nodes are transmitting. Squares are only active if they are required for connectivity.

Sensor Type	RS
	100
	010
	001
	011

Figure 2.7: Reporting group of each node in Figure 2.8. A cycle consists of three rounds and a one in the RS column represents the round at which a node will transmit, e.g., circle nodes will transmit at the first round at each cycle and will not transmit at other rounds.

long to fit in this section and given in Figure 10 in [46]. As mentioned earlier, some nodes from other reporting groups have to be active in order to provide connectivity among all nodes belonging to a certain reporting group. Due to this, some nodes may belong to more than one reporting group.

We have noticed that the proposed algorithm causes routing loops which causes some of the nodes to be unconnected from the network. Thus, the initial aim of all the sensor nodes sending their data within a certain time limit cannot be achieved. The problem occurs due to the post-broadcast update case, in which nodes update their R_{s_i} 's after broadcasting their S . Figure 2.8, which is in the same structure as the scheme used in DRS, shows one possible loop. Figure 2.7 shows the reporting groups of the nodes in Figure 2.8. In this case, sensor s_4 sets sensor s_3 as its next hop and broadcasts a message. From this message, node s_3 sets s_4 as its next hop since it assumes that it has a shorter hop distance to the sink which causes a loop. In order to eliminate this problem, we eliminate the post broadcast messages to avoid routing loops so that after a node determines its next hop, it does not change its decision any more. We call the algorithm with the routing loops eliminated the *Enhanced Data Reporting group Scheduling* EDRS. Also, it is not mentioned clearly what the algorithm does when nodes start to die. Therefore, we assume that the algorithm does not take any action when nodes die when evaluating the performance of it.



Relay	$s_1(RS_{s_1} = 100)$	$s_1(RS_{s_2} = 100)$
Seq.	$S = \{(-1, s_1), (1, s_0), (1, s_0)\}$	$S = \{(-2, s_1), (2, s_0), (2, s_0)\}$
RS_{s_2}	$\{(-1, s_1, s_1), (1, s_0, s_1), (1, s_0, s_1)\}$	—
RS_{s_3}	—	$\{(-2, s_1, s_2), (2, s_0, s_2), (2, s_0, s_2)\}$
RS_{s_4}	—	—
Relay	$s_3(RS_{s_3} = 010)$	$s_4(RS_{s_2} = 001)$
Seq.	$S = \{(1, s_3), (-1, s_3), (3, s_0)\}$	$S = \{(2, s_3), (1, s_4), (-1, s_4)\}$
RS_{s_2}	$\{(-1, s_1, s_1), (-1, s_3, s_3), (1, s_0, s_1)\}$	—
RS_{s_3}	—	$\{(-2, s_1, s_2), (1, s_4, s_4), (-1, s_4, s_4)\}$
RS_{s_4}	$\{(1, s_3, s_3), (-1, s_3, s_3), (3, s_0, s_3)\}$	—

Figure 2.8: The routing loop problem in DRS. s_0 is the sink. a) Shows which nodes are in the transmission ranges of each other. b) The first round in which circle nodes belonging to $RS = 100$ are transmitting. c) The second round in which square nodes belonging to $RS = 010$ are transmitting. d) The last round in which triangular nodes are transmitting which have $RS = 001$. Note that nodes s_3 and s_4 belong to two data reporting groups. They form a loop and cannot reach the sink s_0 .

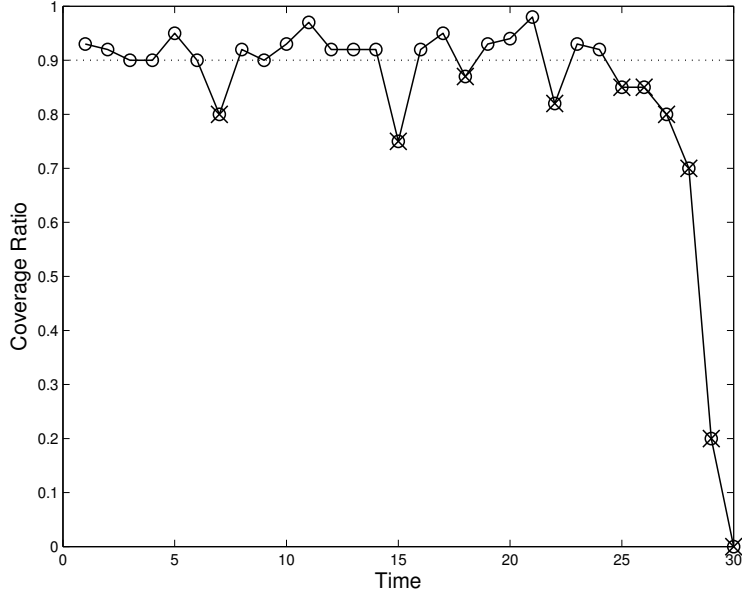


Figure 2.9: GoC-L in a sample coverage plot for $\text{GoC} = 0.9$.

2.8 Definition of Sensor Network Lifetime for Partial Coverage

In [44], lifetime for partial coverage is defined as the time interval until the coverage drops below GoC for the first time and in [47], it is defined as the time it takes for the coverage to drop below, and never exceed a given threshold. Our definition lies between these two definitions.

Due to node failures or temporary disconnectivity because of the scheduling, network coverage can drop below a threshold for a round and then exceed the threshold again. Therefore, we define the *GoC lifetime* (GoC-L) of the network as the number of rounds for which the coverage ratio is greater than or equal to GoC . We exclude the rounds where coverage drops below GoC but add all the rounds to GoC-L where coverage exceeds GoC . Figure 2.9 shows a sample coverage plot in which GoC-L corresponds to the number of circles not marked, i.e., $\text{GoC-L} = 20$. According to the definition in [44], this lifetime is equal to 6 and according to the definition in [47], this lifetime is 24.

2.9 Our Contribution

A practical sleep scheduling algorithm has to be distributed, simple and scalable. The algorithm should be distributed in the sense that nodes should only use local information to decide their states. It should be simple since sensor nodes do not have very powerful processors. Also, it should be scalable and adaptable to many network configurations, independent from the number of nodes and the field size. Nodes scheduled to be active should both satisfy coverage requirements and should be connected.

In the literature, coverage and connectivity issues are considered separately in many studies which is not a practical solution. Also, most of the studies in the area assume precise location information of nodes which is not available in many practical sensor network applications. Most of the studies do not provide an upper bound on the achievable network lifetime and only compare the proposed algorithms with the case when no sleep scheduling is applied or with other existing algorithms.

Also, the fact that the lifetime of a network can be extended by exploiting partial coverage is not extensively studied in the literature and is a new approach in sleep scheduling algorithms. Coverage requirement below one hundred percent increases the possible selection of the active set of nodes. Thus, the sleep scheduling algorithm has more choices when compared with the hundred percent coverage case. It is challenging to choose the most efficient subset of nodes in terms of energy consumption. We address the lack of research in the area of partial coverage and introduce a heuristic sleep scheduling algorithm for this purpose.

First, we provide an analysis of finding the minimum number of sensors that should be deployed to an area in order to provide various coverage levels. We find the minimum set of active sensors among these deployed sensors such that the

coverage requirement is satisfied. Before starting to design any sleep scheduling algorithm, this analysis can be useful in determining how many nodes to deploy to an area and in estimating how many of the deployed nodes are required to satisfy the desired coverage level.

We provide an ILP formulation for finding the optimum sleep scheduling in order to find out for how long can the network sustain a certain coverage level when the nodes are scheduled in an optimum way. In evaluating a sleep scheduling algorithm, an upper bound is very important for observing the performance of the algorithm. Although the ILP formulation is centralized and requires global knowledge about the network, the set of active nodes chosen by the optimum algorithm at each round can be a valuable information for a sleep scheduling algorithm design.

After this pre-deployment analysis, we propose a distributed algorithm for sleep scheduling which does not require location information and is simple and scalable. Nodes only use information they obtain from neighbors within their transmission range. The algorithm uses the sink, which is assumed to have abundant energy, to schedule nodes in the first tier of the network. Sink uses a simple ILP formulation in order to choose the most energy efficient subset of nodes from tier 1 to be active. Using the sink in such a procedure is a novel approach in sleep scheduling. We also allow nodes in the network to schedule other nodes by using local information. Since no location information is available, nodes are chosen with some randomness. The undesired effects of the randomness is reduced by using the sink and other nodes to schedule their neighbors. We compare our algorithm with the optimum solution which provides an upper bound. Also, the algorithm is compared with the DRS algorithm and another algorithm which we propose in Chapter 6.

Chapter 3

The Minimum Connected Set with Sufficient Coverage

Before deploying a sensor network to a remote area, the number of sensors to deploy should be determined since this will directly affect the cost and performance of the network. Also, finding the minimum number of nodes which can satisfy user constraints while being connected among the deployed sensors is an important issue since it reflects the redundancy in the network and is an important data for a sleep scheduling algorithm. A sleep scheduling algorithm designed for a network where this minimum number is very close to the deployed number of nodes may differ completely from a sleep scheduling algorithm designed for a network where this minimum number is very less than the deployed number.

In this section, first, an analysis for finding the minimum number of sensors that should be deployed in order to provide a certain user defined coverage level, named as *Grade of Coverage* (GoC), without considering connectivity, i.e., assuming every node can directly reach the sink node in one hop, is presented. Note that $0 \leq \text{GoC} \leq 1$ and GoC equals 1 corresponds to 100% coverage of the field, GoC equals 0.8 corresponds to 80% coverage of the field and so on. Next,

this analysis is extended to find the number of nodes, with limited transmission ranges, that should be deployed in order to satisfy GoC while being connected with high probability. This way, we get rid of the impractical assumption of a single hop network. Note that, since the deployment of nodes is a random procedure, the number of nodes to be deployed should be selected so as to assure coverage and connectivity with a probability close to 1. Finally, among the deployed nodes, the minimum number of nodes which satisfy both coverage and connectivity constraints is found.

3.1 Minimum Set of Nodes Satisfying Only Coverage

When connectivity of the sensor nodes is not an issue, a theoretical expression for the number of nodes to deploy in order to satisfy a certain GoC can be found assuming a distribution function for the nodes.

As mentioned in Section 2.7, the work in [46] assumes that the sensor nodes are deployed in a field as in Figure 2.4 in which the monitored field is constrained to be at a distance R_s to the deployed field, where R_s is the sensing radius of a node. Thus, the monitored area is not the same with the sensor deployment area which is not a square.

Instead of making this peculiar assumption, we assumed that the nodes are deployed over a square area, and the whole square area is monitored as shown in Figure 3.1. Also, we do not assume that the deployment field has curves at the four edges of the area as in Figure 2.4. This makes the calculations more practical and also applicable to the setting used in this thesis. The analysis is inspired from the analysis of [46], yet extends it to a more practical scenario. As in [46], we assume that the nodes are uniformly distributed in the network field,

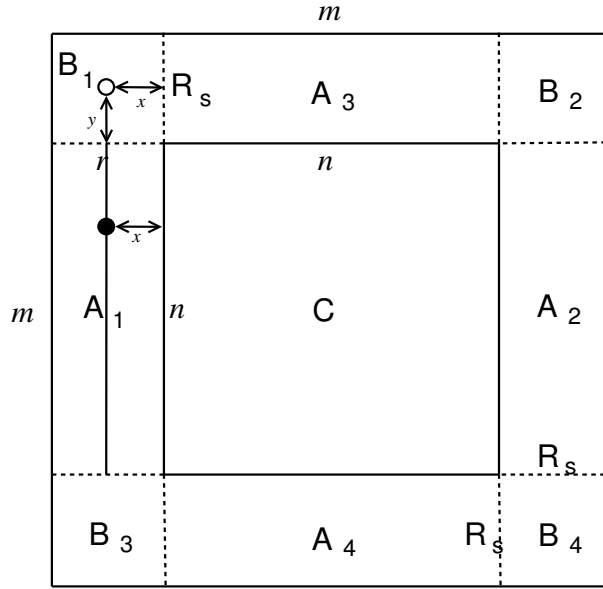


Figure 3.1: The network field divided into separate regions. We assume that the sensor deployed and the monitored areas are the same.

i.e., the horizontal and vertical coordinates of the nodes are two independent random variables with uniform distributions.

The probability of a point in the field not being covered by a sensor node is equal to the probability that the sensor node is at a distance larger than R_s , i.e., the probability that the sensor node is outside the area of a circle with radius R_s centered at the point. However, this is not valid for every point in the field. In a field as in Figure 3.1, points in regions A_1, A_2, A_3 or A_4 (A regions), in regions B_1, B_2, B_3 or B_4 (B regions) and in region C , have different probabilities of not being covered since in the A and B regions, a node should be outside a non-circular area for not covering the point. The probability of a point in area C of not being covered by a sensor is the same as in Eq. (2.1). Note that, because of the assumptions, all the points are in region C in [46] since the monitored area is inside the sensor deployed area with a distance of R_s , and thus a sensor node should always be outside a circular area for not covering a point.

A point in A_1 region is not covered if there is no sensor node in the unshaded region of Figure 3.2. However, this area changes as the distance x shown in

Figure 3.1 changes. The probability of a point in A_1 not being covered will be equal to Eq. (2.1) with S equal to the integral of the unshaded area with respect to x . This probability is the same for all A regions due to symmetry. The unshaded area S' in Figure 3.2 for all points on a line with given distance x can easily be calculated.

$$S'_A(x) = \pi R_s^2 - R_s^2 \arccos(1 - \frac{x}{R_s}) + (R_s - x)\sqrt{R_s^2 - (R_s - x)^2} \quad (3.1)$$

Note that $S'_A(0) = \pi R_s^2$ and $S'_A(r) = \pi R_s^2/2$. Now, we have to average $S'_A(x)$ over all x distances to find the probability of point in region A not being covered.

$$\begin{aligned} A &= \frac{n}{nr} \int_0^{R_s} S'_A(x) dx \\ &= R_s^2 \left(\pi - \frac{2}{3} \right) \end{aligned} \quad (3.2)$$

Similarly, a point in B_1 region is not covered if there is no sensor node in the unshaded region of Figure 3.3. However, this area changes as the distances x and y shown in Figure 3.1 change. The probability of a point in B_1 not being covered is equal to Eq. (2.1) with S equal to the integral of the unshaded area with respect to x and y . This probability is the same for all B regions. The unshaded area in Figure 3.3 for a given distance x and y distance is given as follows.

$$\begin{aligned} S'_B(x, y) &= \frac{\pi/2 + \arcsin(1 - \frac{x}{R_s}) + \arcsin(1 - \frac{y}{R_s})}{2} R_s^2 + \frac{(R_s - x)\sqrt{R_s^2 - (R_s - x)^2}}{2} \\ &\quad + \frac{(R_s - y)\sqrt{R_s^2 - (R_s - y)^2}}{2} + (R_s - x)(R_s - y) \end{aligned} \quad (3.3)$$

We have to average $S'_B(x, y)$ over all x and y distances to obtain the probability of not being covered.

$$\begin{aligned} B &= \frac{1}{R_s^2} \int_0^{R_s} \int_0^{R_s} S'_B(x, y) dx dy \\ &= R_s \left(\frac{\pi}{2} - 1 \right) + R_s^2 \left(\frac{\pi}{4} + \frac{7}{12} \right) \end{aligned} \quad (3.4)$$

Finally, multiplying the probability of a point in a given area (A , B , or C) not being covered by any sensor node by the corresponding area, we obtain P

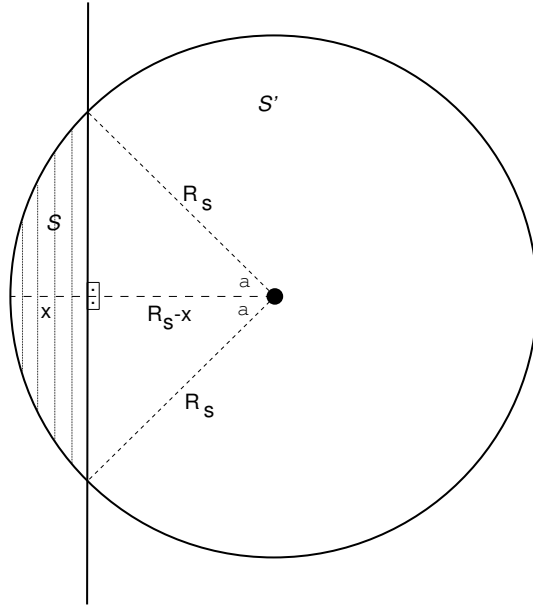


Figure 3.2: Coverage area of a point in area A_1, A_2, A_3 or A_4 .

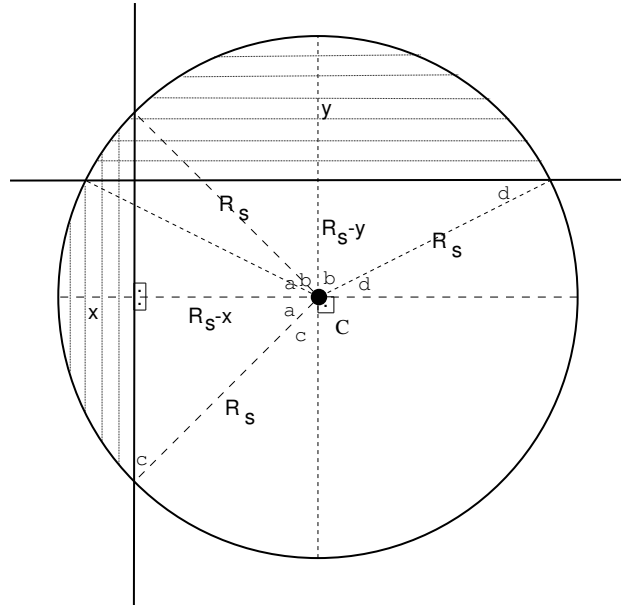


Figure 3.3: Coverage area of a point in area B_1, B_2, B_3 or B_4 .

which is the probability of a general point in the field not being covered.

$$\begin{aligned}
P &= \frac{n^2}{m^2} \left(\frac{m^2 - \pi R_s^2}{m^2} \right) + \frac{4nR_s}{m^2} \left(\frac{m^2 - A}{m^2} \right) + \frac{4R_s^2}{m^2} \left(\frac{m^2 - B}{m^2} \right) \\
&= \frac{1}{m^4} (n^2(m^2 - \pi R_s^2) + 4R_s m^2(n + R_s) - 4R_s(nA + R_s B)) \quad (3.5)
\end{aligned}$$

The probability of a point being covered by at least one sensor node among k nodes is equal to GoC which is also equal to $1 - P^k$ from which the minimum number of sensors to satisfy GoC can be easily found.

$$k = \left\lceil \frac{\log(1 - \text{GoC})}{\log \left(\frac{1}{m^4} (n^2(m^2 - \pi R_s^2) + 4R_s m^2(n + R_s) - 4R_s(nA + R_s B)) \right)} \right\rceil \quad (3.6)$$

Figures 3.4 and 3.5 show the number of nodes obtained from Eq. (3.6) and from simulations to provide various levels of GoC for a 100m-by-100m and a 200m-by-200m network, respectively, for various sensing radiuses. Every point obtained from simulations in the plots are averages of 100 runs. The simulations and Eq. (3.6) greatly overlap.

Although a uniform distribution for the locations of the sensor nodes is assumed, any distribution can be used for finding k . The probability that a point is not covered can be found by integrating the distribution function of the nodes over the areas of Figures 3.2, 3.3 or a circular area depending on the location of the point. Then, with a similar analysis to the one presented above, k can be found.

3.2 Minimum Set of Nodes Satisfying Coverage and Connectivity

In the previous section, connectivity was not considered and it was assumed that every node can reach the sink node in a single hop. In this section, both connectivity and GoC is considered. A limited transmission range equal to R_t for

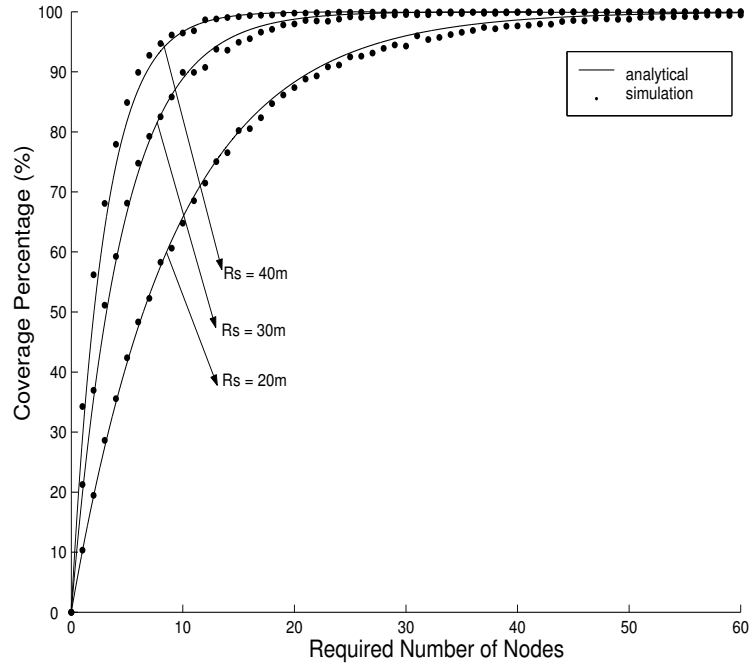


Figure 3.4: Analytical and simulation results for a 100m by 100m network for sensing radii 20m, 30m and 40m.

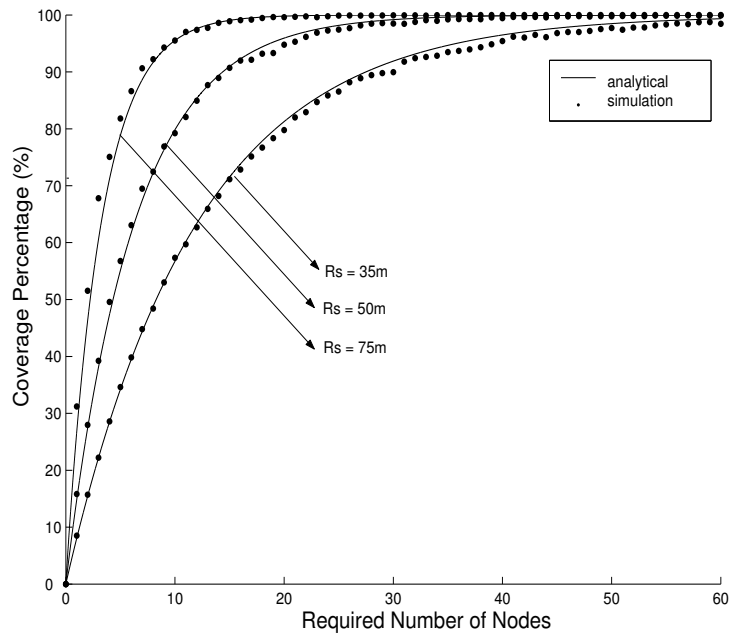


Figure 3.5: Analytical and simulation results for a 200m by 200m network for sensing radii 35m, 50m and 75m.

each node is assumed and the minimum number of sensors which are connected and satisfy GoC is found by using simulations. First, the minimum number of uniformly deployed connected sensors which satisfy GoC with 0.95 probability with 95% confidence is found. This minimum number of sensors is called N_{min} . In other words, when N_{min} sensors are thrown with uniformly distributed locations to a square field, they form a connected set which provides a coverage larger than or equal to GoC with probability 0.95 with 95% confidence. In doing this, we start from an initial guess and then by increasing this number by one at each iteration we converge to N_{min} .

Tables 3.1 and 3.2 show Eq. (3.6) and the N_{min} values for a 200m-by-200m and 300-by-300m network field, respectively, for R_t and R_s equal to 50m. The second column in the tables shows the number of nodes k which satisfy GoC without considering connectivity obtained from Eq. (3.6). The third column labeled by N_{min} represents the number of nodes that satisfies GoC while being connected with the probability value given in the fourth column. Fifth and sixth columns are the 95% confidence intervals for the corresponding probabilities and 1,000 runs are taken in computing these intervals.

There is a big difference between the number of nodes required to satisfy GoC without considering connectivity and N_{min} . For example, for GoC = 0.9 in a 200m-by-200m field, only 14 nodes are required for coverage when connectivity is not considered. However, in order to connect these nodes, we need an extra 29 nodes.

After finding N_{min} , an integer linear program, called *integer linear program finding the minimum connected set satisfying the desired coverage* (ILPMinConCov) is implemented which finds the minimum set of sensors, among these N_{min} deployed sensors, which satisfy coverage and connectivity constraints.

Note that, in deriving the numerical results, it is assumed that the sensor locations are uniformly distributed. However, this approach can be applied to any kind of sensor distribution since the linear program and the preceding analysis only requires the locations of the sensor nodes as inputs and does not assume any specific distribution.

