Design Problems in Large-Scale, Time-Sensitive WSNs

Vom Fachbereich Informatik
der Technischen Universitat Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte

Dissertation

von

Wint Yi Poe

Datum der wissenschaftlichen Aussprache: 14. November 2012

Dekan: Prof. Dr. Arnd Poetzsch-Heffter

Prüfungskommission:

Vorsitz: Prof. Dr. Christoph Garth

Erster Berichterstatter: Prof. Dr. Jens B. Schmitt

Zweiter Berichterstatter: Prof. Dr. Lars Wolf

D 386

Acknowledgements

The successful completion of this dissertation would not have been possible without invaluable supporters.

First of all, I would like to give a heartfelt, special thanks to my "Doktorvater", Prof. Dr-Ing Jens B. Schmitt, for his excellent guidance, patience and encouragement, and for providing me with a wonderful atmosphere for doing research at Distributed Computer and Systems (DISCO) lab from the beginning to the end. His unconditional support and caring made me motivated to overcome anxieties of the challenges. For all these many many things, I am eternally grateful to you.

My gratitude is also extended to the remaining members of my defense committee. I would like to express my sincere gratitude to Prof. Dr.-Ing Lars Wolf, who generously accepted to serve as a second referee, for his willingness to review my dissertation. Special thanks goes to the chair of my defense committee, Prof. Dr.-Ing Christoph Garth for his academic support. I am very grateful to Frau Edith Hofbauer and Frau Babara Erlewine, who have always helped me organizational issues and all the required paperwork. A special acknowledgment goes to Steffen Reithermann for his tremendous help and I can't express how grateful I am.

These acknowledgments would not be complete if I did not mention the past and present members of DISCO group: Frank Zdarsky, Ivan Martinovic, Nicos Gollan, Adam Bachorek, Hao Wang, Matthias William, Steffen Bondorf, Michael Beck, and Daniel Berger for their generous support, willingness to help, and friendship. My special thanks go to Michael Beck for all the exciting joint work that we did together. I'll never forget Matthias William's invaluable support in some lectures. I would like to thank Petra Martinovic who helped me with a quick proofreading whenever I requested. I am so glad to have worked with such wonderful colleagues in DISCO.

I would like to give my deepest gratitude to my parents and sisters for their unconditional love and care. I would not have made the dissertation this far without encouragement and support from my family.

Last, but certainly not least, I would like to deeply thank Gottlieb Daimler- und Karl Benz-Stiftung, TU Kaiserslautern, and DAAD for their generous support in this work.

Abstract

There is a growing trend for ever larger wireless sensor networks (WSNs) consisting of thousands or tens of thousands of sensor nodes (e.g., [91, 79]). We believe this trend will continue and thus scalability plays a crucial role in all protocols and mechanisms for WSNs. Another trend in many modern WSN applications is the time sensitivity to information from sensors to sinks. In particular, WSNs are a central part of the vision of cyber-physical systems and as these are basically closed-loop systems many WSN applications will have to operate under stringent timing requirements. Hence, it is crucial to develop algorithms that minimize the worst-case delay in WSNs. In addition, almost all WSNs consist of battery-powered nodes, and thus energy-efficiency clearly remains another premier goal in order to keep network lifetime high. This dissertation presents and evaluates designs for WSNs using multiple sinks to achieve high lifetime and low delay.

Firstly, we investigate random and deterministic node placement strategies for large-scale and time-sensitive WSNs. In particular, we focus on tiling-based deterministic node placement strategies and analyze their effects on coverage, lifetime, and delay performance under both exact placement and stochastically disturbed placement.

Next, we present sink placement strategies, which constitutes the main contributions of this dissertation. Static sinks will be placed and mobile sinks will be given a trajectory. A proper sink placement strategy can improve the performance of a WSN significantly. In general, the optimal sink placement with lifetime maximization is an NP-hard problem. The problem is even harder if delay is taken into account. In order to achieve both lifetime and delay goals, we focus on the problem of placing multiple (static) sinks such that the maximum worst-case delay is minimized while keeping the energy consumption as low as possible. Different target networks may need a corresponding sink placement strategy under differing levels of apriori assumptions. Therefore, we first develop an algorithm based on the Genetic Algorithm (GA) paradigm for known sensor nodes' locations. For a network where global information is not feasible we introduce a self-organized sink placement (SOSP) strategy. While GA-based

sink placement achieves a near-optimal solution, SOSP provides a good sink placement strategy with a lower communication overhead.

How to plan the trajectories of *many mobile* sinks in *very large* WSNs in order to simultaneously achieve *lifetime* and *delay* goals had not been treated so far in the literature. Therefore, we delve into this difficult problem and propose a heuristic framework using multiple orbits for the sinks' trajectories. The framework is designed based on geometric arguments to achieve both, high lifetime and low delay. In simulations, we compare two different instances of our framework, one conceived based on a load-balancing argument and one based on a distance minimization argument, with a set of different competitors spanning from statically placed sinks to battery-state aware strategies. We find our heuristics outperform the competitors in both, lifetime and delay. Furthermore, and probably even more important, the heuristic, while keeping its good delay and lifetime performance, scales well with an increasing number of sinks.

In brief, the goal of this dissertation is to show that placing nodes and sinks in conventional WSNs as well as planning trajectories in mobility enabled WSNs carefully really pays off for large-scale and time-sensitive WSNs.

Zusammenfassung

In den letzten Jahren ist ein wachsender Trend zu immer größeren Wireless Sensor Networks (WSN), bestehend aus Tausenden oder Zehntausenden von Sensorknoten (z. B. [91, 79]), zu beobachten. Wir glauben, dass dieser Trend anhalten und damit die Skalierbarkeit eine entscheidende Rolle in zukünftigen Protokollen und Mechanismen für WSNs spielen wird. Ein weiterer Trend in modernen WSN-Anwendungen ist die wachsende Zeitsensibilität des Transports von Sensorinformationen an der Senke. Insbesondere sind WSNs ein zentraler Teil der Vision von Cyber-Physical Systems (welche hauptsächlich Regelungssysteme darstellen), d. h. viele WSN-Anwendungen müssen auch unter strengen Zeitanforderungen funktionieren. Daher ist es von entscheidender Bedeutung Algorithmen zu entwickeln, die die Worst-Case-Verzögerung in WSNs minimieren. Darüber hinaus bestehen nahezu alle WSNs aus batteriebetriebenen Sensorknoten, und so ist auch die Energieeffizienz ein wichtiges Ziel um eine hohe Lebensdauer des Netzwerks zu erzielen. Diese Dissertation präsentiert und bewertet Designs für WSNs unter Verwendung mehrerer Senken, um so eine hohe Lebensdauer bei gleichzeitig geringer Verzögerung zu erreichen.

Zunächst untersuchen wir sowohl zufällige als auch deterministische Platzierungsstrategien für Sensorknoten in großen und zeitkritischen WSNs. Insbesondere konzentrieren wir uns auf tiling-basierte, deterministische Platzierungsstrategien und analysieren ihre Auswirkungen auf die Abdeckung, Lebensdauer und Verzögerung des Netzwerks, sowohl für exakte als auch für stochastisch gestörte Platzierungen.

Als nächstes präsentieren wir Platzierungsstrategien für Senken, welche auch den Hauptbeitrag dieser Dissertation bilden. Es werden hierbei sowohl statische als auch mobile Senken betrachtet, die sich entlang geplanter Trajektorien bewegen. Eine gut gewählte Senkenplatzierungsstrategie kann dann die Leistung eines WSNs deutlich verbessern. Im Allgemeinen ist eine optimale Senkenplatzierung mit dem Ziel der Maximierung der Lebensdauer jedoch ein NP-hartes Problem, welches sogar noch schwieriger wird wenn zusätzlich die Minimierung der Verzögerung als Ziel berücksichtigt wird. Um sowohl Lebensdauer- und Verzögerungsziele zu erreichen beschäftigen wir uns mit der Platzierung mehrerer (statischer) Senken,

um die maximale Worst-Case-Verzögerung zu minimieren und gleichzeitig den Energieverbrauch so gering wie möglich zu halten. Unterschiedliche Netzwerktopologien können hierbei angepasste Strategien für die Platzierung unter unterschiedlichen a priori Annahmen erfordern. Deshalb entwickeln wir zunächst einen Algorithmus der auf dem Genetic Algorithm (GA)-Paradigma, unter Verwendung von bekannten Knotenpositionen, basiert. Für Netzwerke in denen solche globalen Informationen nicht verfügbar sind stellen wir eine selbst-organisierte Senkenplatzierungsstrategie (SOSP) vor. Während die GA-basierte Senkenplatzierung eine nahezu optimale Lösung erreicht, bietet SOSP eine gute Strategie mit geringerem Kommunikationsoverhead.

Bis jetzt wurden in der Literatur noch keine Strategien untersucht, die Trajektorien von vielen mobilen Senken verwenden um gleichzeitig Lebensdauer- und Verzögerungsziele in sehr großen WSNs zu erreichen. Deshalb haben wir uns in dieses schwierige Problem vertieft und schlagen ein heuristisches Framework mit mehreren Orbits für die Trajektorien der Senken vor. Das Framework basiert auf geometrischen Überlegungen und erzielt sowohl eine hohe Lebensdauer als auch eine geringe Verzögerung. Mittels Simulationen vergleichen wir zwei Instanzen unseres Frameworks mit einer Reihe von Gegenvorschlägen aus der Literatur, von der statischen Platzierung der Senken bis zu Strategien, die den Batteriezustand der Sensorknoten miteinbeziehen. Unsere zwei Instanzen basieren hierbei entweder Load-Balancing-Überlegungen oder der Minimierung der durchschnittlichen Entfernungen. Unsere Ergebnisse belegen, dass die vorgestellten Heuristiken die Konkurrenz sowohl im Bezug auf Lebensdauer als auch der Minimierung der Verzögerung übertreffen. Weiterhin, und wahrscheinlich noch wichtiger, stellen wir fest, dass die heuristische Strategie mit zunehmender Anzahl von Senken im Bezug auf Verzögerung und Lebensdauer sehr gut skaliert. Kurz gesagt ist das Ziel dieser Arbeit zu zeigen, dass eine planvolle Platzierung von Sensorknoten und Senken in konventionellen WSNs und die Planung von Trajektorien in neuen mobilen WSNs sich vor allem in großen und zeitkritischen Sensornetzwerken auszahlt.

Contents

List of Figures

List of Algorithms

List of Tables

1. Introduction

A Wireless Sensor Network (WSN) provides a wireless communication infrastructure that allows us to react, monitor, observe, and response to phenomena in a natural environment, physical, and cyber infrastructure [113]. It is composed of a number of small, unobtrusive, and autonomous devices called sensor nodes with processing, communication, and sensing capabilities. Sensor nodes can be homogeneous or heterogeneous, i.e., possessing the same or different communication and computation capabilities, respectively. Although some applications require heterogeneous sensors, for many applications it is sufficient to investigate the case of homogeneous WSNs. Less complexity and a better manageability are the most beneficial properties of homogeneity. Some distinct features of WSNs are being data centric and application-specific, having energy constraints, striving to be time-sensitive and scalable. In WSNs, nodes do not have global identifiers, therefore, data is collected based on certain attributes rather than direct addressing of particular nodes, this is called data centricity. Application specificity in WSNs means that one cannot have a solution that fits all problems. Therefore, it is very hard to find standard designs, protocols, and algorithms for all sorts of WSN applications. Due to the battery limitations, sensor nodes are energy-constrained. Therefore, energy-efficient designs, protocols, and algorithms are considered with the intention of lifetime prolongation of WSNs. Besides striving for elongated lifetimes, many applications of WSNs are time-sensitive, i.e., they strongly benefit from bounds on the message transfer delay. Hence, minimizing maximum delay is a primary goal for time-sensitive WSNs. Further, a WSN is composed of a large amount of unsophisticated and low cost sensors to perform the sensing of the information instead of using a few expensive and sophisticated sensors as in a conventional sensor network approach. Thus scalability is a critical issue in WSNs.

In this dissertation, we intend to present insights into designing large-scale and time-sensitive WSNs, by using multiple sinks in WSNs and by planning the trajectories of multiple mobile sinks in order to achieve lifetime and delay goals. In addition, we investigate alternative node placement strategies for various performance metrics.

1.1. Motivation

Lifetime prolongation of WSNs by using multiple static or mobile sinks has already been treated in literature [99, 65, 139, 55, 107, 140, 82, 144, 20, 125]. However, designing WSNs using multiple sinks to achieve both lifetime and delay goals has not been researched yet. In general, there is a tradeoff between lifetime and delay optimization. But existing studies do not explicitly address the optimal sink placement strategy for delay and lifetime goals. Hence, our objective is to design and control both static and mobile sinks for large-scale and time-sensitive WSNs to achieve high lifetime and low delay. In fact, even the optimal static or mobile sink placement for lifetime prolongation is an NP-hard problem. Thus, we can imagine that a solution of the problem for both lifetime and delay goals is very hard to achieve. To solve this problem, heuristics and problem transformations are promising strategies which will be taken into account in this dissertation.

The sink placement problem can be divided into the static sink placement problem and the mobile sink movement problem. The static sink placement problem is related with conventional WSNs where the locations of sinks are fixed. For large-scale and time-sensitive WSNs, scalable and low delay sink placement strategies are desired. It is difficult to find a single solution to all types of problems for application-specific WSNs. For different target networks, appropriate sink placements are required. For example, a sink placement strategy should provide a better performance if a WSN provides global information such as nodes' locations during the design time. Hence, it is necessary to find suitable sink placement strategies for different target networks that guarantee scalability and low delay without degradation of lifetime performance.

In the case of mobile sink placement, the locations of sinks are changing from time to time within the network area. Actually, mobile sink placement problem can be viewed as a sink trajectory problem. Depending on the movement of mobile sinks, a sink trajectory can be defined as discrete or continuous. A sink trajectory is seen as a discrete type if the movement of a sink is abstracted as a sequence of static sink placements assuming that the time scale of sink mobility is much larger than that of data delivery. In contrast, a continuous type of sink trajectory collects the data within a short period of time at each sink candidate location or during the movement without stopping anywhere along the trajectory. With a slow mobility (i.e., a discrete type), the complexity of problems such as packet loss or dynamic topology changes can be reduced effectively. In this dis-

sertation, we investigate the first approach for the sink trajectory problem. Using sink mobility, we face a conflict between lifetime maximization and delay bound minimization in large-scale, time-sensitive WSNs. The challenge thus becomes the search for good trajectories for the sinks such that lifetime and delay goals are met simultaneously.

Additionally, a proper node placement scheme can reduce the complexity of problems in WSNs as, for example, routing, communication, data aggregation, energy consumption, etc. While a random node placement is preferable in many scenarios, deterministic node placement comes to our attention due to its better performance. Therefore it is necessary to analyze deterministic node placement strategies for large-scale and time-sensitive WSNs more deeply. In particular, tiling-based node placement strategies are taken into account in this dissertation. However, we should be aware about deployment errors due to environmental and geographical impacts or node failures. In order to apply realistic node placement strategies which are useful in real world scenarios, network designers should investigate the effects of environmental and geographical disturbances on deterministic node placement strategies.

1.2. Methodology

To evaluate lifetime performance of WSNs, a simple energy model is formulated to study energy consumption at each sensor node. Of course, a very accurate energy estimation would be desirable because knowledge about the node level energy behavior is necessary to evaluate the energy consumption distribution. However, this would need to be investigated starting from the transistor level, taking into consideration leakage, etc. We take a more abstract view and define a rather simple energy model for the assessment and performance comparison of design strategies in WSNs. Our model mainly highlights the energy consumption of the transceiver unit, since the energy consumption of the processing unit is relatively the same for all nodes and, as such, can be taken as a constant. Thus, energy consumption for security, routing, and data aggregation is not taken into account. We define a model which concerns the 1 bit energy consumption of sensing, transmitting, and receiving for all nodes when communicating to their nearest sinks. The energy model is based on the popular MICAz mote [7].

While an average-case analysis is useful in some applications, for time-sensitive WSN applications it must be ensured that messages indicating

dangerous situations are not lost and they should be guaranteed not to exceed a maximum worst-case delay. This can be achieved by a new methodology called sensor network calculus (SNC) [115]. Based on network calculus [71], SNC was proposed and customized as a framework for the worst-case analysis in WSNs. A tool exists called DISCO network calculator [116], which is an open-source toolbox for the worst-case analysis. The network calculus operations and different network analysis algorithms can easily be used from this library: for example, end-to-end service curve calculation, total flow analysis, separated flow analysis, PMOO analysis, etc. Using the foundation of sensor network calculus, we calculate the worst-case end-to-end delays for each flow and find the maximum worst-case delay in the field. A related work on which we build in this research is the previous research on SNC [115, 118, 119].

1.3. Dissertation Overview and Outline

WSNs became mature and smart in many sectors with the support of emerging technologies in embedded systems. At the same time, large-scale and time-sensitive WSN applications are more and more demanding in our environmental and cyber-physical systems. So far, we know that energy-constraints are inevitable in WSNs. For these networks to become integrated in our surroundings, lifetime prolongation with minimal energy consumption and minimizing the worst-case delay bound are crucial. Also, scalability is a fundamental issue for every design. In order to scale well with delay and lifetime, large WSNs require *multiple* sinks. Therefore, it becomes very interesting to investigate how to place multiple sinks in conventional WSNs and how to plan the trajectories of multiple mobile sinks in mobility enabled WSNs such that lifetime and delay goals are met simultaneously. Due to the hardness of the problem, the problems of multiple sinks placement and multiple sinks trajectories are solved with heuristic approaches.

In chapter 2, we survey various design issues for large-scale and time-sensitive WSNs. In each issue, we discuss the existing strategies and their methodologies together with pros and cons. Furthermore, a realistic energy model basing on MICAz motes [7] is introduced. The details of SNC operations and analysis methods will also be discussed there.

In chapter 3, we examine a uniform random node placement and tiling based deterministic node placements are investigated. The goal of this chapter is the introduction of tiling-based node placements for WSNs and

providing alternative placements for the application designers to meet their desired goals. We analyze coverage, energy consumption, and delay of each node placement strategy. We first study the performance of the strategies while assuming an exact placement. Since various difficulties can occur in exactly placing sensor nodes in WSNs due to environmental and geographical constraints, we study a stochastically disturbed version of the strategies. Finally, we analyze the tradeoffs between these performance metrics for each placement strategy with and without disturbance effects to show which strategy is preferable under what factors, e.g., the number of nodes.

In chapter 4, heuristic-based static sink placement strategies are proposed for time-sensitive WSNs. For different target networks we develop two sink placement strategies: a Genetic Algorithm-based sink placement strategy and a self-organized sink placement strategy under different assumptions. Performance is analyzed and compared to other strategies having the same level of abstraction.

In chapter 5, mobile sinks are introduced for large-scale and time-sensitive WSNs. Since finding an optimal sink trajectory is very hard to achieve, we relax the problem by giving it a geometric interpretation, and introduce a heuristic framework which we call orbital sink trajectory. The orbital sink trajectory is built on a geometric reduction of the problem, where the two performance characteristics, delay and lifetime, are amalgamated into minimizing the Euclidean distance between nodes and sinks. The intuition behind this is that both, delay and lifetime, benefit from nodes being closer to their assigned sinks. By simulation, the validity of analytical results are confirmed and the performance of the orbital sink trajectory for multiple sinks is evaluated and compared to several alternatives.

In chapter 6, we summarize the dissertation with its original contributions and discuss the future work.

2. Design Issues in Large-Scale, Time-Sensitive WSNs

2.1. Introduction

Research in the area of WSNs can be classified into three different levels: component level, system level, and application level [113, 112, 45]. Research at the component level is mainly concerned with hardware issues such as sensing, communication, and computation capabilities of sensor devices in WSNs in order to achieve low cost and high performance. The system level research focuses on deployment, communication, and networking issues in an energy-efficient and scalable manner. Research at the application level is very specific. The data collected from the sensors is being processed according to the application's requirements.

Clearly, various design issues arise at each level. In each design, the required information and the level of abstraction is different. An interesting question is: what is the required information for each individual design in order to achieve an optimal design? For application-specific and resources constrained WSNs, the answer of that question is not easy to find. Since the applications of WSNs are very broad, it is very difficult to fulfill the requirements of each individual application. It is debatable which information is important enough to be explicitly considered as a design parameter. Thus, it is very likely that one could argue in favor of adding more information or removing some from others' suggestions. An obvious thing is that there always exists a cost-performance trade-off associated with each design problem. The cost issue not only relates to the hardware and installation cost but also to the requirement of global information, the usage of energy, etc.

Before we discuss the important design issues, we first want to highlight the assumptions taken into account for each design issue. We know that the level of abstraction varies in each design issue. For instance, deployment designs are more abstract than the others because, in general, net-

work information is not known in advance. As a result, the designers have to make several assumptions. During design time, some assumptions are inevitable, while some are flexible and some are strict. Another important thing is that assumptions must be cost-effective under the consideration of large-scale WSNs. One can imagine that the designers can provide an optimal design only if all information of the application is assumed to be given. In general, the more information we have the better the design and the higher performance should be possible to provide. On the other hand, we encounter high computation and communication overheads which are very expensive for large-scale WSNs. For example, in a dynamic routing design that requires global information like nodes' locations, sensor nodes may keep the locations of some of the other nodes or may distribute the locations of itself and/or its neighboring nodes to other sensors. Obviously, such a design has high communication overheads.

To be able to apply a proposed design in large-scale WSNs, the most important feature is its scalability. Scalability plays a vital role in large-scale WSNs. Any design, protocol, and algorithm must be able to maintain its performance with a large amount of sensor nodes since WSNs are expected to be composed of thousands of nodes. There are already WSN deployments, such as [79, 91]. A universal method for solving the scalability issue is the clustering approach [151, 52, 99, 65]. Along with clustering, multiple sinks are considered to maintain performance and to get a better manageability. By dividing the network into smaller sub-networks, a global problem is reduced to several (similar) local problems of a smaller scale. Indeed, such transformation results in a better management, lower communication overheads and energy consumption especially for routing and in-network aggregation of the data in WSNs. The hardest part of clustering is constructing optimal clusters to achieve the desired goals with minimal cost.

In this chapter, we discuss briefly about the important design issues at each level for large-scale and time-sensitive WSNs. Among them, we mainly focus on deployment designs starting from Chapter 3 onwards. Since the goals of our designs are maximizing lifetime and minimizing the worst-case delay, the methodologies to evaluate them are also presented in this chapter.

2.2. Design Issues at the Component Level

A WSN comprises of a collection of autonomous sensing nodes, each equipped with physical sensors, modest computing ability, memory, a wireless transceiver, and a power source [61]. Component level designs are concerned with sensor node hardware. Physical sensors such as pressure, humidity, and sound are used to measure physical and environmental conditions. Sensed data are being processed according to application needs. Memory is used to buffer the intermediate data and store global information, e.g., the locations of sinks. Communication is handled by the radio transceiver, i.e., receiving and transmitting data over the wireless channel. The power required for the above operations is provided by a battery unit. Among a variety of design issues at the component level, we briefly discuss hardware designs and operating system designs in this section. As mentioned above, achieving low cost and high performance are primary goals of these design issues.

2.2.1. Hardware Design

When designing hardware components for a WSN, the designers have to pay attention to reduce size, costs, and energy consumption for processing and wireless radio communication of the sensor node. A small node size is preferable so that nodes are unobtrusive and attachable everywhere. The latter case is necessary for some applications like intrusion detection where sensor nodes should not be noticeable for the intruders. Cost is a very crucial issue not only for the hardware design but also for the usability of the WSN in the real world, especially for large-scale WSN applications. On the other hand, performance and cost effectiveness have to be traded off. The performance in hardware design is mainly concerned with computation and communication capabilities such that these capabilities have to provide an acceptable service. To compromise cost and performance issues, cost-effectiveness and energy-efficient hardware designs are considered for data processing and wireless radio communication. For instance, micro-electromechanical systems (MEMS) sensor generations made a lot of progress in micro sensor technology.

The range of WSN platforms varies from lightweight platforms to high performance platforms. Some of them are compared in Table 2.1. The well known lightweight platforms that are commercially available are Mica2, MicaZ, TelosB, Imote, and BT [5, 2]. These nodes are very popular and widely used in various research organizations. Except for the BT node, the

Table 2.1.: Comparison of wireless sensor network platforms.