3.3 Minimum Connected Set of Nodes among N_{min} Nodes Satisfying GoC (ILPMinConCov)

ILPMinConCov finds the minimum set of nodes among N_{min} uniformly deployed nodes which are connected and satisfy GoC. We assume that the field that will be sensed is represented by a grid structure. A grid i is covered if it is in the sensing range of at least one active node, i.e., at a distance less than or equal to R_s to at least one active node, and it is not covered if it is not in the sensing range of any active node. *Grid Parameter (GP)* represents the number of the grids used to represent the field of the network. In our case, a grid corresponds to a square and each grid is covered if at least one node covers the center of this square, i.e., if the center of the square is in the sensing range of at least one operating sensor node. Decreasing the size of each grid will increase the precision of the coverage percentage calculations. However, it will also cause a huge increase in computation time. We observed that using 4m-by-4m square for each grid is a good choice in terms of computation time and precision for the network field sizes we study. Thus, for a 200m-by-200m field, $GP = 2500$.

It is assumed that nodes which are scheduled to be active sense the environment and forward their data together with their descendants' data towards the sink. In forwarding the data towards the sink, nodes use their hop count with respect to the sink as a reference and send data to their neighbors with smaller hop counts. For example, a node which has an hop count 2, sends its data to

one of its neighbors with hop count 1. This way, routing loops are avoided and nodes send their data by using minimum number of hops to the sink. Definitions of hop count and tier number were given in Section 2.1.

Now, the inputs, decision variables and constraints in ILPMinConCov will be described.

- N : Number of nodes in the network
- T_i : Tier number of node i , $\forall i \in [1, N]$
- s_i : Selection variable of node i , $\forall i \in [1, N]$

$$s_i = \begin{cases} 1, & \text{node } i \text{ is used in the minimum set} \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in [1, N] \quad (3.7)$$

- f_{ij} : The flow parameter

$$f_{ij} = \begin{cases} 1, & \text{node } i \text{ uses node } j \text{ as its next hop} \\ 0, & \text{otherwise} \end{cases} \quad (3.8)$$

$$\forall (i, j) : \{i \in [1, N], j \in [1, N]\}$$

- v_i : Coverage variable for grid i

$$v_i = \begin{cases} 1, & \text{center of grid } i \text{ is covered by at least one active node} \\ 0, & \text{center of grid } i \text{ is not covered by any active node} \end{cases} \quad (3.9)$$

$$\forall i \in [1, GP]$$

- c_{ni} : Coverage matrix of the field

$$c_{ni} = \begin{cases} 1, & \text{center of grid } i \text{ is in the sensing range of node } n \\ 0, & \text{otherwise} \end{cases} \quad (3.10)$$

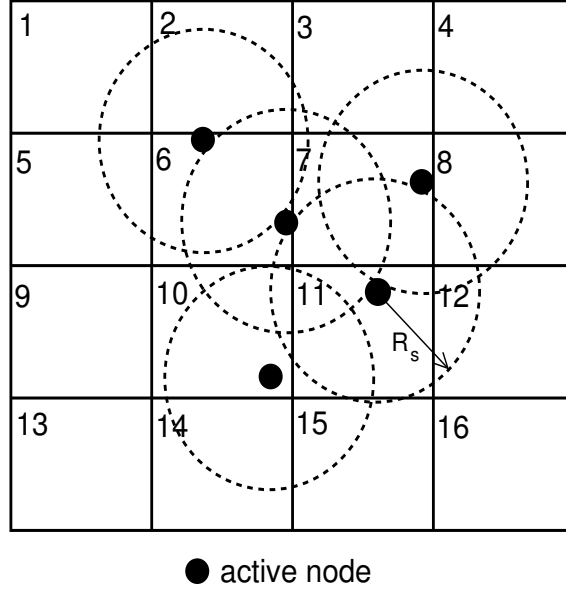
$$\forall (n, i) : \{n \in [1, N], i \in [1, GP]\}$$

Inputs

The input parameters are N, T_i, c_{ni} .

Decision Variables

The decision variables in the linear formulation are f_{ij}, s_i, v_i . Coverage variable v_i is shown for a sample network with a 4x4 grid and 5 active nodes in Figure 3.6.



$$\begin{aligned}
 v_1 &= 0, & v_2 &= 1, & v_3 &= 0, & v_4 &= 0 \\
 v_5 &= 0, & v_6 &= 1, & v_7 &= 1, & v_8 &= 1 \\
 v_9 &= 0, & v_{10} &= 1, & v_{11} &= 1, & v_{12} &= 0 \\
 v_{13} &= 0, & v_{14} &= 1, & v_{15} &= 0, & v_{16} &= 0
 \end{aligned}$$

Figure 3.6: The coverage variable $\{v_i\}$ where R_s is the sensing range of the nodes.

Constraints

A node can only forward to a node with tier number less than its tier number.

$$f_{ij} = 0 \quad \text{if} \quad T_i \leq T_j, \quad \forall i, j \quad (3.11)$$

The sum of the outgoing flows from a node is zero if the node is sleeping and one otherwise.

$$\sum_j f_{ij} = s_i, \quad \forall i \quad (3.12)$$

No flow should be directed to a sleeping node.

$$f_{ij} \leq s_j, \quad \forall i \quad (3.13)$$

The following three equations are for the coverage requirements. If all the nodes covering grid i are asleep, then grid i is not covered.

$$v_i \leq \sum_{n: c_{ni}=1} s_n, \quad \forall i \quad (3.14)$$

If any of the nodes whose sensing area covers grid i is active, then grid i is covered.

$$v_i \geq c_{ni}s_n, \quad \forall i, n \quad (3.15)$$

The next constraint forces the number of covered grids to be larger than a certain percentage, the GoC.

$$\sum_i v_i \geq \text{GoC} \times GP \quad (3.16)$$

Objective

The objective function is to minimize the number of connected sensor nodes which satisfy GoC and is given below.

Minimize

$$\sum_i s_i \quad (3.17)$$

The ILPMinConCov column in Tables 3.1 and 3.2 shows the results of the integer linear program for the corresponding GoC values for a 200m-by-200m and 300-by-300m network field, respectively. The following two columns are the 95% confidence intervals for the 100 runs taken for each GoC. In Table 3.2, the linear program could not provide results for the 100% coverage case in finite amounts of time, so that entry of the table is displayed as not available.

For some GoC values, ILPMinConCov results are below the theoretical numbers displayed in the second column. For example, for the 200m-by-200m network, 14 uniformly distributed nodes are sufficient to provide 90% coverage. However, among 43 nodes, there is a set of nodes with a population of 11.1 which can satisfy 90% coverage and which are connected. Since, these nodes are selected from a larger number of nodes uniformly distributed in the area and since they are not constrained to be uniformly distributed, ILPMinConCov produces lower results than the theoretical value.

Figures 3.7 and 3.8 shows the plot of the minimum number of nodes (obtained by ILPMinConCov) together with the confidence intervals that should be active among N_{min} uniformly deployed nodes in order to satisfy GoC given in the horizontal axis.

GoC	Eq. (3.6)	$N_{\min}$	Ave- rage	Lower Bound	Upper Bound	ILPMin- ConCov	Lower Bound	Upper Bound
0.00	0	0	1.000	1.000	1.000	0.0	0.000	0.000
0.05	1	15	0.975	0.964	0.986	1.0	1.000	1.000
0.10	1	15	0.963	0.952	0.973	1.0	1.000	1.000
0.15	1	15	0.959	0.949	0.969	1.0	1.000	1.000
0.20	1	19	0.985	0.975	0.994	2.0	2.000	2.000
0.25	2	19	0.977	0.967	0.987	2.0	1.990	2.030
0.30	2	19	0.957	0.946	0.968	2.4	2.325	2.535
0.35	3	22	0.973	0.963	0.984	3.1	3.064	3.196
0.40	3	22	0.956	0.946	0.966	3.5	3.387	3.613
0.45	4	24	0.959	0.949	0.970	4.2	4.054	4.306
0.50	4	25	0.954	0.944	0.964	4.7	4.552	4.868
0.55	5	29	0.972	0.963	0.982	5.2	5.109	5.351
0.60	6	29	0.965	0.956	0.974	6.1	5.941	6.279
0.65	7	30	0.950	0.939	0.961	6.7	6.546	6.854
0.70	8	33	0.968	0.958	0.978	7.4	7.203	7.577
0.75	9	34	0.953	0.943	0.963	8.1	7.948	8.272
0.80	10	37	0.971	0.960	0.981	9.1	8.864	9.236
0.85	12	40	0.968	0.957	0.978	9.9	9.751	10.089
0.90	14	43	0.964	0.954	0.974	11.1	10.908	11.259
0.95	19	50	0.965	0.955	0.974	12.5	12.356	12.724
0.99	29	66	0.967	0.956	0.978	14.1	13.889	14.231
1.00	-	92	0.962	0.952	0.971	15.4	15.214	15.506

Table 3.1: Minimum number of uniformly deployed nodes found to satisfy GoC. The results of Eq. (3.6), $N_{\min}$ and ILPMinConCov for a 200m-by-200m network for sensing and transmission radius of 50m for each node.

GoC	Eq. (3.6)	$N_{\min}$	Ave- rage	Lower Bound	Upper Bound	ILPMin- ConCov	Lower Bound	Upper Bound
0	0	0	1.000	1.000	1.000	0.0	0.000	0.000
0.05	1	33	0.954	0.943	0.964	1.0	1.000	1.000
0.10	2	42	0.964	0.954	0.974	2.0	2.000	2.000
0.15	3	48	0.961	0.951	0.970	2.8	2.763	2.917
0.20	3	49	0.957	0.945	0.969	4.0	3.978	4.102
0.25	4	52	0.958	0.948	0.968	5.1	5.014	5.206
0.30	5	53	0.961	0.951	0.970	6.3	6.208	6.452
0.35	6	56	0.968	0.958	0.977	7.5	7.387	7.673
0.40	7	59	0.957	0.946	0.967	8.8	8.647	8.953
0.45	8	62	0.966	0.956	0.976	9.9	9.779	10.041
0.50	10	65	0.955	0.944	0.966	11.6	11.381	11.840
0.55	11	65	0.955	0.944	0.965	12.8	12.660	13.000
0.60	12	69	0.959	0.948	0.970	14.2	13.964	14.336
0.65	14	67	0.963	0.942	0.985	15.9	15.720	16.120
0.70	16	75	0.964	0.954	0.974	17.3	17.010	17.670
0.75	19	77	0.960	0.950	0.969	18.7	18.531	18.909
0.80	22	79	0.962	0.952	0.972	20.7	20.509	20.971
0.85	25	84	0.953	0.943	0.962	22.6	22.349	22.891
0.90	31	92	0.960	0.949	0.971	25.0	24.653	25.247
0.95	40	104	0.959	0.948	0.970	27.3	26.991	27.589
0.99	61	135	0.953	0.943	0.963	30.1	30.348	30.632
1.00	-	218	0.959	0.947	0.970	n/a	n/a	n/a

Table 3.2: Minimum number of uniformly deployed nodes found to satisfy GoC. The results of Eq. (3.6), $N_{\min}$ and ILPMinConCov for a 300m-by-300m network for sensing and transmission radius of 50m for each node.

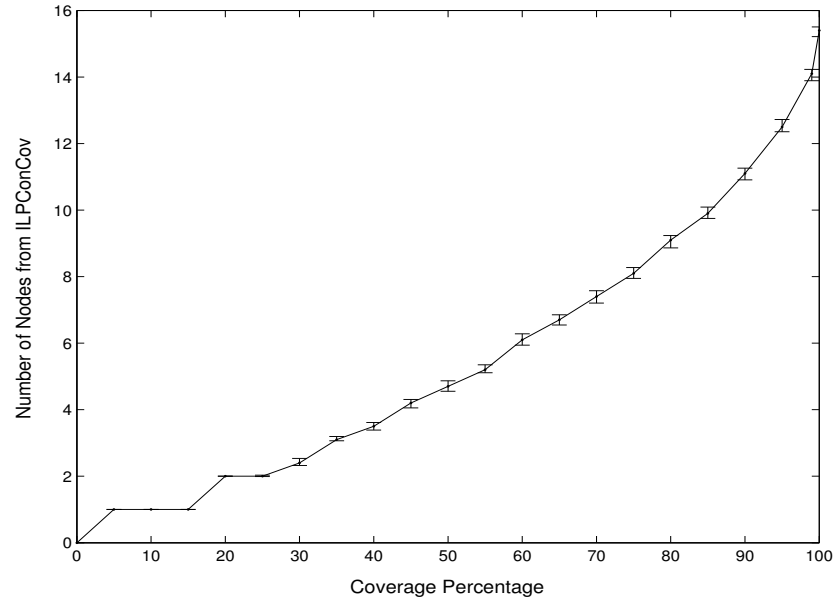


Figure 3.7: Minimum number of active nodes for a given GoC in a 200m-by-200m network where $R_s = 50\text{m}$ and $R_t = 50\text{m}$. Bars represent the 95% confidence intervals of the simulation results.

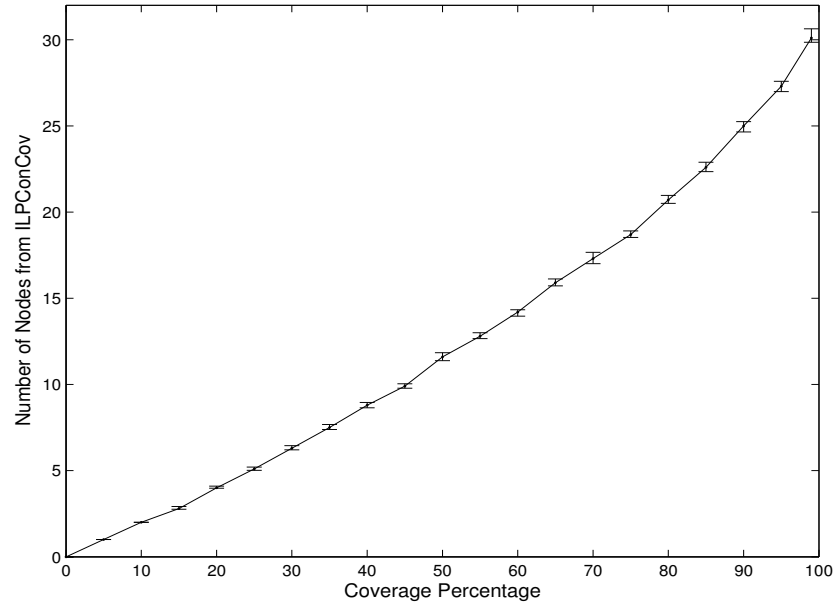


Figure 3.8: Minimum number of active nodes for a given GoC in a 300m-by-300m network where $R_s = 50\text{m}$ and $R_t = 50\text{m}$. Bars represent the 95% confidence intervals of the simulation results.

Chapter 4

Centralized Optimum Sleep Scheduling

The aim of this thesis is to design a sleep scheduling algorithm which ensures connectivity of the operating nodes and at the same time provides a coverage ratio above a user defined level, named as *Grade of Coverage* (GoC), for as long as possible. In this chapter, we find the maximum achievable lifetime of such a network by using integer linear programming techniques. This maximum lifetime is used in Chapter 6 to evaluate our distributed algorithm.

After describing the network model that is used throughout the thesis, two ILP formulations for the cases when there is no aggregation and when there is full aggregation is proposed. These linear formulations are combined together with the algorithm given in Section 4.2.2 for finding the maximum number of rounds for which the network coverage can be held above GoC.

4.1 Network Model

In this section, the basic concepts and assumptions about the network model used in the thesis is introduced.

First of all, it is assumed that the sensor network is uniformly deployed, so that the locations of the nodes are uniformly distributed in the area, to a square area and consists of N nodes and 1 sink. Sink is located at the origin point $(0, 0)$ of the field and has higher computation capability than other nodes, i.e., it has abundant energy and memory resources.

It is assumed that nodes which are scheduled to be active sense the environment and forward their data together with their descendants' data towards the sink. In forwarding the data towards the sink, nodes use their hop count with respect to the sink as a reference and send data to their neighbors with smaller hop counts. For example, a node which has an hop count 2, sends its data to one of its neighbors with hop count 1. This way, the routing loops are avoided and nodes send their data by using minimum number of hops to the sink. The proposed algorithms choose the most energy efficient method in doing this. The terms hop count and tier number are used interchangeably during the thesis. Definitions of hop count and tier number were given in Section 2.1. We allow nodes to forward data only to their neighbors with smaller hop counts because when nodes are allowed to forward data to all their neighbors, the linear program takes very long time. However, since the network is dense, this routing choice will not result in a big performance difference.

The transmission range (R_t) and the sensing range (R_s) are constant, i.e., they cannot be modified by the sensor nodes. Also, the network is static, i.e., nodes are not mobile.

For the energy consumption, it is assumed that every sensor node has an initial energy of E_{init} and every data packet transmitted consumes an energy of E^t and every packet received consumes an energy of E^r . Also, additional energy consumption is associated with the sleep and routing processes which are described in the later sections.

Two schemes are assumed for data forwarding: in the first one, all the packets have to be forwarded without any processing in the sensor nodes (no aggregation) and in the second one, all the packets are assumed to be combined into a single packet before transmitting (full aggregation). No aggregation and full aggregation cases are considered separately in proposing the ILP formulations since in the no aggregation case, data from a node in tier X has to pass from a node in tier $X - 1$, a node in tier $X - 2$, $\dots$, and a node in tier 1 to reach the sink. In order to account for the energy consumption of each individual node, a flow variable which contains the path as well as the source of the data passing through the path has to be defined. However, in the full aggregation case, since every node only transmits a single packet regardless of the number of packets it receives, a flow variable which contains only the path of forwarding is sufficient in calculating the energies. This discussion will become more clear in the description of the flow variable in Section 4.2.1.

The impact of full aggregation in terms of the number of packets each node has to transmit and receive is important. From Figure 4.1, where there is no aggregation, it can be observed that the nodes closer to the sink carry very high loads whereas in the full aggregation case, the number of one hop descendants is more important than the tier number as illustrated in Figure 4.2. Note that, tier 1 nodes, when there is no aggregation, receive $\{6, 8, 11, 13\}$ packets and transmit $\{7, 9, 12, 14\}$ packets respectively after including their own packets. So, a total of $\{13, 17, 23, 27\}$ high energy transceiver operations are required. However, in the full aggregation case, tier 1 nodes receive only $\{2, 2, 3, 4\}$ packets

and transmit $\{1, 1, 1, 1\}$ packets since they are assumed to combine each packet into only one packet. Thus, a total of $\{3, 3, 4, 5\}$ high energy transceiver operations are required. It is easy to notice the enormous difference in terms of energy consumption in the no aggregation and full aggregation cases. Although, full aggregation seems very energy efficient, it is impractical for many sensor network applications since it may not be possible in many cases to combine data in such a way and even if it is possible, individual sensor nodes may not have such high processing capabilities.

We use a time driven data reporting model in which the data reporting operation is periodic and the network operation occurs in rounds. In order to realize such a network operation, clocks in all sensors first need to be synchronized. A MAC protocol, such as S-MAC [48], a commonly used MAC protocol which both provides time synchronization among nodes and allows nodes to have their own scheduling, can be used. Sensor nodes save energy by sleep and listen schedules in the MAC layer in S-MAC.

At every round, active nodes send their data together with their descendants' data to the sink. The proposed algorithms take place at the beginning of each round and schedule nodes to sleep or to be active at that round. For each round, the coverage ratio, which is the proportion of the covered area provided by the set of connected active nodes to the entire sensing area is calculated and compared with GoC and the energies of the nodes are dropped depending on the number of packets they receive and transmit and on their scheduling operations.

4.2 Optimum Scheduling Algorithms

ILP formulations which aim to maximize the number of rounds in which the network coverage is above GoC for the no aggregation and full aggregation cases are described separately.

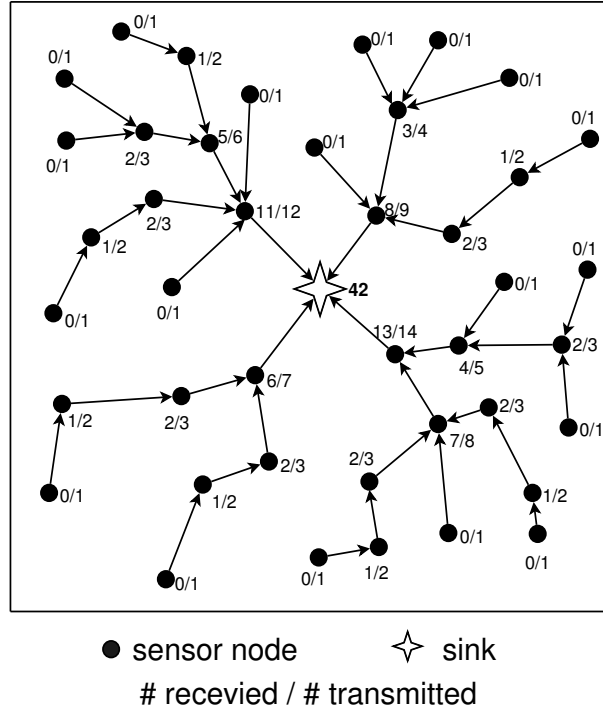


Figure 4.1: Number of packets received and transmitted when there is no aggregation applied.

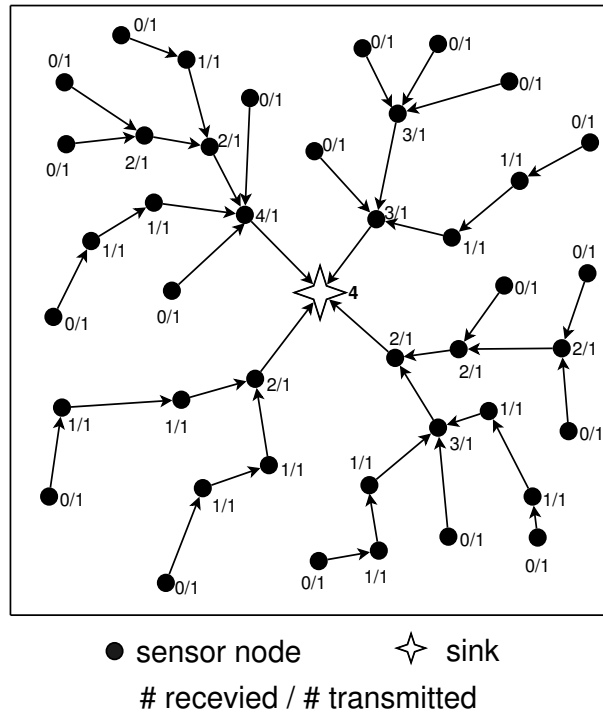


Figure 4.2: Number of packets received and transmitted when there is full aggregation applied.

By the ILP formulations, the optimum sleep schedules of nodes and the optimum routing paths along which the selected active nodes will forward their data are found. These optimum scheduling algorithms require exact positions of sensor nodes, the remaining energies of all the nodes, global knowledge about the network and long run times. They are also centralized, i.e., the schedules of the nodes are computed at a central point with location and energy informations and then somehow all nodes become aware of their schedules with some energy cost. Therefore, these algorithms cannot be used in many practical sensor network applications. However, the results of these ILP schemes will provide the best possible scheduling of nodes and routing of packets while satisfying GoC. These results will also allow us to evaluate and compare the performance of the proposed distributed sleep scheduling algorithm which will be explained in Chapter 5.

4.2.1 Optimum Node Scheduling Without Aggregation (ILPNA)

In this section, an optimum node scheduling and routing scheme, for the case when there is no aggregation, using an integer linear programming approach is proposed. This algorithm is named as *integer linear program with no aggregation* (ILPNA).

The formulation finds an optimum set of nodes which will operate during a round such that the coverage constraint defined by the user is satisfied. In doing this, the program aims to maximize the remaining energies of every node giving priority to nodes with lower tier numbers, which are nodes closer to the sink, and to nodes with lower energies. The energies of nodes reduce depending on their descendant numbers and sleep modes, i.e., whether active or sleeping, at each round. The program assumes global knowledge of the network.

In ILPNA, there is no aggregation so that nodes transmit the data of both themselves and their descendants', which contain their own descendants' data and so on. The total number of data packets which the node will receive has to be known in order to calculate the amount of energy it will consume. Therefore, a 3 dimensional flow parameter for discriminating the sources of the data packets has to be used.

At the beginning of each round, the nodes are sorted according to their energies and tier numbers. First, the nodes are sorted with respect to their tier numbers in ascending order. Then, the nodes with the same tier number are sorted with respect to their energy levels again in ascending order (see Figure 4.3). *Sorted IDs* (SID) contains the node IDs of the sorted nodes.

Node ID	Tier Number	Energy
1	1	3
2	3	6
3	1	7
4	2	1
5	3	9
6	1	4
7	2	5
8	3	2

$$\text{SID} = [\underbrace{1, 6, 3}_{\text{Tier1}}, \underbrace{4, 7}_{\text{Tier2}}, \underbrace{8, 2, 5}_{\text{Tier3}}]$$

Figure 4.3: Forming the SID.

ILPNA first tries to maximize the energy of node SID_1 . After finding a solution for this objective, this result is added as a new constraint to the ILP formulation in the next round. Then, ILPNA tries to maximize the energy of node SID_2 with the additional constraint in the energy of node SID_1 and so on. There are many choices for the energies of other nodes when the energy of node

SID₁ is equal to the maximum value found by the program. Thus, it is better to choose the best among these possible solutions. So, we continue adding the previous solutions as constraints and try to maximize the energy of the next node in the SID until a number called *Energy Optimization Depth (EOD)*. Intuitively, choosing this number as large as possible will give better results in the sense that more nodes' energies will remain as high as possible. Therefore, *EOD* is chosen to be equal to the number of nodes (N) which yields the best possible solution since in this case every nodes' energy is maximized. It is interesting to note that, in many cases, choosing *EOD* equal to the size of the tier 1 nodes yields exactly the same results with choosing it equal to N in shorter run times. This is logical since the bottleneck nodes in the network are the tier 1 nodes.

The algorithm used for the maximization of energies process described here is given in Figure 4.5 and described in Section 4.2.2.

Parameters

Now, the constants and variables used in ILPNA will be described.

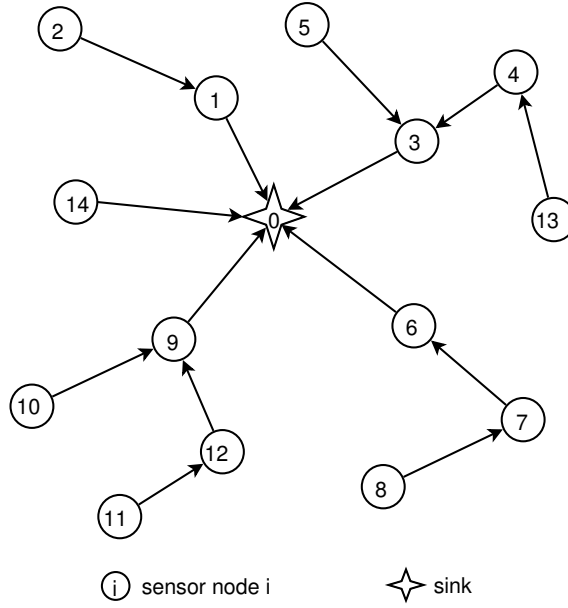
- N : Number of nodes in the network
- T_i : Tier number of node i , $\forall i \in [1, N]$
- e_i : Energy of node i , $\forall i \in [1, N]$
- e'_i : Energy of node i after the round, $\forall i \in [1, N]$
- s_i : Sleeping variable of node i , $\forall i \in [1, N]$

$$s_i = \begin{cases} 1, & \text{node } i \text{ is active} \\ 0, & \text{node } i \text{ is sleeping} \end{cases} \quad \forall i \in [1, N] \quad (4.1)$$

- f_{ij}^n : The flow parameter (see Figure 4.4)

$$f_{ij}^n = \begin{cases} 1, & \text{node } n \text{ uses path } i \rightarrow j \\ 0, & \text{otherwise} \end{cases} \quad (4.2)$$

$$\forall(i, j, n) : \{i \in [1, N], j \in [1, N], n \in [1, N]\}$$



$$\begin{array}{cccccc} f_{1,0}^1 & f_{2,1}^2 & f_{1,0}^2 & f_{3,0}^3 & f_{4,3}^4 & f_{3,0}^4 \\ f_{5,3}^5 & f_{3,0}^5 & f_{6,0}^6 & f_{7,6}^7 & f_{6,0}^7 & f_{8,7}^8 \\ f_{7,6}^8 & f_{6,0}^8 & f_{9,0}^9 & f_{10,9}^{10} & f_{9,0}^{10} & f_{11,12}^{11} \\ f_{12,9}^{11} & f_{9,0}^{11} & f_{12,9}^{12} & f_{9,0}^{12} & f_{13,4}^{13} & f_{4,3}^{13} \\ & & f_{3,0}^{13} & f_{14,0}^{14} & & \end{array}$$

Figure 4.4: The flow variable f_{ij}^n . The f_{ij}^n 's shown are equal to 1, others are equal to 0.

- v_i : Coverage variable for grid i , see Figure 3.6

$$v_i = \begin{cases} 1, & \text{center of grid } i \text{ is covered by at least one active node} \\ 0, & \text{center of grid } i \text{ is not covered by any active node} \end{cases} \quad (4.3)$$

$$\forall i \in [1, GP]$$

GP is defined in Section 3.3.

- c_{ni} : Coverage matrix of the field

$$c_{ni} = \begin{cases} 1, & \text{center of grid } i \text{ is in the sensing range of node } n \\ 0, & \text{otherwise} \end{cases} \quad (4.4)$$

$$\forall(n, i) : \{n \in [1, N], i \in [1, GP]\}$$

- x_{ij} : Variable to force each node to forward its own data and all received data only to one link, i.e., to avoid splitting of traffic

$$x_{ij} = \begin{cases} 1, & \text{node } i \text{ uses node } j \text{ as its next hop} \\ 0, & \text{otherwise} \end{cases} \quad (4.5)$$

$$\forall(i, j) : \{i \in [1, N], j \in [1, N]\}$$

- E^t : Transmit energy per packet
- E^r : Receive energy per packet
- E^s : Energy per round consumed for sleep scheduling, which depends on whether the node sleeps or not, the tier number of the node and the parameters ASD and NAS of the heuristic algorithm described in Chapter 5
- $E^{routing}$: Energy spent by each node at each round for exchanging routing messages.

Constraints

Nodes can only forward to nodes with tier numbers less than its tier number.

$$f_{ij}^n = 0 \quad \text{if } T_i \leq T_j, \quad \forall i, j, n \quad (4.6)$$

The next constraint expresses the conservation of flows. For the sink, the outgoing flow is zero and the incoming flow corresponding to a source depends on the sleep mode of that source. For the node itself, there is no incoming flow from itself but an outgoing flow if the node is not sleeping. Otherwise, the input flow

and the output flow should be equal.

$$\sum_j f_{ij}^n - \sum_j f_{ji}^n = \begin{cases} s_n, & i = n \\ -s_n, & i = \text{sink} \\ 0, & \text{otherwise} \end{cases} \quad \forall n, i \quad (4.7)$$

If the node sleeps, there should be no incoming and no outgoing flow through it.

$$f_{ij}^n \leq s_i, \quad \forall n, i, j \quad (4.8)$$

$$f_{ji}^n \leq s_i, \quad \forall n, i, j \quad (4.9)$$

The energy of a node will drop in direct proportion with the number of received and transmitted data packets. Also, some additional energy is consumed depending on the sleep schedule together with a constant energy consumption for routing.

$$e'_i = e_i - \sum_n \sum_j f_{ij}^n \cdot E^t - \sum_n \sum_j f_{ji}^n \cdot E^r - E^s - E^{\text{routing}} \quad (4.10)$$

The energy of each node cannot become negative.

$$e'_i \geq 0, \quad \forall i \quad (4.11)$$

The next three equations force every node to forward all its data to only one node. Thus, nodes are not allowed to split the traffic. We observed that allowing nodes to split the traffic does not have a major effect in terms of the number of rounds achieved. The recent work in [49] supports our observation by showing that the energy savings of a splittable traffic strategy is relatively small when compared to the unsplitted strategy. However, the scope of this work is totally different from ours and they consider neither sleep scheduling nor coverage. But even in this case, the effects of splitting the traffic are similar.

$$x_{ij} \leq \sum_n f_{ij}^n \quad \forall i, j \quad (4.12)$$

$$x_{ij} \geq f_{ij}^n \quad \forall n, i, j \quad (4.13)$$

$$\sum_j x_{ij} \leq 1, \quad \forall i \quad (4.14)$$

The following three equations are for the coverage requirements. If all the nodes covering grid i are asleep, then grid i is not covered.

$$v_i \leq \sum_{n: c_{ni}=1} s_n \quad \forall i \quad (4.15)$$

If any of the nodes whose sensing area covers grid i is active, then grid i is covered.

$$v_i \geq c_{ni}s_n \quad \forall i, n \quad (4.16)$$

The last constraint forces the number of covered grids to be larger than a certain percentage, the GoC.

$$\sum_i v_i \geq \text{GoC} \times GP \quad (4.17)$$

ILPNA described here has to be implemented many times as explained in Section 4.2.1 and the algorithm that will be used for this purpose will be explained next. Since this algorithm is the same for the aggregation and no aggregation schemes, we first describe the algorithm and then proceed with the ILP formulation for the full aggregation case.

4.2.2 The Centralized Algorithm (CA)

The centralized algorithm (CA) uses ILPNA multiple times in each round in order to find the maximum number of rounds in which the coverage is above GoC. As explained in Section 4.2.1, the initial objective of ILPNA is to maximize the energy of node SID_1 , where SID contains the node IDs of the nodes sorted according to their tier numbers and remaining energies. Then, the result of this ILP is added as a constraint at the next round and the objective is modified to maximize the energy of node SID_2 . Next, the result of this run is also added as a constraint to the ILP and the objective is now to maximize SID_3 . This process continues till the last node in the network. This way, the optimum value for the energy of each node is found. The reason for employing such an iterative

process was explained in Section 4.2.1 and will be rephrased here. The energy of node SID_1 is maximized for possibly multiple alternatives of the energies of other nodes. Therefore, adding this result as a constraint and maximizing the energy of the next node in the SID chooses the best solution among this many alternatives. When this process is repeated for all nodes, we find the best solution for the energies of each node. Figure 4.5 shows how to use ILPNA to simulate the network performance.

At the beginning of the algorithm, SID is formed as described in Section 4.2.1, i.e., the nodes are sorted according to their energies and tier numbers (Step 1).

If a solution cannot be found for the *LinearProgram*, the program is terminated. Note that if a solution to *LinearProgram* (Set Objective : Maximize e'_{SID_1}) can be found, which means that the *LinearProgram* has objective of maximizing e'_{SID_1} which is the value that the energy of node 1 will take after the round (see Section 4.2.1), then there is also a solution to *LinearProgram* (Set Objective : Maximize e'_{SID_x}) for $x = 2, \dots, EOD$, (Steps 2-6). The reason will become clear shortly. Note that *LinearProgram* corresponds to ILPNA.

The algorithm then adds the most recent solution, i.e., solution of *LinearProgram* (Set Objective : Maximize $e'_{SID_{x-1}}$), as a constraint to the problem, (Steps 7 and 11), and changes the objective to Maximize e'_{SID_x} (Step 10) until energies of all the nodes are maximized with additional constraints coming from previous solutions. Note that EOD equals number of nodes N , (Step 9).

When a solution to *LinearProgram* (Set Objective : Maximize e'_{SID_1}) is found (let *result* denotes this solution), this means that there exists at least one solution for which $e'_{SID_1} \geq result$, i.e., at least the solution with $e'_{SID_1} = result$. So, when *LinearProgram* (Set Objective : Maximize e'_{SID_x} s.t. $1 < x \leq EOD$) is called, we certainly have a solution if we have a solution for node 1.

Next, the algorithm updates the energies of each node as in equation (4.18) (Step 14).

$$e_i = e_i - \sum_n \sum_j f_{ij}^n \cdot E^t - \sum_n \sum_j f_{ji}^n \cdot E^r - s_i \cdot E^s - E^{routing} \quad (4.18)$$

The tier numbers of nodes have to be updated when necessary, i.e., when the energies of nodes are not sufficient to participate in any network operation, in other words when they die. When the energy of a node drops below a certain level, named as *CriticalEnergy*, that node is deleted from the problem formulation and then the tier numbers are updated accordingly, (Steps 15-18). After a node's energy drops below a certain level, the ILP may result in an infeasible solution whereas a solution exists without that node. Applying Steps (15-18) remedies this problem.

Finally, the *SID* is formed again, i.e., nodes are sorted with respect to their new residual energies and tier numbers, the algorithm is run all over again until no solution exists, (Step 20).

4.2.3 Optimum Node Scheduling With Full Aggregation (ILPFA)

In this section, an optimum node scheduling and routing scheme, for the case when there is full aggregation, using an integer linear programming approach is proposed. This algorithm is named as *integer linear program with full aggregation* (ILPFA).

ILPFA is very similar to ILPNA except that there is full aggregation in the network. Every node first receives their neighbors' data. Then, every node adds its own data and combines all this data into a single packet. For example, a node can perform simple operations such as finding the average, minimum or maximum of the received data and its own data and only send one packet in

Centralized Algorithm:

```

1: Sort the Nodes According to their Remaining Energy and Tier Number (Form the
    $SID$ );
2: while 1==1 do
3:    $result \leftarrow LinearProgram$  ( Set Objective : Maximize  $e'_{SID_1}$  );
4:   if  $LinearProgram$  is Infeasible then
5:     Terminate the Program;
6:   else
7:      $LinearProgram$  ( Add Constraint :  $e'_{SID_1} \geq result$ );
8:   end if
9:   for  $x = 2$  to  $EOD$  do
10:     $result \leftarrow LinearProgram$  ( Set Objective : Maximize  $e'_{SID_x}$  );
11:     $LinearProgram$  ( Add Constraint :  $e'_{SID_x} \geq result$ );
12:  end for
13:  for  $i = 1$  to  $N$  do
14:    Update the Energy of Sensor Node  $i$ ;
15:    if  $e_i \leq CriticalEnergy$  then
16:      Delete Node  $i$  from the Problem
17:      Recompute Tier Numbers of Each Node
18:    end if
19:  end for
20:  Sort the Nodes According to Their Remaining Energy and Tier Number (Form
   the new  $SID$ );
21: end while

```

Figure 4.5: Centralized algorithm for finding the optimum sleep schedule

applications where individual data sensed from each sensor is not important. Energy consumption is defined depending on this situation as in Eq. (4.26).

In the full aggregation case, since the amount of data a node will receive depends only on the number of its one hop neighbors which decide to forward data to it, there is no need to introduce one more dimension to f_{ij} as in the no aggregation case.

The objective function is exactly the same with the objective function of ILPNA and SID is used in the same way as in ILPNA. The same procedure as described in Section 4.2.1 is applied for maximizing the remaining energies.

Parameters

Now the constants and variables used in ILPFA are described.

- N : Number of nodes in the network
- T_i : Tier number of node i , $\forall i \in [1, N]$
- e_i : Energy of node i , $\forall i \in [1, N]$
- e'_i : Energy of node i after the round, $\forall i \in [1, N]$
- s_i : Sleeping variable of node i , $\forall i \in [1, N]$

$$s_i = \begin{cases} 1, & \text{node } i \text{ is active} \\ 0, & \text{node } i \text{ is sleeping} \end{cases} \quad \forall i \in [1, N] \quad (4.19)$$

- f_{ij} : The flow parameter

$$f_{ij} = \begin{cases} 1, & \text{node } i \text{ uses node } j \text{ as its next hop} \\ 0, & \text{otherwise} \end{cases} \quad (4.20)$$

$\forall (i, j) : \{i \in [1, N], j \in [1, N]\}$

- v_i : Coverage variable for grid i , see Figure 3.6

$$v_i = \begin{cases} 1, & \text{center of grid } i \text{ is covered by at least one active node} \\ 0, & \text{center of grid } i \text{ is not covered by any active node} \end{cases} \quad (4.21)$$

$\forall i \in [1, GP]$

GP is defined in Section 3.3.

- c_{ni} : Coverage matrix of the field

$$c_{ni} = \begin{cases} 1, & \text{center of grid } i \text{ is in the sensing range of node } n \\ 0, & \text{otherwise} \end{cases} \quad (4.22)$$

$$\forall(n, i) : \{n \in [1, N], i \in [1, GP]\}$$

- E^t : Transmit energy per packet
- E^r : Receive energy per packet
- E^s : Energy per round consumed for sleep scheduling, which depends on whether the node sleeps or not, the tier number of the node and the parameters ASD and NAS of the heuristic algorithm described in Chapter 5
- $E^{routing}$: Energy spent by each node at each round for exchanging routing messages.

Constraints

Nodes can only forward to nodes with tier numbers less than its tier number.

$$f_{ij} = 0 \quad \text{if} \quad T_i \leq T_j, \quad \forall i, j \quad (4.23)$$

The sum of the outgoing flows from a node is zero if the node is sleeping and one otherwise.

$$\sum_j f_{ij} = s_i, \quad \forall i \quad (4.24)$$

No flow should be directed to a sleeping node.

$$f_{ij} \leq s_j, \quad \forall i \quad (4.25)$$

The receive energy depends on the number of neighbors using node i as their next hops and the transmit energy is constant since every node sends only a single packet if it is not sleeping. Also, some additional energy is consumed depending on the sleep schedule together with a constant energy consumption for routing.

$$e'_i = e_i - s_i \cdot E^t - \sum_j f_{ji} \cdot E^r - E^s - E^{routing}, \quad \forall i \quad (4.26)$$

The energy of each node cannot become negative.

$$e_i' \geq 0, \quad \forall i \quad (4.27)$$

The following three equations are for the coverage requirements. If all the nodes covering grid i are asleep, then grid i is not covered.

$$v_i \leq \sum_{n: c_{ni}=1} s_n, \quad \forall i \quad (4.28)$$

If any of the nodes whose sensing area covers grid i is active, then grid i is covered.

$$v_i \geq c_{ni}s_n, \quad \forall i, n \quad (4.29)$$

The last constraint forces the number of covered grids to be larger than a certain percentage, the GoC.

$$\sum_i v_i \geq \text{GoC} \times GP \quad (4.30)$$

Similar to ILPNA, ILPFA has to be run multiple times for each round. The algorithm that is run using ILPFA is given in Figure 4.5 and further described in Section 4.5. The only difference is in the energy update step, i.e., Step 19 in Figure 4.5. Note that *LinearProgram* in Figure 4.5 corresponds to ILPFA in this case. The energy spent at each round for the full aggregation case should be calculated as follows.

$$e_i = e_i - s_i \cdot E^t - \sum_j f_{ji} \cdot E^r - E^s - E^{\text{routing}}, \quad \forall i \quad (4.31)$$

This concludes our discussion for the optimum sleep schedules of a network. The proposed distributed approach to the problem is described next.

Chapter 5

Distributed Adaptive Sleep Scheduling Algorithm (DASSA)

Distributed Adaptive Sleep Scheduling Algorithm (DASSA) schedules sensor nodes in a dense sensor network to sleep or to operate while ensuring connectivity of the operating nodes and keeping the coverage ratio of the sensor field above a certain user specified value, named as *Grade of Coverage* (GoC), without any knowledge of sensor locations. The algorithm uses a novel approach for scheduling nodes closer to the sink, which are the critical nodes in a network since all traffic has to pass through them to reach the sink. The algorithm is adaptive in the sense that it reconfigures itself at the beginning of each round depending on the residual energies of sensor nodes and continues operation until none of the nodes in the network are able to reach the sink.

The primary objectives of any algorithm designed for a sensor network can be classified as energy efficiency, computational simplicity and scalability.

Energy efficiency DASSA schedules many of the nodes in the network not to operate as long as GoC is satisfied. This not only reduces the overall energy consumption, but also the network traffic and packet collisions. Since the total number of data forwarded in the network decreases and the sleeping

nodes do not spend any energy at all, the overall energy consumption of the network decreases. Also, DASSA chooses the set of nodes with highest residual energies to be active which further contributes to its energy efficiency.

Computational Simplicity Each node only performs simple calculations such as finding the maximum number in a small set of numbers which makes DASSA computationally simple.

Scalability Sensor networks consist of a large number of nodes and therefore any practical algorithm designed for these networks must be scalable. DASSA can be applied to dense and high population networks with only tuning some of its parameters without any increase in the complexity.

As mentioned earlier, sensor nodes sense and forward data periodically in rounds. At the beginning of each round, DASSA determines which nodes will sleep and which nodes will operate during the round. Nodes which do not sleep will sense data and transmit their own data together with other data received from their neighbors. The overhead of DASSA is quite small since nodes make simple computations and transmit small sized broadcast messages.

DASSA consists of five sequential steps. These steps are :

- 1. Neighbor discovery** In this step, every node gathers information about the nodes in their transmission range. This information includes the hop counts, the remaining energies and the IDs of the neighbor nodes.
- 2. Scheduling tier 1 nodes** Since every packet in the network passes through one of the tier 1 nodes before reaching the sink, tier 1 nodes are the first nodes in the network to die. Also, after all tier 1 nodes die, none of the remaining nodes can reach the sink so the network dies. Therefore, it is

crucial to maximize the lifetime of the tier 1 nodes. This step is devoted only to the scheduling of tier 1 nodes in the best possible way.

3. Scheduling intermediate nodes Although not as crucial as tier 1 nodes, nodes which are closer to the sink but not in tier 1 are also very important for the same reason explained for tier 1 nodes. Therefore, this step deals with the scheduling of nodes close to the sink.

4. Scheduling far away nodes Nodes which are at the edges of the field are not very effective in determining the lifetime of the network since these nodes do not have many descendants and in many cases only forward their own data. This step schedules these nodes in a simple way.

5. Transmitting and forwarding data After DASSA schedules each node, nodes forward their data to their neighbor which is closest to the sink and which has highest energy. The final step concludes the round.

In Chapter 4, centralized optimum scheduling algorithms which require exact sensor locations, the remaining energies of all nodes, global network knowledge and long run times were presented. However, as mentioned earlier, location information for a typical sensor network is not available and a centralized approach cannot be used. These optimum algorithms are used in Chapter 6 to evaluate the performance of DASSA.

The rest of this chapter describes each step of DASSA in detail. The flowchart of DASSA is given in Figure 5.10.

5.1 Step I : Neighbor Discovery

In this step, all nodes in the network learn their tier numbers, their neighbor IDs, and their neighbors' remaining energy levels. For this, each node broadcasts a packet which contains the fields of information shown in Figure 5.1 (see Figure 5.2).

Node ID	Residual Energy	Tier Number
---------	-----------------	-------------

Figure 5.1: Broadcast message format.

Initially, the sink broadcasts a message with *Tier Number* field equal to 0. The nodes receiving this packet set their tier number value to 1. Then, these nodes broadcast a packet including the fields given in Figure 5.1 with the *Tier Number* field set to 1. Nodes receiving packets with *Tier Number* 1, set their tier numbers to 2. This way, the sink has *Tier Number* equal to 0, the one-hop neighbors of the sink have *Tier Number* equal to 1, and so on.

To generalize, nodes receiving a broadcast packet with value *Tier Number* set their tier number value to $Tier\ Number + 1$ and after waiting for a certain time enough to receive the broadcast messages from all its neighbors from lower tier numbers, they broadcast their own message containing the fields given in Figure 5.1. When nodes receive broadcast messages from nodes with equal or higher tiers, they do not modify their tier number value but store the information about their neighbors (*Node ID*, *Residual Energy* and *Tier Number*). Figure 5.3 shows the tier numbers of each node after the first step of the algorithm.

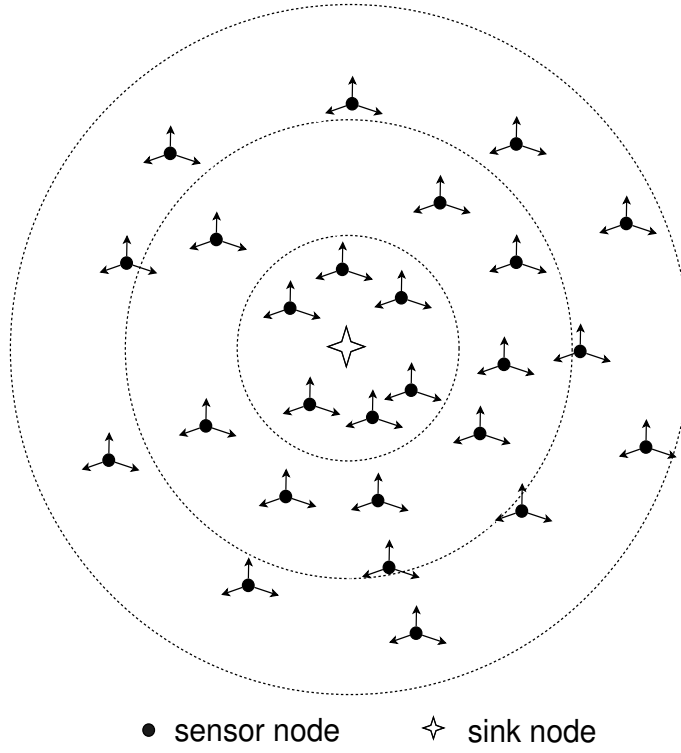


Figure 5.2: Illustrating first step of DASSA; Nodes broadcast their ID, tier number and remaining energy level.