Mote	Microcontroller	Data/Storage Memory	Radio	Data Rate
Mica2	ATmega128l/8-bit	4 / 512 KB	CC1000	38.4 Kbps
MicaZ	ATmega128l/8-bit	4 / 128 KB	CC2420	250 Kbps
TelosB	TI MSP430/16-bit	16 +10 RAM / 48 KB	CC2420	250 Kbps
Imote	ARM7/32-bit	11 / - KB	Zeevo TC2001	723.2 Kbps
BT node	ATmega128l/8-bit	64 / 180 KB	Zeevo ZV4002	723.2 Kbps
XYZ	OKI ML67Q5002/32-bit	32KB+2MB RAM/256 KB	CC2420	250 Kbps
Stargate	Intel XScale/32-bit	64 / 32 MB	IEEE 802.11b	1–11 Mbps

above mentioned platforms are from Crossbow Technologies [5]. In contrast to these general purpose platforms, some platforms focus on a specific application. For example, in [36], the authors presented the design of the eXtreme Scale Mote, a new sensor network platform for reliably detecting, classifying, and quickly reporting rare, random, and ephemeral events in a large-scale, long lived, and retaskable manner. The Yale XYZ platform, which is a typical example of ranging above the lightweight platforms, has more memory and processing resources than the lightweight platforms. However, intermediate platforms are still equipped with low bandwidth radios (e.g., the IEEE 802.15.4 compliant CC2420). Thus, real-time high volume data streaming is not possible with these platforms either. At the higher performance level, there are PDA-class platforms (e.g., Stargates) which are more powerful than the intermediate platforms. PDA-class platforms equipped with IEEE 802.11 radios are suitable for real-time high volume data streaming due the highly capable processors and large memory available on board. However, the energy dissipation of these devices is more than an order of magnitude higher than the energy dissipation of lightweight platforms. Furthermore, it consumes considerably more energy for communication. For that reason, these nodes are suitable to use as gateway nodes or sinks because they are usually assumed to be resource unconstrained.

Although, many improvements can be found in this research area, on the one hand, low power processor and radio designs for computation and communication, and cost-effective physical sensors and processing unit are still desirable under the inevitable limited battery problem. On the other hand, long lived battery and sensor nodes integrated with energy harvesting circuits (e.g., solar cell) designs are active research areas [120, 26, 12].

2.2.2. Operating System Design

The operating system for WSNs can be viewed at two levels: node-level or local systems and network-level or distributed systems [108]. The important issues related to node-level systems are limited resource management, concurrency handling, and power management whereas issues related to both systems are inter-node communication, failure handling, heterogeneity, and scalability.

As we know that energy is limited, an operating system has to provide the necessary mechanisms to optimize the power consumption for prolonging the lifetime of the WSN. Some efficient methods are the periodic sleeping of nodes, controlling scheduling policies, and adjusting the duty cycle. In general, sensor nodes operate in three sleep modes: idle, power down, and power save. In idle mode, only the processor shuts down. In power down mode, everything is shut down except the watch dog timer and interrupt logic necessary to wake up the node. The power save mode is similar to the power down mode except that a timer is running. By allowing periodic sleeping of nodes, the nodes can save energy effectively. Note that switching modes between sleep and wake up should be energy-efficient in case an application needs to wake up frequently. Otherwise, the complexity and cost for switching modes will be negatively effected. An alternative method is that the operating system should properly schedule the tasks to the processor to optimize the processing power. At the same time, the operating system should be able to handle concurrent operations at a given point of time since a sensor node may be doing more than a single task, for example, sensing data from itself and collecting data from others while transmitting data to the sink. Operating systems should handle these situations in an energy-efficient way. Otherwise, tasks are lost and a node consumes more energy for repeating the same task again. Adjusting the duty cycle is a simple but efficient way of saving energy in sensor nodes. This method is relevant to periodic data sensing applications.

Besides node-level issues, network-level issues are equally important in operating system designs. In a distributed environment, an operating system should provide an optimal inter-networking among components of each individual node. The operating system designs should be robust, i.e., able to handle disconnections between nodes in case of node failures or topology changes. That means the running application should not be effected under these conditions. Like other design concerns, heterogeneity and scalability have to be carefully considered in the operating system design. Further important characteristics are a flexible architecture, an efficient execution

model, real-time features, resource management, etc.

Existing operating systems for sensor nodes can be classified as event-driven, thread-driven, and hybrid designs according to their execution model. The choice of the operating system should be dependent on the application's desired goals. In an event-driven system, tasks are executed in FIFO order as scheduled. Preemption is not possible. The most popular event-driven operating system for sensor nodes is TinyOS [3]. In a thread-driven system, tasks are preempted by the scheduler for other high priority tasks to execute. An example of a thread-driven system is MOS [22]. We can see that preemption is a key difference between event-driven and thread-driven systems. An event-driven system is assumed to have a better performance in constrained environments whereas a thread-driven system has advantages with high concurrency. Although preemption is a key advantage in thread-driven system, it usually has higher energy consumption. A hybrid approach combines the advantages of event- and thread-based operating systems. An example of a hybrid approach is the Contiki operating system [35]. A highly concurrent operating system design is desirable for real-time WSN applications, but at the same time, energy consumption should also be minimized. Although there are studies on this topic [145], advanced designs and methodologies are still required.

2.3. Design Issues at the System Level

2.3.1. Deployment Design

Deployment is defined as setting up an operational sensor network in a real world environment [109]. The primary design issues where major research activities are going on in deployment designs are node placement and sink placement.

Node placement can be a one-time activity or a continuous process. In the former case, sensor node positions are fixed, no replacement of failed nodes is considered. In contrast, the latter case takes additional node deployment at any time during operation into account in order to maintain the performance of the system. In a fixed node placement, nodes are distributed in a random or deterministic fashion. In a random deployment, sensor nodes can be deployed either by placing one after another in the area of interest or by dropping them from a vehicle, e.g., a plane. Random node distribution is not desirable with respect to system performance but it is assumed to be a cost effective and unsophisticated deployment. Post-processing of random distribution for high performance such as self-

configuration of sensor networks without human intervention is considered in [74, 142]. Such an option has several challenges for applications, e.g., due to environmental conditions. As a performance-oriented design, a deterministic node placement comes into attention. Regular pattern-based node placement is proposed in WSNs as a good deterministic node placement [14, 16, 15, 17]. In most aspects, a regular pattern placement outperforms a random placement. For this reason, many works take the assumption of deterministic node placement in their designs, protocols or algorithms. In some applications, post-processing of node placement is mandatory. To that end, node mobility is a solution to perform this task in order to increase or recover coverage or connectivity of the network. Research in node mobility can be found in [33, 19, 134, 97]. Node placement strategies play a very important role in providing better QoS, for example, coverage performance like, how well each point in the sensing field is covered. However, due to severe resource constraints and hostile environmental conditions, it is nontrivial to design an efficient deployment strategy that would minimize cost for communication and the required number of sensors, while maximizing the area coverage, and maintaining a globally connected network.

Coverage or area coverage is one of the fundamental problems related to node placement design. It has been observed that different applications require different degrees of coverage in the sensing field. Some applications require a high degree of area coverage, for example, a military surveillance application would want a region to be monitored by multiple nodes simultaneously in order to protect from node failures and capture high quality information. Whereas some monitoring applications like a temperature sensing application might require only a low degree of area coverage. Besides, some applications may need to adjust the coverage degree dynamically, for instance, an intrusion detection system may need to change the degree of coverage at a restricted region in case a threat or an intrusion takes place. Nevertheless, high coverage is a key for robust systems and makes it feasible to extend the lifetime by allowing power-saving sleep nodes. A drawback is the increasing node redundancy, which is an expensive option for a large-scale network. The problem of guaranteeing coverage while achieving application requirements often time rises to NP-hard problems [37, 38]. To solve the coverage problem, many works proposed methods relying on computational geometry and graph theory [38, 75, 89, 138, 90]. In computational geometry, the problem of coverage based on the sensor deployment is related to the traditional art gallery problem (AGP) [98]. In graph theory, the coverage problem is defined

based on exposure paths. The well known methods of exposure paths coverage problems are minimal/maximal exposure path, maximal breach path, and maximal support path [75, 89, 138, 90]. For example, Meguerdichian et al. [90] suggests constructing a Voronoi diagram for the set of nodes in order to compute the maximal breach path such that the path that maximizes the distance between the target and the nearest sensor node. The points on the edges of the Voronoi diagram provide the maximum distance to the given sensors. Among various coverage types, the k-coverage [69] or minimum coverage problem has received much attention in the literature. Along with coverage, node density may vary widely, ranging from very sparse to very dense.

A significant deployment problem in WSNs is sink placement. Many studies [99, 65, 139, 55, 107, 140] have already proven that the locations of sinks strongly influence the performance of the WSN applications. The number of sinks is lower bounded by one, but can be arbitrarily high, depending on the size of the network and the application demands. For large-scale WSNs, using multiple sinks is a key idea for scalability and better manageability. In conventional WSNs, sinks' locations are fixed. The data gathered by the nodes is forwarded to the sinks via multi-hop communication paths and thus is forming a many-to-one traffic flow problem. Due to the many-to-one architecture, the sensor nodes nearer to the sink deplete their battery quickly by carrying heavy traffic loads from other sensors. As a result, the so called energy hole or hot-spot problem appears, which causes that no more data can be delivered to the sink and the WSN is not able to continue its operation properly. Some studies [148, 80] suggest to deploy non-uniform node distribution or non-uniform battery assignment to solve this energy hole problem; however these approaches cannot eliminate the problem. Recently, an elegant approach of sink mobility was introduced to eliminate the hot-spot problem: by moving the sink location from time to time, the lifetime of a WSN improves significantly [21, 121, 82, 92, 144, 29, 44, 125]. In the general case, the primary objective of static sink placement or mobile sink placement is lifetime maximization. In addition to the lifetime goal, the worst-case delay bound plays a vital role among other performance metrics for time-sensitive WSN applications. In order to control the performance of end-to-end information transfer delay, sink placement becomes the most promising strategic option in WSNs because the locations of sinks heavily effect the traffic flows and hop-distance between sensor nodes and sinks. Other important characteristics for sink placement designs are scalability, load balancing, mobility related issues, synchronization, and so forth.

2.3.2. Communications Design

Along with deployment designs, the notion of connectivity is equally important in WSNs because the goal of an optimal sensor deployment strategy is to have a globally connected network. Hence, some studied connectivity issues in deployment designs, for example, the connectivity problem related with coverage issue is addressed in [14, 157]. Since connectivity is a foundation for communication functionalities, we discuss the connectivity issue under system level designs. Two nodes are connected if and only if they are located inside each others communication range. In other words, there exists a connection between two nodes if both can hear each other. Three types of connectivity are available in WSNs [112]. A network is known to be *fully connected* if there exists a single-hop or multihop communication path between any two nodes. By ensuring that the network is fully connected, it is also ensured that the sensed information can be transmitted to other nodes and finally to a sink which is a necessity for WSN applications. A network has an *intermittent connectivity* if there exist partitions in the network. This type of connectivity requires relay nodes or gateway nodes between partitions in order to be fully connected. If nodes are outside of the others' communication ranges most of the time, such connectivity is called *sporadic*. Connectivity strongly effects the design of communication protocols and data gathering algorithms in order to provide high robustness and throughput for a sensor network. Like k-coverage, k-connectivity is considered in WSNs. k-connectivity means that at least k other nodes fall within the transmission range of each node. A sensor network is said to have k-connectivity if removal of any $k-1$ nodes does not render the underlying communication graph disconnected. Different approaches of the connectivity problem have been addressed in the literature. One way is assigning different transmission ranges to the sensors to guarantee a connected network. The problem of finding the critical transmission range has been addressed in [114] under homogeneous node distributions. An alternative approach is using deterministic node placement, where each node has k-connectivity [14]. Almost all available connectivity models use simplifying assumptions such as a disc-based transmission range, uniform transmission ranges, etc. Since real radios are much more complex than the simple model, these complexities have a strong impact on the behavior of communication protocols and algorithms [159]. A detailed study of simplifying assumptions for radio propagation has been addressed in [66]. Although such assumptions are unavoidable at the design level, a realistic stochastic propagation models should be developed to suit different

environments and to verify the protocols or algorithms.

With a connected network, a good routing design should consider uncongested routes from nodes to the sink for a reliable data transfer. At the same time, it should minimize end-to-end delay and energy consumption. Therefore, the key idea is to avoid congestion by balancing the load over nodes along the shortest route. In earlier work [110, 111], the routing problem is modeled as a classical graph flow problem such as the shortest path tree (SPT) or spanning tree (ST) problems. The minimum total energy routing algorithm in [110] finds a path with minimum total energy consumption. In [111], a minimum hop algorithm is introduced for energy-efficient routing. Later, a new paradigm of cross-layer design, an opportunistic routing, has been developed in WSNs, e.g., [158, 160]. The opportunistic routing schemes use the network and MAC layer simultaneously such that the MAC layer selects a node to forward the data to, and the network layer selects the best candidate nodes to receive the data. Although the opportunistic routing is an efficient routing scheme to overcome the problems of unreliable links or unstable end-to-end connections, a drawback is that it consumes a lot of energy due to multiple broadcasts. Recently, geographic routing has been shown to be a promising method for efficient point-to-point routing in large-scale WSNs. In order to apply such a routing algorithm, a node must have knowledge about the positions of its 1-hop neighbors, the destination, and itself. Then the node forwards the data to the neighbor which is the closest to the sink. Such a kind of routing is suitable for event driven execution. The advantage of a geographic routing is its statelessness and that forwarding is performed by local decisions. Being a localized algorithm, geometric routing scales better in dynamic networks, where a global algorithm would need to track every topology change. However, the drawback of geographic routing is the requirement of nodes' locations. Additionally, local minima (or) dead ends may arise. With the help of GPS or other localization methods, nodes can estimate their 1-hop neighbors, however, this is expensive for large-scale networks. The dead end problem or unreliable end-to-end connections will occur when a node does not have a neighbor closer than itself to the destination. To avoid the dead end problem, additional non-local state or auxiliary mechanisms are considered in geographic routing (see in [43, 133, 25, 62]). When a packet reaches communication holes or a dead end node [62], for example, uses a distributed perimeter forwarding algorithm that routes around such obstacles. An alternative approach is based on the idea of divide and conquer: the network is decomposed into components [132, 63] where greedy routing is likely to perform well, and then

a global structure is used to assist inter-component routing. One of the simplest globalized routing protocols is shortest path routing. Obviously, shortest path routing has higher overhead at each node than any localized scheme. Each node has to update the global information of other nodes whenever the network topology changes. On the other hand, such a routing design can avoid dead ends and can handle frequent topology changes. Due to the dynamic and unpredictable nature of wireless communication and mobility, reliability and self-organizing functions have to be carefully planned in advanced routing designs.

2.3.3. Network Design

The network topology severely affects many important network characteristics such as delay, robustness, and capacity. The complexity of routing and data gathering also relies on the topology. We can use several network topologies to coordinate nodes and sinks in WSNs. In the simplest form, where each node is able to communicate to the sink via a single hop, a WSN has a star topology. This topology is applicable for small-scale WSNs applications. Although it is simple, long distance communication is not desirable in WSNs due to energy constraints. A multi-hop network may form an arbitrary topology such as a tree or a set of connected stars. If multiple sinks are available in a multi-hop network, one approach is to implement a cluster or tree topology. Under a tree topology, it is clear that some nodes act as routers and thus they consume more energy than leaf nodes. A drawback is that all downstream nodes lose their connections to the sink if their root router node depleted its energy. To solve such problems, mesh topologies can be used in WSNs, where redundant communication paths are available to increase the reliability of the system. A mesh topology in WSNs can be viewed as a set of connected star topologies. By providing multiple paths from nodes to the sink, the network is able to automatically reroute the packets to the sink in case one router node fails to forward the data [43]. This way, the network can prolong the lifetime to some degree. A drawback is that mesh topologies maximize the information transfer delay which is undesirable for time-sensitive WSNs. An alternative approach is a backbone-based topology which is introduced to guarantee the connectivity of the whole network. In fact, it is a type of bus topology. Every node except the backbone nodes must have at least a connection to one of the backbone nodes thus guaranteeing 1-connectivity. The problem is also known as connected dominating set. Finding the minimum connected dominating set is shown to be a NP-hard problem [155]. [149] proposes a

distributed algorithm for the k-connected m-dominating set problem with time complexity $O((m + \triangle) \times Diam)$ where $\triangle$ is the maximum node degree and $Diam$ is the network diameter, i.e., the maximum number of hops between any two nodes. Since the complexity of related design issues is pretty much dependent on the selected topology, the designers should carefully choose the best topology for the application. Further, topology problems become much more complicated if the WSN is dynamic due to mobility enabled nodes.

2.4. Design Issues at the Application Level

2.4.1. Data Gathering

Popular data reporting models are time-driven, query-driven, or event-driven. In a time-driven model, sensors periodically sense the data from the environment and transport the data to the sink. Such a kind of data reporting can be found in environmental monitoring applications, for example, CO_2 emission level is measured every day to control air pollution of industrial areas. In a query-driven model, data is reported to the sink according to the query, for example, the control center wants to know the path of military vehicles (e.g., tanks) with tracking applications [6]. When the query enters the network, nodes collaborate in estimating the path and tracking results are transmitted to the control center. An event-driven data reporting model is similar to a query-driven model. Nodes are activated by the event instead of a query. A typical application using the event-driven model is intrusion detection where sensors report whenever any suspicious event is detected. Applications like tracking and detecting have to guarantee data reporting from sensors to sinks within a given deadline.

Although different data reporting methods are available for individual WSN applications, data gathering is the main task of every application. A typical data gathering involves a systematic collection of sensed data from all sensors and transmitting the data to the sink via efficient routes. Actually, data gathering is strongly effected by the design of the routing protocol. Previous work in routing algorithms was mainly concerned with energy-efficient and uncongested paths without considering data aggregation. Since the data generated from sensors is often redundant and also the amount of data generated may be very huge for the sink to process it, additional methods are required to improve the data gathering process. Recently, many studies [39, 72, 78, 49] addressed the routing problem

together with mechanisms to gather and to process data more efficiently. Among them, in-network aggregation is known to be the most efficient way to reduce redundant transmissions from multiple sensors by estimating the desired data and providing aggregated information to the sink.

In-network aggregation is the global process of gathering and routing information through a multihop network, processing data at intermediate nodes with the objective of reducing resource consumption (in particular energy), and thereby increasing network lifetime [39]. While, in-network aggregation solves redundant transmissions and reduces energy consumption at each node, it can be considered a relatively complex functionality because the aggregation algorithms should be distributed over the network which requires coordination among nodes in order to achieve better performance. A typical way of data gathering is using a gathering tree from nodes to the sink where the sink acts as the root, e.g., [72]. Usually, flooding is used to construct such a gathering tree. Using a beaconless protocol flooding is very simple, but, it is not energy-efficient because nodes have to forward data as long as they receive message from their upstream nodes. Previous work in [49, 51, 78, 77] focused on preserving energy from sensors during the process of data gathering. In the directed diffusion protocol [49], the gradients are set up for data flows from source to sink during the interest/query dissemination from the sink. By aggregating the data at intermediate nodes, this approach achieves significant energy savings. An alternative approach is used in the spin protocol [51] where metadata negotiations are used between sensors to eliminate redundant data transmissions through the network. In [77], sensors form chains so that each node transmits and receives from a nearby neighbor. Gathered data moves from node to node, gets aggregated and is eventually transmitted to the sink. Next, a clustering approach is introduced in data gathering. LEACH [52] was an early proposal for a randomized clustering protocol that demonstrated some of the gains of in-network compression and its relation to routing. The LEACH protocol is a cluster-based approach where each cluster head participates in data gathering. In order to balance energy consumption, the cluster head is reselected from time to time. In [78], the authors propose a hierarchical scheme based on [77] that reduces the average energy and delay incurred in gathering the sensed data. Note that all the above studies work well in static networks where the topology is fixed. In order to handle data gathering in a dynamic environment such as under node mobility, alternative approaches have been proposed [43, 13]. Even multi-path algorithms may be able to deal with topology changes, but the efficiency is limited [94]. A reactive and local-

ized routing protocol is proposed in [13] where MAC layer issues are taken into account. A hybrid approach can be found in [86], which combines the properties of tree-based and multi-path schemes. Nevertheless the existing data gathering approaches are only able to deal with limited topology changes and thus several issues are still open with respect to more dramatic topology changes as seen in mobility enabled WSNs. Although some initial work addressed this topic, a deeper analysis of the aggregation functions is not sufficiently well studied.

2.4.2. Other Technical Issues

In WSNs, many technical issues still need to be addressed more comprehensively and be solved. Among them, mobility, self-organization, and heterogeneity are discussed here.

Enabling mobility in WSNs changes the whole infrastructure of some important functionalities. Most severely effected research areas are topology, routing, data gathering and aggregation, connectivity, and deployment. While the degree of network dynamics has a large impact on the design of networking protocols and distributed algorithms, the trajectory and the speed of movement strongly relies on the deployment, data gathering, and connectivity issues. On the other hand, a mobile enabled WSN gains significantly in lifetime prolongation which, in fact, is the most prioritized issue among energy-constrained sensor nodes. Naturally mobility can apply to both nodes and sinks. To trade off between network dynamics and lifetime, partial mobility (rather than full mobility) of nodes and sinks are considered. Furthermore, the degree of mobility, i.e., the types and velocity of movement, may also vary from discrete to continuous movement. The key idea of using node mobility is to reconfigure the coverage and connectivity in order to solve the energy hole problem. For large-scale WSNs, only partial node mobility is considered due to complexity, cost, and environmental conditions. Although several challenges are created by sink mobility, a great advantage is the maximizing of the lifetime of WSNs by eliminating the classical hot-spot problem which is inevitable in traditional WSNs. Hence, mobility has to be considered as a parameter in every design of WSNs.

Along with mobility, self-organizing designs and algorithms are preferable due to the dynamically changing environment. Self-organization can be applied at all design levels. In the component level designs, self-organization is considered to accomplish specific functions such as sharing and managing processing and communication capacity. In the system level design, self-

organization helps in forming and maintaining structures and in adapting behavior associated with routing. If a network needs post-processing after deploying nodes across a geographic area, self-organization could form an initial topology by adjusting the power level of nodes or by injecting replacement sensors. In case of losing connectivity due to, for example, node failures, self-organization could provide topology reconfiguration by moving nodes to new locations or by adjusting the transmission ranges of some nodes. In the application level design, self-organization provides adaptive processing functionalities associated with disseminating and querying for information, with assigning tasks, configuring software components and resilience by repairing faults and resisting attacks. A drawback of self-organization is that it produces a higher complexity and overhead.

Another important characteristic of future WSN applications is heterogeneity. Many prototypical systems available today are composed of a variety of different devices and sensors. That means some nodes may support powerful computation; some nodes may have special hardware like Global Positioning System (GPS), some nodes may have special sensing devices such as cameras, some nodes may have higher battery levels to act as gateways, etc. The degree of heterogeneity is an important factor for designs and management of the whole system.

Although a large amount of existing work is available, WSNs still remain an exciting and open field in the research area.

2.5. Two Objectives in Large-Scale, Time-Sensitive WSNs

Research activity in the area of WSNs has grown dramatically in the past few years and was motivated by a vast array of potential applications. It includes for example, environmental monitoring, intrusion detection, security and surveillance, precision agriculture, utility plant monitoring, health monitoring, battlefield scenarios, building management, and disaster recovery. Unlike traditional networks, WSNs may consist of several thousands of sensor nodes for unattended operations. Consequently, various design and control techniques used in traditional networks cannot be applied directly to WSNs.

Typical scenarios for large-scale WSNs contain multiple sources and multiple sinks. Compared to a single sink, the use of multiple sinks results in a better manageability of large-scale WSNs. In a general WSN application, it is desired to collect the information acquired by sensors for processing,

archiving and other purposes. A sink normally has a higher capacity as well as cost than usual sensor nodes. Sinks can be sensors themselves or devices such as PDAs or gateways to other larger networks [61]. In WSNs, sensor nodes are not only sources but also act as routers. If a sensor node acts as a router in a large-scale WSN, it can be experiencing quite high amounts of data flows from other sensors or routers. Clearly the router nodes deplete their battery rapidly. The condition is even worse if the sinks are distributed in improper locations. In case of an improper deployment of sensor nodes and sinks, it is difficult to control a running network and to that end energy consumption and system performance are negatively affected. Additionally, many nodes would be far away from the sink and thus many hops must be traversed before the sink is reached under multi-hop communication scheme. As a result, response times become excessive and the lifetime of the WSN becomes very short and a difficult replacement of batteries if sensor nodes are deployed for example in a forest or in the ocean has to be performed. If sensor nodes use high transmission power which means they enforce as direct communication to the sinks as possible, the high delay problem can be avoided but energy will be depleted very fast. Therefore, it is sensible to deploy multiple sinks as optimally as possible so that messages reach their destination with less hops and consequently response times are decreased and energy is saved.

Energy consumption is often assumed to be the most critical issue in WSNs due to operation on batteries, which probably constitutes one of the main differences from traditional networks. The sensor node TelosB, for example, mote has a lifetime of roughly ~ 90 hours with $100\,\%$ duty cycle with the initial capacity provided by energizer ultra+ batteries [95]. Here 100% duty cycle means the node's radio is always on, no sleep and no communication at all. Therefore the lifetime of a WSNs is very limited. One may argue that there are several promising methods for energy scavenging which, however, need additional circuitry and cost. For that reason, all design and control processes are usually focused on minimizing energy consumption as much as possible as we discussed in previous sections. A protocol is called energy-efficient if it minimizes the cumulative energy consumption while completing its task. Note that an energy-efficient protocol does not necessarily maximize the network lifetime. In WSNs, communication for data reception and transmission between nodes is the highest energy consumer. In order to reduce the energy consumption of communication, WSNs were improved by the use of multi-hop communication instead of direct communication. Adjusting the duty cycle is also a powerful way of optimizing energy consumption.

Additionally, finding the optimal location of sinks in a way that the total energy consumption or the total load from nodes is minimized is considered for lifetime prolongation [99, 65, 139, 55, 107, 140]. Furthermore an effective way of lifetime prolongation by mobile sinks is introduced in [82, 20, 144, 125]. By changing the sinks' locations from time to time, the set of nodes nearer to a sink is changing as well and thus balancing energy consumption at each node. Along these lines, lifetime of WSNs can be maximized. By carefully designing WSNs, we prolong the lifetime which can be seen in the next chapters. In order to evaluate the lifetime of WSNs, we define an energy model in Section 2.6.2 which concerns the 1 bit energy consumption of sensing, transmitting, and receiving for all nodes when communicating to their destination sinks. A realistic energy model is considered basing on MICAz motes [7].

Depending on different criteria, network designers have to plan WSNs, which do not only optimize energy consumption but also achieve a high system performance. Performance issues in WSNs play a vital role in many applications. There exist time-sensitive applications where the response time is crucial in order to ensure a timely actuation in case of certain events. For instance, in a production surveillance or fire detection application, it must be ensured that messages indicating dangerous situations arrive at the control center with minimum delay. Thus the maximum allowable message transfer delay must be bounded and consequently it is crucial to develop algorithms to minimize the maximum worst-case delay in WSNs. For that reason, the dissertation takes the worst-case delay as one of the primary metrics in WSNs into account. In particular, we focus on strategies to minimize the maximum worst-case delay, which is important for any timely actuation based on the information collected by a WSN.

To model and consequently control the worst-case delay of a given WSN we build upon the so-called Sensor Network Calculus (SNC) (a recent methodology introduced in [115]). Network calculus [71] is a framework for worst-case analysis which allows to calculate maximum message transfer delays, maximum buffer requirements at each sensor node and lower bounds on duty cycles. SNC customizes the conventional network calculus to the typical setting in WSNs. Some details of SNC operations and analysis methods will be discussed in Section 2.7.

2.6. Lifetime

In WSNs, the lifetime is defined as the time span until the network cannot fulfill its task anymore. It is likely that the concrete lifetime definition may differ from one application to another. Among various lifetime definitions, the following are known to be the most often used ones.

2.6.1. Lifetime Definitions

Definition 1: The time span starting from initial deployment phase until any node depletes its battery.

This is the most common lifetime definition for WSNs applications. This type of definition is applicable for some WSNs applications where area coverage is the heart of the application requirements. In military applications such as battlefield surveillance, area coverage is the most sensitive issue where the application will totally fail in case of a single node failure. Although the definition is somewhat arguable, it is assumed that the rest of the nodes deplete their batteries soon right after the first node dies. This holds especially in homogeneous network.

Definition 2: The time span starting from initial deployment phase to the first loss of coverage of a region of interest.

This type of definition is well suited for applications where a single node failure is not so dramatic. For example, in an environmental application such as pollution monitoring, final data represents aggregated data of a certain region but not the sensed data from a single node. The point is that the pollution level such as CO_2 emission is roughly the same within a certain region and thus the nodes in that region will have redundant data. In this case, a single node failure within a certain region does not necessarily effect heavily on the data accuracy of the region of interest. The network cannot fulfill its task only if all nodes within a certain region die. In fact, this definition falls together with the first one in case a network only guarantees 1-coverage.

Definition 3: The time span starting from initial deployment phase to the time when only m sensor nodes are still alive.

This definition is similar to the previous definition. The only difference is that the failure of a tolerable number of sensors is accepted. If a network is composed of heterogeneous nodes, node failures can be tolerable. As long as a specific number of tasks in the application survive with m remaining sensor nodes, the network is assumed to be still alive.

2.6.2. Energy Model

A sensor node is composed of a sensing unit, a processing unit, a transceiver unit, and a power unit. Each unit consumes a different amount of energy. Usually, the main consumers of energy are the transceiver unit and the processing unit. The sensing unit consumes energy for a variety of sensors and for ADC converters. The processing unit requires energy to aggregate data, compute routing, maintain security, etc. Since the purpose of the transceiver unit is to both transmit and receive data, there is no doubt that it consumes quite a lot of energy. Thus energy consumption by transmitting and receiving messages has to be analyzed based on a hop-by-hop communication scheme.

In fact, a highly accurate energy estimation is desirable because a node level energy behavior model is necessary to evaluate the energy consumption distribution. However, this would need to be investigated starting from the transistor level, taking into consideration leakage, etc. We take a more abstract view and define a simple energy model for the assessment and performance comparison of WSN designs. Our model mainly highlights the energy consumption of the transceiver unit, since the energy consumption of the processing unit is largely the same for all nodes and, as such, can be taken as a constant. Thus, energy consumption for security, routing, and data aggregation is not taken into account. Regarding wireless signal propagation, one should be aware of path loss. Typically, the path loss exponent, τ, varies from 2 to 6. If the environment is a free space, then $\tau = 2$ is considered based on the Friis free space model. Otherwise $\tau = 5$ to 6 can be considered for shadowed areas and obstructed indoor scenarios [61]. We intend to use this model for the analysis of MICAz motes.

The model concerns the total energy consumption of 1 bit data transmissions from all nodes to their nearest sinks. The formulation of the total energy consumption, E_{total}, is given in Equation 2.6.1. It is the sum of total energy consumption per group, E_i, in which the number of groups corresponds to the number of sinks, s.

$$E_{total} = \left(\sum_{i=1}^{s} E_i \right) \tag{2.6.1}$$

In Equation 2.6.2, we describe the total energy consumption of a group, E_i. It amounts to the total energy consumption of the data flows whose number is the number of nodes, n'. A data flow is considered to be routed on the shortest paths between its source node and its nearest sink.

Obviously, the number of hops in each flow of interest, l_j, depends on the nodes' locations. Moreover, the energy consumption per hop, e_k, needs to be analyzed separately due to its distance-dependent characteristic.

$$E_i = \left(\sum_{j=1}^{n_i'} \sum_{k=1}^{l_j} e_k \right) \tag{2.6.2}$$

To calculate the energy consumption of a single hop, we must know the energy used by a transmitter, a receiver, and a sensor for 1 bit of data, which is shown in Equation 2.6.3. Each component can easily be calculated by multiplying the consumed power and their individual time required for 1 bit of data.

$$e_k = e_{tx} + e_{rec} + e_{sense} \tag{2.6.3}$$

Equations 2.6.4 and 2.6.5 are used to calculate the energy consumption of receiver electronics, e_{rec}, and energy consumption of sensing, e_{sense}, respectively. Taking the values from the MICAz data sheet, we can calculate the power consumed by the receiver electronics, $P_{recElec}$. The time spent on receiving, t_{rec} is independent of the transmitted data rate, but it varies according to the user-defined duty cycle. In most works, it is assumed that the power consumed by sensors, P_{sense} is negligible. However, it should not be neglected for a considerable amount of sensors in the sensing unit. In that case t_{sense} expresses the time required to sense 1 bit of data. From a given data rate the time required to sense 1 bit of data is obtained.

$$e_{rec} = P_{recElec} * t_{rec} \tag{2.6.4}$$

$$e_{sense} = P_{sense} * t_{sense} \tag{2.6.5}$$

There are two components that consume energy in the transmitter part. The formula is described in Equation 2.6.6. The first part represents power used in transmitter electronics, P_{txElec}, while the remaining part is expressed as the transmission power of the RF signal generation, P_{amp}.

$$e_{tx} = (P_{txElec} + P_{amp}) * t_{tx} \tag{2.6.6}$$

$$P_{amp} = V * I_{tx} \tag{2.6.7}$$

Basically, P_{txElec} can be assumed to be a constant, whereas we define P_{amp} in Equation 2.6.7. Let us discuss the second component in detail. Although it looks simple, the choice of a current consumption depends on the transmitted output power setting that relies on the distance and

the selected modulation scheme. It is impossible to directly use a typical current because with MICAz it does not report a connection between them. Therefore, we must check the relationship (in dB) between RF power, P_{tx}, and the received signal power at distance d, P_d.

We define the transmission model based on the specifications of the CC2420 RF transceiver of a MICAz mote [1] using reference [126]. First, we study the effect of the path loss variation over the distance between two nodes. The path loss occurs due to the dissipated power at transmitter op-amp and channel propagation. For a general analysis of the system design, the transmission power is derived from the mean path loss which is measured in dB, as shown in Equation 2.6.8. The mean path loss, $PL(d)$ can be computed using the mean path loss at reference distance d_0, $PL(d_0)$, and the path loss exponent,τ[1].

$$PL(d) = PL(d_0) + 10\tau log_{10}\left(\frac{d}{d_0}\right) \tag{2.6.8}$$

Based on the free space radio propagation environment, Equation 2.6.9 is used to compute the value of $PL(d_0)$.

$$PL(d_0) = 20log_{10}\left(\frac{4\pi d_0}{\lambda}\right) \tag{2.6.9}$$

where,

$$\lambda = c/f$$

$c := speed\ of\ light$
$f := frequency\ of\ the\ transmitted\ signal.$

We now compute the received signal power at a distance d based on the transmitted signal in dB with the following Equation.

$$P(d) = P_{tx} - PL(d) + \sigma \tag{2.6.10}$$

Based on the above equation, a distance-dependent corresponding power level for the MICAz mote is introduced to get a satisfactory power level for a given distance, d, [126]. By referring to the Chipcon CC2420 output power setting for the MICAz mote, we get the typical current consumption, and thus P_{amp}.

$$P(d) = \begin{cases} P_{tx} - 40.2 - 20log_{10}(d),\ d < 8m \\ P_{tx} - 58.5 - 33log_{10}(\frac{d}{8}),\ d > 8m \end{cases} \tag{2.6.11}$$

[1] A wide range of $1km$ is considered for cellular system and a short range of $1m$ is considered for WLANs [61].

Note that transmitting uses less energy than receiving even at the highest output power of the transceiver chip. The reason is that the receiver consumes a considerable amount of power due to idling in the receive mode. So, a duty cycle is a good way to control energy consumption of a receiver.

2.7. Worst-Case Delay Bound

Sensor network calculus is based on network calculus [71], which is a framework for worst-case analysis allowing to calculate maximum message transfer delays, maximum buffer requirements at each sensor node, and lower bounds on duty cycles. Sensor network calculus customizes the conventional network calculus to the typical setting in WSNs.

Many WSNs can be modeled as feed-forward (FF) networks. This has several advantages. FF networks are stable, and fast to compute, because they do not contain cycles. A tree topology is an example of a FF network. If WSNs do not have FF characteristics, sensor nodes have many possible data paths and as a result they may create cyclic dependencies which makes analysis difficult and infeasible for higher network loads. To avoid cyclic dependencies, the turn prohibition algorithm [40] is a very effective way to make general topologies FF.

2.7.1. Basic Sensor Network Calculus

This is a basic overview of the Sensor Network Calculus (SNC). Detailed explanations of the SNC can be found in [115, 118, 119].

To apply SNC, the network topology has to be known to some degree. For example, a tree-structured network topology with a sink at the root and n sensor nodes can be used. Next, the network traffic has to be described in terms of the so-called arrival curves for each flow. An arrival curve defines an upper bound for the input traffic of a node. Leaf nodes in the network must handle traffic according to the sensing function they perform; for example, a node might sense an event and create a data packet at the maximum rate of one packet every second. This sensing pattern can be expressed as an arrival curve α_i. Non-leaf nodes handle traffic according to their own sensing pattern and the traffic they receive from other nodes.

To calculate the output, a so-called service curve β_i is used. The service curve specifies the minimum forwarding capabilities of a node. The unavoidable forwarding latencies are defined by the nodes' forwarding characteristics.

With the knowledge of arrival and service curves, it is possible to calculate the output bound for each node. Using those bounds, it is possible to compute the effective input $\bar{\alpha}_i$ for each node. After that, the local per-node delay bounds D_i for each sensor node i can be calculated according to the basic network calculus result given in [70]:

$$D_i = h(\bar{\alpha}_i, \beta_i) = \sup_{s \geq 0}\{\inf\{\tau \geq 0 : \bar{\alpha}_i(s) \leq \beta_i(s+\tau)\}\}$$

To compute the total information transfer delay $\bar{D}_i$ for a given sensor node i, the per-node delay bounds on the path $P(i)$ to the sink need to be added:

$$\bar{D}_i = \sum_{j \in P(i)} D_j$$

Clearly, a bound on the maximum information transfer delay in the sensor network can then be calculated as $D = \max_{i=1,\ldots,N} \bar{D}_i$. The whole procedure is called *total flow analysis* (TFA) because the entire traffic arriving at a given node is treated in an aggregate fashion.

Examples for the use of this calculus can be found, e.g., in [67, 124, 130].

2.7.2. Advanced Sensor Network Calculus

Because TFA is a straightforward method for applying network calculus in the domain of wireless sensor networks, there is room for improvement with respect to the quality of the calculated performance bounds. This is due to the fact that the concatenation result for consecutive nodes offering service curves is not exploited by TFA. In particular, we can exploit and even extend the concatenation result towards the so-called Pay Multiplexing Only Once analysis (PMOO) described in [119], to compute an end-to-end service curve for the specific flow of interest from one sensor node to the sink. Due to the sink-tree structure of the network, all flows that join the flow of interest remain multiplexed until the sink, making it possible to calculate the total information transfer delay $\bar{D}_i$ for a given sensor node i by using a flow-specific end-to-end service curve. PMOO was shown to deliver a tight bound for sink-trees of homogeneous nodes [117]. When compared to the addition of the nodal delay bounds, as done by TFA, this results in considerably less pessimistic as each interfering flow's burst has to be taken into consideration only once.

2.7.3. DISCO Network Calculator

In our work, we use the DISCO network calculator which is an open source toolbox for worst case analysis written in $Java^{TM}$. The network calculus operations and different network analysis algorithms can easily be used from this library [116].

3. The Effect of Sensor Node Placement on Performance Metrics

3.1. Introduction

Node placement is a fundamental issue to be solved in Wireless Sensor Networks (WSNs). Nodes can be deployed over a network in a random or deterministic fashion. While a random node placement is preferable in many applications, if possible, other placements should be investigated since an inappropriate node placement can increase the complexity of other problems in WSNs. A proper node placement scheme can reduce the complexity of problems in WSNs as, for example, routing, communication, energy consumption, etc. For instance, since a node stops working when it depletes its battery, the tasks of a WSN cannot continue if many sensors deplete their battery. The battery depletes faster if a sensor node needs to forward a large amount of data packets either from itself or other sensors. In other words, some nodes may have an unbalanced load and this situation can occur if sensor nodes are placed in inproper locations of the sensor field. At the same time, related problems, for example, coverage loss and topology changes arise. These examples show that a careful node placement is preferable in order to meet applications' desired goals. For example, a triangular lattice is known to be optimal for coverage performance [64], such an optimal node placement uses $O(log\, n)$ times fewer sensors than a random node placement [69]. However, optimal node placement is a very challenging problem. It has been proven to be NP-hard in [30, 104].

In this chapter, we examine a uniform random node placement and six deterministic node placements for large-scale WSNs. Tiling based deterministic node placements are investigated, in particular, three regular tilings: a triangular tiling, a square tiling, and a hexagonal tiling and three semi-regular tilings: a trihexagonal tiling, an elongated triangular tiling, and a snub square tiling are taken into account. The goal of this chapter is to introduce tiling-based node placements for WSNs and to help network

designers alternative placements to meet their desired goals.

Since the priority of performance metrics varies in application-specific WSNs, it is worthwhile to investigate a set of them. For instance, [152] surveys the effects of sensor node placements under several performance metrics. Among them, we analyze three performance metrics: coverage, energy consumption, and information transfer delay. We define each metric as follows.

- Coverage: For the surveillance kind of applications, the minimum coverage performance, or so called k-coverage must be targeted for data accuracy and for sleeping nodes in an unreliable network. A network is said to have k-coverage if every point in it is covered by at least k sensors. Instead of focusing on the minimum k-coverage, we propose a novel strategy for calculating the relative frequency of the exactly k-covered points, which uses k-coverage maps for each deterministic node placement as presented in Section 3.6. Based on this, we measure the average k-coverage and the standard deviation of exactly k-covered points.

- Energy consumption: The simple energy model presented in Section 2.6.2 in Chapter 2 is used to study energy consumption for each deployment strategy. Since energy is often the most critical issue in WSNs, it is necessary to optimize energy consumption in various ways. Using a proper node placement scheme, energy consumption can be reduced and thus the lifetime of WSNs can be extended.

- Worst-case delay: Often the maximum allowable message transfer delay must be bounded in order to enable time-sensitive applications of WSNs. Using the foundation of sensor network calculus as mentioned in Section 2.7 in Chapter 2, we calculate the worst-case end-to-end delays for each flow in every sink tree of the entire network and find the maximum worst-case delay among them.

We first study the performance of the strategies under the assumption of an exact placement. Various difficulties can occur when sensor nodes shall be placed in exact locations within the network. In fact, an exact node placement strategy may not be applicable to some WSN applications due to environmental and geographical constraints. In order to make node placement strategies realistic and usable in real world scenarios, the network designer should investigate such disturbances for the deterministic node placement strategies. For this reason, we also investigate stochastically disturbed versions of the strategies. Finally, we analyze the tradeoffs

between these performance metrics for each placement strategy with and without disturbance effects to show which strategy is preferable under what factors, e.g., the number of nodes.

3.1.1. Contributions

The contributions of this chapter are

- To the best of our knowledge, we are the first to introduce semi-regular tiling-based node placements in large-scale, time-sensitive WSNs. (→Section 3.5)
- We introduce the k-coverage map to assess all possible coverage areas and to analyze the relative frequency of exactly k-covered points. (→Section 3.7)
- We thoroughly investigate tiling-based node placements under exact and disturbed placements for coverage, energy consumption, and worst-case delay. (→Section 3.8)

3.1.2. Outline

The remainder of for the node placement problem chapter is organized as follows: Section 3.2 describes the related work. The network model and assumptions are presented in Section 3.3. In Section 3.4 and 3.5, we explain the selected placement strategies and their characteristics. Since we have already discussed the worst-case delay and lifetime performance metrics in Chapter 2, the remaining coverage metrics especially the relative frequency of the exactly k-covered points and the formulation of k-coverage maps, a new approach of evaluating coverage performance, for the deterministic node placement will be presented in Section 3.6 and 3.7. In Section 3.8, the properties of these three performance metrics for the exact placement strategies are compared. Under the same experimental set-up, we evaluate these three performance metrics for stochastically disturbed placements as can be found in Section 3.9 and 3.10. The chapter is concluded in Section 3.11.

3.2. Related Work

Finding the optimal locations for sensor nodes is a very hard problem and it has been proven to be NP-hard in [30, 104]. Ways of finding sub-optimal

solutions by heuristic methods have been addressed in [34, 32] where the nodes are placed according to the desired performance metrics of their specific applications, e.g., coverage performance. Besides, they focus on static node placement. In reality, nodes' locations can change dynamically depending on the network states and various external factors to improve the performance of the application. Repositioning of the nodes during the operation of the network is addressed in [143, 142, 146]. However, the complexity of dynamic adjustment of nodes' locations for large-scale WSNs applications is very high.

3.2.1. Placement Methodology

In two dimensional (2D) space or three dimensional (3D) space, sensor nodes are distributed either randomly or deterministically. Yet, a random placement is not an efficient distribution because the performance of the random node placement relies pretty much on the node density, some WSNs applications such as [91, 135] use random deployment due to the nature of applications and/or geography. [58] investigates three random node distribution functions with respect to fault-tolerance properties of the stochastic placements. A good thing about random placement is having no effects of placement errors because a random distribution with a placement error is still a random distribution. However, sensor failures in random deployments affect the density of the network. In addition, a random node placement is a common distribution not only for the deployment purpose but also for evaluation of various algorithms in the network.

On the other hand, nodes can be placed in a deterministic fashion in order to meet desired performance goals. It is also known that if sensor nodes are placed at exact locations and they are reliable, an optimal deterministic placement will need $O(log\, n)$ times fewer sensors than a random placement where n is the number of sensors used in random placement [69]. Such kind of node distributions are usually desirable for civilian WSN applications, e.g., structural health monitoring. In [100, 88], sensor nodes are deterministically placed to monitor corrosion and overstressed beams that can endanger structure's integrity. Deterministic node placement can also be found in 3D applications, e.g., underwater acoustics [10] and range-finders [48]. In [10], nodes are necessary to be placed at each other's line-of-sight whereas [48] introduced an art-gallery model for which the lowest amount of guards is to be placed to monitor a gallery. For these applications, nodes' candidate locations are very limited due to the applications' requirements. Another form of deterministic node placement is

grid-based or polygon-based placement which, in fact, is suitable for wide area monitoring applications [60, 131, 150, 15]. This kind of deterministic node placement becomes our interest in this chapter. The 3D space extension from 2D space can also be found in [11, 34].

3.2.2. Relevant Performance Metrics

Regarding the node placement issue, the first priority metric that came to the attention is area coverage in order to provide a high quality of information on the region of interest. Since an optimal node placement is a difficult task in WSNs, the complexity of the problem is reduced to the coverage problem where the goal is to minimize the number of sensors to cover the region of interest [34, 32, 27]. The optimal node placement to achieve full coverage when deploying sensors deterministically and critical density needed to achieve full coverage when deploying sensors randomly has been studied extensively in [15, 141, 69]. The triangular lattice is known to be the optimal placement pattern to achieve full coverage [15]. However, deployment errors and sensor failure issues have not been discussed in this work. We know that deployment errors have a strong change of on the performance of deterministic node placements. The problem is extensively studied under two deterministic and one random placement in [18]. To overcome sensor failures and deployment errors, enough sensors are placed at each lattice point in the first strategy. In the second strategy, only one sensor is placed at each lattice point but the lattice spacing is sufficiently shrunk to achieve full coverage under the presence of failures and placement errors. The third strategy is a random deployment with appropriate density. Their study showed that the deterministic placement with placement error of half the sensing range and a failure probability of 50% has better coverage than random placement. Under this condition, the random deployment needs around 10% higher node density to achieve the same coverage performance. In that paper, the authors only considered 1-full coverage performance. The authors in [69] proposed the appropriate number of sensors to provide k-coverage for a mostly sleeping sensor network. An interesting point is that the conditions for the deterministic deployment are very similar to the conditions for the random deployment to achieve k-coverage. [131] studies a generic approach to evaluate the average performance of coverage under inevitable random errors for all kinds of grid-shapes and different sensing models. Another study on such environmental effects can be found in [73] where the authors pointed out that the performance of the grid-based deployment depends

only on the environmental conditions instead of the coverage performance. A similar placement error is considered for our tiling-based node placement strategies. The sensing model plays a vital role in evaluating coverage performance. While [24] studies a more practical model of irregular sensing range, in literature, many works abstracted to a disc-based sensing range [57, 15, 18, 34, 60, 150]. This level of abstraction is hardly avoidable during the design phase. Like in [57], we compute each k-coverage as the ratio of the covered area to the region of interest.

The second metric, which has been the most critical issue for WSNs, is the network lifetime since the positions of nodes significantly affect it. The sensors near the sink usually consume more energy than others thus the lifetime becomes shorter. To solve this energy hole problem, the authors in [58] showed the R-random placement, in which the density of the nodes nearer the sink is higher, is a good random distribution with respect to high fault tolerance with a high probability of having nodes that can sense. It, however, does not completely eliminate the problem but leads to unbalanced traffic load and causes bottlenecks. Maximizing the lifetime by sensor nodes with coverage constraints has been addressed in [33] where the idea of mobile nodes as relay nodes is introduced. The authors transformed the problem into minimizing the average energy consumption by a sensor in each communication in order to balance the load among the sensors. A heuristic approach is proposed to select the relay nodes among sensors where the selected relay nodes are relocated to the positions which are carefully chosen in order to form the most efficient topology. Recently, sink mobility [83, 29, 44, 125] has been introduced to prolong the lifetime of WSNs. Details will be discussed in Chapter 5.