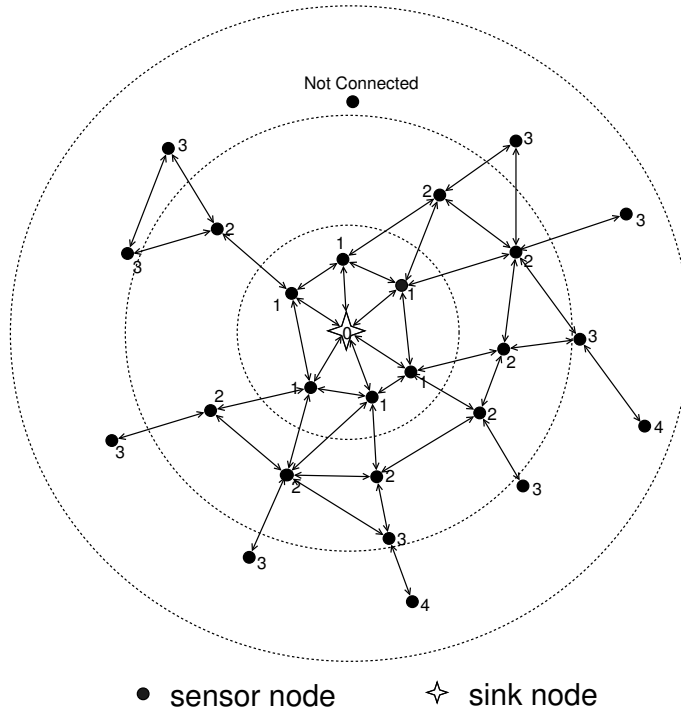


Figure 5.3: Tier numbers of the network in Figure 5.2 after the setup messages are exchanged. Nodes at the edges of a two sided arrow are in the transmission ranges of each other.

5.2 Step II : Scheduling Tier 1 Nodes

After each node discovers its local neighborhood, DASSA continues with the next step in which the sink decides which nodes from tier 1 will be operating by solving an integer linear program called *ILPSink*.

5.2.1 ILP implemented by the Sink (ILPSink)

After the neighbor discovery step ends, nodes in tier 1 broadcast the IDs of their neighbors belonging to tier 2. This broadcasting is only done in the first step of DASSA at the first round. Since we assume a static network, this process is not repeated at each round.

In this ILP formulation, the sink, after receiving the node IDs, energy levels and neighbor lists of tier 1 nodes, finds the minimum number of nodes from tier 1 with maximum remaining energy such that these nodes cover α percentage of the nodes in tier 2.

There are three objective functions defined for ILPSink. Depending on the network type, one can give better results than the others.

Parameters

For convenience, we will now describe the constants and variables used in ILPSink.

- α : Fraction of nodes to be covered from tier 2 by tier 1 nodes, where $0 \leq \alpha \leq 1$
- e_i : Remaining energy of node i
- E_{init} : Initial energy of every sensor node

- $\epsilon : 10^{-1}/E_{init}$
- T_i : The set of nodes in tier i
- C_i : Neighbors of node i from tier 2, $\forall i \in T_1$
- n : Size of T_2 , i.e., the number of nodes in tier 2
- c_{ij} : Coverage of tier 2 nodes from tier 1 nodes

$$c_{ij} = \begin{cases} 1, & i \in T_1, j \in T_2, j \in C_i \\ 0, & \text{otherwise} \end{cases} \quad \forall (i, j) : \{i \in [1, N], j \in [1, N]\} \quad (5.1)$$

- s_i : The sleep variable

$$s_i = \begin{cases} 1, & \text{if } i \in T_1 \text{ is active} \\ 0, & \text{if } i \in T_1 \text{ is sleeping} \end{cases} \quad \forall i \in [1, \text{size of } T_1] \quad (5.2)$$

- l_i : Indicates whether a node in tier 2 is at least in the neighbor list of one non-sleeping node from tier 1

$$l_i = \begin{cases} 1, & \text{if } i \in T_2 \text{ is in the neighbor list of at least one active node in } T_1 \\ 0, & \text{otherwise} \end{cases} \quad (5.3)$$

$\forall i \in [1, \text{size of } T_2]$

- w : Variable representing the node with minimum energy when objective 2 is used (see Eq. (5.9)).

Constraints

If at least one non-sleeping node from tier 1 has node j from tier 2 in its neighbor list, then this equation forces the corresponding variable l_j to be equal to one.

$$s_i \cdot c_{ij} \leq l_j, \quad \forall i \quad (5.4)$$

If none of the active nodes from tier 1 cover node j from tier 2, then l_j should be zero meaning that node j is not covered.

$$l_j \leq \sum_i s_i \cdot c_{ij}, \quad \forall j \quad (5.5)$$

The following equation determines what percentage of nodes from tier 2 will be covered by the operating nodes in tier 1.

$$\sum_i l_i \geq \alpha \cdot n \quad (5.6)$$

Objectives

There are three objective functions considered in this thesis which can be used by ILPSink. These are represented by the *Objective* parameter of DASSA, which can be equal to 1, 2 or 3 if the objective function chosen is given by Eq. (5.7), Eq. (5.8) or Eq. (5.10), respectively. The first objective function Eq. (5.7) tries to minimize the number of active nodes from tier 1 and to maximize the sum of the remaining energies of these active nodes. ϵ is used to give priority to the objective of minimizing the number of active nodes. The second objective function Eq. (5.8) tries to minimize the number of active nodes from tier 1 and to maximize the minimum energy of the node which is selected to be active from tier 1. Again ϵ is used for giving higher priority for minimizing the number of active sensors. Finally, the last objective function Eq. (5.10) is very similar to the first one except for a scaling factor. In this equation, the sum of the remaining energies of the selected active sensors from tier 1 is given more importance as compared to the first objective function. In our simulations, we observed that the third objective function generally gives the best results. However, the other two objective functions also give very close results and in some cases beat the third objective function.

1. Minimize

$$\sum_i s_i - \epsilon \sum_i s_i \cdot e_i \quad (5.7)$$

2. Minimize

$$\sum_i s_i - \epsilon w_i \quad (5.8)$$

Also the additional constraint

$$w_i \leq s_i \cdot e_i, \quad \forall i \quad (5.9)$$

3. Minimize

$$\sum_i (E_{init} - e_i) \cdot s_i \quad (5.10)$$

Note that, as the energy of a node from tier 1 reduces below a certain level such that it cannot participate in the network operation anymore and is dead, the sink learns this information from the broadcast messages at Step I. Accordingly, it reconstructs the list of tier 1 nodes which can continue operation and the list of tier 2 nodes that the new set of tier 1 nodes cover. This brings no extra cost since the sink knows which nodes in tier 2 are in the neighborhood of which nodes in tier 1 by the message exchange occurring in the first round, Step I of DASSA. Then, the sink solves ILPSink with its updated entries.

5.2.2 Transmitting the Schedules

After the sink finds the minimum number of nodes which will be active from tier 1 by ILPSink, it broadcasts a packet in the form of Figure 5.4 containing the node IDs of these nodes. When nodes from tier 1 receive this message, they operate in the current round if their IDs are included or otherwise sleep as shown in Figure 5.5.

Node ID 1	Node ID 2	· · · · ·	Node ID n
-----------	-----------	-----------	-----------

Figure 5.4: Broadcast packet transmitted by the sink to the nodes in tier 1. n represents the number of nodes which are selected to be active from tier 1.

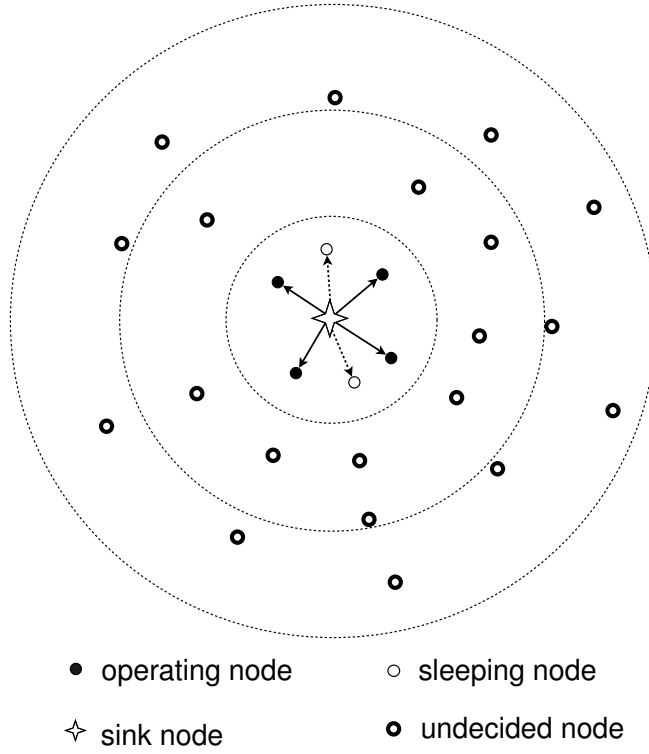


Figure 5.5: Illustrating second step of DASSA; The sink decides which nodes will operate from tier 1.

With this step, a balanced selection of operating nodes from tier 1 is provided without any location information. The reason for such a balanced choice of operating nodes from tier 1 is the requirement to cover at least some α percentage of nodes from tier 2 in the ILPSink formulation implemented in the sink (see Eq. (5.4) - (5.6)). A balanced structure such as in Figure 5.6 a) is provided rather than an undesired structure such as in Figure 5.6 b) with the help of the sink. Since the most critical nodes in such network configurations belong to tier 1, such a structure greatly enhances performance as compared to a random or other non-feedback based method when there is no location information. Also, the sink not only chooses a balanced structure but also chooses the active nodes from tier 1 which have the largest remaining energy. Since this procedure is applied at each round, the algorithm provides balanced energy consumption for tier 1 nodes. Recall that the sink was assumed to have abundant energy resources and high computational capabilities. Thus, using the sink in such a process is reasonable.

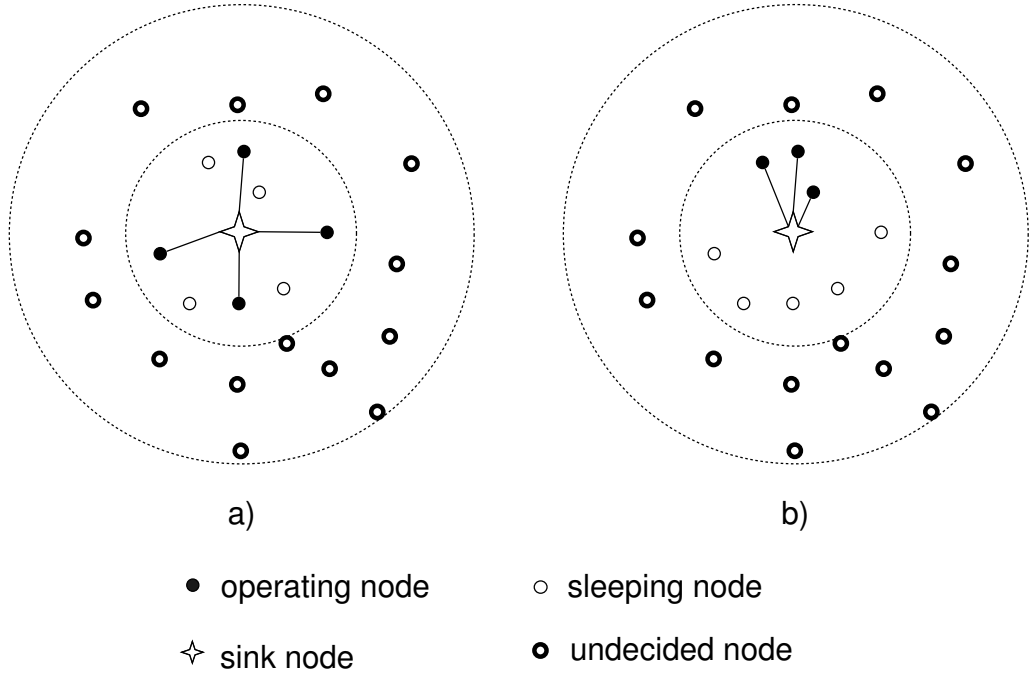


Figure 5.6: The balanced structure of DASSA. a) Balanced structure obtained by feedback from the sink when there is no location information. b) Unbalanced structure which could result without any feedback from the sink and when there is no location information.

5.3 Step III : Scheduling Intermediate Nodes

In the third step of the algorithm, nodes from tier 1 which were decided to be active in the previous step choose which nodes will be active from the next tier, i.e., tier 2, depending on the residual energy levels of these nodes. Every node in tier 1 broadcasts the node ID of its neighbor with the highest remaining energy (see Figure 5.7). After a node from tier 2 receives these messages, it decides to operate if any of the messages contains its node ID and decides to sleep if none of the messages contains its node ID.

In small scale networks, only the node with the greatest remaining energy is selected, whereas in larger networks, the first two or more nodes with the highest energies are selected to be active (recall that the energy information of the neighbor nodes was retrieved in the first step of the algorithm). This number is represented by the parameter *Number of Selected Descendants (NSD)* in the

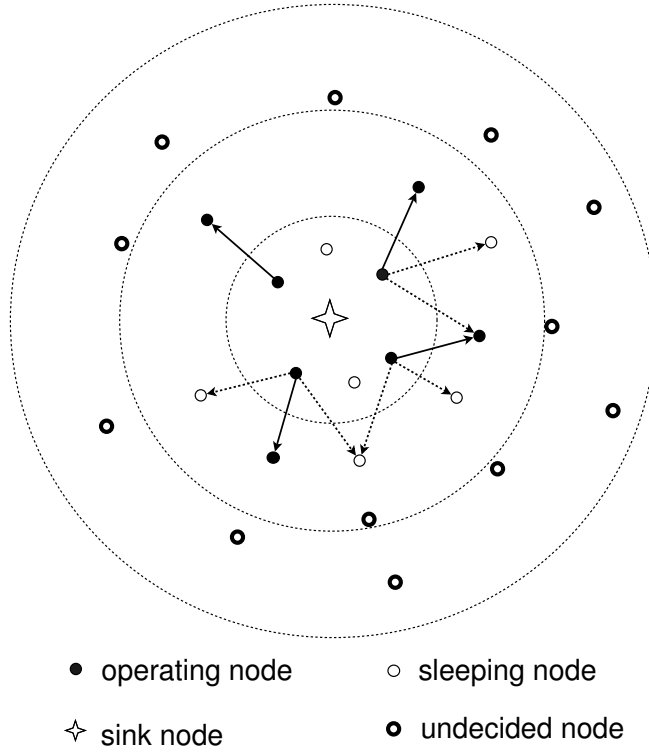


Figure 5.7: The third step of DASSA; Nodes which were decided to be active in the second step of SSA, schedule their neighbors from the next tier. Dashed arrows indicate that the received packet does not contain the receiver's node ID but intended to another node.

algorithm. Nodes from tier 1 broadcast a packet (see Figure 5.8), from which the nodes in tier 2 learn their sleep state. NSD parameter can be different for the nodes belonging to different tiers.

Node ID 1	Node ID 2	⋯ ⋯ ⋯	Node ID NSD
-----------	-----------	-------	---------------

Figure 5.8: Packet transmitted from the nodes in tier 1 to the nodes in tier 2. NSD is the node with NSD^{th} highest energy.

Nodes from higher tier numbers than 1 can also schedule nodes from their subsequent tiers to sleep or not to sleep. A parameter named *Adaptive Scheduling Depth* (ASD) is used to determine how further this process will continue. The nodes with tier number less than or equal to ASD schedule their neighbors from the next tier depending on the energies of these neighbors. In some network configurations, employing this structure only in the first tier gives good results

whereas in some network configurations, employing this structure up to the last tier gives better results. Nodes with tier number up to ASD transmit a packet to their subsequent tier with the structure given in Figure 5.8. Nodes in tier number $ASD + 1$, which are the last nodes which this scheduling mechanism goes up to, broadcast a packet if they will be active to the next tier since these nodes will not determine their subsequent tier's sleep schedule. This packet only contains the node ID of the node transmitting it. The nodes with tier number $ASD + 2$ will understand that the node with that node ID will not sleep from this broadcast message. The nodes which decide to sleep do not transmit such a packet.

In some cases, inactive nodes from tier 1 may also schedule nodes from tier 2. This will only reduce some energy from these nodes. The same argument can be extended to other tiers as well, e.g., all the nodes in tier 2, whether decided to be active or not, may schedule nodes from tier 3. Usually, employing such a procedure in only tier 1 nodes is sufficient. *Not Active Scheduling (NAS)* parameter determines whether this option is used ($NAS = 1$) or not used ($NAS = 0$) for tier 1 nodes.

DASSA schedules nodes with high residual energies to be active at each round. This way the energy consumption among nodes is balanced. For example, assume that we are at the beginning of the network operation. The energies of the active nodes will become lower than the energies of their neighbors which are scheduled to sleep. Thus, in the next round, one of the neighbors which was sleeping in the previous round will be selected to be active since its energy will be higher. Thus, the active role will be rotated among the nodes and the energies of the nodes will be consumed in a balanced manner.

5.4 Step IV : Scheduling Far Away Nodes

In the last step of scheduling, nodes with tier numbers greater than or equal to $ASD + 2$ randomly decide whether to sleep or not to sleep. They generate a random number uniformly distributed in the interval $[0, 1]$ and then compare this number with p_s . If the number is less than p_s , they sleep, otherwise, they decide to operate. After making a decision, the non-sleeping nodes broadcast a packet containing their node IDs. Nodes from higher tier numbers use this information in the network layer to route packets to non-sleeping nodes.

5.5 Step V : Transmitting and Forwarding Data

After each node decides its activity state, it forwards its data and its neighbors' data according to a simple routing procedure which will be explained next. Nodes forward their data to their neighbors from the previous tier which has the highest remaining energy. In addition to the other steps in which nodes with higher remaining energies are scheduled to be active, this step ensures that nodes with the maximum energies have higher loads. So, we apply a two-fold energy balancing scheme; both in scheduling and in routing. Figure 5.9 illustrates the last step of DASSA.

The flowchart of the DASSA is provided in Figure 5.10. Every node except the sink runs this algorithm at the beginning of every round and then senses and forwards data or sleeps for a round.

An example operation of DASSA for a 200 node network deployed in a 200m-by-200m field with R_t and R_s equal to 50m is provided in Figure 5.11. The nodes which are dead are marked with X. The empty circles are scheduled to sleep and the black circles are scheduled to be active at the corresponding round. At the last round, since all the tier 1 nodes connecting the sink to other nodes

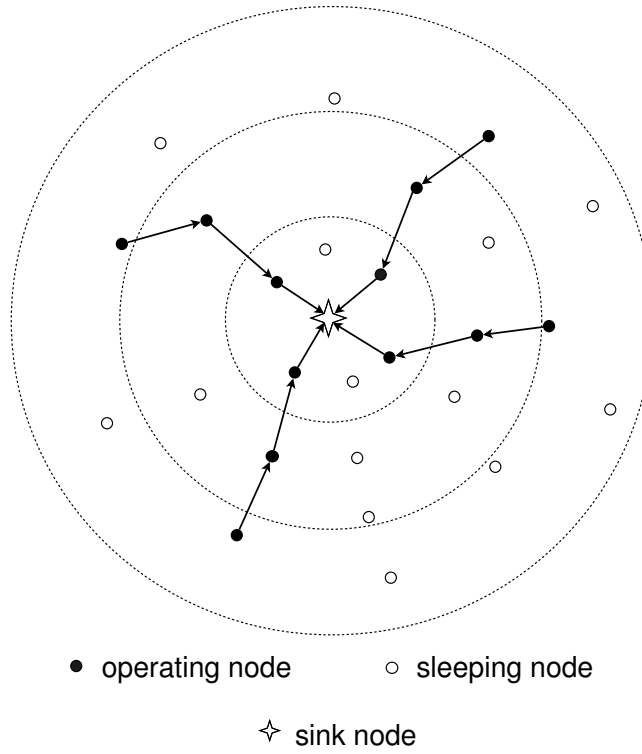


Figure 5.9: The final step of DASSA. Nodes which decide to operate forward their data to the sink.

are dead, network operation ends although many of the nodes are still alive. This reemphasizes the importance of the scheduling of tier 1 nodes.

In the following chapter, the performance of DASSA is evaluated and compared with the optimum sleep scheduling algorithm, an oblivious scheduling algorithm and an existing algorithm in the literature.

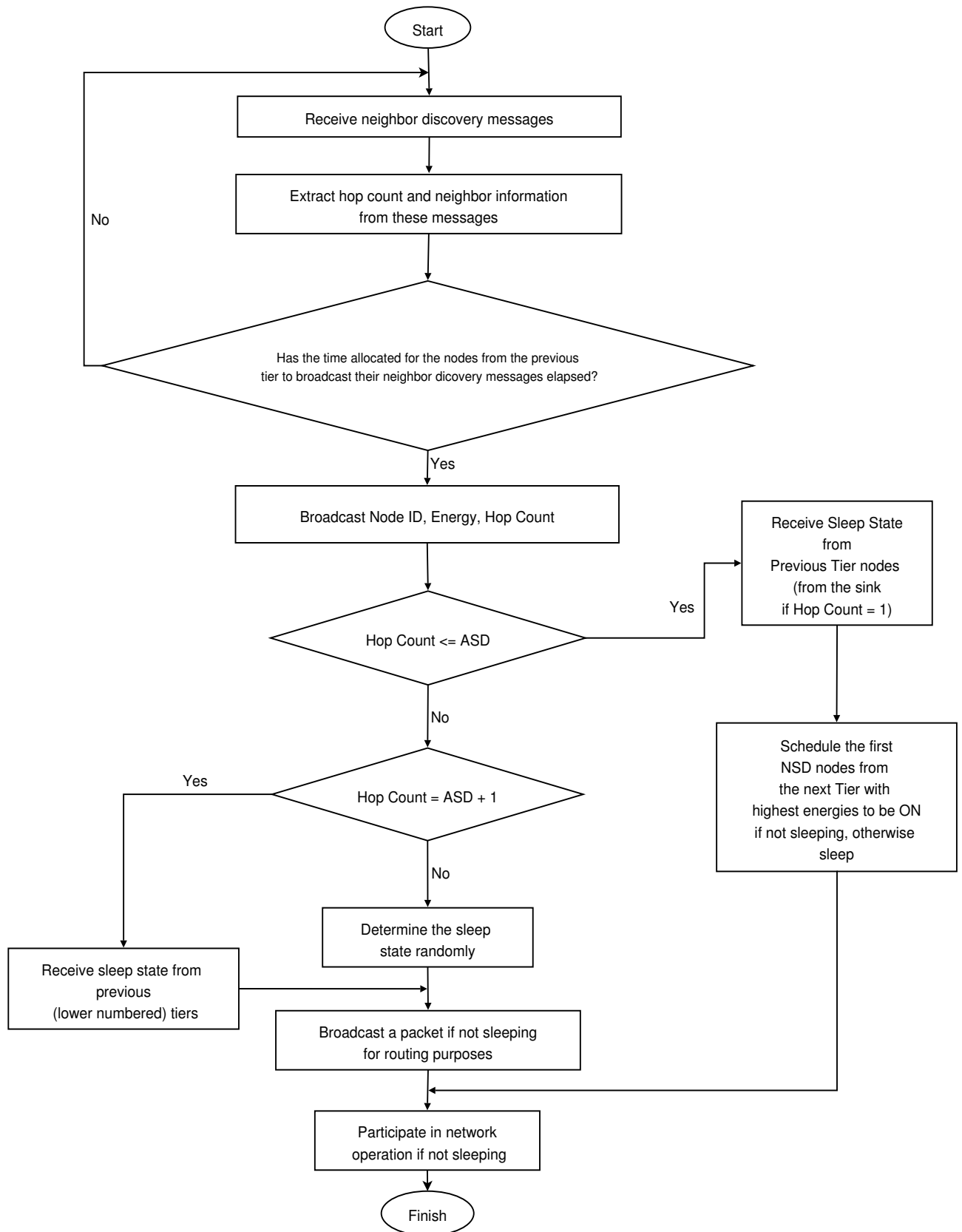
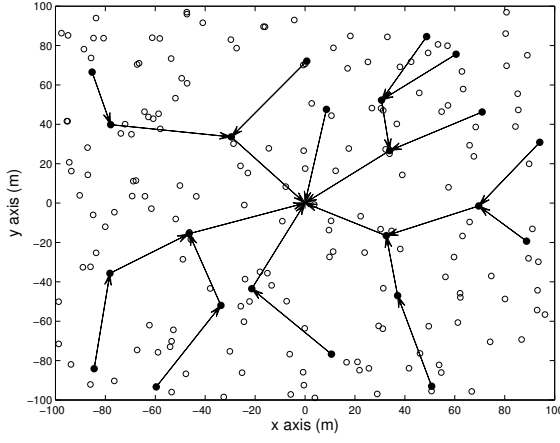
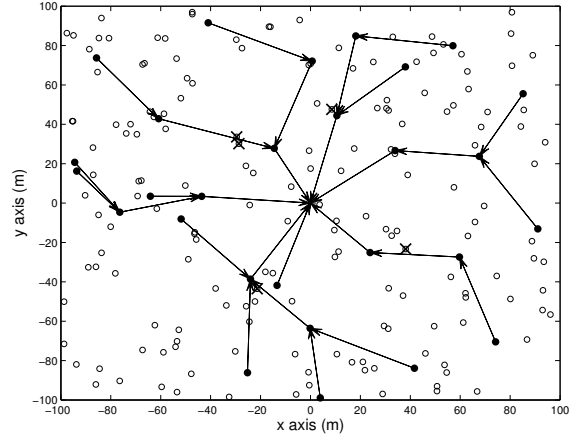


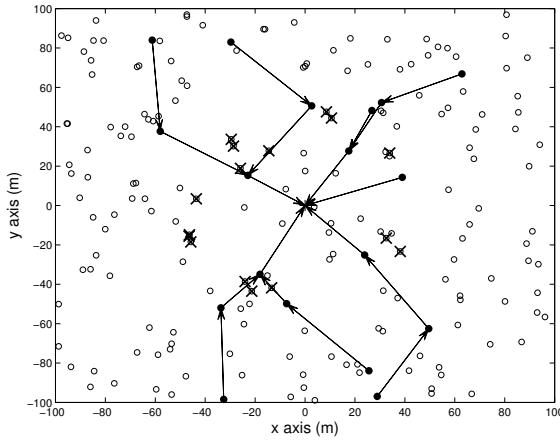
Figure 5.10: Flowchart of DASSA. This algorithm is implemented by every node at the beginning of each round.



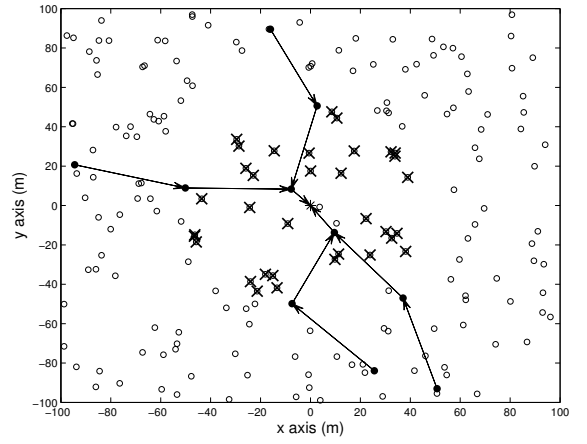
(a)



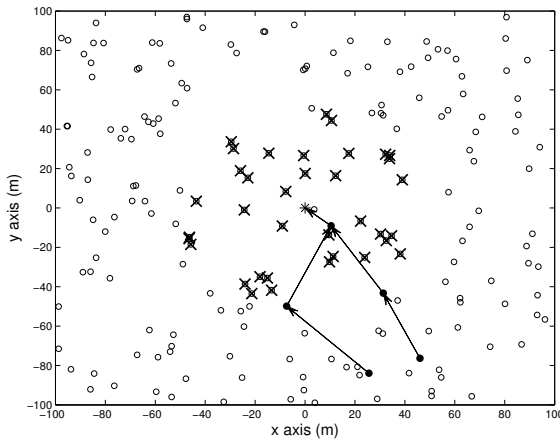
(b)



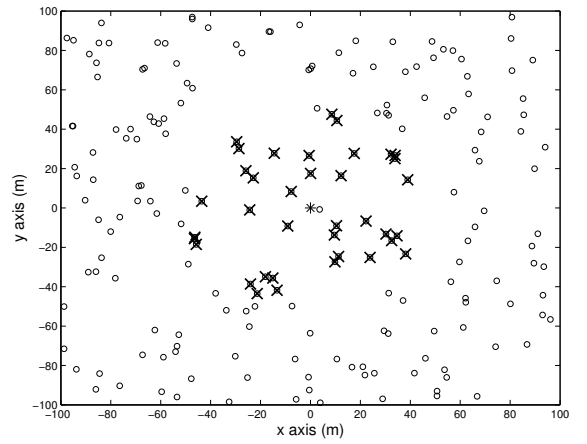
(c)



(d)



(e)



(f)

Figure 5.11: Operation of DASSA for a sample network. $\alpha = 0.95$, $ASD = 2$, NSD for tier 1 = 2, NSD for tier 2 = 1, $p_s = 1$, Objective = 3, $NAS = 0$. (a) Round 6, Coverage = 0.95. (b) Round 30, Coverage = 0.96. (c) Round 51, Coverage = 0.92. (d) Round 87, Coverage = 0.72. (e) Round 96, Coverage = 0.41. (f) Round 99, Coverage = 0.

Chapter 6

Performance Evaluation of Sleep Scheduling Algorithms

In this chapter, the performance of DASSA is investigated and compared with the following algorithms: ILP based optimum algorithm, a second scheduling algorithm we propose, the algorithm in [46] (both the original version and the corrected version) and an algorithm which employs no sleep scheduling.

First, the results are compared for the no aggregation case for several network topologies where R_t and R_s are equal. Then, the results for unequal R_s and R_t are provided. The performances of the algorithms are also compared for different network sizes and different node populations. Finally, the results for the full aggregation case will be presented.

Now, the scheduling algorithms that will be compared in this chapter are summarized.

No Sleeping It is assumed that none of the nodes sleep and they all are sensing and transmitting data at every round. After using exchange messages as in step I of DASSA (see Section 5.1), nodes send data to their neighbor with the highest remaining energy and which has less hop count to the

sink. This algorithm provides the number of rounds that the network can sustain without any sleep scheduling algorithm.

Optimum The algorithm finding the optimum results is centralized and knows the locations and remaining energies of all the nodes in the network. By using this information in an ILP based algorithm as described in Chapter 4, the maximum possible number of rounds that can be achieved is computed. This provides us with an upper bound on evaluating our proposed algorithm.

DASSA This is a distributed, simple and energy efficient algorithm which is proposed and described in Chapter 5.

OSSA In *Oblivious Sleep Scheduling Algorithm* (OSSA), nodes are chosen randomly with some probability p to sleep in each round oblivious of the states of neighboring nodes. However, after deciding their sleep schedules, nodes forward data using exchange messages as in step I of DASSA (see Section 5.1) and the procedure described in step V of DASSA (see Section 5.5), i.e., nodes send data to their neighbor with the highest remaining energy and which has less hop count to the sink. Therefore, only the scheduling is random, but the routing is performed in an energy efficient manner. Also, p is optimized for best performance.

DRS This algorithm is proposed in [46] in its original form. As mentioned earlier, the algorithm suffers from routing loops which makes it impractical to implement. In order to provide the results, it is assumed that the nodes in a loop somehow learn this situation and do not spend any energy. Only the data sent from the nodes not in routing loops is counted in calculating the coverage.

EDRS This algorithm is the algorithm proposed in [46] with the correction we proposed to avoid loops.

6.1 No Aggregation

The results presented in this section are for the case when there is no aggregation scheme applied, i.e., nodes forward every packet they receive without any processing. This is more practical than full aggregation, which will be evaluated later, since in most cases the data of many individual sensors are required and aggregating the data requires high processing capabilities which may not be available at each sensor node.

Figure 6.1 shows the network topologies of the five networks used for the comparisons. The first 4 network consist of 100 nodes and the fifth network consists of 150 nodes. The nodes are uniformly deployed in a 200m-by-200m network field. There is a sink located at the origin point (0,0) of the field for all deployments to which the nodes scheduled to be active forward their sensed data. The optimum results obtained for these networks will give us a clear idea of the performance of our heuristic algorithm. Larger networks with various field dimensions and node populations are analyzed later. For finding the optimum solutions, the CPLEX solver [50], which implements the simplex algorithm to solve linear problems, is used.

We use a similar energy consumption model with [4] and [14]. Each sensor reports a 2000-bit report message to the sink at each round and each message transmission and reception consumes 0.1mJ. The exchange messages used at the beginning of each round are 200 bits long and cost $10\mu\text{J}$. Receiving a sleep control message which is 100 bits long costs $5\mu\text{J}$ and transmitting a sleep control message which is 100 bits long costs $5\mu\text{J}$. If E_{init} is chosen too low, the comparisons will not be very accurate since the network will die very quickly. On the other hand, if E_{init} is chosen too large, the optimum results will require very long run times. We observed that $E_{init} = 10\text{mJ}$ is a good choice in terms of accuracy and run times for the first five topologies.

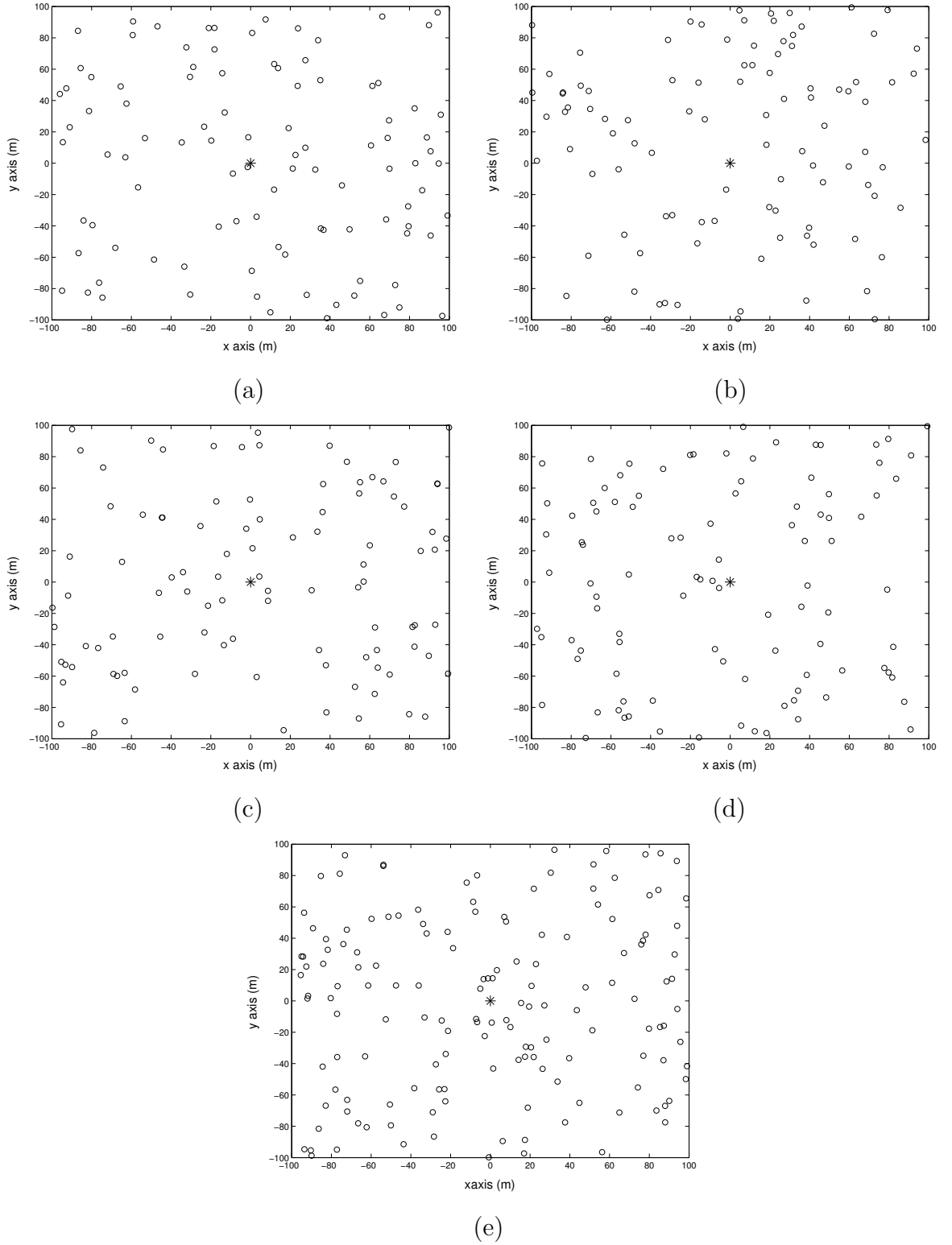


Figure 6.1: Network topologies used in the simulations. (a) Topology 1. (b) Topology 2. (c) Topology 3. (d) Topology 4. (e) Topology 5. (a)-(d) have 100 nodes and (e) has 150 nodes.

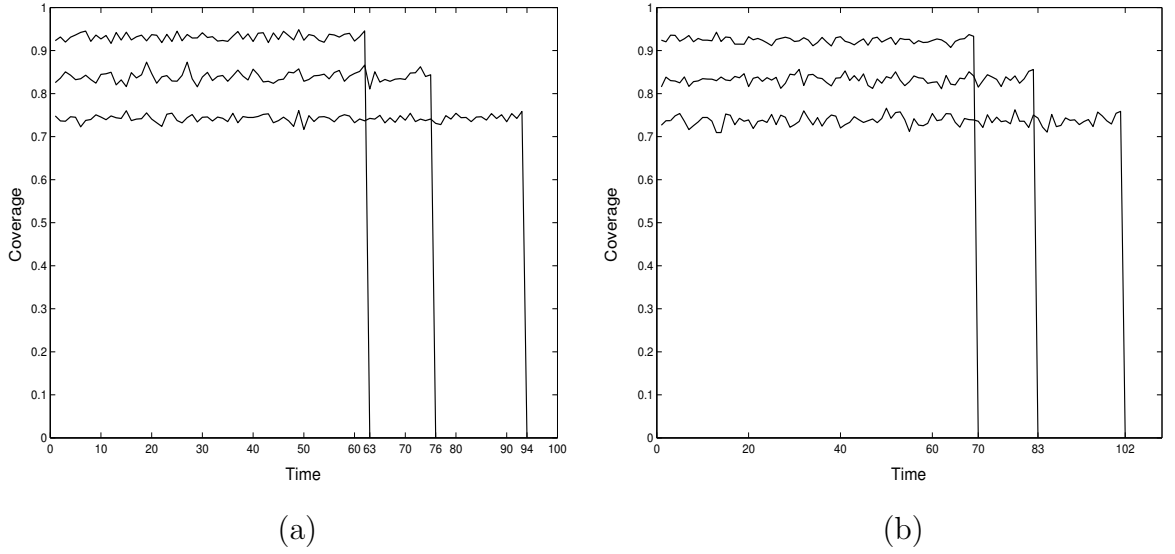


Figure 6.2: Optimum Results for (a) Topology 1. (b) Topology 2.

For the first comparisons, $R_t = 50\text{m}$ and $R_s = 50\text{m}$. In Section 6.1.2, the results for unequal transmission and sensing ranges are also presented.

Figure 6.2 shows the optimum coverage plots for $\text{GoC} = 0.9, 0.8$, and 0.7 for Topology 1 and Topology 2. As expected, the number of rounds the network can sustain increases as GoC decreases. The optimum algorithm holds the coverage above GoC in all rounds before the final round and then suddenly the network operation stops. The values in the graphs are slightly above $0.9, 0.8$, and 0.7 since the resolution used at the linear program was 1 grid per 4 meters for feasible run times, and the coverage values were calculated for 1 grid per 1 meter for better accuracy.

For OSSA, we report the results obtained with $p = p^*$ where p^* corresponds to the optimum choice of p which maximizes the number of rounds for which GoC constraint is satisfied. For Topology 1, Figure 6.3 illustrates the average lifetime with respect to p . Increasing p above a certain level causes the network to become disconnected and show poor performance. For this topology, $p^* = 0.60, 0.70, 0.76$ for $\text{GoC} = 0.9, 0.8$ and 0.7 , respectively. Note that OSSA is both

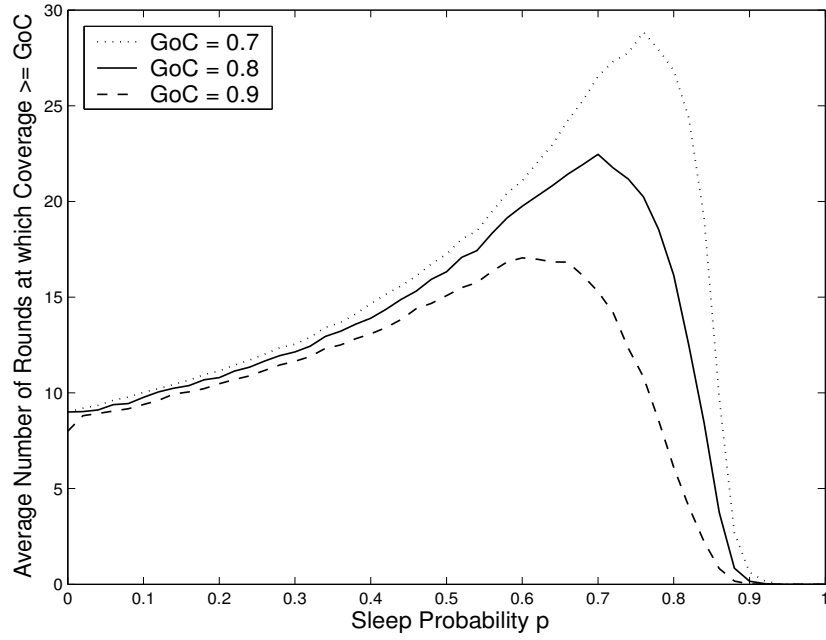


Figure 6.3: The optimum p for OSA for Topology 1.

optimized and uses an energy efficient routing algorithm at each round which makes it a good algorithm for comparison purposes.

Figures 6.4, 6.5 and 6.6 show the coverage plots of DASSA and OSA together with the optimum results for $\text{GoC} = 0.9, 0.8$ and 0.7 , respectively, for Topology 1. As GoC decreases, the gap between DASSA and the optimum result decreases since the importance of exact location information decreases as GoC decreases. Actually, for $\text{GoC} = 0.8$ and 0.7 , the performance of DASSA is very close to the optimum results. Also, it is important to note that DASSA provides a balanced coverage plot in which the coverage maintains almost all the time above GoC up to the time where it decreases below GoC . However, in OSA, the coverage values can fall well below GoC even at the first rounds due to randomness. Furthermore, in DASSA, the coverage is held very close to GoC , whereas in OSA the coverage value is above GoC most of the time.

Figures 6.7, 6.8 and 6.9 show the coverage plots of DASSA, optimum algorithm and OSA for the second topology. The results are very similar when compared with the results for the Topology 1. Again, OSA does not provide

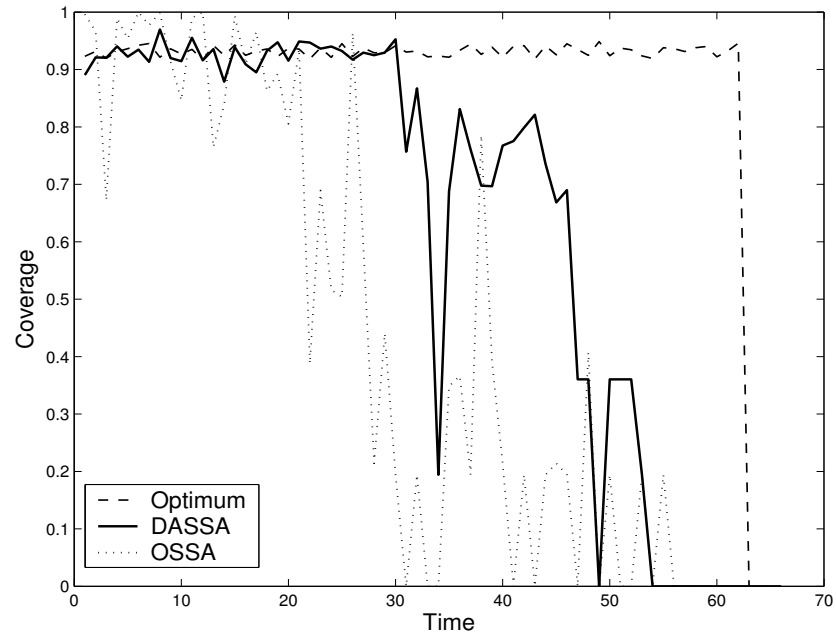


Figure 6.4: DASSA, OSSA and the optimum results for GoC = 0.9, Topology 1.

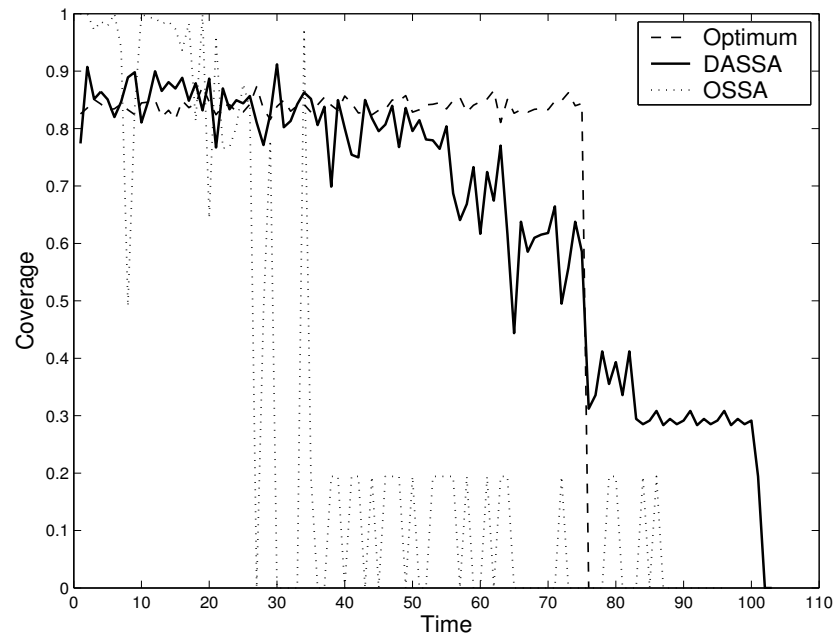


Figure 6.5: DASSA, OSSA and the optimum results for GoC = 0.8, Topology 1.

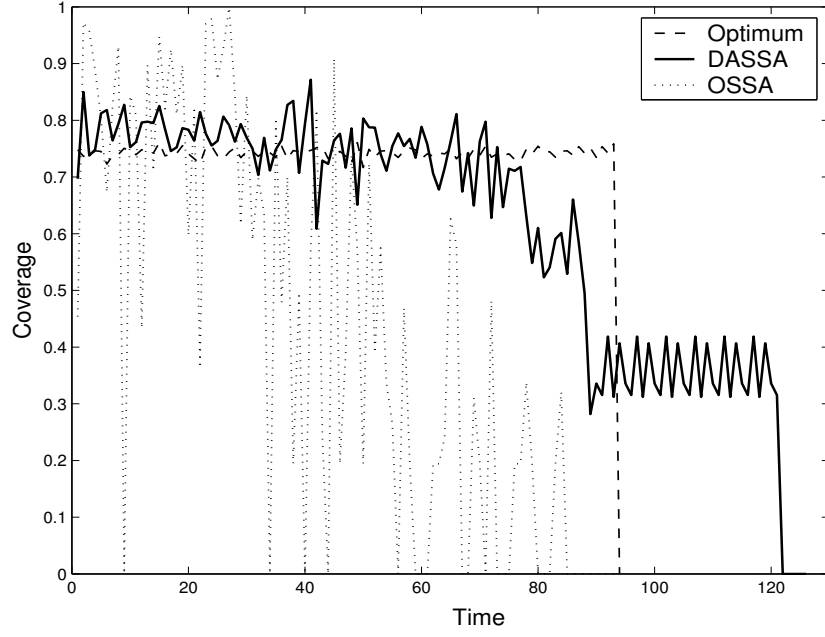


Figure 6.6: DASSA, OSSA and the optimum results for GoC = 0.7, Topology 1.

a balanced coverage plot, whereas, DASSA keeps the coverage percentage above GoC almost in all rounds up to which the coverage drops below GoC for the first time. In OSSA, the coverage may drop very low, even to %0, which means that the network cannot provide any data to the sink. DASSA does not suffer from such a problem since the nodes are selected as to assure connectivity and sufficient coverage. Also, the performance of DASSA is quite remarkable for Topology 2 for GoC = 0.7.

Before presenting further results, we define *Effective Coverage* (EC) as the area under the coverage plot where the coverage percentage is above GoC.

$$EC = \sum_k c'_k \quad (6.1)$$

where

$$c'_k = \begin{cases} 0, & c_k < \text{GoC} \\ c_k, & c_k \geq \text{GoC} \end{cases} \quad (6.2)$$

c_k is the coverage percentage at round k .