Thirdly, connectivity is a typical networking requirement. Recently, the connectivity issue has been addressed in node placement [64, 59, 14, 15, 17, 16]. It is known that a triangular lattice is asymptotically optimal with respect to minimizing the number of discs needed to achieve full coverage [64]. Besides, when the ratio of the transmission range r_{tx} and the sensing range r_{sense} is greater than or equal to $\sqrt{3}$, the triangular lattice has 6-connectivity. Recently, [59] proved that the strip-based pattern is optimal when $r_{tx} = r_{sense}$. Based on the strip-pattern, the authors in [14] proved 1-full coverage and 1- and 2-connectivity for all ranges of r_{tx}/r_{sense}. The authors continued their work for a complete set of deployment patterns for all ranges of r_{tx}/r_{sense} that achieve 1-full coverage and k-connectivity where $k \leq 6$ [15, 17]. In [16], the authors discussed the pattern mutation phenomenon that contradicts the conjecture presented in [14, 15, 17] that there exists a universal elemental pattern among optimal pattern evolution

and that pattern evolution is continuous. All these works assume disc-based sensing and transmission ranges.

Last but not least, the message transfer delay can also be improved with the help of a proper node placement. Since the end-to-end delay depends highly on the hop-distance between the sensor and the sinks and the traffic load along the flow of interest, the nodes' locations as well as the sinks' locations are very important. In literature, some have addressed delay minimization by sink placement but not using node placement. For this reason, we also study the effect of node placement on the delay performance in this chapter.

3.3. Node Placement Model and Assumptions

During the design phase of WSNs, the designer knows the number of sensor nodes, n, which are deployed in a given field in either a random or a deterministic fashion. Finding the optimal number of sensors to achieve the desired goal will not be considered in this chapter. A circular field with radius R is considered in our experiments. We introduce one node placement strategy for random node placement and six strategies for tiling-based deterministic node placements together with their characteristics. Although [11, 31] address node placement in 3D space, we investigate node placements strategies over 2D space which seems more realistic and usable in most WSN applications. However, in a further study we also investigate the 3D effects due to realistic 2D imperfections like, e.g., a hilly terrain. We make the following assumptions for random and deterministic node placements:

- A circular field network with radius R is considered for n sensor nodes, where n and R are assumed to be given. In each tiling, each of the n grid points hosts a sensor.
- A disc based sensing model r_{sense} which, in fact, is a good abstraction from the real world especially when certain theoretical foundations are to be established.
- The radius of r_{sense} is defined as the distance between any two vertices in a tiling-based node placement. It is obvious that each tiling scheme uses different r_{sense} in order to deploy n given sensors inside a circular field with radius R.

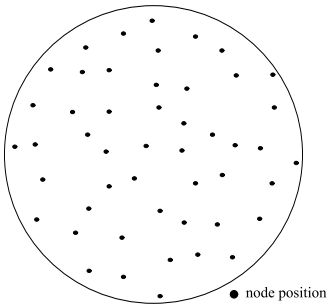


Figure 3.4.1.: Random node placement.

- Since finding the optimal r_{sense} inside a circular field for each tiling method is not an easy task, we provide approximate formulations. The idea is that the total area of the network having radius R is equally subdivided into n areas where each area represents each node. Then we transform each area into a corresponding shape for each tiling. The corresponding shape is achieved by connecting the midpoints of the surrounding polygons of a vertex, for example, a hexagon is a corresponding shape for a triangular tiling. In brief, the area of a corresponding shape which is a function of r_{sense} is approximately equivalent to $\frac{\pi R^2}{n}$. In this way, an approximate formulation for the relationship among the parameters r_{sense}, R, and n is achieved.

3.4. Uniform Random Node Placement

A uniform random node placement is considered to be a common node placement strategy in WSNs such as for analyzing the performance of the algorithms and protocols where the effect of node placement is not sensitive. We choose a uniform random placement as one of the competitors and as a reference strategy. In the uniform random placement, each of the n sensors has equal probability of being placed at any point inside the given field, as shown in Figure 3.4.1. Consequently, the nodes are scattered on locations which are known with uncertainty. For example, such a placement can result from throwing sensor nodes from an airplane. In general, a uniform random placement is assumed to be easy as well as cost-effective. As we mentioned before, WSN applications often prefer a

random node placement, which is why we assess its performance metrics here. Besides, [69] claimed that a uniform random placement outperforms both the grid and the Poisson distribution placements for k-coverage under mostly sleeping sensor nodes.

3.5. Deterministic Node Placement

In deterministic node placement, the entire network is filled with a basic placement pattern. In fact, tiling, also known as tesselation, is the covering of an infinite space (usually Euclidean) with figures that neither overlap nor leave any gaps. Although a pattern-based node placement may be problematic for placing nodes at exact locations in real applications, its performance characteristics may justify the additional effort. Among different tilings, we focus on regular and semi-regular tiling in Euclidean plane. The name of each tiling can be expressed according to the vertex configuration which is a short-hand notation for representing the vertex figure of a tiling as the sequence of faces around a vertex. A regular tiling is a highly symmetric tessellation made up of congruent regular polygons. A regular polygon has the same side lengths and interior angles. There exist three regular polygon tilings made up of equilateral triangles, squares, or hexagons. Semi-regular tiling composes of more than one basic placement polygon. There are eight possible semi-regular tiling in a two dimensional plane where the arrangement of polygons at every vertex point is identical. Among them, we investigate a trihexagonal tiling (THT), an elongated triangular tiling (ETT), and a snub square tiling (SST).

3.5.1. Triangular Tiling (TT)

Triangular tiling is a tiling of equilateral triangles where each vertex has a 3^6 pattern. As we mentioned before the name comes from going around a vertex and listing the number of sides each regular polygon has, as illustrated in Figure 3.5.1(a). The distance of each side of polygon r_{sense} is defined as d_t in the triangular tiling.

For coverage performance, a triangular tiling is asymptotically optimal with respect to the number of discs needed to achieve full coverage [64]. Under the same r_{sense}, the density of triangular cell is the highest among the three regular tilings. In other words, triangular tiling has the largest r_{sense} for the same number of nodes n. Having the highest sensing range, it is obvious that triangular tiling consumes more energy for sensing than the other two regular tilings. However, each vertex has six neighbor nodes

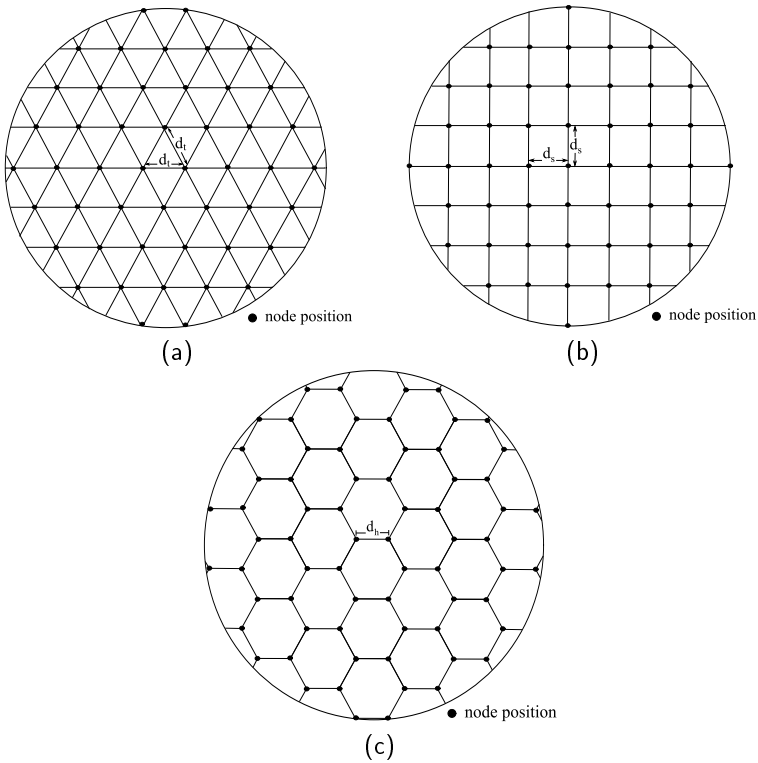


Figure 3.5.1.: Three regular tilings: (a) a triangular tiling, (b) a square tiling, and (c) a hexagonal tiling.

within its sensing range regardless of the transmission range of the sensor. Yet, the node degree cannot guarantee to minimize the worst-case delay, though it helps to improve the worst-case delay by providing more possible paths to the sink.

The approximate length d_t can be calculated in the following way. Remember that each tiling point hosts a node in all tilings. We first check the corresponding shape for the area of each node (i.e., $\frac{\pi R^2}{n}$). By connecting the centroids of the surrounding triangles of a node, the corresponding shape becomes a regular hexagon with edge lenght $d_{h'}$. We know that the area of the regular hexagon is $\frac{3\sqrt{3}}{2} d_{h'}^2$. The correlation between $d_{h'}$ and d_t is $d_{h'} = \frac{d_t}{\sqrt{3}}$. Finally, the relationship among n, d_t, and R is given in

Equation 3.5.1

$$d_t = \sqrt{\frac{2\pi R^2}{3n}}\,. \tag{3.5.1}$$

3.5.2. Square Tiling (ST)

A square tiling is a natural placement strategy over a unit square. As illustrated in Figure 3.5.1(b) a square tiling placement, which is also often called a grid, is a popular placement not only in WSN but also in conventional networks. A square tiling has a 4^4 configuration since each vertex is surrounded by four polygons with four sides each. In square tiling, we define d_s as r_{sense}.

A square tiling has a higher sensing range d_s than a triangular tiling because the density of a square tiling is not as high as a triangular tiling due to its node degree of four. In particular, the square grid uses about $5\,\%$ of r_{sense} less than that of a triangle grid. As a result, it consumes less energy for sensing but has lower minimum coverage than a triangular tiling.

$$d_s = \sqrt{\frac{\pi R^2}{n}} \tag{3.5.2}$$

Equation 3.5.2 provides the approximate formula for the relationship among the parameters n, d_s, and R. The corresponding shape for each node in a square tiling is a square with edge length d_s after connecting the midpoints of four squares around a vertex. Thus the corresponding shape area d_s^2 is equivalent to the area assigned for each node which can be achieved by dividing the total area of the circular network having radius R with the number of nodes n.

3.5.3. Hexagonal Tiling (HT)

Another regular tiling is a hexagonal tiling where each vertex has a 6^3 pattern as shown in Figure 3.5.1(c). We define one side of each hexagon as d_h which becomes r_{sense} to construct a hexagonal tiling.

The hexagonal tiling provides the lowest maximum k-coverage among the three regular tilings due to its node degree of three. It has some nice properties such as having the smallest r_{sense} for a given n and R compared to the triangular and square tiling. In a hexagonal grid, r_{sense} is about $17\,\%$ less than in the triangle grid. Therefore, the hexagonal tiling consumes the lowest energy for sensing among the three regular tilings

under the assumption that r_{sense} is equal to one side of each regular polygon.

$$d_h = \sqrt{\frac{4\pi R^2}{3\sqrt{3}n}} \tag{3.5.3}$$

Equation 3.5.3 allows the approximate computation of any one parameter out of n, d_s, and R given the other two parameters. With the same approach, we search a corresponding shape to represent the area of each node $\frac{\pi R^2}{n}$ in the hexagonal tiling. Since triangular tiling and hexagonal tiling are dual to each other, the corresponding shape for a single vertex of hexagonal tiling is a triangle having edge length $d_{t'} = \sqrt{3}d_h$. Knowing the area of equilateral triangle to be $\frac{\sqrt{3}}{4}d_{t'}^2$, the approximate edge length of each hexagon d_h can be computed according to Equation 3.5.3.

3.5.4. Trihexagonal Tiling (THT)

A THT uses a triangle and a hexagon in a two dimensional plane, the so-called 3-6-3-6 configuration as shown in Figure 3.5.2(a). The sensing range of THT is defined as d_{THT} from now on. A THT combines the advantages of coverage performance from the triangular grid and the lowest r_{sense} from a hexagonal grid. In general, the THT placement uses $13\,\%$ of r_{sense} less than the triangle grid while providing nearly double of the 3-coverage area (more details can be found in Section 3.7.4). Among the selected three semi-regular tilings, THT uses the least sensing range. Besides, the node degree of THT within the sensing range d_{THT} becomes four.

In a similar way to the regular tilings, an approximate formulation for r_{sense} can be found for THT. Although semi-regular tilings use more than one regular polygon, there exists a single corresponding shape to represent the area of each node. For a THT tiling, the area for each node corresponds to a parallelogram with height d_{THT} where each corner is either at the midpoint of a hexagon or a triangle. The base length of the parallelogram is $\frac{4}{2\sqrt{3}}d_{THT}$ which is the sum of half of a hexagon's height and one-third of the height of the triangle. Then the area of the parallelogram becomes $d_{THT} \times \frac{4}{2\sqrt{3}}d_{THT}$. An approximate solution for d_{THT} can be computed using Equation 3.5.4

$$d_{THT} = \sqrt{\frac{2\sqrt{3}\pi R^2}{4n}}\,. \tag{3.5.4}$$

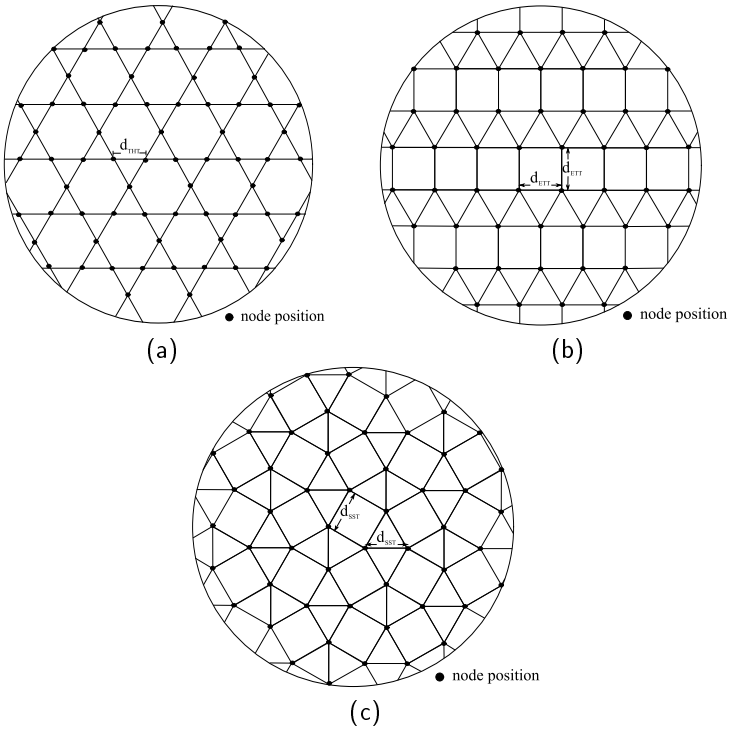


Figure 3.5.2.: (a) A trihexagonal tiling, (b) an elongated triangular tiling, and (c) a snub square tiling.

3.5.5. Elongated Triangular Tiling (ETT)

An ETT tiles up squares and equilateral triangles where each vertex is composed of three triangles and two squares forming a 3-3-3-4-4 vertex configuration as illustrated in Figure 3.5.2(b). The d_{ETT} is called the sensing range for ETT. The coverage performance of ETT is a bit better but it also has a higher sensing range than a square tiling due to the existence of the triangular cells in ETT. For a given number of n and R ETT use $12\,\%$ lower energy consumption for sensing than the THT node placement. In addition, ETT has five neighboring vertices within d_{ETT}.

The approximate value of r_{sense} for ETT placement is presented in Equation 3.5.5

$$d_{ETT} = \sqrt{\frac{2\sqrt{3}\pi R^2}{(2\sqrt{3}+3)n}}. \tag{3.5.5}$$

The corresponding shape for the area represented by each vertex of ETT is a pentagon. We compute the area of the pentagon which is a combination of two symmetric trapezoids. Each trapezoid has the height $\frac{1}{2}d_{ETT}$ and the lengths of the two bases $\frac{3+\sqrt{3}}{6}d_{ETT}$ and $\frac{2+\sqrt{3}}{2\sqrt{3}}d_{ETT}$. Hence, the total area of the pentagon becomes $\frac{2\sqrt{3}+3}{2\sqrt{3}}d_{ETT}^2$ which is roughly the same as $\frac{\pi R^2}{n}$.

3.5.6. Snub Square Tiling (SST)

An SST is another tiling of triangles and squares in the two dimensional plane. Again each vertex is composed of three triangles and two squares (as ETT), but the configuration order of SST is 3-3-4-3-4 as illustrated in Figure 3.5.2(c). Such a configuration improves the coverage performance of SST to guarantee a minimum 3-coverage. We define d_{SST} as the sensing range for SST tiling which, in fact, is the same value as for ETT node placement.

Although SST and ETT have different configurations, both use the same combination of polygons. Therefore, the cell length d_{SST} is the same as d_{ETT} given in Equation 3.5.5.

3.6. Coverage: a Primary Objective for Node Placement

As we mentioned in Section 3.2.2, area coverage is a primary objective for node placement in WSNs. The simple reason for caring about coverage is to provide a high quality of information on the region of interest. This is also known as area coverage which is important for most WSN applications. Full coverage and partial coverage are both notions considered for WSN applications. To satisfy full coverage of a given region of interest, every point in it must be covered by at least one sensor without allowing any uncovered points. However, there may be exceptions when partial coverage can be assumed as a full coverage. For example, in temperature or pressure sensing in environmental monitoring applications, reading at one point is adequate for a wide region since it may have the same readings in its

surrounding area. In any case, the overall coverage pretty much depends on both the sensing ranges and the placement scheme of the nodes. To achieve a desired coverage of a region, adjusting the sensing range has its limitations due to expensive energy consumption and restricted node capabilities. Therefore, node placement becomes very important. Among various types of coverage, calculating the minimum coverage so called k-coverage is a usual way of specifying conditions on coverage.

3.6.1. Minimum k-coverage

In [69, 122], the authors formulate the k-coverage of a region for mostly sleeping large-scale WSNs. In [69], it is claimed that the value of the expression $np\pi r^2_{sense}/log(np)$, where n is the number of sensors and p is the probability of active sensors, equals 1 forms a sufficient condition for k-coverage. Although minimum k-coverage is good for surveillance type of applications, other kinds of coverage, such as average k-coverage or maximum k-coverage, may be more meaningful in other WSN applications. Moreover, it seems inappropriate to measure k-coverage for performance comparisons due to its sole interest in the minimum coverage area of the network. For this purpose we investigate the relative frequency of the exactly k-covered points in node placement strategies.

3.6.2. Exact k-Coverage

We define a new term called the exact k-coverage, which means the total area of the field covered by k sensor nodes. In other word, we can check how much percentage of the region of interest is covered by exactly k sensors. This way we can compute a minimum, an average, or a maximum coverage of the network. In this chapter, we use the term "exact k-coverage" for the k-coverage map to avoid confusion with k-coverage.

3.7. k-coverage Map

We introduce the notion of a k-coverage map, which is used to check all possible coverage areas and to analyze the relative frequency of exactly k-covered points. Using the idea of the k-coverage map we measure the quality of coverage performance of node placement strategies. We model the k-coverage map for six tiling placements. A k-coverage map for each tiling represents the smallest cell which provides all possible coverage area. We can easily model the k-coverage map by using basic geometry. For

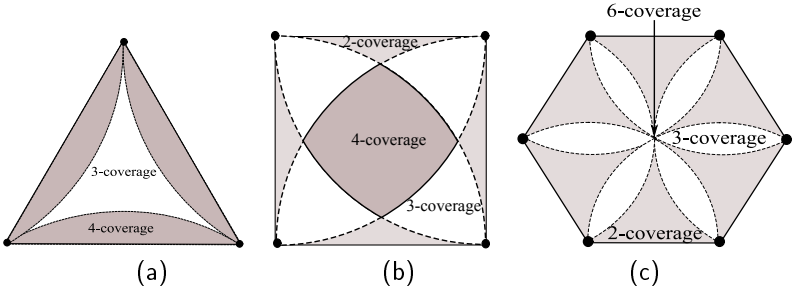


Figure 3.7.1.: k-coverage map for (a) a triangular cell, (b) a square cell, and (c) a hexagonal cell.

a uniform random placement, it can be achieved by applying systematic sampling over a given field.

We make further assumptions to model the k-coverage map for each tiling:

- A point is covered by a node if it lies either within a disc of sensing range, r_{sense}, or exactly at circumference of the disc.
- No boundary conditions are considered which seems reasonable for large-scale WSN scenarios.

3.7.1. Triangular Cell

A single triangular cell is sufficient to check the k-coverage map for triangular tiling which can be seen in Figure 3.7.1(a). Each triangular cell has exact 3- and 4-coverage areas. The middle region is covered by exactly 3 nodes. The shaded regions are covered by four nodes where the three nodes lie at the vertices of the triangle and the fourth node is the third node of the neighboring tiled triangles for each edge of the triangle under study. Under the same sensing range d_t each shaded region is symmetric. The coverage area of exact k-coverage can be calculated by using Equations 3.7.1 to 3.7.3. Equation 3.7.1 is used to compute one shaded region of Figure 3.7.1(a) which is the difference between one-sixth of the area of a circle having radius d_t and the area of the equilateral triangle which occupies the exact 3-coverage area. Then, the total area of exact 4-coverage is three times A_1^t and the difference between the equilateral triangle and the total exact 4-coverage is the area of exact 3-coverage that we can compute by Equation 3.7.2 and 3.7.3 accordingly. In general, $62.75\,\%$ of

the triangular cell is exact 4-covered while $37.25\,\%$ of the area is occupied by exact 3-coverage.

$$A_1^t = \left[\frac{2\pi - 3\sqrt{3}}{12}\right] * d_t^2 \tag{3.7.1}$$

$$A_t^4 = \left[\frac{2\pi - 3\sqrt{3}}{4}\right] * d_t^2 \tag{3.7.2}$$

$$A_t^3 = \left[\frac{2\sqrt{3} - \pi}{2}\right] * d_t^2 \tag{3.7.3}$$

3.7.2. Square Cell

Figure 3.7.1(b) shows the k-coverage map of all possible exactly k-covered points of a square grid cell. In the square grid cell, nodes are placed at the corners and their sensing ranges' intersections form a tesselation of the region. As it is assumed that the sensing range is equal to the length of a cell, a square grid cell has exact 2-, 3- and 4-coverage regions. For instance, the middle region has exact 4-coverage because it forms the intersection region of all nodes. Since the radii of circles are the same, some tesselations are symmetric. Therefore a square cell has four symmetric gray-regions near the border lines and four symmetric white-regions covered by exactly 2 and 3 sensor nodes, respectively.

Using Equation 3.7.4 to 3.7.8, we compute the total area of exact k-coverage of a grid cell.

$$A_1^s = \left[\frac{4\pi - 3\sqrt{3}}{6}\right] * d_s^2 \tag{3.7.4}$$

$$A_2^s = \left[\frac{\pi - 2}{2}\right] * d_s^2 \tag{3.7.5}$$

With Equation 3.7.4 we formulate the intersection area between two circles if the circumference of one circle passes through the origin of the other circle and vice versa. In Equation 3.7.5, the area between two circles, $x^2 + (y - r)^2 = 1$ and $(x - r)^2 + y^2 = 1$ is calculated, where r is the radius.