The results of all algorithms are presented in Table 6.1 for Topology 1. The GoC-L column shows the number of rounds for which the coverage is above GoC,

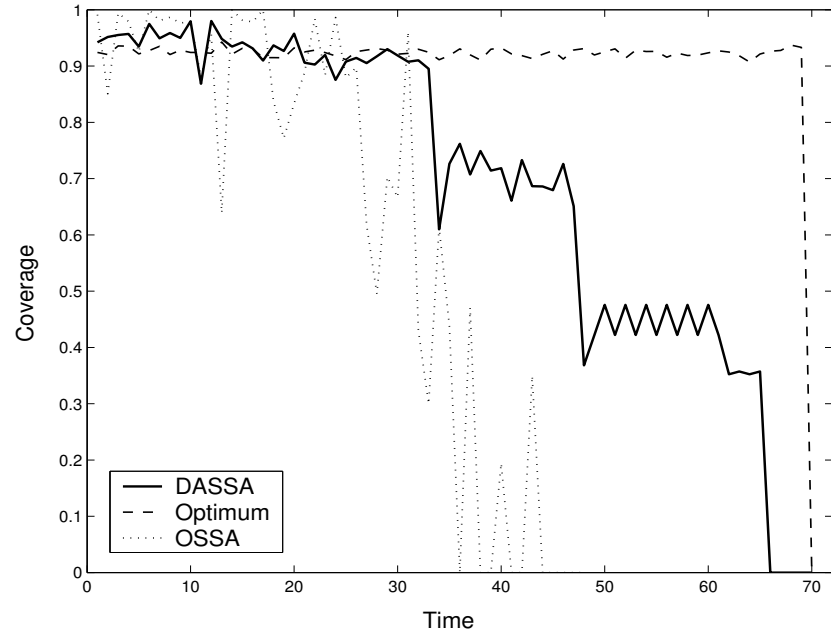


Figure 6.7: DASSA, OSSA and the optimum results for $\text{GoC} = 0.9$, Topology 2.

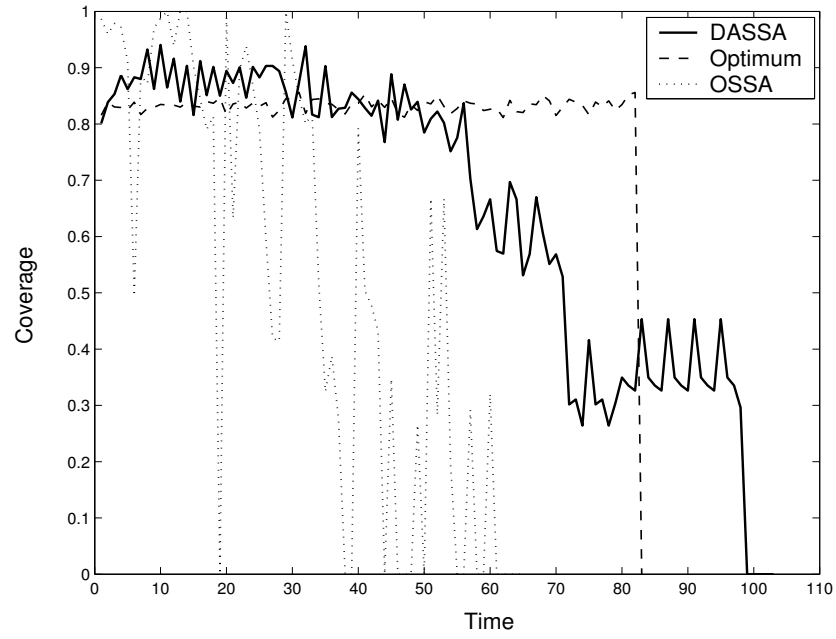


Figure 6.8: DASSA, OSSA and the optimum results for $\text{GoC} = 0.8$, Topology 2.

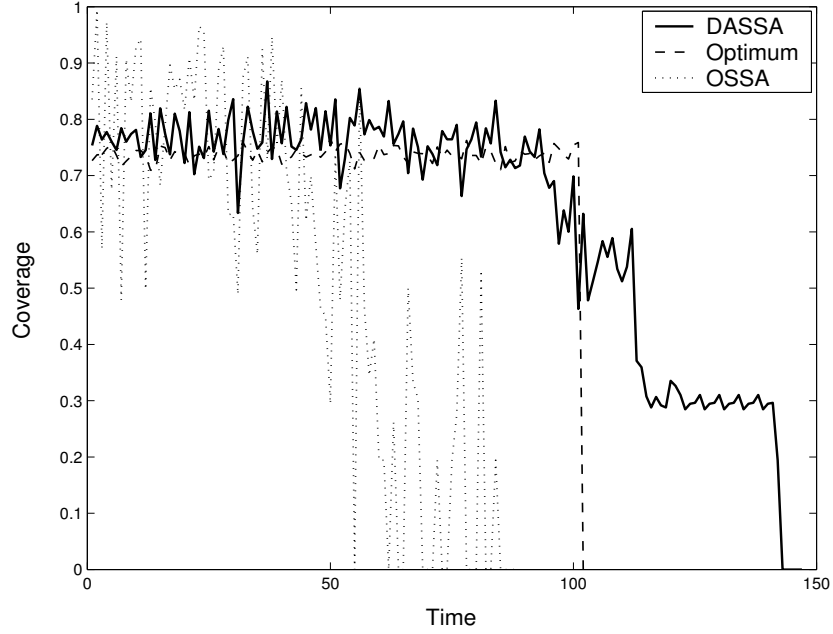


Figure 6.9: DASSA, OSSA and the optimum results for GoC = 0.7, Topology 2.

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	9	889	89.1	9	889	89.1	8	889	96.2
Optimum	93	6901	8.9	75	6284	10.8	62	5778	12.7
DASSA	69	5306	10.2	43	3647	12.6	27	2516	20.4
OSSA	28.9	2483	21.2	22.3	2087	29.5	17.0	1657	35.7
DRS	22.1	1805	13.1	16.9	1506	16.3	11.2	1066	21.7
EDRS	24.2	1993	13.4	17.7	1585	16.6	12.7	1215	21.9

Table 6.1: Results of a 100 node network (Topology 1) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field. $E_{init} = 10\text{mJ}$.

EC is calculated from Eq. (6.1) and *Number of Reporting Nodes* (NRN) is the average number of nodes whose packets reach the sink for the rounds where coverage is above GoC. For the no sleeping case, NRN is below 100 since the coverage is still above GoC when some of the nodes die.

Table 6.1 shows that DASSA outperforms OSSA, and DASSA and OSSA outperform DRS and EDRS. Recall that, GoC-L is the number of rounds for which the coverage is above GoC. Although the results of DRS and EDRS are similar, one must use EDRS since DRS has routing loops and if these loops exist the initial aim of [46] cannot be reached. There is a remarkable gain in DASSA

when compared with the no sleeping case. OSSA is an adaptive algorithm since it chooses nodes with highest remaining energies as next hops, whereas DRS and EDRS do not use energy information. Therefore, OSSA achieves much better performance. The subset of active nodes are chosen randomly for each round in DRS and EDRS at the beginning of the algorithm and a node reporting in a round does not report in other rounds in a cycle if it is not required for connectivity purposes. Usually, the set of nodes randomly selected to be active at only one round is not sufficient to provide GoC which accounts for the poor performance of DRS and EDRS.

As GoC decreases, the gap between DASSA and the optimum algorithm decreases, whereas the gap between DASSA and other algorithms increases even more. The reason is that as GoC decreases, OSSA, DRS and EDRS need to open more nodes in order to assure connectivity. In other words, as GoC decreases, the necessary number of nodes to assure GoC also decreases but in order to provide connectivity of these randomly selected nodes, the probability of sleeping cannot be increased very much. For OSSA, this can also be observed from Figure 6.3 where p^* does not increase very much when GoC decreases from 0.9 to 0.7. Therefore, the random algorithms cannot benefit from the fact that less nodes should be activated for less GoC values. However, DASSA only activates the necessary number of nodes for satisfying GoC and these nodes are connected due to the nature of the algorithm. As GoC decreases, the importance of the locations of the nodes becomes less important and thus DASSA can perform better as compared to the optimum algorithm.

NRN column is important for evaluating the performance of the algorithms. For $\text{GoC} = 0.8$ and 0.7 , DASSA manages to keep the number of active nodes close to the optimum number. However, for $\text{GoC} = 0.9$, since DASSA does not have location information, it opens some redundant nodes in order to assure GoC at each round. Also, OSSA, DRS and EDRS suffer from the same problem. Yet,

DASSA keeps the number of active nodes closer to the optimum number when compared with the other algorithms.

The EC column shows us how well the coverage behaves with time since it is the sum of the coverage values at the rounds for which coverage is above GoC. Since this value is directly related to the number of rounds, same comments made for the number of rounds are valid. As GoC decreases, the area comes close to the optimum case, thus DASSA performs closer to the optimum. Also, from EC, we can comment on how much is the coverage percentage held above GoC. For example, for $\text{GoC} = 0.9$, EC of DASSA is 2516 and the number of rounds is 27. Thus, the average coverage per round is $2516/27 = 93.2$ whereas for OSSA the same value is $1657/17 = 97.5$. We observe that OSSA cannot maintain a coverage value close to the desired GoC.

Tables 6.2, 6.3 and 6.4 show the results of the algorithms for Topology 2, 3 and 4, respectively. Similar results are obtained as in Topology 1. As long as the nodes are densely deployed so that they are scattered in a balanced way in the area, such as in a uniform distribution, the algorithm gives similar results as compared to the optimum case. For these tables, DASSA is individually tuned for each topology, i.e., we tune the parameters of the algorithm for the best performance for the given topology, in order to compare with the optimum algorithm. In other words, the best possible performance of DASSA is compared with the optimum solution. Also, the performance of DASSA when it is commonly tuned for every topology is evaluated. The parameters of DASSA is tuned for a 100 node network deployment. Table 6.5 shows the performance of DASSA for this case. There is a little drop in the performance of DASSA. However, the gain is still high with respect to OSSA and for GoC lower than 0.9, the gain is remarkable.

Figure 6.10 shows the percentage of GoC-L obtained by DASSA and by OSSA with respect to the optimum GoC-L. We observe that, both of the algorithms

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	10	975	85.6	10	975	85.6	9	892	91.0
Optimum	101	7449	8.6	82	6830	10.4	69	6376	12.8
DASSA	90	6930	9.0	52	4476	14.3	30	2806	22.4
OSSA	32.5	2757	19.7	25.2	2332	27.6	18.3	1773	33.0
DRS	23.9	1940	12.5	17.1	1507	15.5	10.9	1034	21.2
EDRS	24.3	1972	12.6	17.3	1522	15.8	11.1	1047	21.2

Table 6.2: Results of a 100 node network (Topology 2) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field. $E_{init} = 10\text{mJ}$.

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	13	1252	82.2	12	1175	85.2	11	1090	88.8
Optimum	109	8017	9.1	91	7581	10.8	77	7147	12.8
DASSA	75	5713	9.4	50	4322	14.7	31	2903	21.0
OSSA	33.0	2764	19.4	25.3	2331	27.7	18.1	1770	36.7
DRS	26.7	2191	13.0	18.8	1677	16.3	13.5	1292	21.5
EDRS	30.0	2476	13.2	21.6	1916	16.2	14.7	1403	21.6

Table 6.3: Results of a 100 node network (Topology 3) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field. $E_{init} = 10\text{mJ}$.

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	9	863	76.4	8	786	76.4	8	786	79.4
Optimum	84	6215	8.9	67	5596	10.9	54	4997	13.8
DASSA	61	4705	8.9	34	2985	15.0	23	2154	21.4
OSSA	22.5	1968	23.0	17.0	1593	28.1	13.3	1299	34.3
DRS	23.4	1937	13.1	16.5	1481	16.3	12.1	1156	21.4
EDRS	25.8	2160	13.4	19.3	1543	16.5	13.2	1260	21.7

Table 6.4: Results of a 100 node network (Topology 4) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field. $E_{init} = 10\text{mJ}$.

Top. #	Algorithm	GoC = 0.7			GoC = 0.8			GoC = 0.9		
		GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
1	Optimum	93	6901	8.9	75	6284	10.8	62	5778	12.7
	DASSA	66	4932	8.8	43	3647	12.6	22	2058	21.3
	OSSA	28.1	2374	19.7	21.7	2025	28.1	16.7	1630	34.9
2	Optimum	101	7449	8.6	82	6830	10.4	69	6376	12.8
	DASSA	85	6576	9.2	45	3890	14.0	24	2251	19.3
	OSSA	32.5	2757	19.7	25.2	2332	27.6	18.1	1757	34.6
3	Optimum	109	8017	9.1	91	7581	10.8	77	7147	12.8
	DASSA	71	5525	10.3	50	4322	14.7	25	2331	20.7
	OSSA	33.0	2764	19.4	25.3	2331	27.7	17.9	1741	34.8
4	Optimum	84	6215	8.9	67	5596	10.9	54	4997	13.8
	DASSA	56	4428	9.6	31	2759	15.8	20	1902	21.6
	OSSA	22.1	1883	19.9	17.0	1593	28.1	13.3	1309	35.1

Table 6.5: Performance of the algorithms when parameters are same for all topologies.

show a steady behavior. DASSA achieves almost two times GoC-L when compared to OSSA and the performance of the algorithm with respect to the optimum case does not change much with topology. Figure 6.10 (a) shows the performance when DASSA is individually tuned for each topology and Figure 6.10 (b) shows the performance of DASSA when it uses common parameters for every topology. There is some drop in the performance for $\text{GoC} = 0.9$ but DASSA still performs better than OSSA. For, $\text{GoC} = 0.8$ and 0.7 , DASSA performance does not drop much and there is noticeable gain with respect to OSSA. The parameters of DASSA and OSSA used in obtaining the results are given in Tables A.1 and A.2, respectively.

When the number of nodes in the network is increased, the performance of DASSA is not affected much. Table 6.6 shows the performance of all algorithms for a 150 node network deployed in a 200m-by-200m field which corresponds to Topology 5. Therefore, we can say that the performance of DASSA is not affected by node population as long as the network is dense.

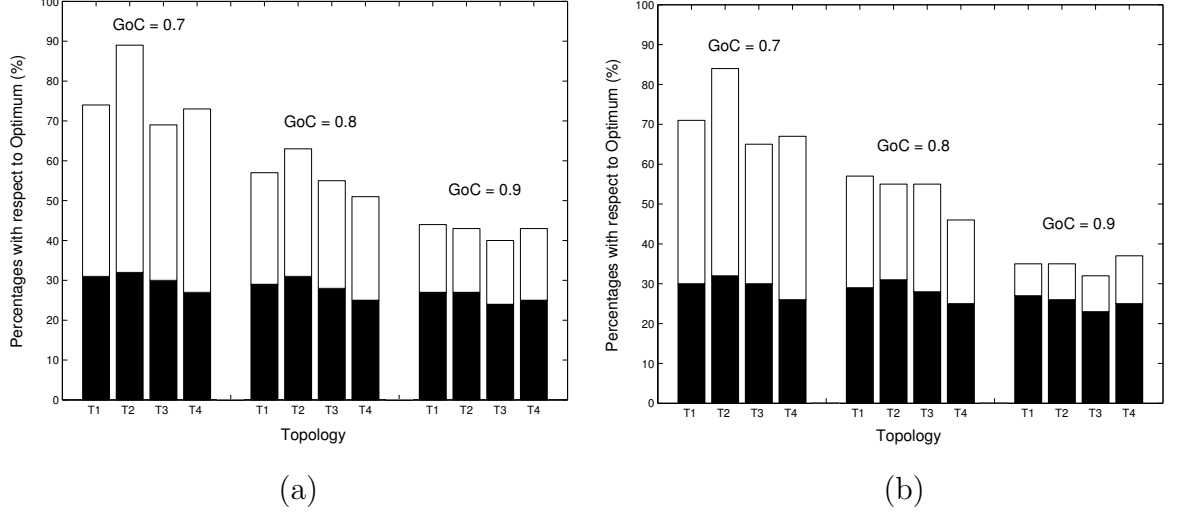


Figure 6.10: Black bars are the percentages of the number of rounds obtained by OSSA and the white and black bars together are the percentages of the number of rounds obtained by DASSA with respect to the number of rounds obtained by optimum case. T1, 2, 3, 4 represent Topology 1, 2, 3, 4, respectively. a) DASSA and OSSA are individually optimized for each topology. b) Same parameters are used in DASSA and OSSA for every topology.

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	16	1559	114.3	15	1483	119.8	15	1483	119.8
Optimum	172	12662	8.2	140	11657	10.4	120	11101	12.9
DASSA	125	9552.4	9.5	80	7076	14.8	49	4410	23.5
OSSA	61.2	5241	22.1	45.2	4214	30.2	33.7	3297	38.8
DRS	44.7	3641	12.8	31.2	2775	16.4	19.0	1808	20.6
EDRS	47.3	3872	13.0	33.2	2967	16.5	20.7	1969	20.8

Table 6.6: Results of a 150 node network (Topology 5) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field. $E_{init} = 10\text{mJ}$. The parameters used for DASSA and OSSA are given in Table A.3.

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	11.7	1148	163.0	11.4	1126	166.5	11.1	1107	171.7
DASSA	129.8	10065	9.0	86.2	7490	14.0	55.7	5340	22.1
OSSA	67.6	5807	21.4	52.4	4830	26.4	39.7	3886	38.2
DRS	40.7	3294	12.4	31.9	2819	15.8	19.7	1872	20.8
EDRS	44.6	3611	12.5	32.8	2910	15.9	20.9	1983	20.8

Table 6.7: Results of a 200 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field. All values are the averages of the results of the algorithms for 100 random (uniformly distributed) deployment of the network and $E_{init} = 10\text{mJ}$. The parameters used for DASSA and OSSA are given in Table A.4.

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	109.7	10933	191.3	109.3	11386	192.6	108.8	10867	193.3
DASSA	1377.2	108158	9.6	937.5	81987	14.8	623.8	60023	23.3
OSSA	748.5	65073	22	635.0	58799	27.1	473.7	46464	39.3
DRS	449.5	36312	12.4	330.4	29252	16.0	211.7	20111	20.6
EDRS	427.7	34702	12.6	330.8	29311	16.0	220.8	20968	20.7

Table 6.8: Results of a 200 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field. All values are the averages of the results of the algorithms for 100 random (uniformly distributed) deployment of the network and $E_{init} = 100\text{mJ}$. The parameters used for DASSA and OSSA are given in Table A.4.

For a 200m-by-200m field with 200 nodes, 100 network deployments are simulated and the performance of DASSA and other algorithms are compared for two different initial energy values of the nodes: $E_{init} = 10\text{mJ}$ and $E_{init} = 100\text{mJ}$. Note that the parameters of the algorithms used for each deployment are the same. Table 6.7 shows that DASSA performs much better than OSSA and the other algorithms. When E_{init} is 100mJ (see Table 6.8), the results are similar with the $E_{init} = 10\text{mJ}$ case.

Finally, for a 300m-by-300m field with 400 nodes, 100 network deployments are simulated and the performance of DASSA and other algorithms are compared for $E_{init} = 50\text{mJ}$. Again, the parameters of the algorithms are optimized once and these same parameters are used in every deployment. Table 6.9 shows that DASSA performs better than OSSA, and DASSA and OSSA perform much better

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	23.6	2324	358.8	22.8	2265	365.0	22.2	2214	370.6
DASSA	203.8	16247	31.7	124.0	11212	55.0	89.0	8559	67.4
OSSA	122.5	10457	49.6	102.3	9605	67.8	82.8	8056	78.1
DRS	54.8	4363	26.8	43.7	3829	33.8	33.4	3156	44.5
EDRS	76.7	6166	27.4	59.5	5240	34.1	41.8	3160	45.3

Table 6.9: Results of a 400 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 300m-by-300m field. All values are the averages of the results of the algorithms for 100 random (uniformly distributed) deployment of the network and $E_{init} = 50mJ$. The parameters used for DASSA and OSSA are given in Table A.5.

than DRS and EDRS. As the field size increases with R_t and R_s being constant, the number of tiers increases. In DASSA, nodes in the first tier are selected in a balanced manner by using the feedback from sink. However, as the number of tiers increases, the randomness in DASSA increases due to the lack of location information. Thus, DASSA starts to open redundant nodes in order to assure good performance at each deployment and because of this, its performance gain with respect to OSSA decreases. Yet, there is still an improvement with respect to OSSA and as GoC decreases, the gap between DASSA and OSSA increases even more since location information becomes less important.

The results for the last network shows that as the number of tiers in the network increases, OSSA starts to approach the performance of DASSA. Thus, for larger network field sizes, employing DASSA with a clustered approach using multiple sinks or a heterogenous deployment structure would be better. In heterogenous deployments, two types of nodes are deployed to a field where one type of nodes (Type 0) has higher energy than the other type of nodes (Type 1). In such clustered schemes, every cluster will have small number of tiers and every clusterhead, a sink or a Type 1 node, will implement ILPSink and schedule its descendants. This way, the problem due to lack of information can be eliminated.

6.1.1 Further Analysis of DASSA and OSSA

Up to here, we have seen that DASSA outperforms OSSA in almost all the cases. Now, the reasons underlying this performance gain will be investigated and it will be shown how DASSA manages to yield results close to the optimum case.

The following parameters are compared for DASSA and OSSA.

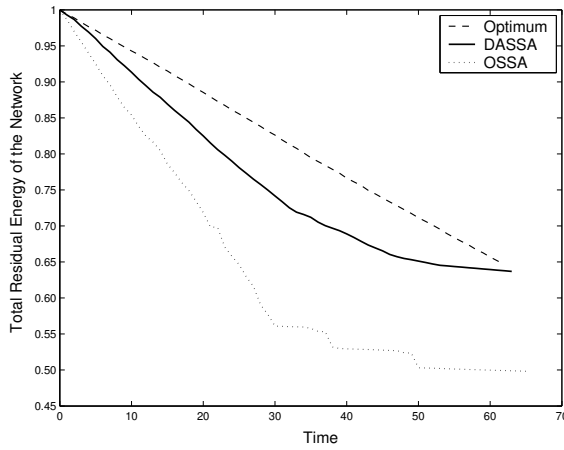
- *Total residual energy of the network* is the sum of the remaining energies of all the nodes in the network.
- *The consumed energy at the network* is the amount of total energy consumed at each round and is found from the total residual energy difference at two consecutive rounds.
- *Number of nodes scheduled to be on* is the number of nodes each algorithm schedules to be active at each round.
- *Number of connected nodes* is the number of nodes which can reach the sink at each round. Some of the nodes selected to be active may not be able to reach the sink. That is why we define this parameter in addition to the number of nodes scheduled to be active.
- *Number of dead nodes* at each round is the number of nodes that have exhausted their batteries so far.

Figure 6.11 shows the parameters described previously in order. All the plots are for Topology 1 and $\text{GoC} = 0.9$. The optimum algorithm consumes about the same amount of energy at each round in a perfectly balanced manner. DASSA consumes energy much efficiently than OSSA. Also, DASSA schedules nodes to be active much better than OSSA. Figures 6.11 (c) and 6.11 (d) show that OSSA cannot maintain the connectivity of the nodes which are scheduled to be active, whereas DASSA shows little difference in the amount of nodes scheduled

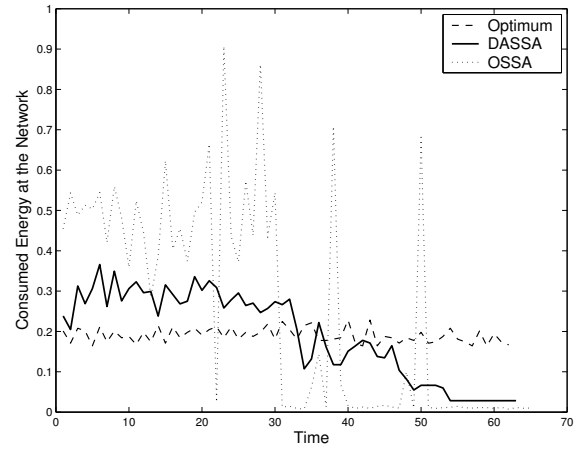
to be active and the amount of nodes which can actually function since they are connected. Without using location information, DASSA achieves very close results to the optimum. From Figure 6.11 (e), we observe that the number of useless nodes increases faster in OSSA. For the optimum algorithm, the number of useless nodes increases in late rounds compared to DASSA and OSSA since the optimum algorithm has global knowledge of all the nodes' remaining energies and therefore consumes energy in a more balanced way. Note that Figure 6.11 is in correspondence with the coverage plot given in Figure 6.4 and the parameters of each algorithm are plotted up to the round for which they can maintain operation. For example, since the coverage plot for the optimum algorithm ends in round 63, the number of dead nodes is drawn up to round 63 in Figure 6.11 (e).

Figure 6.12 is for $\text{GoC} = 0.8$ and Figures 6.13 is for $\text{GoC} = 0.7$. The consumed energy for DASSA when $\text{GoC} = 0.8$ and 0.7 is lower than the optimum because at those rounds, coverage provided by DASSA is below the coverage provided by the optimum algorithm which can be observed from Figures 6.5 and 6.6. As for $\text{GoC} = 0.9$, the energy is much more efficiently consumed than OSSA and is in fact very close to the optimum energy consumption. Similarly, the number of connected nodes are close to the optimum results. This is in direct relation with the good performance of DASSA for $\text{GoC} = 0.8$ and 0.7 .

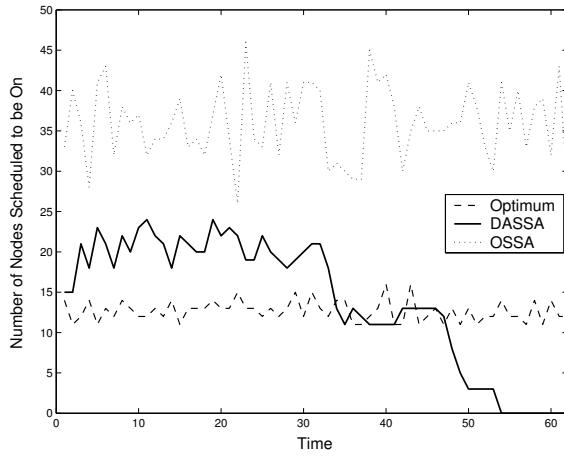
Figure 6.14, 6.15 and 6.16 show the number of rounds each location in the field is covered, i.e., the coverage count of each location, by DASSA, OSSA and the optimum case for $\text{GoC} = 0.9$, 0.8 and 0.7 , respectively. OSSA makes a peak at the center of the field. The reason is that, after nodes start to die, OSSA can only maintain the coverage of one or two nodes which are at a single hop to the sink. Thus, the area in the proximity of the sink is covered more. Figure 6.14, 6.15 and 6.16 are in correspondence with the coverage plots for Topology 1 given in Figures 6.4, 6.5 and 6.6. We observe from these figures that OSSA maintains very low coverage levels at the last rounds which causes peaks to occur around



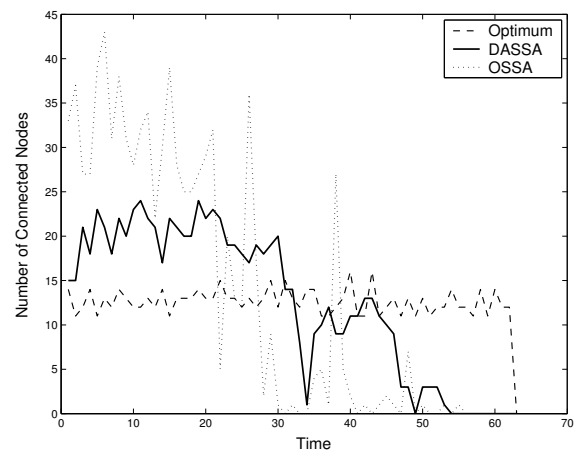
(a)



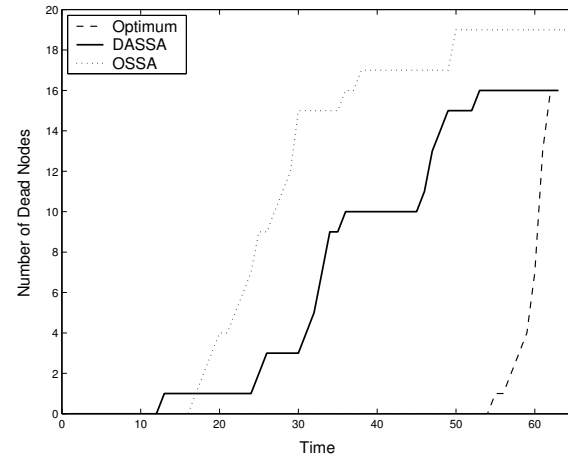
(b)



(c)

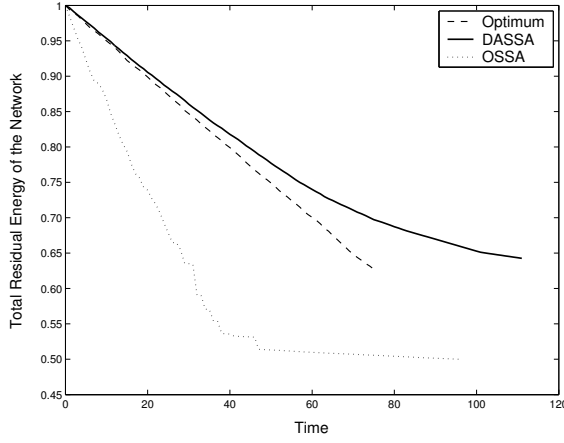


(d)

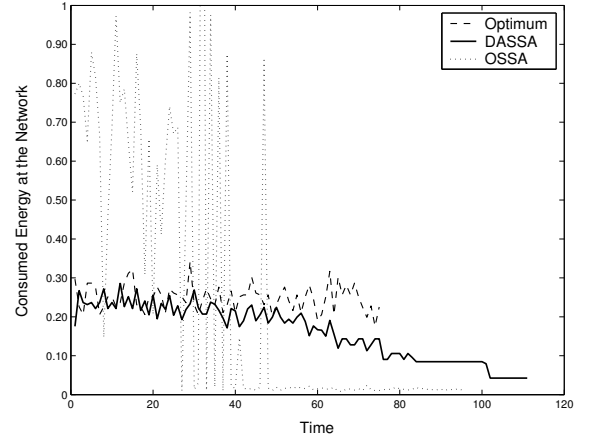


(e)

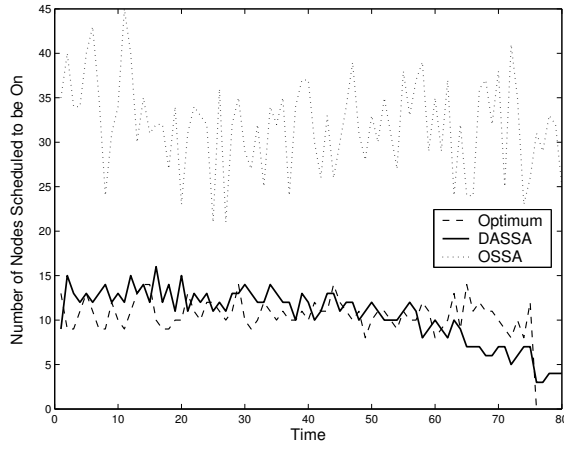
Figure 6.11: Energy and scheduling plots for $\text{GoC} = 0.9$ for Topology 1; (a) Normalized total residual energy of all nodes with respect to 1J. (b) Normalized total consumed energy at all nodes with respect to 30mJ. (c) Number of nodes which are scheduled to be active at each round. (d) Number of nodes connected to the sink at each round among the nodes which are scheduled to be active. (e) Number of nodes which are dead at each round.



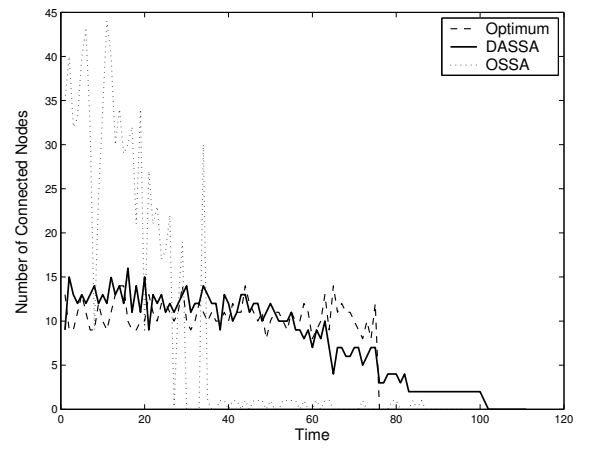
(a)



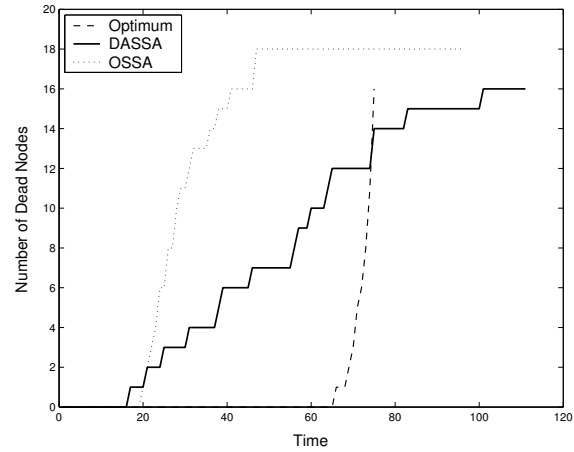
(b)



(c)

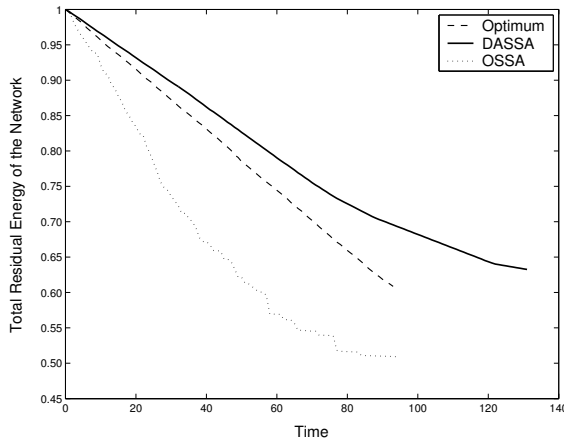


(d)

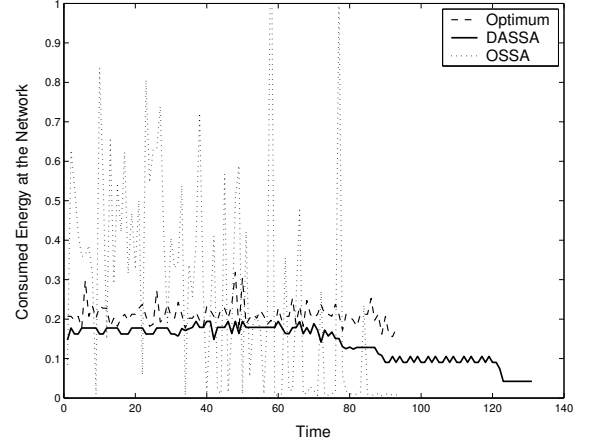


(e)

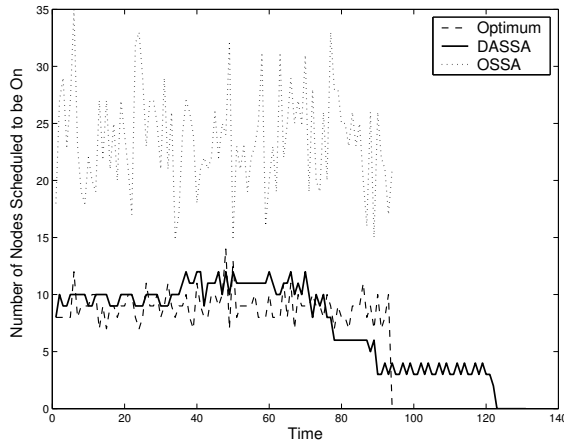
Figure 6.12: Energy and scheduling plots for $\text{GoC} = 0.8$ for Topology 1; (a) Normalized total residual energy of all nodes with respect to 1J. (b) Normalized total consumed energy at all nodes with respect to 20mJ. (c) Number of nodes which are scheduled to be active at each round. (d) Number of nodes connected to the sink at each round among the nodes which are scheduled to be active. (e) Number of nodes which are dead at each round.



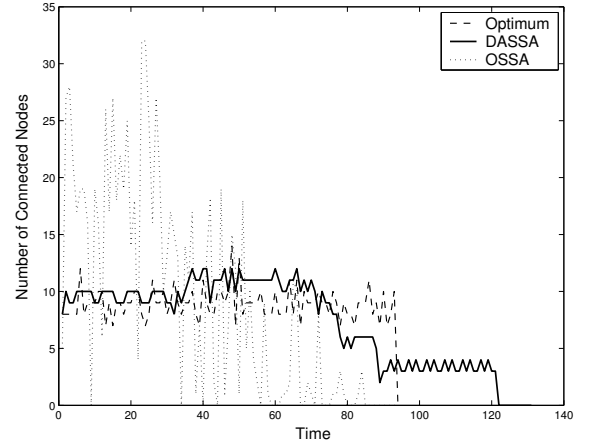
(a)



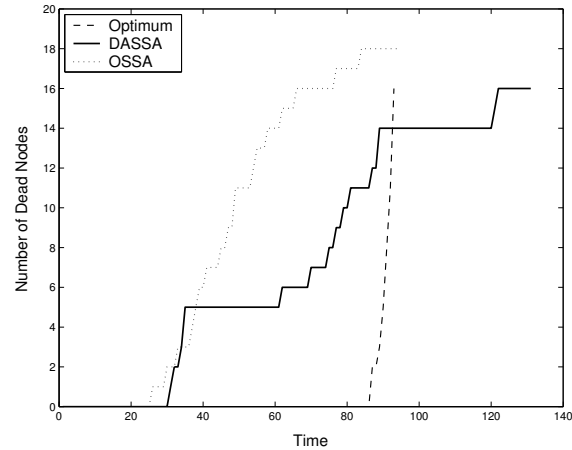
(b)



(c)



(d)



(e)

Figure 6.13: Energy and scheduling plots for $\text{GoC} = 0.7$ for Topology 1; (a) Normalized total residual energy of all nodes with respect to 1J. (b) Normalized total consumed energy at all nodes with respect to 20mJ. (c) Number of nodes which are scheduled to be active at each round. (d) Number of nodes connected to the sink at each round among the nodes which are scheduled to be active. (e) Number of nodes which are dead at each round.

the sink located at the origin $(0, 0)$ of the field. The optimum algorithm balances coverage very well in the field for every GoC even though the initial aim was to maximize the number of rounds the coverage is above GoC and no constraint for balanced coverage were introduced in the ILP formulations. Similarly, the aim of DASSA is to maximize the number of rounds for which the coverage is above GoC without any concern in balancing the coverage. However, also DASSA maintains a balanced coverage at each location and achieves far better coverage counts than OSSA. The far away locations are less covered in DASSA due to the parameters chosen.

Finally, Figure 6.17 shows the percentage of the scheduling overhead at each round for Topology 1. When GoC decreases, the energy consumption decreases faster than the decrease in the overhead. That is why the overhead percentage increases as GoC decreases. We observe that the scheduling overhead of DASSA is around 5-15% as compared to the total energy consumption. Figure 6.17 is in correspondence with Figures 6.4-6.6.

6.1.2 Unequal Transmission and Sensing Ranges

So far, it has been assumed that the sensing and transmission ranges are equal to each other. However, this might not be the case all the time. Sensing range being larger than transmission range is a rare case. Therefore, we analyze the effect of increasing the transmission range with respect to the sensing range. The sensing range is kept constant at 50m and the transmission range is changed to 60m and 75m. Topology 1 is used in the simulations.

Since DASSA is unaware of the locations of the sensor nodes, we do not expect it to be affected by the changes in the transmission range. Actually, when the transmission range increases with respect to the sensing range, we only need to increase the number of nodes that a node will schedule to be operating

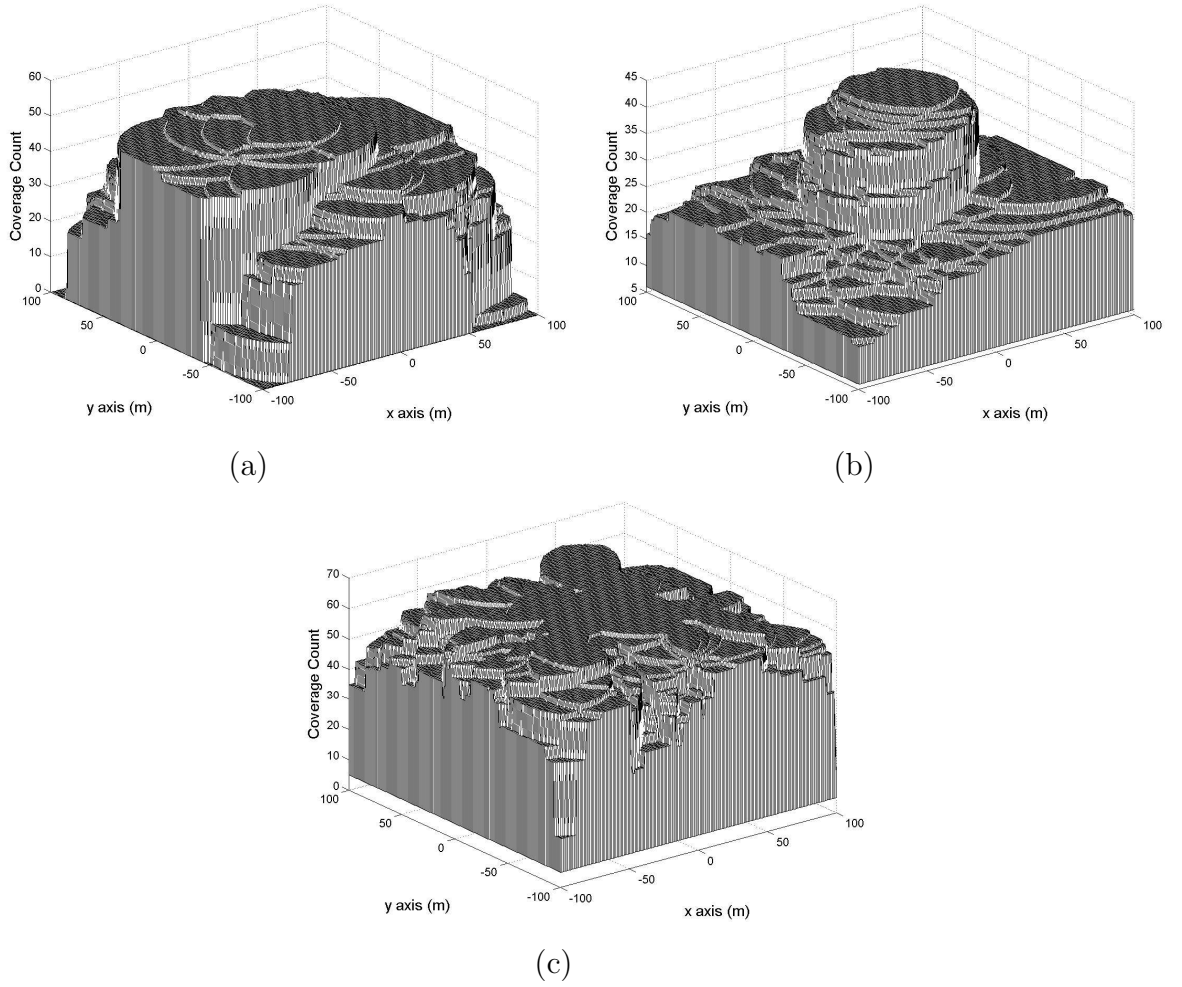


Figure 6.14: The x and y axis correspond to the network field 200m-by-200m size and the z axis is the number of rounds each of the locations in the field are covered. The algorithms are run up to the number of rounds obtained from the optimum algorithm, in this case for 62 rounds. The coverage counts of (a) DASSA, (b) OSSA and (c) Optimum algorithm. The results are for Topology 1 and $\text{GoC} = 0.9$.

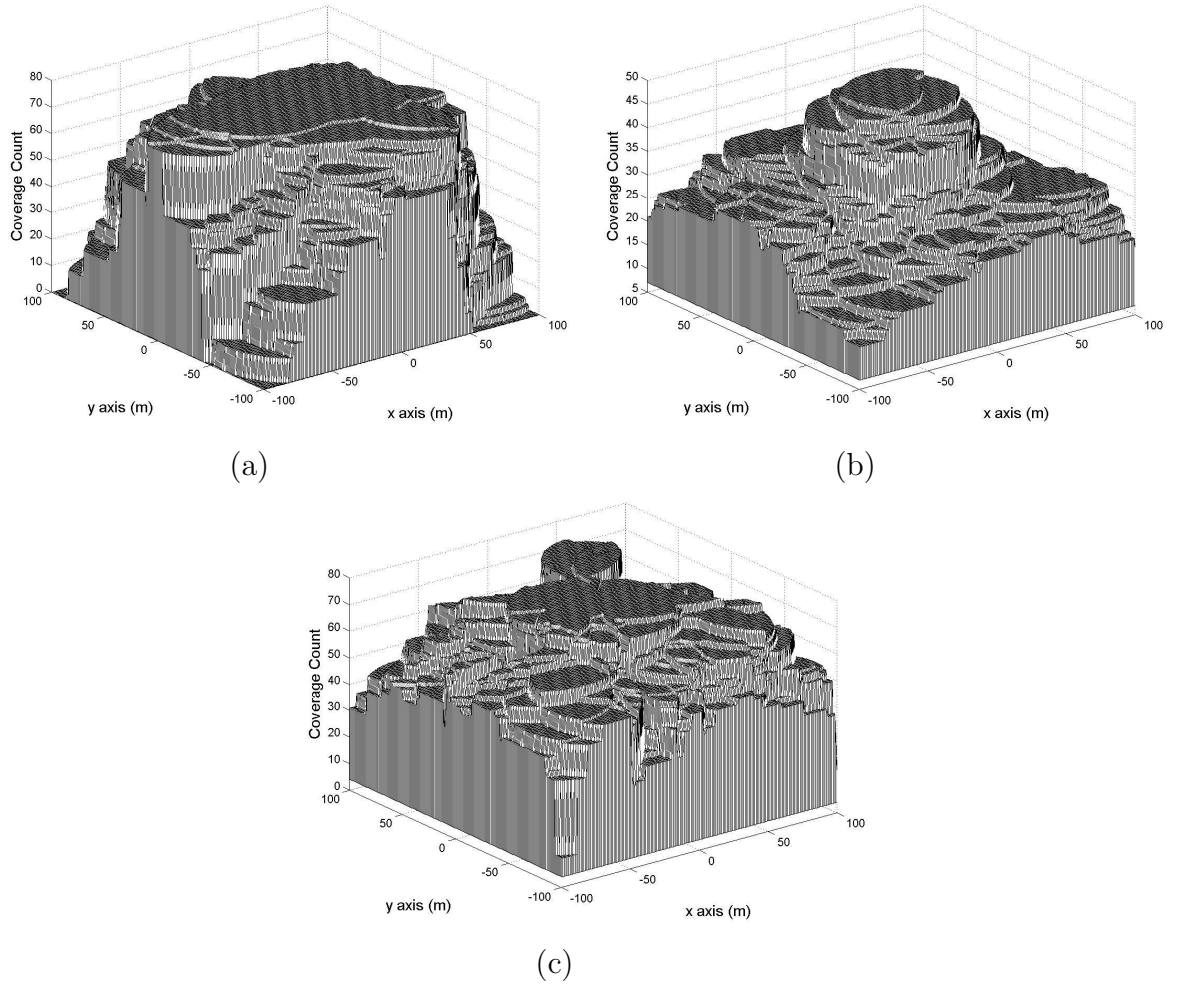
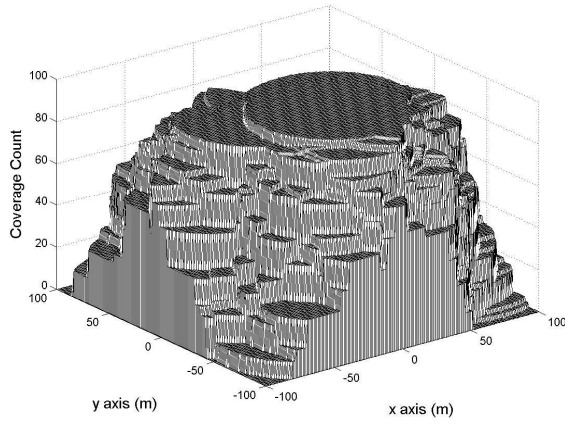
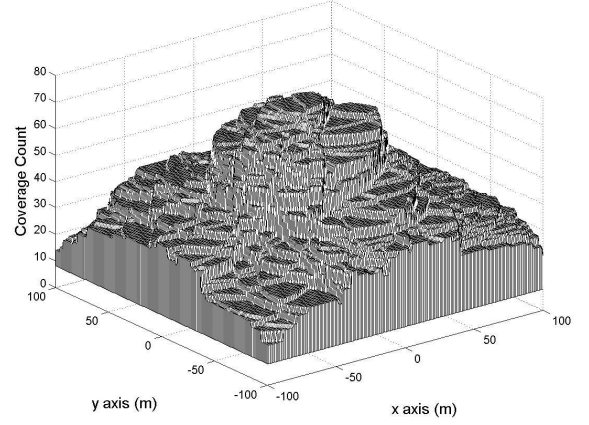


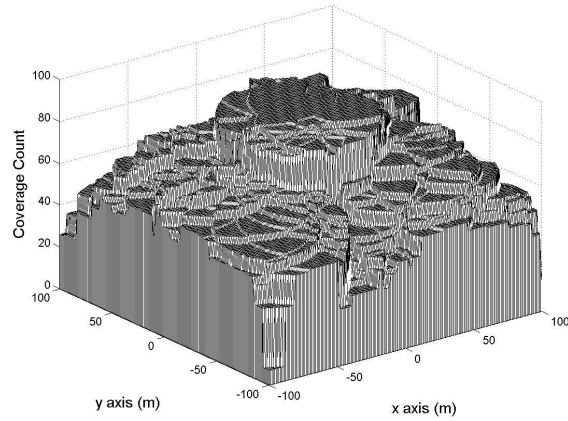
Figure 6.15: The x and y axis correspond to the network field 200m-by-200m size and the z axis is the number of rounds each of the locations in the field are covered. The algorithms are run up to the number of rounds obtained from the optimum algorithm, in this case for 75 rounds. The coverage counts of (a) DASSA, (b) OSSA and (c) Optimum algorithm. The results are for Topology 1 and $\text{GoC} = 0.8$.