Based on Equations 3.7.4 and 3.7.5, the required tesselations are formulated. With Equation 3.7.6, we compute the area of the combination

of 2- and 3-coverage, which is the difference of a quarter circle area and a half of area A_1^s. Equation 3.7.7 is used to calculate the 2-coverage area near the border line. Therefore, total 2-coverage regions inside a square grid cell are four times A_4^s as presented in Equation 3.7.8.

$$A_3^s = \left[\frac{\pi d_s^2}{4} - 0.5A_1^s\right] \tag{3.7.6}$$

$$A_4^s = \left[d_s^2 - \frac{\pi d_s^2}{4}\right] - A_3^s \tag{3.7.7}$$

$$A_s^2 = \frac{12 - 2\pi - 3\sqrt{3}}{3} * d_s^2 \tag{3.7.8}$$

Knowing A_3^s and A_4^s, we calculate the exact 3-coverage which is four times the difference of A_3^s and A_4^s as presented in Equation 3.7.9. Finally, the exact 4-coverage area is computed by using Equation 3.7.10. In summary, each square cell has 17.4% of 2-coverage, 51.1% of 3-coverage, and 31.5% of 4-coverage.

$$A_s^3 = \frac{\pi + 6\sqrt{3} - 12}{3} * d_s^2 \tag{3.7.9}$$

$$A_s^4 = \frac{\pi + 3 - 3\sqrt{3}}{3} * d_s^2 \tag{3.7.10}$$

3.7.3. Hexagonal Cell

As shown in Figure 3.7.1(c), there exist three possible exact k-coverages in a hexagonal cell: 2-coverage, 3-coverage and 6-coverage. Note that a hexagonal cell is composed of six symmetric tesselations where each tesselation is an equilateral triangle as shown in Figure 3.7.1(a). As there is no center point in the hexagonal cell each equilateral triangle has exact 2-coverage in the center and exact 3-coverage at the border of two edges of the triangle. Finally, Equation 3.7.11 and 3.7.12 are presented to compute exact 2-coverage and 3-coverage for each hexagonal cell. From this coverage map, we know that about $58.16\,\%$ of the hexagonal cell area is covered by 2 nodes and the remaining $41.84\,\%$ is covered by exactly 3 sensors. The center of a regular hexagon has exact 6-coverage because it can be reached by six sensor nodes. As a singular point it has area 0. Nevertheless, such a point is a good candidate for sink placement in

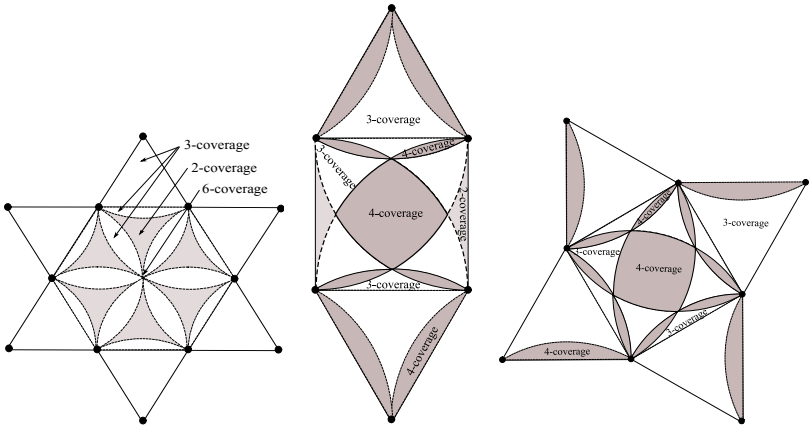


Figure 3.7.2.: k-coverage map for (a) a THT cell, (b) a ETT cell, and (c) a SST cell.

order to have many 1-hop neighbors, thus reducing energy consumption and worst-case delay.

$$A_h^3 = \left[2\pi - 3\sqrt{3}\right] * d_h^2 \tag{3.7.11}$$

$$A_h^2 = \left[\frac{9\sqrt{3} - 4\pi}{2}\right] * d_h^2 \tag{3.7.12}$$

$$A_h^6 = \varepsilon \tag{3.7.13}$$

3.7.4. THT Cell

The THT cell is illustrated in Figure 3.7.2(a), which is composed of two basic placement patterns: six equilateral triangles and one regular hexagon, where each of the tiling point hosts a node. The THT tiling has three possible exact k-coverages: 2-, 3-, and 6-coverage. The area of each equilateral triangle is fully covered by three nodes, thus having exact 3-coverage. Inside a regular hexagon, the only difference with the hexagonal cell is the border region and the rest are the same. The equilateral triangles

on each side of the hexagon produce the exact 3-coverage tesselations at the border of the hexagon. The coverage area of the hexagon is now a composition of six equilateral coverage maps as shown in Figure 3.7.1(a) but only for exact 2- and 3- coverage due to a lack of a center node inside the hexagon.

Basing on the concepts of triangular and hexagonal cells, total areas of exact k-coverages for the THT cell can be calculated by using Equations 3.7.14 to 3.7.16. Equation 3.7.14 is used to compute the exact 3-coverage for the THT cell which in fact is the area of six equilateral triangles and the white regions of a hexagonal cell as shown in Figure 3.7.2(a). Equation 3.7.15 computes the total exact 2-coverage. As discussed for the hexagonal cell, the center point of the THT cell is reachable by 6 sensors and therefore we denote the area of exact 6-coverage as ε. In summary, $18.6\,\%$ and $81.4\,\%$ of the THT cell area are used up by 2- and 3-coverage, respectively.

$$A^3_{THT} = \left[3\pi - 3\sqrt{3}\right] * d^2_{THT} \tag{3.7.14}$$

$$A^2_{THT} = \left[6\sqrt{3} - 3\pi\right] * d^2_{THT} \tag{3.7.15}$$

$$A^6_{THT} = \varepsilon \tag{3.7.16}$$

3.7.5. ETT Cell

The smallest cell of the ETT tiling for which we check all possible exact k-coverages is presented in Figure 3.7.2(b). An ETT cell has a square and two equilateral triangles which result in exact 2-, 3-, and 4-coverage. Each triangular cell has exact 3- and 4-coverage as shown in Figure 3.7.1(a) except that one of the 4-coverage tesselations near the border of the square is unavailable. The square in the ETT cell has exact 2-, 3-, and 4-coverage. Unlike the square cell from Figure 3.7.1(b), the tiled triangles make a better coverage area near the borders of a square in the ETT cell. As a result, a square in the ETT cell increases the exact 3- and 4-coverage area by eliminating half of the exact 2-coverage from the original square cell. Let A^{ETT}_1 be one exact 4-coverage inside the square. Equation 3.7.17 provides the area of A^{ETT}_1 which is half of the difference between the area computed by Equations 3.7.1 and 3.7.7 with the sensing range d_{ETT}. Using the k-coverage maps of triangular and square cells, the corresponding area coverage of ETT cell can be computed from Equations 3.7.18 to 3.7.20.

In addition, an ETT cell has $4.65\,\%$ of 2-coverage, $53.98\,\%$ of 3-coverage, and $41.37\,\%$ of 4-coverage.

$$A_1^{ETT} = \left[\frac{\pi - 3}{6}\right] * d_{ETT}^2 \tag{3.7.17}$$

$$A_{ETT}^2 = \left[\frac{12 - 2\pi - 3\sqrt{3}}{6}\right] * d_{ETT}^2 \tag{3.7.18}$$

$$A_{ETT}^3 = \left[\frac{9\sqrt{3} - 4\pi}{3}\right] * d_{ETT}^2 \tag{3.7.19}$$

$$A_{ETT}^4 = \left[\frac{5\pi - 6\sqrt{3} - 3}{3}\right] * d_{ETT}^2 \tag{3.7.20}$$

3.7.6. SST Cell

Though SST tiling has the same composition of basic placement patterns as ETT, the smallest possible k-coverage map of SST composes of four equilateral triangles and a square as illustrated in Figure 3.7.2(c). Accordingly, the coverage performance of SST is better than for ETT after eliminating the exact 2-coverage by surrounding the equilateral triangles in a square. Based on the tesselations from ETT cell, Equation 3.7.21 and 3.7.22 provide the total area of exact 3- and 4-coverage. While $68.3\,\%$ of SST cell is 3-covered, $31.7\,\%$ of the SST cell is occupied by 4-coverage area.

$$A_{SST}^3 = \left[\frac{9\sqrt{3} - 7\pi + 12}{3}\right] * d_{SST}^2 \tag{3.7.21}$$

$$A_{SST}^4 = \left[\frac{7\pi - 6\sqrt{3} - 9}{3}\right] * d_{SST}^2 \tag{3.7.22}$$

3.8. Performance Evaluation Under Exact Placement

In this section we conduct a performance evaluation for our exact node placement strategies. Note that random placement is used as a benchmark strategy for our exact node placements. In the random placement, we generate 10 scenarios and take the average value for analysis of each performance metric. The primary factors for all experiments are: the number of nodes, the number of sinks, and the sensing range. Nodes are distributed over a circular field and sinks are placed at the center of gravity of a sector of a circle (CGSC) [106]. The routing topology we use here is based on Dijkstra's shortest path algorithm, which produces the shortest hop distance from a source to a sink. We also assume that r_{tx} is twice r_{sense} in all strategies. All the selected values for the experiments are based on a realistic model of MICAz mote [7] running under TinyOS.

3.8.1. Coverage

The network size is varied from 100 to 500 nodes for different scenarios. Under the same number of nodes n, each tiling uses the best sensing range that fits n nodes inside the circular shape with radius R. Under this circumstance, it is clear that the sensing range slightly differs for each tiling.

We do not consider boundary conditions in deterministic node placement. In the case of the random placement, we do a systematic sampling over the area that equals a square tiling without boundary condition. Like a square tiling did, an $11\,m$ sensing range is considered. Taking a smaller granularity does not significantly change the results, so we chose $0.5\,m$ for the experiments. Under 10 distributions of uniform node placement, we investigate the exact k-coverage and take average results. In fact, the distributions of exact k-coverage are relatively the same in all scenarios. The exact k-coverage varies from 0 to 8 as shown in Figure 3.8.1. In all scenarios, $5\,\%$ of the network is not covered by any node. Most of the area is covered by exact 1- to 4-coverage and exact 3-coverage has the highest covered area varying from $21.6\,\%$ to $23\,\%$ of the network. The random placement has an average 3.00-coverage with a standard deviation of 1.7.

In the regular tilings, no matter what amount of nodes is analyzed, a single cell is sufficient for computing the whole network coverage since it has symmetric cells. Moreover the sink location does not affect coverage performance. The relative frequencies of exactly k-covered points of three

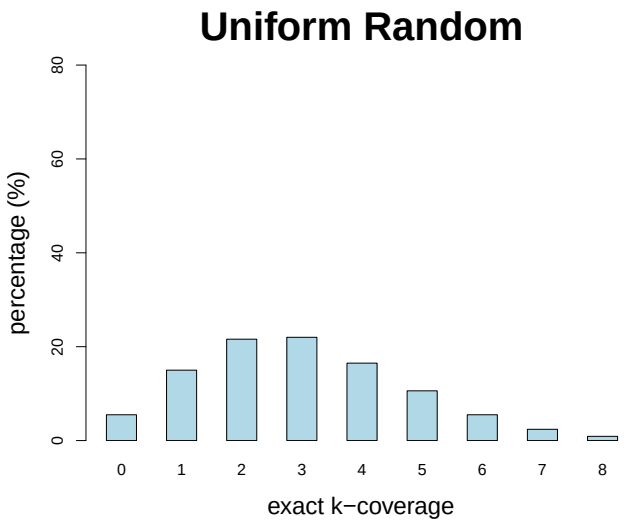


Figure 3.8.1.: The exact k-coverages of the uniform random placement.

regular tilings are shown in Figure 3.8.2.

In Figure 3.8.2(a), a triangular tiling uses $11.75\,m$ sensing range. About $62.75\,\%$ of the network is occupied by exact 4-coverage area while $37.25\,\%$ of the area has exact 3-coverage. A triangular tiling has an average of 3.63-coverage and a standard deviation of 0.48.

A square tiling uses $11\,m$ sensing range. The exact k-coverage of a square tiling is presented in Figure 3.8.2(b) where about half of the network is covered by three sensor nodes while the other half is covered by exact 2- and 4-coverage. The exact values are mentioned in Subsection 3.7. In general, the square tiling has an average 3.14-coverage with a standard deviation of 0.68.

The exact k-coverage distribution of a hexagonal tiling can be seen in Figure 3.8.2(c) where 58.16% of the area has exact 2-coverage and 41.84% of the total area is covered by 3 sensors. A hexagonal tiling has an average 2.42-coverage with a standard deviation of 0.49. While the coverage performance of a hexagonal tiling is the worst among our selected strategies, it has the lowest sensing range of $9.7\,m$.

Regarding the average coverage performance metric, the triangular tiling placement outperforms the other regular tiling placements.

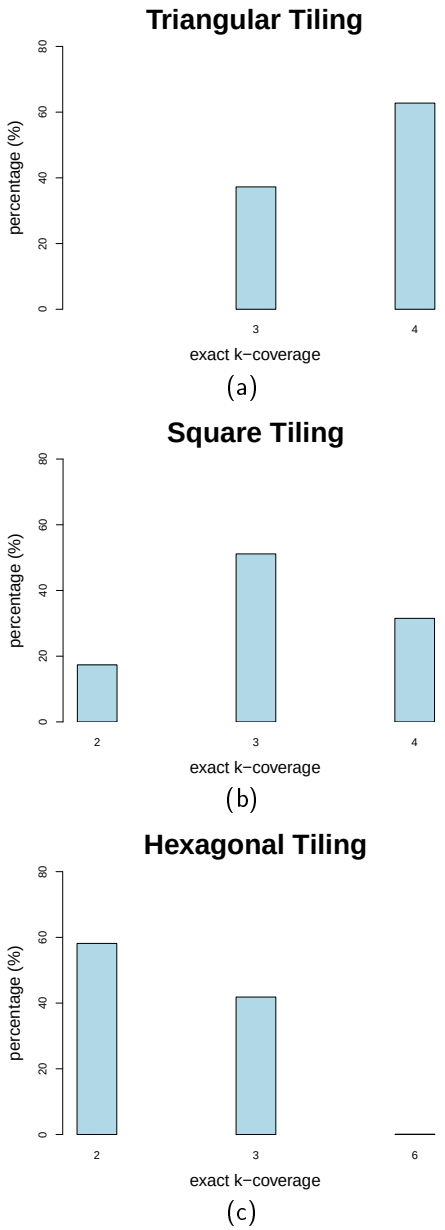
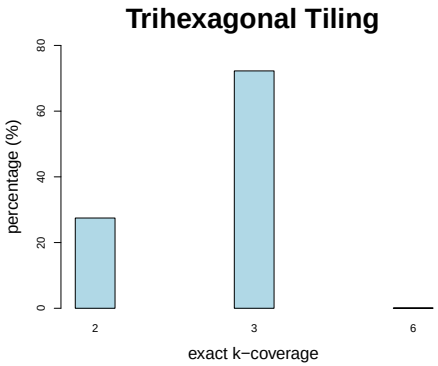
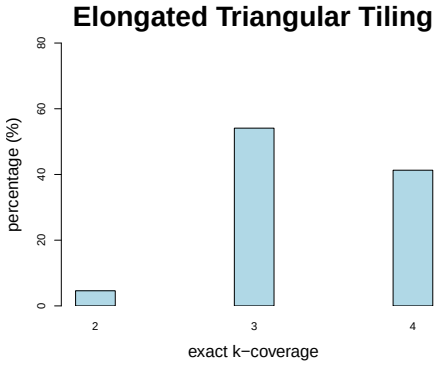


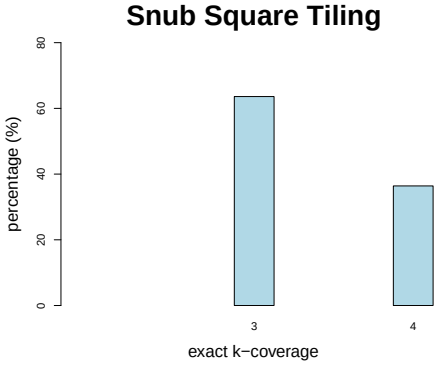
Figure 3.8.2.: The exact k-coverages of (a) TT, (b) ST, and (c) HT placements.



(a)



(b)



(c)

Figure 3.8.3.: The exact k-coverages of (a) THT, (b) ETT, and (c) SST placements.

In a semi-regular tiling, a single k-coverage cell is not sufficient to judge the exact k-coverage of the network. In this case, we first compute the total amount of triangle and hexagon cells inside a given circular field having radius R. After that we compute the relative frequency of exactly k-covered points by using the k-coverage map as shown in Figure 3.8.3. Although the ratio of cells does not remain exactly equal when increasing n, the result for exact k-coverages are nearly the same.

For THT almost two-thirds of the network are 3-covered whereas the rest is the exactly 2-covered. THT has an average 2.7-coverage with a standard deviation of 0.47. What is more, THT needs less sensing range than other strategies. While others except hexagonal tiling placement use more than $11\,m$ sensing range, THT requires only $10.25\,m$.

The results of an ETT cell are presented in Figure 3.7.2(b). From the results shown in Figure 3.8.3(b), $54.1\,\%$ of the network area is covered by 3 sensors whereas $4.6\,\%$ and $41.4\,\%$ of the network area are occupied by 2- and 4-coverage. ETT has an average 3.36-coverage with a standard deviation of 0.48.

Both ETT and SST use an $11.4\,m$ sensing range. In an SST placement, about $63.6\,\%$ of the area has 3-coverage while the remaining $36.4\,\%$ is covered by exactly 4 sensors. With respect to the minimum k-coverage performance, SST is better than ETT because SST eliminates the exact 2-coverage from the network. SST, however, roughly has the same average k-coverage and standard deviation as ETT.

3.8.2. Energy Consumption

We investigate between 100 and 1000 nodes with up to 30 sinks. Among them the experimental results of three scenarios are presented in Figure 3.8.4. Under the assumption of the free space propagation, we apply Equation 2.6.11 in order to get the current consumption. All strategies require a current consumption of $8.5\,mA$ with $-25\,dBm$ for distances up to $12.5\,m$ and $9.9\,mA$ for distances between $12.5\,m$ and $23\,m$ with $-20\,dBm$. A constant voltage of $3\,V$ is used for transmit and receive modes. Then we compute the power, P_{amp}, for a 1 bit data transmission. Since the power used in transmitter electronics, P_{txElec}, is relatively the same, we assume it as a constant. To apply Equation 2.6.6, we compute t_{tx}. To simplify, in our experiment we assume that the same data rate is used for transmission and sensing. A data rate of $250\,kbps$ is used, which takes $t_{tx} = 4\,\mu s$ for a 1 bit data transfer. Again, we also assume equal energy consumption in sensing. It means that Equation 2.6.3 contains only

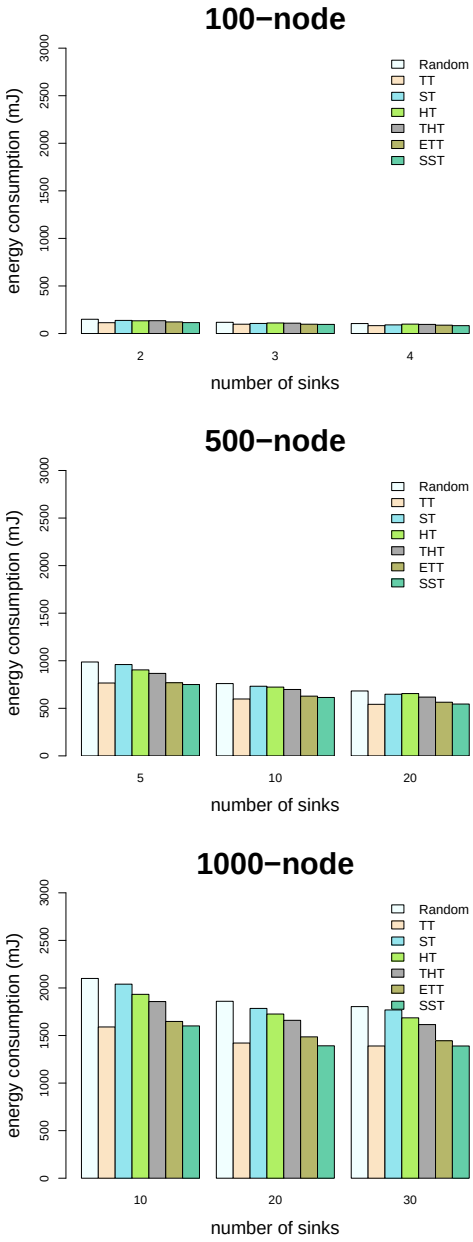


Figure 3.8.4.: 1 bit energy consumption comparison of three strategies in different scenarios.

e_{tx} and e_{rec}. We use a current of $19.7\,mA$ for the consumed power by the receiver electronics with a $1\,\%$ duty cycle for receiving 1 bit of data. The results from Equation 2.6.3 are used to analyze Equation 2.6.2, and finally in Equation 2.6.1, we compute the total energy consumption for the 1 bit data transfer.

In all scenarios, the triangular tiling and SST outperform the other strategies. The energy consumption of ETT placement is comparable to them. Surprisingly, the uniform random placement is comparable to the square tiling for energy consumption performance. This is due to the sink placement strategy where the sinks are placed at CGSCs, which is known to be good for a uniformly distributed network [106].

3.8.3. Worst-Case Delay

The analytical results of the worst-case delay comparison among strategies are shown in Figure 3.8.5. For SNC computations, the popular token-bucket arrival curve and rate-latency service curves are considered. In particular, for the service curve we use a rate-latency function that corresponds to a duty cycle of $1\,\%$. For the $1\,\%$ duty cycle, it takes $5\,ms$ time-on-duty with a $500\,ms$ cycle length which results in a latency of $0.495\,s$[1]. The corresponding forwarding rate is $2500\,bps$. The results of 100-, 500-, and 1000-node networks under a varying number of sinks are shown in Figure 3.8.5. The worst-case delay of SST under the exact placement outperforms the other strategies in all scenarios. In fact, TT and ETT have a very similar delay performance as SST. An interesting point is that the uniform random placement is very comparable to the ST placement.

3.9. Performance Evaluation Under Environmental Disturbances

Various difficulties may occur in deploying exact node placement strategies in WSNs. In fact, an exact node placement strategy may not be applicable in many WSN applications due to environmental and geographical impacts. In order to make node placement strategies realistic and usable in real world scenarios, a network designer should investigate such disturbances for the deterministic node placement strategies.

An environmental disturbance is an event having the potential to affect the deployment. It can be caused naturally or by human action. In

[1] The values are calculated based on CC2420AckLpl.h and CC2420AckLplP.nc.

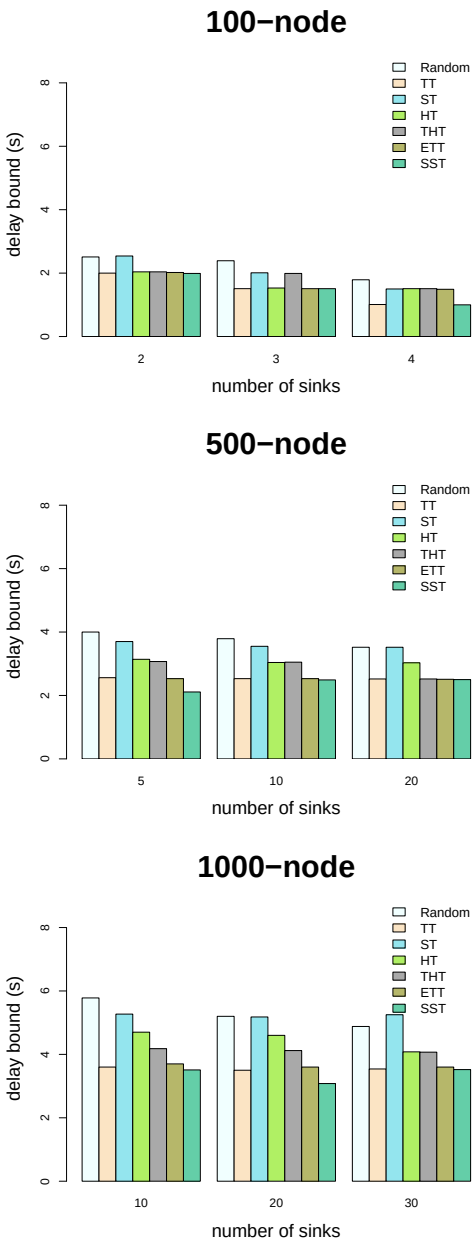


Figure 3.8.5.: Worst-case delay comparison of three strategies in different scenarios.

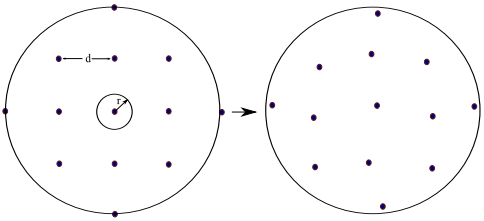


Figure 3.9.1.: An example of a square tiling with gentle disturbance.

WSN applications, where sensor nodes are embedded in natural environments, nodes are subject to environmental disturbances like fire, flooding, and cyclones. Furthermore, the positions of nodes may also be altered by human action or the deployment process is inexact due to human errors. Intuitively, the environmental disturbance completely affects the whole network and can occur during the deployment process or during the runtime of the network. Thus, we undertake this investigation in order to see the performance of deterministic node placement strategies under a disturbance effect. Although, some environmental disturbances may destroy a WSN entirely, in our study, we focus on a gentle environmental disturbance like a slight meteorological disturbance or placement inaccuracies. Obvious examples of the first kind are wind and rain that may disturb the original deterministic nodes' positions by moving them to new locations. An example of a gentle disturbance is shown in Figure 3.9.1 where each node moves to a new position, which is a uniform random point within a disturbance radius r centered at the original node position.

We create a gentle disturbance for six tiling placements by slightly perturbing nodes n from their original positions, as shown in Figure 3.9.1. By varying the disturbance radius r, we examine the effect of a physical disturbance on tiling-based node placement strategies under the same experimental setup used in Section 3.8. The disturbance radius is varied up to $100\,\%$ of each cell length r_{sense}, for instance, $11\,m$ with step size of $1.1\,m$ for square tiling. Among several experiments, the results of a 500-node network are presented. Moreover, the same performance metrics are analyzed for each strategy under such disturbances.

3.9.1. Coverage

A systematic sampling with a granularity of $0.5\,m$ is used in all coverage experiments. The coverage distributions of tilings under gentle disturbance

up to 100% of the sensing range are shown in Figure 3.9.2 and 3.9.3. In most cases, two of the highest k-coverage are decreasing while increasing the other exact k-coverage except for THT. Accordingly, the average coverages are gradually decreased and, in contrast, the standard deviation is increased. The higher the disturbance radius, the lower the average coverage appears in all tilings except for HT and SST. It is obvious that all tilings have coverage holes, i.e., exact 0-coverage, as well as new exact k-coverages ($k = 5$ or 6) appeared after the disturbance effect. Among them, HT and THT have the highest amount of exact 0-coverage due to the hexagonal cells.

Figure 3.9.2(a) represents the exact k-coverage distirbutions for the triangular tiling which has an average of 3.56-coverage with a standard deviation of 0.72 at 10% disturbance radius while having an average of 3.50-coverage with a standard deviation of 1.17 at 100%. The variation of the square tiling coverage due to disturbance can be seen clearly in Figure 3.9.2(b). The square tiling has an average of 3.09-coverage with a standard deviation of 0.79 at $r = 1.1\,m$. At $r = 11\,m$, the square grid has an average of 3.03-coverage with a standard deviation of 1.07. Relative frequencies of exactly k-covered points of the disturbed hexagonal tiling are shown in Figure 3.9.2(c). An interesting thing is that the HT maintains almost the same average coverage but dramatically increases the standard deviation. At $r = 0.97\,m$, HT has an average of 2.36-coverage while increasing its standard deviation from 0.64 to 0.92. The coverage results of the disturbed THT strategy are shown in Figure 3.9.2(d). At $r = 1.0\,m$, THT has an average of 2.64-coverage with a standard deviation of 0.64. Also, the standard deviation increases gradually. For example, at $r = 1.0\,m$, the average coverage is 2.63 with a standard deviation of 0.98. The exact k-coverage distributions of ETT and SST under gentle disturbance are very similar. However, the average coverage performance of ETT decreases from 3.25- to 3.24 with a respective standard deviations of 0.77 and 1.11. In constract, SST increases slightly after the disturbance is applied. In particular, SST has an average of 3.24-coverage with a standard deviation of 0.72 at 10% of disturbance radius. At $r = 11.4\,m$, SST has an average of 3.25-coverage with a standard deviation of 1.09.

According to the experimental results, the original coverage performance of each tiling is slightly affected by gentle disturbances. An interesting point is that the average coverage of SST increases slightly after the disturbance is applied.The HT has almost constant average coverage under gentle disturbance.

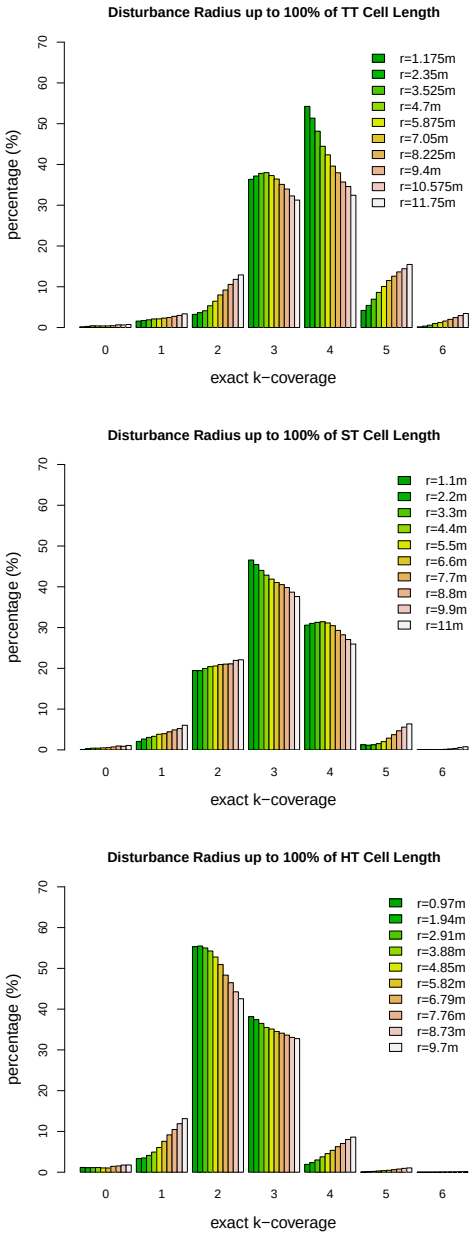


Figure 3.9.2.: The distributions of exact k-coverage at various disturbance radii under TT, ST, and HT placements.

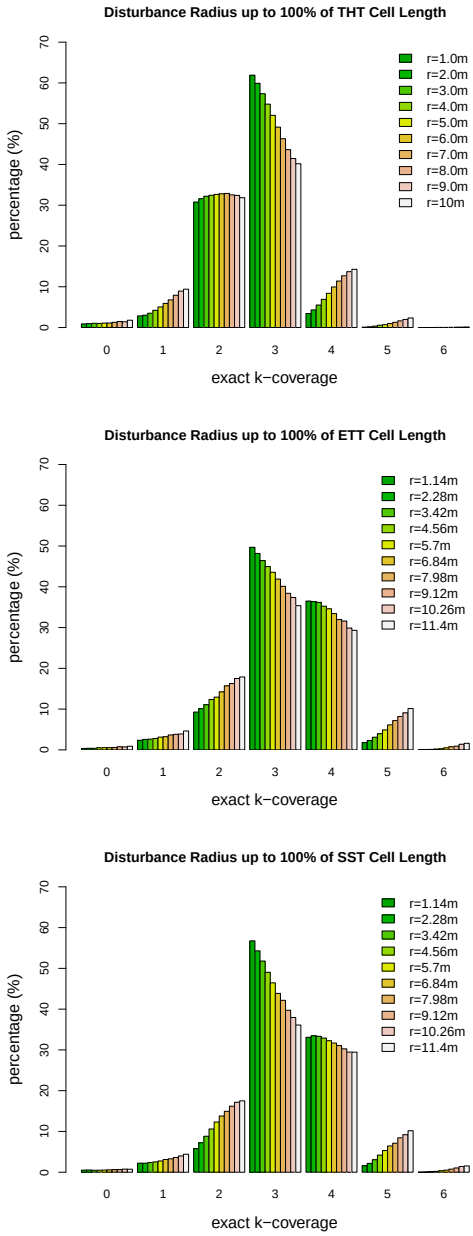


Figure 3.9.3.: The distributions of exact k-coverage at various disturbance radii under THT, ETT, and SST placements.

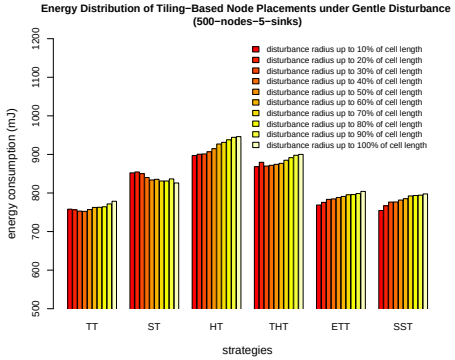
3.9.2. Energy Consumption

We investigate the energy consumption of each tiling under gentle distrubance effects. For comparison, we consider the same parameters used in Section 3.8.2. The results of a 500-node network with 5-, 10-, and 20-sinks are presented in Figure 3.9.4. It is interesting to note that in all the experiments the energy consumption of the square tiling under gentle disturbance becomes lower than the exact square tiling placement.

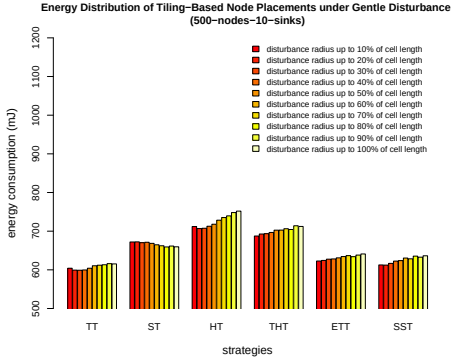
We first look at the energy consumption of the disturbed triangular tiling placement. The confidence interval lengths of 5-, 10-, and 20-sink scenarios are 4.45, 3.92, and 3.94, respectively. Under the same scenario, the square tiling has the confidence interval lengths 6.98, 4.92, and 4.17 whereas the hexagonal tiling has the confidence interval lengths of 6.67, 5.63, and 3.53 respectively.

The experimental results show that the energy consumption of THT is rising. In 5-, 10-, and 20-sink scenarios, the confidence interval lengths for all disturbance radii are 7.28, 5.16, and 3.55. The energy consumptions of ETT and SST with gentle disturbance increase gradually. While ETT has confidence interval lengths of 5.83, 4.54, and 3.45, the SST has 6.02, 4.43, and 4.37 for 5-, 10-, and 20-sink scenarios.

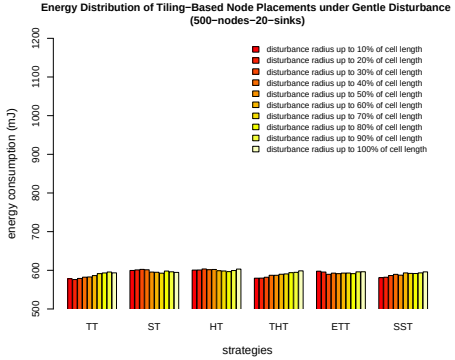
In conclusion, the energy consumption of the square tiling strategy is better after a slight perturbation of the exact node placement. The conjectured reason for it is that the node degree (i.e., the number of neighbors) increases when disturbance is applied. It implies that each node has a shorter path, which means a lesser hop count to the nearest sink. Consequently, the energy consumption is lower according to the hop-based energy model. Besides, the positive impact is more evident in the square grid strategy than in other strategies due to its symmetrical nature. It seems that the symmetric cells of square tiling are good for a disturbance to minimize the energy consumption. The results of TT, ETT, and SST are very similar under gentle disturbances up to 100% of the respective cell lengths. Among all strategies, HT has the highest energy consumption under gentle disturbance.



(a)

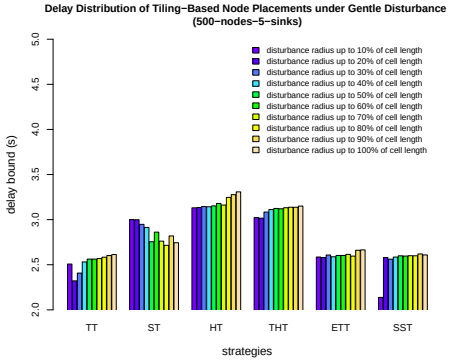


(b)

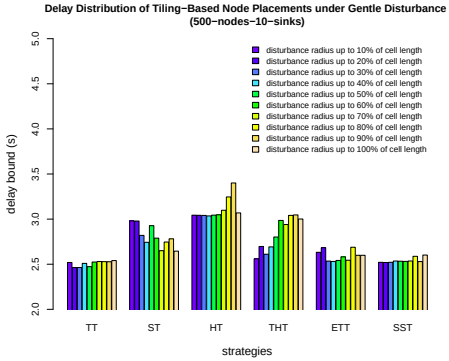


(c)

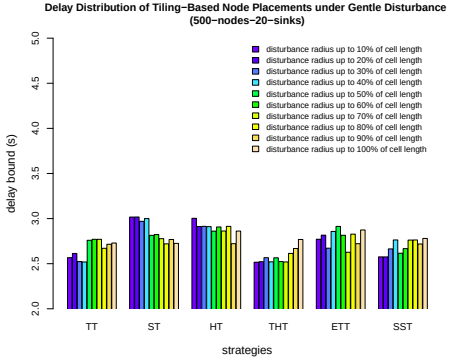
Figure 3.9.4.: The distributions of energy consumption at various disturbance radii under tiling-based node placement strategies (a) a 500-nodes-5-sinks, (b) a 500-nodes-10-sinks, and (c) a 500-nodes-20-sinks.



(a)



(b)



(c)

Figure 3.9.5.: The distributions of the worst-case delay of tiling-based node placements at various disturbance radii (a) a 500-nodes-5-sinks, (b) a 500-nodes-10-sinks, and (c) a 500-nodes-20-sinks.

3.9.3. Worst-Case Delay

Figure 3.9.5 presents the results of the worst-case delay of tilings with disturbance. However, the disturbed square tiling has a better worst-case delay than the original exact placement. The worst-case delay of the square tiling decreases sharply in the disturbed version. Other strategies are affected negatively in their delay performance except the disturbed HT placement for the 500-nodes-20-sinks scenario. Among all strategies, SST has the most stable delay performance for 500-nodes with 5- and 10-sinks cases except a significant delay change from $2.13\,s$ to $2.58\,s$ between $r = 1.14\,m$ and $r = 2.28\,m$ in the 500-nodes-5-sinks case.

3.10. Performance Evaluation Under 3D Effects

In WSNs, a disturbance can also be caused by obstacles that may block the location of the exact node placement positions. The amount and type of obstacles depends on specific WSN applications. Depending on the environment, these obstacles can be stationary, movable, or moving. For the sake of simplicity, stationary obstacles that cannot be moved, neither by themselves nor by others, such as buildings and hills, are taken into account. In order to make the node placement strategy for real world scenarios realistic and usable, blocking obstacles should be studied in designing WSNs.

Among the variety of obstacles, we first focus on ground contours, which prevent deploying a precise pattern of node placement. Obviously, geographic node placement strategies should investigate the ground contours caused by geographic nature. Furthermore, such an investigation is necessary for some WSN applications, such as environmental and agricultural monitoring, where varieties of ground contours may exist. In fact, such a type of disturbance, is also suitable for analysis of node placement strategies, especially for WSN applications in mountainous areas. An illustration of such a 3D disturbance is presented in Figure 3.10.1. We evaluate the performance of all tilings with 3D disturbance under the same experimental setup from Subsection 3.9.

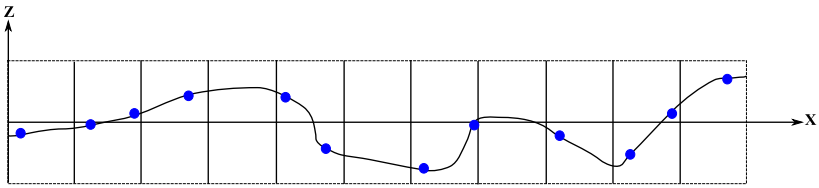


Figure 3.10.1.: A cross sectional view of 3D disturbance.

3.10.1. Coverage

The coverage distributions of regur and semi-regular tiling-based node placement under 3D disturbance can be seen in Figure 3.10.2(a)-(f). The average coverage performance of each tiling strategy decreases dramatically with an increasing standard deviation with respect to an ascending order of disturbance radii. The triangular tiling has an average 3.49-coverage at $r = 1.175\,m$. At $r = 11.75\,m$, it has only an average of 1.97-coverage. The corresponding standard deviations are 0.66 and 1.15 respectively. The average coverage performance of the square tiling strategy decreases dramatically from 3.04- to 1.7-coverage with an increasing standard deviation from 0.77 to 1.05. The hexagonal tiling reduces its average coverage performance from 2.3- to 1.3-coverage with standard deviations of 0.58 and 0.9, respectively. Also, the THT strategy with 3D disturbance decreases dramatically from 2.6- to 1.48-coverage.The corresponding standard deviation also goes from 0.6 at $r = 1.0\,m$ to 0.98 at $r = 10\,m$. The ETT and SST also have very similar average coverages under 3D disturbance. The average coverage performance dramatically decreases from 3.22- to 1.86-coverage for ETT and 1.84-coverage for SST under an increasing disturbance radius. While ETT has corresponding standard deviations of 0.72 and 1.09, the SST has standard deviations of 0.67 and 1.1. From this experiment, it is clear that the coverage performance is significantly influenced by the 3D disturbance.

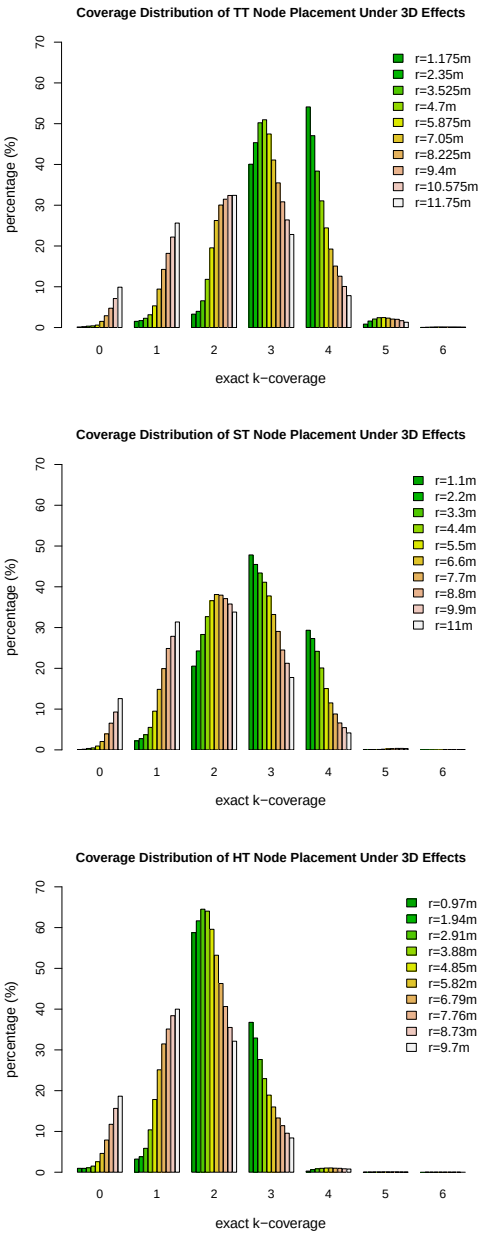


Figure 3.10.2.: The distributions of exact k-coverage at various 3D disturbance radii under TT, ST, and HT placements.

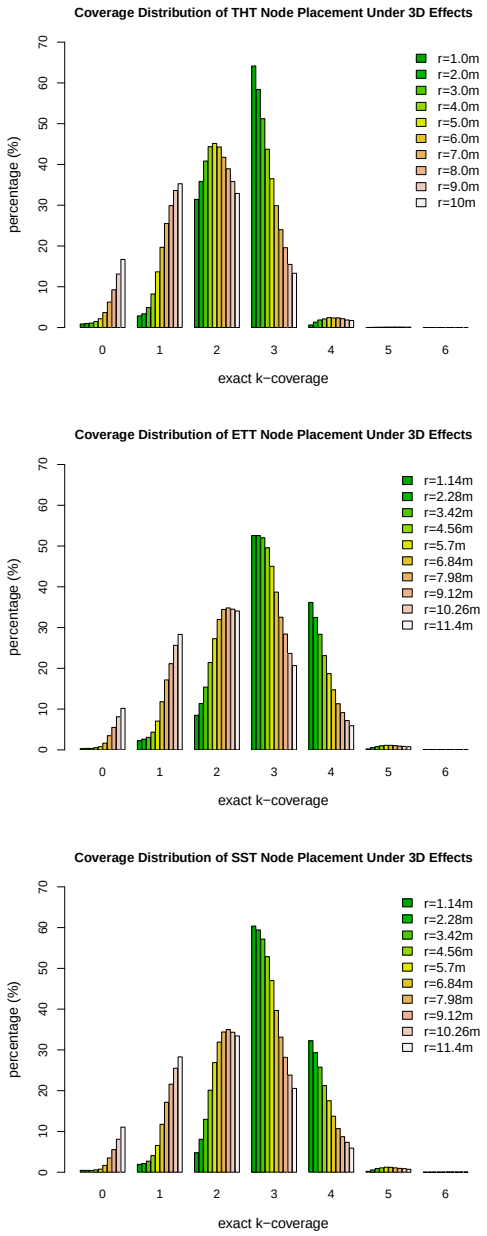


Figure 3.10.3.: The distributions of exact k-coverage at various 3D disturbance radii under THT, ETT, and SST placements.

3.10.2. Energy Consumption

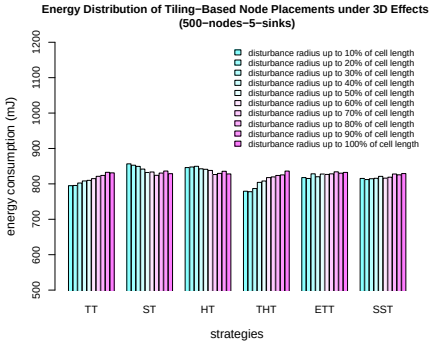
The energy consumptions of 500-node tiling-based node placements with 3D disturbance are presented in Figure 3.10.4. Similar to the gentle disturbance, the energy consumption is better after applying 3D disturbance in the square grid strategy. Under 3D disturbance, HT node placement also improves its energy consumption in all scenarios. The energy consumptions of HT and THT under 3D disturbance are better than for 2D disturbance. In contrast, TT, ETT, and SST consume more energy under 3D disturbance than for 2D gentle disturbance. An interesting point is ST maintains stable energy consumption under 2D and 3D disturbance.

3.10.3. Worst-Case Delay

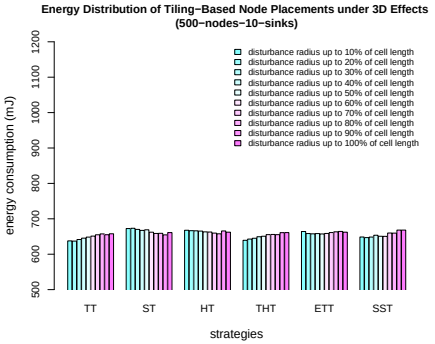
The analytical results of worst-case delay computations under 3D disturbed tilings are shown in Figure 3.10.5.

In triangular tiling, the worst-case delay bounds are negatively influenced after 3D disturbance is applied. As soon as 3D disturbance is applied, the worst-case delay of the square and hexagonal tilings decrease in all scenarios. It can be seen clearly for hexagonal tiling. In most scenarios, the worst-case delay of HT is decreasing with increasing disturbance radii, e.g., from $3.09\,s$ at $10\,\%$ of disturbance to $2.62\,s$ at $100\,\%$ of disturbance radius in 500-nodes-5-sinks scenario. Under this experiment, the worst-case delay bounds of THT are not bad. For most disturbance radii, THT provides the lowest worst-case delay bounds. In a 5-sink scenario, $r = 2\,m$ has the minimum worst-case delay of $2.58\,s$ whereas the minimum worst-case delay of $2.51\,s$ is found at $r = 4\,m$ for the 20-sink scenario. The more sinks, the lower the worst-case delay is.

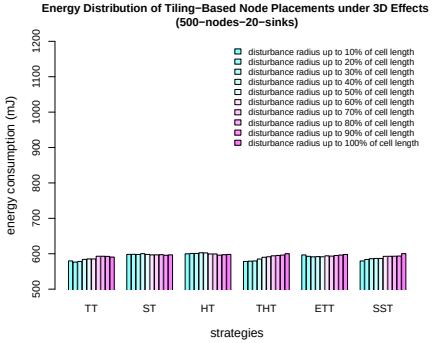
The worst-case delay performance of ETT and SST are not so similar to the energy consumption performance under 3D disturbance effects. Their worst-case delay bounds are quite chaotic in this experiment. Their delay bounds seem increasing in 500-nodes-5-sinks scenario but both have decreasing and increasing tendencies in 500-nodes-10-sinks scenario. In 500-nodes-10-sinks scenario, ETT has better worst-case delay than SST.



(a)

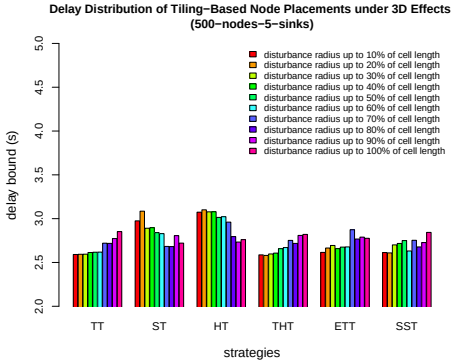


(b)

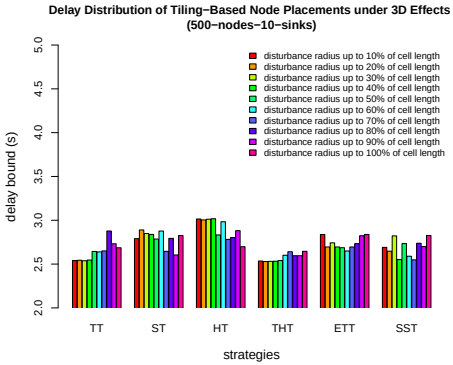


(c)

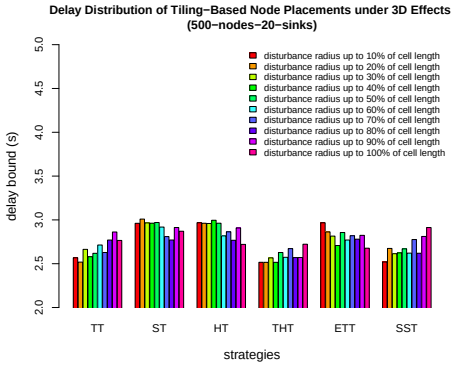
Figure 3.10.4.: The distributions of energy consumption of tiling-based node placements at various 3D disturbance radii (a) a 500-nodes-5-sinks, (b) a 500-nodes-10-sinks, and (c) a 500-nodes-20-sinks.