(a)



(b)



(c)

Figure 6.16: The x and y axis correspond to the network field 200m-by-200m size and the z axis is the number of rounds each of the locations in the field are covered. The algorithms are run up to the number of rounds obtained from the optimum algorithm, in this case for 93 rounds. The coverage counts of (a) DASSA, (b) OSSA and (c) Optimum algorithm. The results are for Topology 1 and $\text{GoC} = 0.7$.

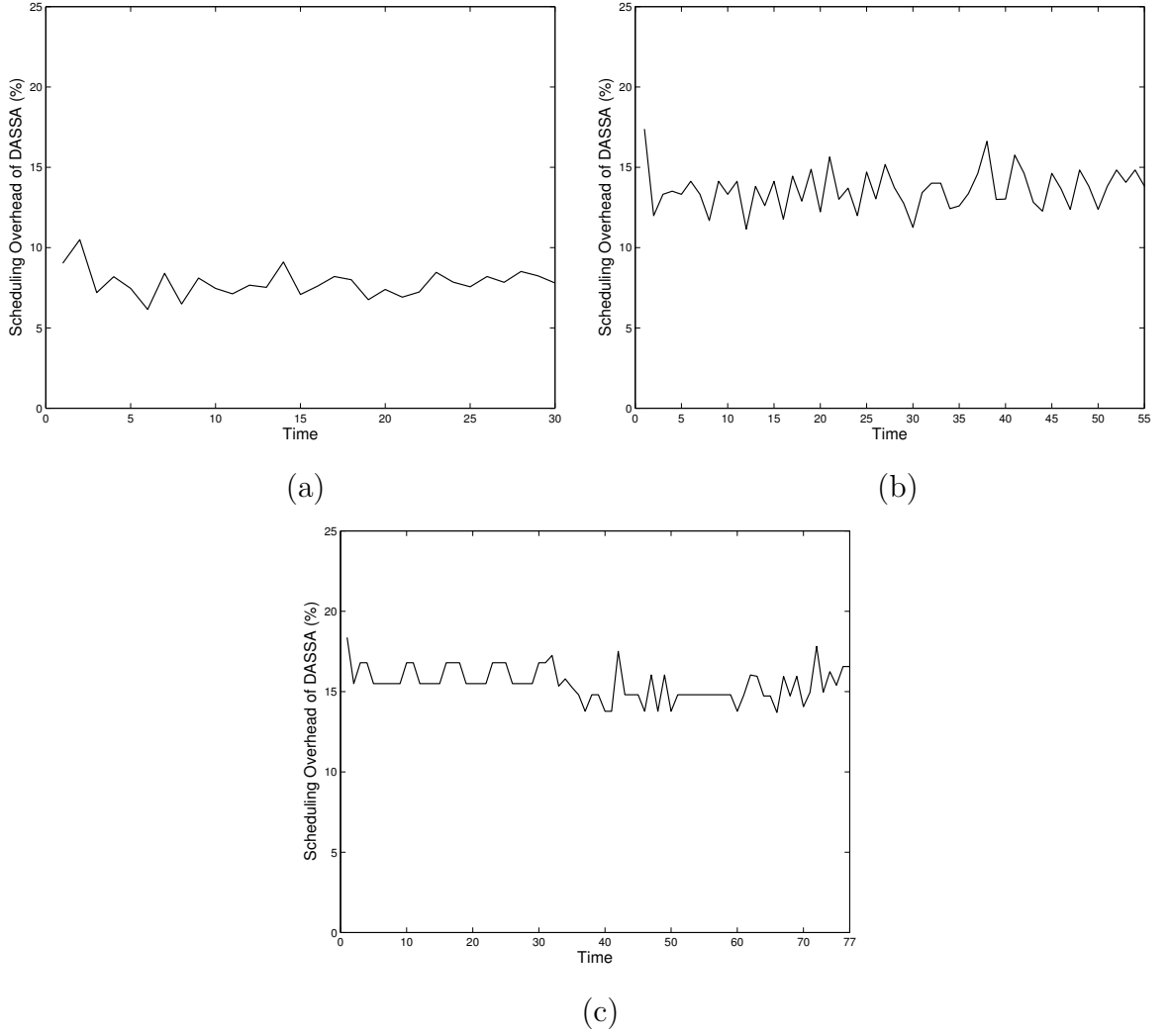


Figure 6.17: Scheduling overhead of DASSA for Topology 1. Percentages are with respect to the total consumed energy at each round. The plots are for (a) $GoC = 0.9$, (b) $GoC = 0.8$ and (c) $GoC = 0.7$.

from its next tiers (by increasing NSD and decreasing p_s parameters in DASSA). Although impractical, if the transmission range becomes less than the sensing range, then DASSA should be adopted to make more nodes to go into sleep (by lowering NSD and increasing p_s parameters of the algorithm). Also, ASD parameter should be modified accordingly.

Tables 6.11 and 6.12 show the results of the DASSA, OSSA and optimum case together with the no sleeping case for transmission ranges of 60m and 75m, respectively. First, the number of rounds in the no sleeping case increases with

respect to the case when $R_t = 50\text{m}$. This is expected since more nodes are in the single hop of the sink and thus the relaying burden on the tier 1 nodes decreases as reported in Table 6.10. Next, there is not much difference in the ratio of the number of rounds achieved by DASSA and the optimum results with respect to $R_t = 50\text{m}$ case. Also, when $R_t = 75\text{m}$, OSSA gives very close results to DASSA. This is expected since when $R_t = 75\text{m}$, the network consists of only two hops and the chosen set of nodes by OSSA are with high probability connected since GoC is 0.9 and many nodes are opened. However, the gap still remains wide open for GoC = 0.7 since as mentioned before, OSSA activates redundant nodes in order to provide connectivity and cannot benefit from the lowering of GoC. The number of nodes belonging to each tier number when R_t is changed is shown in Table 6.10.

R_t	50m	60m	75m
# of tier 1 nodes	17	24	42
# of tier 2 nodes	38	63	58
# of tier 3 nodes	41	13	0
# of tier 4 nodes	4	0	0

Table 6.10: Tier sizes for Topology 1 for different R_t .

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	14	1395	89.4	14	1395	89.4	14	1395	89.4
Optimum	145	10755	7.4	115	9613	9.4	96	8887	12.0
DASSA	116	9152	9.4	70	6327	15.3	48	4578	22.3
OSSA	54.9	3840	18.0	42.0	3362	20.0	32.1	2889	31.0

Table 6.11: Results of a 100 node network (Topology 1) for $R_t = 60\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field. $E_{init} = 10\text{mJ}$. The parameters used for DASSA and OSSA are given in Table A.6.

6.2 Full Aggregation

Although the full aggregation case is somewhat very optimistic in terms of practicality, the performance of DASSA is also evaluated for this case. Table 6.13

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	27	2684	90.6	27	2684	90.6	26	2595	93.3
Optimum	243	17929	6.8	193	16093	8.2	175	16250	9.7
DASSA	222	18245	8.2	124	11387	12.8	85	8057	13.3
OSSA	142.9	12264	14.1	111.4	10267	18.2	81.4	7933	25.1

Table 6.12: Results of a 100 node network (Topology 1) for $R_t = 75\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field. $E_{init} = 10\text{mJ}$. The parameters used for DASSA and OSSA are given in Table A.7.

	GoC = 0.7			GoC = 0.8			GoC = 0.9		
	GoC-L	EC	NRN	GoC-L	EC	NRN	GoC-L	EC	NRN
No Sleeping	16	1565	89.9	15	1494	89.9	15	1494	94.5
Optimum	104	7628	10.2	95	7890	12.1	82	7613	14.0
DASSA	76.8	6429	18.4	60.9	5517	23.3	42.3	4113	31.5
OSSA	58.2	5155	24.2	48.6	4558	29.2	38.7	3778	36.9
DRS	34.4	2834	13.4	29.2	2615	16.6	22.8	2178	21.8
EDRS	36.9	3051	13.5	31.0	2773	16.6	26.0	2490	22.1

Table 6.13: Results of a 100 node network (Topology 1) for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field when there is full aggregation. $E_{init} = 5\text{mJ}$. The parameters used for DASSA and OSSA are given in Table A.8.

presents the performance of all the sleep scheduling algorithms described at the beginning of the chapter. In contrast with the results obtained for no aggregation case, OSSA now provides better results. However, for $\text{GoC} = 0.8$ and 0.7 , there is still a big improvement over OSSA when DASSA is employed. Note that, for the full aggregation case $E_{init} = 5\text{mJ}$ is used and the other energy costs are the same as in the no aggregation case.

OSSA performs better in this case because opening more nodes for connectivity is not a big disadvantage anymore for OSSA since nodes combine all received data and transmit only one packet. Note the difference in the NRN values for DASSA and OSSA. Still, DASSA chooses nodes much more efficiently but since there is full aggregation, this does not cause a big difference in energy consumption.

Chapter 7

Conclusions

In this thesis, the problem of sleep scheduling in wireless sensor networks is investigated. Sleep scheduling activates only a subset of nodes which can maintain user defined constraints in an energy efficient manner to prolong network lifetime.

In order to gain some insight into the problem, a theoretical analysis on the number of nodes that should be deployed for various coverage levels without considering connectivity is presented. Furthermore, the minimum number of sensors that should be deployed in order to satisfy a given partial coverage target with a certain probability while maintaining connectivity is computed and an ILP formulation is presented for finding the minimum number of sensors that should be activated within the set of deployed sensors. This pre-deployment analysis can provide valuable information to a sleep scheduling algorithm design.

The optimum scheduling of nodes is found by using an ILP formulation which provides the maximum number of rounds the network can satisfy a certain coverage level. This approach is centralized and requires global knowledge of the network. Next, a distributed, simple and scalable sleep scheduling algorithm called DASSA is proposed. The main objective of DASSA is to find the minimum set of nodes which can satisfy the desired coverage without using any location information and only using local information.

DASSA is compared with the optimum scheduling results, an oblivious algorithm called OSSA and an existing work in the literature for various network sizes and various number of nodes. DASSA and OSSA outperforms the work in [46] and DASSA outperforms OSSA in all cases we considered. DASSA can reach up to the 43%, 63% and 89% of the optimum algorithm for coverage levels of 90%, 80% and 70%, respectively, when tuned properly. DASSA is a very flexible algorithm and with the fine tuning of its parameters depending on the network size and number of nodes, it can achieve performances close to optimum.

As the number of tiers in the network increases, the gain provided by DASSA algorithm decreases with respect to OSSA since the randomness in DASSA increases. Thus, DASSA would benefit from a clustered network structure. A multi-sink scenario where each sink employs DASSA could increase the performance of the algorithm. DASSA can also be used in an heterogenous network, where a subset of nodes have higher capabilities than the other type of nodes. In such a network, high powered nodes can employ DASSA algorithm within their clusters.

For future work, the performance of DASSA can be analyzed for a clustered network and compared with OSSA. Also, the ILP based formulation can be adjusted for finding the optimum scheduling for clustered networks. Partial coverage is a new subject in the context of sleep scheduling and extending the study in this thesis to the clustered case would be of value. Also, a more adaptive version of DASSA where the parameters of the algorithm are modified as nodes die may be studied. Finally, the parameters of DASSA should be tuned to find the real optimum point of the algorithm by using exhaustive search. We only reported the best possible runs we could find in limited time. Therefore, the performance of the algorithm when the parameters are at the real optimum values can be better when compared with the results we provide.

Sleep scheduling is a flourishing area in extending network lifetime and together with partial coverage, it could be the key to very long lifetime sensor networks.

APPENDIX A

Algorithm Parameters

In this section, the parameters used in DASSA and OSSA for obtaining the results in Chapter 6 are presented. Note that the NSD column is divided into more than one columns in some of the tables. For that cases, ASD parameter is larger than 1, and as we mentioned in Chapter 5, the NSD parameter differs for each tier which are involved in scheduling their neighbors from the next tiers. Thus, the number of columns under NSD should go up to ASD since only nodes from tiers 1 to ASD schedule their neighbors. In other words, first column under the NSD parameter in the tables is the number of nodes that each node from tier 1 schedules, second column is the number of nodes that each node from tier 2 schedules and so on.

Top. #	GoC	α	<i>Objective</i>	p_s	<i>NAS</i>	<i>NSD</i>	<i>ASD</i>
1	90	0.65	3	1.00	1	2	1
	80	0.93	3	1.00	1	1	1
	70	0.99	3	1.00	0	1	1
2	90	0.99	2	1.00	1	2	1
	80	0.97	3	1.00	1	1	1
	70	0.93	1	1.00	0	1	1
3	90	0.51	3	1.00	1	2	1
	80	0.93	3	1.00	1	1	1
	70	0.93	3	1.00	0	1	1
4	90	0.96	2	1.00	1	2	1
	80	0.96	2	1.00	1	1	1
	70	0.91	2	1.00	0	1	1
Common	90	0.58	1	1.00	1	2	1
	80	0.93	3	1.00	1	1	1
	70	0.94	3	1.00	0	1	1

Table A.1: Parameters of DASSA for Topology 1, 2, 3 and 4.

Top. #	GoC = 0.9	GoC = 0.8	GoC = 0.7
1	0.60	0.70	0.76
2	0.66	0.70	0.78
3	0.62	0.70	0.78
4	0.65	0.70	0.74
Common	0.64	0.70	0.78

Table A.2: p^* of OSSA for Topology 1, 2, 3 and 4.

GoC	α	<i>Objective</i>	p_s	<i>NAS</i>	<i>NSD</i>		<i>ASD</i>	p^*
					1	2		
0.9	0.87	3	1.00	1	1	1	2	0.73
0.8	0.95	3	1.00	0	1	1	2	0.78
0.7	0.75	1	1.00	0	1	1	2	0.83

Table A.3: Parameters of DASSA and OSSA for Topology 5.

GoC	α	<i>Objective</i>	p_s	<i>NAS</i>	<i>NSD</i>		<i>ASD</i>	p^*
					1	2		
0.9	0.93	3	1.00	0	2	1	2	0.80
0.8	0.90	3	1.00	0	2	-	1	0.82
0.7	0.87	3	1.00	0	1	-	1	0.86

Table A.4: Parameters of DASSA and OSSA for 100 random deployments of a 200 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 200m-by-200m field.

GoC	α	<i>Objective</i>	p_s	<i>NAS</i>	<i>NSD</i>			<i>ASD</i>	p^*
					1	2	3		
0.9	0.99	3	0.85	0	2	1	-	2	0.80
0.8	0.90	3	1.00	0	3	1	1	3	0.84
0.7	0.95	3	1.00	0	2	1	1	3	0.86

Table A.5: Parameters of DASSA and OSSA for 100 random deployments of a 400 node network for $R_t = 50\text{m}$, $R_s = 50\text{m}$ and a 300m-by-300m field.

GoC	α	<i>Objective</i>	p_s	<i>NAS</i>	<i>NSD</i>	<i>ASD</i>	p^*
0.9	0.94	3	1.00	1	2	1	0.68
0.8	0.95	3	1.00	1	1	1	0.80
0.7	0.95	1	1.00	0	1	1	0.81

Table A.6: Parameters of DASSA and OSSA for $R_t = 60\text{m}$, $R_s = 50\text{m}$ and Topology 1.

GoC	α	<i>Objective</i>	p_s	<i>NAS</i>	<i>NSD</i>	<i>ASD</i>	p^*
0.9	0.98	2	1.00	1	1	1	0.75
0.8	0.98	2	1.00	1	1	1	0.82
0.7	0.97	1	1.00	0	1	1	0.86

Table A.7: Parameters of DASSA and OSSA for $R_t = 75\text{m}$, $R_s = 50\text{m}$ and Topology 1.

GoC	α	<i>Objective</i>	p_s	<i>NAS</i>	<i>NSD</i>	<i>ASD</i>	p^*
0.9	0.80	3	0.55	0	2	1	0.62
0.8	0.86	3	0.62	0	1	1	0.69
0.7	0.74	3	0.66	0	1	1	0.73

Table A.8: Parameters of DASSA for Topology 1 with full aggregation.

Bibliography

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Comput. Networks*, vol. 38, no. 4, pp. 393–422, 2002.
- [2] L. Hu and D. Evans, “Localization for mobile sensor networks,” in *Tenth Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2004.
- [3] N. Bulusu, J. Heidemann, and D. Estrin, “GPS-less low-cost outdoor localization for very small devices,” *IEEE Personal Communications*, no. 2–3, pp. 28–34, 2000.
- [4] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *HICSS ’00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, (Washington, DC, USA), p. 8020, IEEE Computer Society, 2000.
- [5] A. Manjeshwar and D. Agrawal, “TEEN: A routing protocol for enhanced efficiency in wireless sensor networks,” in *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pp. 2009–2015, 2001.
- [6] B. Krishnamachari, D. Estrin, and S. B. Wicker, “The impact of data aggregation in wireless sensor networks,” in *ICDCSW ’02: Proceedings of the*

- 22nd International Conference on Distributed Computing Systems*, (Washington, DC, USA), pp. 575–578, IEEE Computer Society, 2002.
- [7] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, “Building efficient wireless sensor networks with low-level naming,” in *SOSP ’01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, (New York, NY, USA), pp. 146–159, ACM Press, 2001.
 - [8] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, “Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks,” in *MobiCom ’01: Proceedings of the 7th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 272–287, ACM Press, 2001.
 - [9] J. Liu, X. Koutsoukos, J. Reich, and F. Zhao, “Sensing field: coverage characterization in distributed sensor networks,” in *IEEE ICASSP*, vol. 5, pp. 173–176, 2003.
 - [10] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, “Grid coverage for surveillance and target location in distributed sensor networks,” *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1448–1453, 2002.
 - [11] S. Meguerdichian and M. Potkonjak, “Low power 0/1 coverage and scheduling techniques in sensor networks,” Technical Reports 030001, University of California Los Angeles, January 2003.
 - [12] S. Slijepcevic and M. Potkonjak, “Power efficient organization of wireless sensor networks,” in *IEEE International Conference on Communications, ICC*, pp. 472–476, 2001.
 - [13] C. Huang and Y. Tseng, “The coverage problem in a wireless sensor network,” in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications (WSNA)*, pp. 115–121, 2003.

- [14] D. Tian and N. Georganas, “A coverage-preserving node scheduling scheme for large wireless sensor networks,” in *First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.
- [15] A. Boukerche, X. Fei, R. B. Araujo, and P. Patnaik, “A local information exchange based coverage-preserving protocol for wireless sensor networks,” in *Proceedings of IEEE ICC*, 2006.
- [16] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, “Coverage problems in wireless ad-hoc sensor networks,” in *Proceedings of IEEE INFOCOM*, 2001.
- [17] Y. Xu, J. Heidemann, and D. Estrin, “Geography-informed energy conservation for ad hoc routing,” in *Proceedings of ACM MOBICOM*, 2001.
- [18] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, “Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks,” in *Mobile Computing and Networking*, pp. 85–96, 2001.
- [19] A. Cerpa and D. Estrin, “ASCENT: Adaptive self-configuring sensor networks topologies,” in *Proceedings of IEEE INFOCOM*, June 2002.
- [20] Y. Xu, J. Heidemann, and D. Estrin, “Adaptive energy-conserving routing for multihop ad hoc networks,” research report 527, USC/ Information Sciences Institute, October 2000.
- [21] H. Gupta, S. Das, and Q. Gu, “Connected sensor cover: Self organization of sensor networks for efficient query execution,” in *ACM Mobile Adhoc Network Symposium (MOBIHOC)*, pp. 189–199, 2003.
- [22] S. Shakkottai, R. Srikant, and N. Shroff, “Unreliable sensor grids: Coverage, connectivity and diameter,” in *Proceedings of IEEE INFOCOM*, April 2003.

- [23] V. P. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff, "A minimum cost heterogeneous sensor network with a lifetime constraint," *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 4–15, 2005.
- [24] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. 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*, (Washington, DC, USA), p. 28, IEEE Computer Society, 2003.
- [25] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proceedings of Sensys*, 2003.
- [26] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *International Journal of Wireless Ad Hoc and Sensor Networks*, vol. 1, no. 1–2, pp. 89–124, 2005.
- [27] M. Cardei and J. Wu, "Energy-efficient coverage problems in wireless ad hoc sensor networks," *To be published in Computer Communications, special issue on Sensor Networks*.
- [28] R. Iyer and L. Kleinrock, "QoS control for sensor networks," in *Proceedings of the IEEE International Conference on Communications*, 2003.
- [29] M. Tsetlin, *Finite Automata and Modeling the Simplest Forms of Behavior*. PhD thesis, V.A. Steklov Mathematical Institute, 1964.
- [30] J. Frolik, "QoS control for random access wireless sensor networks," in *IEEE Wireless Communications and Networking Conference*, March 2004.
- [31] J. Kay and J. Frolik, "Quality of service analysis and control for wireless sensor networks," in *First IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, October 2004.

- [32] B. Liang, J. Frolik, and X. S. Wang, "A predictive QoS control strategy for wireless sensor networks," in *Second IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, November 2005.
- [33] J. Misic, S. Shafi, and V. Misic, "Cross-layer activity management in an 802-15.4 sensor network," *Communications Magazine, IEEE*, vol. 44, pp. 131–136, January 2006.
- [34] B. Liang, J. Frolik, and X. S. Wang, "Maintaining reliability through activity management in 802.15.4 sensor networks," in *Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, August 2005.
- [35] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," *Wireless Communications, IEEE*, vol. 11, pp. 6–28, December 2004.
- [36] K. Akkaya and M. Younis, "A survey of routing protocols in wireless sensor networks," *Elsevier Ad Hoc Network Journal*, vol. 3, no. 3, pp. 325–349, 2005.
- [37] C. J. Leuschner, "The design of a simple energy efficient routing protocol to improve wireless sensor network lifetime," MS thesis. Faculty of Engineering, University of Pretoria. April 2005.
- [38] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless Networks*, vol. 8, no. 2–3, pp. 169–185, 2002.
- [39] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Mobile Computing and Networking*, pp. 56–67, 2000.
- [40] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *Proceedings of IEEE INFOCOM*, pp. 22–31, 2000.

- [41] U. Monaco, F. Cuomo, T. Melodia, F. Ricciato, and M. Borghini, “Understanding optimal data gathering in the energy and latency domains of a wireless sensor network,” *To appear in Computer Networks (Elsevier)*, 2006.
- [42] Y. Liu and W. Liang, “Approximate coverage in wireless sensor networks,” in *Proceedings of of 30th Annu. IEEE Conf. on Local Computer Networks, IEEE Computer Society*, pp. 68–75, November 2005.
- [43] Y. Xu, J. Heidemann, and D. Estrin, “pCover: Partial coverage for long-lived surveillance sensor networks,” tech. rep., Department of Computer Science, Michigan State University, November 2005.
- [44] H. Zhang and J. Hou, “On deriving the upper bound of α -lifetime for large sensor networks,” in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, (New York, NY, USA), pp. 121–132, ACM Press, 2004.
- [45] W. Choi and S. K. Das, “Trade-off between coverage and data reporting latency for energy-conserving data gathering in wireless sensor networks,” in *First IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, October 2004.
- [46] W. Choi and S. K. Das, “A novel framework for energy-conserving data gathering in wireless sensor networks,” in *Proceedings of IEEE INFOCOM*, March 2005.
- [47] F. Ye, G. Zhong, S. Lu, and L. Zhang, “Energy efficient robust sensing coverage in large sensor networks,” tech. rep., UCLA, 2002.
- [48] W. Ye, J. Heidemann, and D. Estrin, “An energy-efficient MAC protocol for wireless sensor networks,” in *Proceedings of IEEE INFOCOM*, pp. 1567–1576, 2002.

- [49] S.Tai, R.Benkoczi, H.Hassanein, and S.Akl, “A performance study of splittable and unsplittable traffic allocation in wireless sensor networks,” in *Proceedings of IEEE ICC*, 2006.
- [50] “Cplex solver.” [http : //www.ilog.com/products/cplex/](http://www.ilog.com/products/cplex/), June 2005.