(a)



(b)



(c)

Figure 3.10.5.: The distributions of worst-case delay of tiling-based node placements at various 3D disturbance radii (a) a 500-nodes-5-sinks, (b) a 500-nodes-10-sinks, and (c) a 500-nodes-20-sinks.

3.11. Discussion and Conclusion

We made the following findings for exact placement schemes of six tilings and a unifrom random distribution.

- TT has the best coverage performance among the selected node placement strategies. A triangular tiling has average 3.63-coverage whereas ETT and SST have the same average 3.36-coverage.
- TT and SST outperform other strategies for energy consumption and worst-case delay.
- It can also be seen that random deployment is not a bad strategy and it is comparable to the popular square grid deployment for the worst-case delay.

In case the exact placements are stochastically disturbed by 2D and 3D effects, we found out the following:

- The original coverage performance of each tiling is slightly affected under gentle disturbance. However, the coverage performance is significantly influenced under 3D disturbance.
- The square tiling is influenced positively after applying disturbances in terms of the energy and worst-case delay.
- The energy consumptions under 2D and 3D disturbance effects are a bit chaotic. The energy consumptions of HT and THT under 3D disturbance are better than for 2D disturbance. In contrast, TT, ETT, and SST consume more energy in 3D disturbance than for 2D disturbance. An interesting point is that ST maintains a stable energy consumption for 2D and 3D effects of disturbance.
- Another interesting finding is that the node degree (i.e., the number of neighbors) increases in the disturbed versions of exact placements. This implies that many node have a shorter path (i.e., a lesser hop count to the nearest sink). As a result, energy consumption is lower in some tilings such as ST according to the hop-based energy model. Also, it affects the performance of worst-case delay.
- ETT and SST have a similar performance for all metrics under by 2D and 3D disturbances.
- Note that the HT placement uses the smallest sensing and transmission ranges compared to other placements in our models.

In each specific WSN application, the performance metrics of interest may differ from each other. In this chapter, we investigated the effects of tiling-based node placement on the most common performance issues of coverage, energy consumption, and worst-case delay.

4. Placing Multiple Static Sinks in Large-Scale, Time-Sensitive WSNs

4.1. Introduction

4.1.1. Background and Motivation

In many WSN applications, it is desired to collect the information acquired by sensors for processing, archiving and other purposes. The station where that information is required is usually called a sink or base station. A sink normally has higher capacity as well as cost than usual sensor nodes. Sinks can be sensors themselves or devices such as PDAs or gateways to other larger networks [61]. For large-scale WSNs, a single-sink model is not scalable since message transfer delays as well as energy consumption of the sensor nodes become prohibitive, due to the fact that most of the nodes would be far away from the sink and thus many hops must be traversed before the sink is reached. As a result, response times become excessive and the lifetime of the WSN becomes very short. Therefore, it is sensible to deploy multiple sinks so that messages reach their destination with less hops and consequently response times are decreased and energy is saved.

In WSNs, data packets traverse from the sensor nodes to a sink through multi-hop communication owing to expensiveness or even infeasibility of direct communication. Therefore, a sensor node is not only a source but can also be a router. For large-scale WSNs, if a sensor node acts as a router close to a sink, it can experience quite high amounts of data from other nodes flowing through it. As a result, the traffic intensity in the network becomes high and data transfer may be considerably delayed. What is more, if the sink is located far apart from some nodes, their hop distances increase, simutaneously amplifying the message transfer delay. Sink placement at the right position can minimize maximum message transfer delay and can increase the lifetime of the network since the loads are shared and sensors nodes can choose the nearest sink so that communication to a sink

node is faster and saves energy as well.

In general, optimal sink placement is a hard problem like many other location problems (it exhibits particular similarity to the uncapacitated warehouse location problem, an NP-complete problem). Moreover, the location problem is even more complex if the problem is an uncapacitated problem. Obviously, our problem is such a kind of an uncapacitated problem.

In WSNs, sinks can be either mobile or stationary. In this chapter, we will focus on stationary sink placement. We design two types of sink placement strategies under different assumption. An obvious distinction between the two strategies is node locations information. The proposed sink placement strategies in this chapter are mainly intended for time-sensitive WSN applications. We developed a genetic algorithm sink placement (GASP), which performs well and delivers near-optimal solution. If it is feasible to provide global information such as nodes' locations, GASP is a good choice. GASP is based on a discretization of the originally continuous search space into a finite search space, which forms a contribution of our work of independent interest. The idea is we first introduce a method to discretize the search space resulting in a set of candidate locations for the placement of sinks in a WSN. These candidate locations are based on the concept of *regions of indifference*, that is regions for which wherever a sink is placed the routing topology will not be altered and thus the objective function for the worst-case delay does not change. Hence any location in such a region of indifference can be chosen as a candidate location. Nevertheless, for large-scale WSNs the number of candidate locations still grows very fast and an exhaustive search becomes quickly prohibitive. That is why we consider a GA-based strategy, although, of course, it cannot guarantee to arrive at a provably optimal solution. The set of candidate locations is the very heart of the GASP algorithm. The GASP initially picks random sink candidate locations as initial individuals and computes the fitness function. By applying the well-known GA operators such as mutation, crossover and selection, GASP operates the evolutionary loop until either a good solution is found or the maximum number of generations is reached.

Avoiding the costly design of using the nodes' locations information, we also introduce a self-organized sink placement (SOSP) algorithm with lower computation and communication overhead to minimize the maximum worst-case delay. In brief, the novel algorithm for SOSP is solely based on rough knowledge about the number of sensor nodes and the number of sinks being used. The sensor field shape is assumed to be circular, although generalization from that should be straightforward. Based on this, we choose the initial sink locations at the center of gravity of the

sector of a circle (we call CGSC further on). Each sink represents a group. Moreover, we consider movable devices for the selection of sink locations. The initially located sinks now communicate with their 1-hop neighbors by broadcasting a message. Using trilateration-based localization, we estimate the distances and thus locations of the 1-hop neighbor sets. Then we suitably define fixed candidate locations in the 1-hop neighbors' regions (i.e., in one of the intersection sets of 1-hop neighbors' transmission ranges). Finally, the moved sinks select the best sink location among the set of candidate locations which minimize the maximum worst-case delay. For the best sink locations, SOSP consider stationary sinks to collect data packets from the sensor nodes via multi-hop communication.

4.1.2. Contributions

We contribute the following in this chapter:

- One contribution is the set of candidate locations for sink placement by discretizing the original continuous search space into a finite search space based on the concept of regions of indifference, that is regions for which wherever a sink is placed the routing topology is not altered. (→Section 4.4.1)
- We develop a near optimal heuristic sink placement called genetic algorithm sink placement (GASP) for time-sensitive WSNs where global information is available. (→Section 4.4.3)
- Avoiding the costly assumption of using the nodes' locations information, we introduce a self-organized sink placement (SOSP) algorithm with lower computation and communication overhead to minimize the maximum worst-case delay.
  (→Section 4.5)
- A thorough simulative investigation and comprehensive comparison with alternative approaches inspired by literature is presented. (→Section 4.6)

4.1.3. Outline

The goal of this chapter is to develop good strategies for the sink placement problem that minimizes the maximum worst-case delay in WSNs based on the sensor network calculus framework. For different target networks we

develop two sink placement strategies under different assumptions and analyze their performance.

The organization of this chapter is as follows: In Section 4.2 we discuss the related work. The assumptions and problem formulation are presented in Section 4.3. In the next section, we introduce the heuristic sink placement of the GASP strategy alongside with the method for discretization of the search space. After that we introduce a self-organized sink placement of SOSP strategy alongside with deployment, grouping, sink placement selection, and its operational phases in Section 4.5. Next, we analyze the performance of the GASP and SOSP by comparing it to other strategies having the same level of abstractions in Section 4.6. One of the competitors for GASP is an optimal strategy based on an exhaustive search which serves as an upper bound for the performance achievable by the GASP, and the other one is a Monte-Carlo based strategy that should serve as lower bound on the performance that can be expected from the GA. For the SOSP strategy, we compare it against previously proposed benchmarks called GSP and the near optimal GASP. During the performance analysis of the sink placement strategies, the question of a "good" number of sinks given a certain number of sensor nodes is also discussed. Finally, we conclude this chapter in Section 4.11.

4.2. Related Work

In this section, we discuss the existing work on the sink placement problem in WSNs.

Although a sink location can be fixed or mobile in WSNs, we will focus on the static sink placement problem in this chapter. WSNs with mobile sinks will be discussed in Chapter 5. First, we will discuss the related work of single sink and multiple sink placement problem. Since multiple sinks are preferrable for large-scale WSNs, we will discuss two main categories of sink placement strategies in WSNs: (1) using the global knowledge of the sensor nodes' locations and (2) based on the estimates or no use of the nodes' locations. In the first group, the node locations may be obtained from GPS receivers or are simply known from a planned deployment process. Obviously, it suits only small-scale WSNs because of the corresponding computation and communication overhead. The second category is sink placement based on the estimation of node locations, which uses, for example, anchor points. Yet, these approaches still seem infeasible for large-scale WSNs. A geometric way of sink placement does not require

the nodes' locations information. Obviously, the efficiency of such kind of sink placement is typically lower compared to known or estimated nodes' information based techniques.

If the nodes are unifromly distributed and only a single static sink is available, in general, the sink is placed at the centroid of the network, for example, in the center if the network is circular or square shape. This is proved by [82], where it is shown that the center of the circle is the optimum position for a single sink in WSNs under the uniform distribution of sensor nodes. Nevertheless, a single sink is not sufficient for large-scale WSNs because nodes near the sink always have a burden of forwarding a huge amount of data from the other nodes.

In fact, the multiple sink location problem is NP-complete, so finding the exact optimal sink placement is very hard. In [23], the sink placement problem is reduced to the dominating set problem on a unit disk graph and is proven to be NP-complete. Recently several studies [99, 65, 139, 55, 107, 140] handle locating multiple sinks with different approaches which include integer linear programming, exhaustive search and iterative clustering. The problem is even addressed as a flow problem in [23].

A number of works solve the problem in the way of integer linear programming [55, 107]. In [55], the authors studied the joint problem of energy provisioning and relay node placement (EP-RNP) for the upper tier aggregation and forwarding nodes (AFNs) to increase network lifetime. Due to the hardness of the EP-RNP problem, the authors developed an efficient polynomial-time heuristic algorithm, SPINDS, that solves the EP-RNP problem as an iterative LP problem. Moreover, the authors considered a one-time energy provisioning for the network with the objective of maximizing network lifetime. In [107], the authors explored the placement problem under three wireless link models. The authors formulated the placement problem for each link model as an integer linear program.

In [23], two heuristic approaches are considered for sinks' location selection to minimize the power used to collect the data. The greedy scheme deploys the sinks incrementally. Basically, the algorithm picks the position of sinks one-by-one in a greedy manner to improve the data rate by means of the power efficiency. Although the greedy algorithm is considered as a good strategy, it generally does not perform well. The second approach is a local search technique which is more powerful than the greedy algorithm. In the local search, the sinks are initially placed randomly. In the next steps, each sink tries to locate the positions of the neighboring sensors to find a better position in order to maximize the data rate. The algorithm stops if no improvement is possible and returns the best solution. In fact,

the local search is an exhaustive search which actually is a drawback for large-scale WSNs. Another tight restriction is that the sink location must be one of the nodes' locations. Accordingly, the search space is limited and the optimal position for sink placement is infeasible under this restriction.

Another heuristic sink placement can be found in [154, 9]. In [154], the authors propose the Genetic Algorithm for Hop Optimization (GAHO) approach which is based on artificial neural networks for optimal sensor grouping by reducing the number of hops from sensors to sinks with the intension of minimizing the delay. We also introduce a Genetic Algorithm based sink placement for optimal sinks' locations selection in this chapter but not for grouping purpose. In fact, [154, 9] considered a clustering technique in WSNs. There are many good clustering algorithms in literature, which are mainly divided into hierarchical and non-hierarchical methods [151]. If a clustering method is applied in WSNs, the cluster head or the centroid of a cluster is usually considered as a sink (see, e.g., [99] and [65]). Instead of choosing the centroid of the cluster for sink placement, the GAHO approach in [154] focuses on the centroid to find a nearby position such that the sink has the highest number of neighbors in its communication range within the cluster. Clearly, it is a computationally quite expensive approach. The sinks' location selection in our SOSP algorithm is very similar with this approach. However, we do need only 1-hop neighbors of initial sink locations while they need the global information of nodes' locations for sink location. Another advantage is that the SOSP algorithm optimizes the actual delay instead of the hop count. A lighweight version of GAHO called Genetic Algorithm for Distance Optimization (GADO) is proposed in there, where the idea is to optimize the Euclidean distance. Using a geometric approach for sink placement in clustering, GADO reduces the complexity of the algorithm but it still needs the nodes' locations.

While [99] and [65] focus on energy-efficiency as optimization criterion, [154, 9] intended to minimize the delay perfromance under clustering technique. In order to fulfill both optimization criteria, we focus on the delay while achieving energy-efficiency goals by setting duty cycles accordingly. By using clustering, in most cases, one cannot guarantee to obtain the optimal location because most clustering algorithms choose the clusterhead location according to the Euclidean distance. On the other hand, clustering is a kind of geometric placement.

In addition, a number of different approaches are considered for sink placement problem in WSN. Facility location problems are also related to sink placement problem, and have been considered extensively in the

field of operations research (e.g., [68, 137, 41]). In [150], three random deployment strategies for relay nodes, namely, connectivity-oriented, lifetime-oriented, and hybrid deployment are presented. A hybrid approach of using static and mobile sinks is presented in [147] to solve the "offset" problem caused by the networking-wide broadcasting. The idea is that mobile sink broadcast their locations with limited transmission range. Whichever nodes received the broadcast message send their data to this mobile sink whereas the other nodes are assigned to communicate to the static sink. However, their work only focuses on a single static sink and a mobile sink which is not applicable for large-scale WSNs.

All the above work assumes global knowledge of the network such as where the nodes are located, what the energy levels of the nodes are, and so forth. Thus position-awareness of the sensor nodes becomes a critical issue for large-scale WSNs.

Although many work consider sensor nodes equipped with global positioning system (GPS) hardware, for many applications this is too expensive. It may be even infeasible for some applications due to environment issues. In this case, other localization techniques are preferrable. Of course, there are plenty of localization algorithms [93, 128, 50] that can work quite well under some a priori knowledge of a few nodes' positions and provide an approximation of others. A computational geometry based approach is considered in [101, 102] where the sink location problem is formulated as finding the minimum enclosing circle problem for 1-tiered and 2-tiered WSNs. Such a circle can be formed if three locations of sensors are known at most. In [140], the authors introduced a mathematical model to determine the sink locations that minimize the average hop distance with the intension of lifetime maximization. An interesting part of the paper is that the authors proposed a 1-hop algorithm using the locations of the neighboring sensors against a global algorithm using the locations of the sensors. The 1-hop algorithm is similar to our SOSP algorithm but the difference is that the authors considered relocation of the sinks' position within 1-hop neighbors while we are interested to use movable sinks to get the best locations during the design phase.

The primary objective of most of the above sink placement strategies is lifetime maximization. In contrast, we intend to optimize the maximum message transfer delay as our primary goal.

4.3. Assumptions and Problem Statement

As discussed in Subsection 4.1.1, if sinks are placed in good locations, this can reduce traffic flow and energy consumption for sensor nodes. Although energy is usually considered the most critical resource in WSNs, some applications depend heavily on performance characteristics such as message transfer delays or buffer requirements. In particular, message transfer delay is the most critical issue for time-sensitive WSN applications like fire or intrusion detection systems in order to ensure a timely actuation in case of a detection event. Therefore, we focus on strategies to minimize the maximum worst-case delay, which is important for any timely actuation based on the information collected by a WSN. Hence, a worst-case analysis approach is considered. While average-case analysis is useful in some applications, for many WSN applications like production surveillance or fire detection, it must be ensured that messages indicating dangerous information are not lost (with high probability) and arrive at the control center with minimum delay.

However, that brings up the question how these sinks can be placed optimally in order to achieve a minimum message transfer delay for time-sensitive applications. In mathematical terms we can pose the problem as follows:

$$min. \max_{i \in \{1,\ldots,n\}} \{d_i\}$$

with

$$d_i = f\left(\tau | \vec{\alpha}, \vec{\beta}\right)$$

$$\tau = g(\vec{s} | \vec{p}, \mathcal{R})$$

$$\vec{p} = \left(p_i^{(x)}, p_i^{(y)}\right)_{i=1,\ldots,n}$$

$$\vec{s} = \left(s_j^{(x)}, s_j^{(y)}\right)_{j=1,\ldots,k}$$

$$p_i, s_i \in \mathcal{F}$$

Here, n denotes the number of sensor nodes and $\vec{p}$ is the vector of their locations in the sensor field $\mathcal{F}$, these locations are assumed to be given. The values d_i are the worst-case delays for each sensor node i. By minimizing the maximum worst-case delay in the field, it is ensured that response times are balanced as far as possible. k is the number of sinks

and the vector $\vec{s}$ contains their locations, these locations are the actual decision variables of the optimization problem. This is somewhat hidden by the fact, that the delays d_i are only indirectly affected by the choice of the sink locations, because as a first-order effect, the d_i are a function of the topology τ in which the WSN organized the data flow towards the sinks and the arrival and service curves of the sensor nodes, denoted as $\vec{\alpha}$ and $\vec{\beta}$ in the formulas above. While the arrival and service curves are parameters for a given WSN scenario, the topology is itself a function of the nodes' locations, the routing algorithm $\mathcal{R}$ and the sinks' locations $\vec{s}$, where however only the sinks' locations are variable and the other two are again given parameters. Note, in particular, that we assume the routing algorithm to be given and not to be subject to the optimization. Although this could in principle be done, it would aggravate the problem further and is therefore left for future study.

So, in principle, we face a continuous optimization problem where the objective function is to minimize the maximum worst-case delay in the field subject to constraints that ensure that each sensor node is connected to a sink (possible via multi-hop communication, determined by the routing algorithm) as well as some geographic constraints. Due to the highly non-linear, jumpy behaviour of the worst-case delay function f, and thus of the objective function, which results from the formulas derived by sensor network calculus (see [115]), the direct solution of that optimization problem is practically infeasible.

For this reason, we introduce alternative solutions for different target networks in the next sections: a heuristic-based sink placement for known sensor nodes' locations and a self-organized sink placement strategy for unknown sensor nodes' locations.

4.4. Genetic Algorithm Sink Placement

In this section, we first propose a discretization method that reduces the continuous search space into a finite set of candidate locations. Each candidate location represents a so-called region of indifference, in which wherever we place a sink the routing topology is not changed and thus also the value of the objective function, the maximum worst-case delay, is not altered. Hence any location in such a region of indifference can be chosen as a candidate location. So, interestingly, the discretization of the search space does not result in a loss of optimality. Yet, as we still face a hard combinatorial optimization problem similar to warehouse

location problems (which are known to be NP-complete) where however the objective function is highly non-linear in contrast to those, the only resort for larger problem instances is a heuristic search technique. In the following subsections, we discuss the method for the discretization of the search space and the interior workings of the Genetic Algorithm sink placement (GASP) for known sensor node locations.

4.4.1. Discretization of the Search Space

We assume the locations of the sensors and their transmission ranges to be given. In a first step, we use the nodes' locations to calculate the total number of regions of indifference, so that we know the cardinality of the candidate location set. To do so, we look at the intersection regions of sensor nodes' transmission ranges. In that context, nodes are called neighbours if their transmission areas intersect. By the sensor nodes' transmission areas we obtain a tesselation of the sensor field. The atomic regions forming that tesselation become the regions of indifference for the sink placement problem, because inside such a region it does not matter where we place a sink since the routing topology remains invariant and thus the objective function remains unchanged. Thus, within such a region of indifference we can choose any location as candidate location and consequently discretize the search space.

To the best of our knowledge, there has been no explicit formula for the number of intersecting regions when their locations and transmission ranges are given. In Figure 4.4.1, we discuss possible circle intersections and the way to calculate the total number of regions as given by our formula which is given in Equation (4.4.1). In our method, first, we pick a node i with all intersecting neighbors and for each neighbor we add two regions to the total number of regions. Then we eliminate node i from the scenario. Based on this a loop invariant can be formulated which shows the correctness of this procedure. The same procedure will continue until the last node, which has no further neighbors. Notice that the network is connected so that no further neighbors node means the last node of our scenario and its neighbors have already been counted. Finally, we add a region to the total number of regions for the last node.

The formula to calculate the total number of regions of indifference, $N(n)$, for multi-circle intersections is given in Equation (4.4.1) and we give evidence for it with some graphical illustration in Figure 4.4.1.

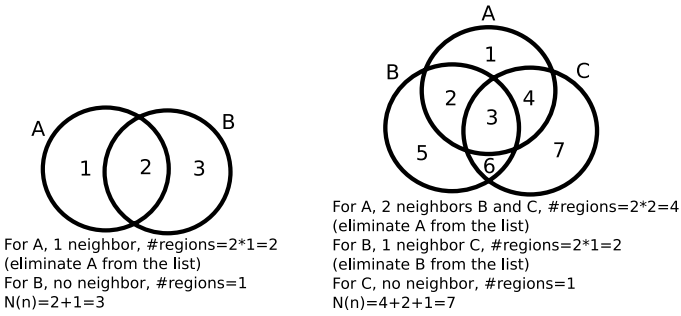


Figure 4.4.1.: Multi-circle intersection regions.

$$N(n) = \left(\sum_{i=1}^{n-1} 2 * nb_i \right) + 1 \tag{4.4.1}$$

where

$n :=$ number of sensor nodes

$nb_i :=$ number of neighboring nodes after elimination of nodes $1, \ldots, i-1$

$N(n)$ is important as it controls a sampling process of the sensor field, where we iteratively increase the sampling as much as necessary until the maximum number of candidate locations is found. The sampling process works by laying a grid over the sensor field, determining for each grid point its neighborhood in terms of the transmission areas in which it is contained and then merging grid points which have identical neighborhoods. If the sampling accuracy is eventually high enough, all regions of indifference have been identified and the respective remaining grid points can be used as candidate locations for the sink placement. While the iterative grid-based procedure may seem computationally intensive and a direct method for calculating the indifference regions based on the locations and transmission ranges of the sensor nodes seems appealing, we had to find out after intensive literature research that the basic geometric problem has not been addressed so far and that even the easier problem of counting the intersection regions had not been addressed, to the best of our knowledge. However, the discretization never was a bottleneck in the practical scenarios we investigated and further has the advantage that it also works if the transmission properties cannot be captured by the typical circular structure of free-space signal loss models.

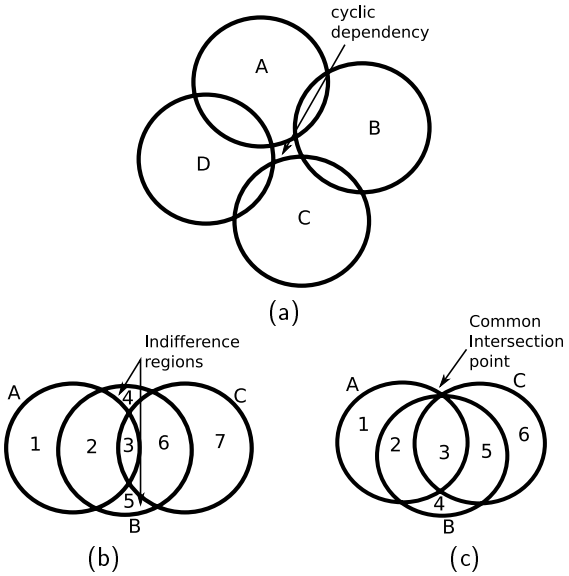


Figure 4.4.2.: (a) Cyclic dependency, (b) indifference regions, and (c) common intersection point.

The alert reader might have noticed that there are some special cases of node constellations which do not agree with Equation (4.4.1) resulting in unwanted or duplicated regions. These special cases are shown in Figure 4.4.2. One such case is a cyclic dependency which results in an unwanted region that is useless for sink placement, because sensors cannot reach that region as shown in Figure 4.4.2(a). Moreover, there is the possibility of scattered regions of indifference, where two or more regions of indifference result in the same routing topology so that actually one of them is sufficient which is illustrated in Figure 4.4.2(b). The discretization disposes with such areas in order to keep the search space minimal. The last case in Figure 4.4.2(c) does not agree with Equation (4.4.1) because some regions of indifference might actually degenerate to a single point in the plane if we have a common intersection point between three or more transmission areas of sensor nodes. This case we rule out based on the observation that in practice this is very unlikely to happen. So, the exact number of regions of indifference may differ from Equation (4.4.1) due to some cyclic dependencies and scattered indifference regions but is still under control of the discretization process. In the end, the search space becomes a finite set of candidate locations for the sinks.

4.4.2. Some Background on GAs

Based on the set of candidate locations for the sinks an exhaustive search can be implemented that tries all possible combinations of sink placements. However, such an exhaustive search, while ensuring to find the optimal sink placement, becomes computationally too expensive for even moderately sized WSNs. More specifically, if the cardinality of the set of candidate locations is denoted by L and k sinks shall be placed, then the number of combinations to be checked becomes $\binom{L}{k}$, which grows quickly with L. So, the only resort from this is to use heuristic techniques. We decided to design a genetic algorithm based heuristic because the GA paradigm is known to be very flexible and allows to integrate domain-specific knowledge well [46].

John H. Holland [54] introduced the term and basic concepts of Genetic Algorithms in order to solve complex optimization problems. While being a random search technique, like a Monte Carlo strategy, a GA attempts to learn as much as possible from its "experience" with the function to be optimized and can actually be reasoned to do so in an optimal fashion due to its inherent parallelism (see [46] for details). While not ensuring optimality for the solution found, it usually at least provides very good approximations as long as the so-called building block hypothesis is satisfied, i.e., if a good solution can be composed from good partial solutions (yet not in a very strict sense). The art of GAs therefore lies in the definition of the chromosomes of the individuals that represent the possible solutions for the problem at hand such that the building block hypothesis is fulfilled.

To implement a GA, the first step is to select initial individuals from the overall search space. Each individual represents a solution of the problem and is coded in the so-called chromosome. The selected set of individuals becomes the initial "population". The population size may vary depending on the search space and the problem characteristics. The next step is to generate new individuals by reproduction, which is often called the crossover operator of GAs, and by mutation. How to exactly implement these operators is problem-specific [46], although a large set of variants to choose from exists. In the last step each individual's "fitness" is computed using the objective funtion, before the final operator of GA, the selection, is invoked to create the next "generation" of individuals which form the new population. Again, also for the selection operator there are many variants one can choose from. Which is best depends again on the problem and is usually subject to experimentation. The termination of a GA can be done in two fashions, either the population is converged, meaning that the

individuals are all the same or at least very close to it, or, more commonly, the number of generations, i.e., the number of evolutionary loops is limited to some number. The latter criterion gives better control on the amount of computation that is invested.

4.4.3. The GASP Algorithm

The GASP algorithm is given in Algorithm 4.1. Initially, it computes the candidate location set based on the method from Section 4.4.1. Each candidate location is given an index number where the indices are assigned according to an increasing distance towards the left upper corner of the sensor field, thus, to some degree, keeping track of the geographical positions of the candidate locations.[1]

Next, an initialization step is performed: select N individuals randomly by choosing a random subset of the candidate location set with the cardinality of the subset being equal to the number of sinks to be placed. These subsets are then transformed into an ordered list where the ordering criterion is according to the distance of the left upper corner of the sensor field, i.e., the chosen locations are ordered according to their respective index numbers. This chromosome encoding shall ensure that building blocks can actually be formed and that for the GA crossover operators (as described below) such building blocks are destroyed with low probability. Now, all individuals of the initial population are calculated for their fitness invoking the sensor network calculus computations to provide the maximum worst-case delay for the given sink placement.

Afterwards, the evolutionary loop that is composed of several GA operations can be started. Before actually performing the crossover operation, we need to pick two parents. The details of the problem-specific crossover operator we propose are discussed below, yet the idea of the crossover is to combine good partial solutions to achieve a better total solution. After the crossover, the mutation operator is invoked for each individual (old and new ones). Again the specifics of the mutation operator are deferred to the next subsection. The idea of the mutation operator is to integrate the explorative power of random search by trying new options in the search space which were not explored so far. Yet, it must be used with care, since otherwise the GA might degenerate into a pure random (Monte Carlo) search. As last step in the evolutionary loop, the selection operator chooses the N best individuals from the whole set of old and new

[1] This is under the assumption of a rectangular field, which, however, does not pose a restriction as the sensor field can always be embedded into a rectangle.

Algorithmus 4.1 GASP algorithm.

Given: location of the n sensor nodes, transmission ranges of the nodes, number of sinks to be placed (k)
1. Calculate the candidate locations set
2. Init: N individuals
   (i) Place the number of sinks ($s_1^{(i)}$,$s_2^{(i)}$,...,$s_k^{(i)}$) randomly from candidate locations set
    oriented from the left upper corner of the network field
   (ii) Calculate fitness function of maximum worst-case delay
3. Evolutionary loop: G generations
   (i) Crossover
   (ii) Mutation
   (iii) Selection
4. Go back to 3.

individuals to form the next generation (elitist strategy). The number of individuals per generation drives the degree of parallelism in the search and is again problem-specific, but in general the larger the search space, the larger should be the population size in order to avoid a premature convergence. After the selection of the next generation the evolutionary loop is repeated for the new population until a certain number of generations is reached. We choose to control the termination via an explicit number of generations in order to keep the amount of computation invested under control, as discussed above. This also eases the comparison with other strategies. As will be discussed in the performance evaluation section, for larger problems we need to choose a large number of generations in order to achieve good solutions.

4.4.4. The GASP Operators

In this section, we discuss in more detail how the problem-specific GA operators we designed for the GASP strategy work. After the individuals of the initial population are determined and their fitness is evaluated, the crossover, mutation, and selection operators are applied to drive the search in the right direction.

Crossover. The first step for the crossover operation is to select which individuals are chosen as mating partners. Although there are many known methods with respective benefits, we used the tournament and random variants. The tournament method selects two individuals randomly from

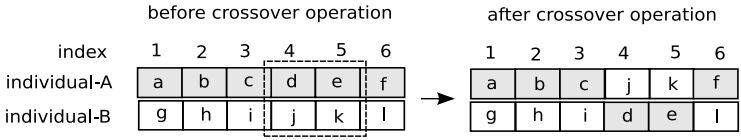


Figure 4.4.3.: Illustration of the crossover operation.

the current population. Then it compares their fitness values, i.e., in our case their maximum worst-case delays, and chooses the one with the better value. The same procedure is repeated for the other mating partner. The random method is even simpler: two mating partners are selected randomly from the current population. Note that an individual might be selected more than once for a crossover operation. We experimented with both methods and found the random method to be more effective in our problem setting, probably because otherwise the explorative character of the search is lost too quickly. Once two individuals are selected the actual crossover operation can take place in order to produce new individuals by recombination of the parents. The crossover operation is at the very heart of a GA-based search and distinguishes it from other search techniques. It is very important for the crossover operation when recombining the chromosomes of the parent individuals that building blocks are not destroyed with high probability. Under these general guidelines we designed the crossover operation as follows: As mentioned above, each individual is represented by an ordered set of indices, one index value for each sink. Now we choose a random start position in that list and choose a random number of positions such that from the starting point the chosen number of indices are exchanged between the individuals (see Figure 4.4.3 for an illustration). In doing so we make sure that this operation does not wrap around. This procedure is similar to the standard 2-point crossover with binary encoded chromosomes, but makes sure that indices are not cut through at intermediate positions during crossover. Furthermore, and more importantly, in most cases geographically close sinks remain together because they tend to be near each other on the chosen chromosome encoding. This interplay between chromosome encoding and crossover seemed to be inevitable to satisfy the building block hypothesis, because when we experimented with other crossover operators and chromosome encodings we achieved significantly less satisfying results.

Mutation. Although the crossover operation is generally considered the more important operator of a GA, the mutation also plays an important role

Algorithmus 4.2 Mutation algorithm.

1. for a given index/sink location: choose random number r from [0,1]
2. if $(r > 0.8)$ then sink is moved up a small random number
   else if$(0.6 < r < 0.8)$ then sink is moved up/down a small random number
   else sink remains at its position

in the search process, since it does not only recombine existing solutions but tries to find new interesting areas of the search space. Especially with the above crossover operator mutation is indispensable since otherwise sink locations different from the one in the initial population would never be tried. The mutation operator we implemented works as follows: For each individual and each sink location (represented on the chromosome by the respective index) Algorithm 4.2 is applied. First it is checked whether a mutation should be applied or not, this is controlled by the mutation probability. This probability should not be too high because otherwise good partial solutions are destroyed too often by the mutation. However, in our case we use a relatively high mutation probability because the crossover operation is a pure recombination as already mentioned. Again by experimentation, we found a value of 0.4 for the mutation probability to deliver the best results in our scenarios. Now, the actual mutation operation consists of moving the index and thus the sink location up or down (with equal probability) in index space. This movement is restricted to a relatively small, compared to the index space, random number (less than 1% of the index space). This, in fact, weakens the disruptive nature of the high mutation probability we apply. Again, we experimented with different versions of mutation operators and only present the one that we found to deliver good results.

4.5. Self-Organized Sink Placement (SOSP)

The position-awareness of the sensor nodes becomes a critical issue in large-scale WSNs. Sensor nodes equipped with global positioning system (GPS) hardware are for many applications too expensive. Of course, there are plenty localization algorithms [93, 50, 96] that can work quite well under some a priori knowledge of a few nodes' positions and provide an approximation of others. Nonetheless, even this might often be inconvenient to assume in large-scale WSNs due to the related communication

overhead.

In this section we present the algorithm of SOSP in detail. SOSP is intended for a network for which it is infeasible to provide the global information of nodes' locations. In SOSP, the assumptions on a priori knowledge we make are the number of sinks, the number of nodes and their transmission ranges. Sensor nodes are deployed over a circular field shape since we consider the initial sinks at the center of gravity of the sector of the circle (CGSC) under a sectorization-based grouping. Due to its simplicity and efficiency we introduced CGSC for initial sink placement. Note that a circular shape network is not a strong assumption. For any shape of network with arbitary initial sink placement is feasible to apply the SOSP algorithm. For instance, in the case of a very long rectangular network, an elipse will be a generalization from a circular field shape and the center of gravity of the sector of the elipse could be an option for initial sinks. We consider movable sinks for the initial selection of the best sink placement, and later use stationary sinks to collect data packets from the sensor nodes via multi-hop communication.

4.5.1. Initial Sink Placement: Geometric Sink Placement (GSP)

We call a sink placement at the center of gravity of the sector of the circle (CGSC) a geometric sink placement (GSP). GSP requires only the number of the sinks and the radius of the field to calculate the centers of gravity. No further infomation is needed. The center of gravity of a sector with angle α always lies on the middle radial line ($\alpha/2$) of the sector. Equation (4.5.1) calculates the ratio where to place the sink at the middle radial line of a sector and the center of gravity is simply found by multiplying with radius R. It can be calculated with Equation (4.5.2). The value of α must be within the range 0 to $\frac{\pi}{2}$ if it is given in radians.

$$CGSC = F(\alpha) \times R \tag{4.5.1}$$

$$F(\alpha) = \frac{\frac{4}{3} sin(\frac{\alpha}{2})}{\alpha} \tag{4.5.2}$$

where;

α is in radians,

$$0 \leq \alpha \leq \frac{\pi}{2}$$

$$R = radius$$

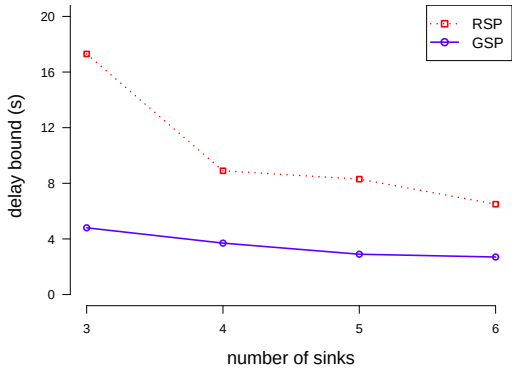


Figure 4.5.1.: Delay bound comparison of GSP vs. RSP in a 500 nodes network.

The above equations allow to compute the center of gravity of a sector and the degree of a sector can be simply obtained from Equation (4.5.3). The degree depends on the number of sinks that shall be deployed. Obviously, a single sink WSN places the sink at the center of the circle. For two sinks placement, sinks are placed at the center of gravity of the semi-circles. In fact, the center of gravity is approximately between 0 to 2/3 of the radius on the middle radial line of each sector (0 to 360 degree). The following simple formula gives a sector degree ($sDegree$) for a given number of sinks.

$$sDegree = \frac{2\pi}{\#sinks} \tag{4.5.3}$$

Though we mainly consider GSP as initial sink placement, it is a simple and efficient sink placement strategy to minimize the maximum delay for the uniformly distributed networks when there is no information about sensor nodes' locations. The delay bound comparison of GSP against random sink placement (RSP) can be seen in Figure 4.5.1. We can see that the performance gap between RSP and GSP is quite large. Furthermore, the GSP strategy is obviously very computationally efficient. It must be kept in mind that it is designed for applications where N sensor nodes are deployed and S sinks shall be placed in a network with radius R without being given any further information.

4.5.2. SOSP Algorithm Overview

After the (random) node deployment phase, the algorithm chooses the initial sink placements at CGSC by applying the GSP sink placement. The mathematical expression for the CGSC is given in Subsection 4.5.1. Next, the sinks, now located at CGSCs, perform grouping, starting from a 1-hop neighbor set in a wave to an n-hop neighbor set. The value of n varies according to the number of sensor nodes, sinks, node density, and transmission ranges. In our algorithm, we assume that sensor nodes are homogeneous and that both sensors and sinks use the same transmission range with a given node density. The simulation results of the n-hop values having different sensor nodes and sinks are discussed in more detail in the subsection on the grouping phase. Note that the maximum value of n must be considered to obtain a fully connected network. Furthermore, we estimate the locations of the 1-hop neighbors, which are required for selecting sink candidate locations. Without any supporting GPS hardware, we estimate the locations by using trilateration with the time of arrival (TOA) method [61]. In TOA, the distances to the nodes are estimated based on the time spent by the signal transmitted from the sink. The time spent in transit is converted into the distance travelled to calculate the distance between a sink and a node. Note that this method is better suited for outdoor WSNs since signal interference with obstacles causes lower accuracy. Knowing the estimated distance, we can calculate the location of a node, which is demonstrated in the subsection on the sink location selection phase. In SOSP, we need to estimate locations only for the 1-hop neighbors sets. From this location information, we define a set of fixed candidate locations for sink placement inside the search space of the 1-hop neighbors region. Finally, the best candidate is selected for sink placement. Here, we assume that a mobile sink performs the tasks of determining the 1-hop neighbor nodes' locations (by doing the TOA measurements from at least three different positions) and the computation of the best sink placement for each group. In the following subsections, we discuss the four operational phases of the SOSP algorithm in detail.

4.5.3. Deployment

During the design phase of WSNs, the designer knows only the number of sensor nodes and sinks to be deployed. Both sinks deployment and nodes deployment affect the performance of the WSN. The nodes deployment can be either random or deterministic. In a random node deployment, sensor nodes are positioned at locations which are not known with

certainty whereas for a deterministic node deployment, sensor nodes are manually placed so that their locations are known. Deterministic node deployments are usually only suitable for small-scale WSNs because of cost issues. Moreover, deterministic node placement has to be careful about placement errors which we discussed in Chapter 3 for tiling-based node placement strategies. Due to our interest in large-scale WSNs, in this work we focus on random node deployments. To illustrate where a random node deployment would be inevitable, let us assume that sensor nodes are distributed by an airplaine, for example, in an environmental monitoring application.

4.5.4. Grouping

Grouping is an important issue for large-scale WSNs for manageability by keeping operations local. Although conventional clustering is considered as a usual way of grouping, we focus on self-organized grouping, which improves in terms of computation and communication overhead. In order to build a group, it is necessary to have an inital location which should be a good starting point for sink placement. Since typical large-scale WSNs are designed with multiple sinks, it is possible to create groups whose number corresponds to the number of sinks. Based on this, we initially choose the sink placement at each CGSC. To minimize the maximum worst-case delay a sink placement at CGSC produces a fair result as presented in Figure 4.5.1. So, we first place at CGSCs mobile sinks that connect to their 1-hop neighbor nodes by broadcasting a message. (Any node that can hear a message from the sink is defined as a 1-hop neighbor, also referred to as a 1-hop distance neighbor.) Upon receiving this message, the nodes within the sink's transmission range communicate back to their sink (we use a random send time to avoid collisions for these replies). This self-organized grouping of neighboring nodes continues to the next hops up to the set of n-hop neighbors, in which n is set according to simulative results (provided in Subsection 4.5.8). A priori knowledge of the n-hop value helps to minimize the energy consumption while keeping the balance of the group sizes. An example of grouping in a 50-node network with 3 sinks is illustrated in Fig. 4.5.2.

In our algorithm, a node is not allowed to communicate with more than one sink to avoid nodes' duplication and to maintain a fully connected network, as shown in Figure 4.5.2. In the grouping phase, if a node is already in a group it is then owned by that group, although it may have a link to another group (perhaps, it may even have a lower hop distance to

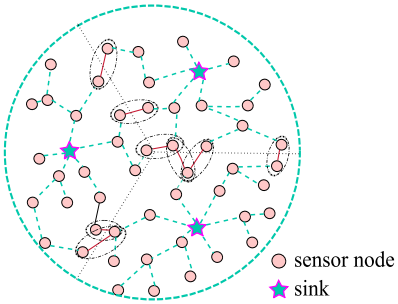


Figure 4.5.2.: Grouping for 50-node network with three sinks.

the other sink). For those nodes that act as gateways to other groups, we have the chance to choose the nearest sink (i.e., the one with shorter hop distance) during the operation phase. Such gateway nodes are marked inside an oval in Figure 4.5.2. What is more, this characteristic is very important for fault-tolerance because a node may take an alternative route to another sink in case of a node failure inside the local group.

In addition, the determination of the 1-hop neighbors' locations is necessary for the selection of the best sink placement during the grouping phase. How to determine the locations of the 1-hop neighbors from estimated distances and the maximum n-hop distances with respect to related factors, such as the number of sensor nodes, sinks, transmission range, and node density are discussed in Subsection 4.5.7 and 4.5.8.

4.5.5. Sink Location Selection

The step following the grouping phase is to choose the optimal sink placement. The goal of SOSP is to design an algorithm for the sink placement which minimizes the maximum worst-case delay. At the same time, the algorithm should have an acceptable computation and communication effort for an on-line operation. Due to these design criteria we focus on the area near CGSC, especially within the 1-hop neighbor region, for sink location selection. As we face a large, continuous search space, the computation time for an on-line operation of SOSP indicates a purely local search. Because of the continuous search space, we first discretize and determine a set of candidate locations for sink placement.

We propose two methods of a candidate location selection. The first method is based on the discretization of the originally continuous search space into a finite search space as described in Subsection 4.4.1, whereas

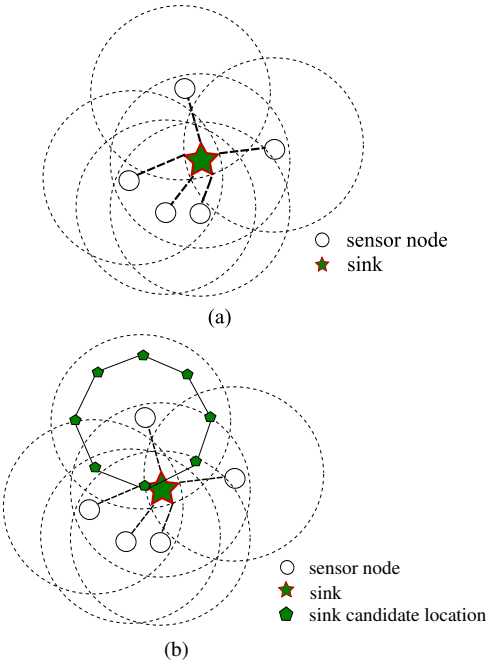


Figure 4.5.3.: (a) Discretizing of candidate locations among 1-hop neighbors, and (b) fixed candidate locations at circumradius of a regular octagon.

the latter is based on a fixed set of candidate locations. In order to determine the candidate locations within the 1-hop neighborhood we must know the 1-hop neighbor node locations that are obtained from the grouping phase for both methods. The first method simply samples the search space and chooses a candidate from each indifference region as we discussed in Subsection 4.4.1. Using the sensor nodes' locations and their transmission ranges, the sampling takes place over all possible regions and collects a point from each of them. Figure 4.5.3(a) illlustrates all possible regions for sink placement. From the intersection of the nodes' transmission ranges we obtain a tesselation of the sensor field. If it were for a global search, the whole space would need to be discretized in that fashion assuming that the nodes' locations and transmission ranges are known to determine the candidate locations, however, for our purposes we discretize only the area of the 1-hop neighbors' transmissions intersection. The candidate locations may vary with respect to the number of neighbors

and their locations. Although this method confines to the 1-hop neighborhood, the computational effort for sampling is still pretty high and may strain the on-line operation of SOSP.

As opposed to the sampling method, we introduce a fixed candidate location generation, which has considerably lower communication overhead and computational effort. Again, we focus on the 1-hop neighbors' positions and their transmission ranges in this method. Instead of sampling the search space, we place the candidate locations at fixed points inside the transmission range of each neighbor node. The fixed candidate locations are placed at the corners of a regular octagon as illustrated in Figure 4.5.3(b). Those points can easily be calculated from the sensor nodes' locations. We use three-quarters of the node's transmission range as a distance between the node and the candidate location (circumradius of octagon). For example, we place a candidate location at 12m distance (which is three-quarters of our typically assumed 16m transmission range) from the node at a corner of the regular octagon. The closer this distance becomes to the transmission range, the better the chance for more 1-hop neighbors. As illustrated in Figure 4.5.3(b), some candidate locations may be duplicated with respect to covering indifference regions. This is unavoidable yet, it must not constitute a drawback since the indifference regions may be different when factoring in 2-hop neighbors. For this reason, we also consider the distance that is as far as possible from the node location. In this method, the number of the candidate locations depends only on the number of neighbor nodes. In particular, the total candidate locations will be eight times the number of 1-hop neighbors if the fixed candidate is based on a regular octagon shape.

Figure 4.5.4 shows a quantitative comparison between fixed and sampled candidate locations in the SOSP strategy. We analyze this experiment based on the assumptions in Section 4.6.1.

As shown in Figure 4.5.4, the fixed candidate location scheme performs almost as good as the exhaustive sampling scheme. With the advantages of lower communication overhead and computational expensiveness we opted for the fixed candidate locations scheme in SOSP.

When knowing the candidate location set for sink placement, we can start with the sink location selection. We assume that the mobile sink traverses from one candidate location to another and computes the maximum worst-case delays which are a function of the routing topology. After these calculations, the sink location minimizing the maximum worst-case delay is selected. All of these actions can be done in parallel for each of the mobile sinks.

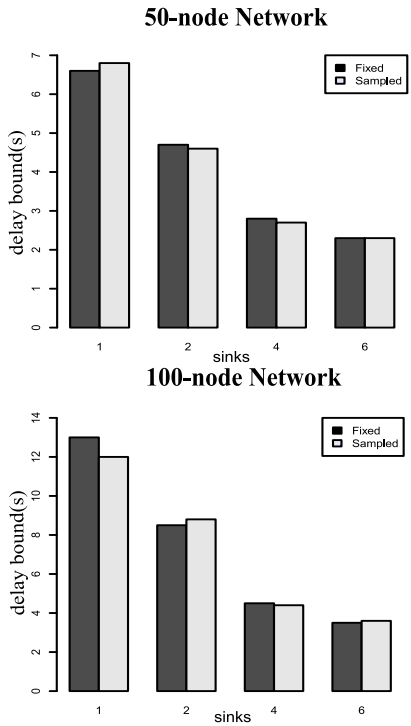


Figure 4.5.4.: Comparison between fixed and sampled candidate locations in SOSP under 50- and 100-node networks.

4.5.6. Operation

After selecting the best sink location in each group, the sinks are made stationary at those locations. In fact, it is well conceivable that sink locations are recomputed from time to time in order to deal with network changes, which would require the sinks to become mobile again, but we leave this for Chapter 5. The actual operation of the WSN can now start, for example, after broadcasting a query to all sensor nodes. A node receiving the message forwards it to the others until all nodes are reached. The sensor nodes send back the message to the nearest sink in order to fix the routing. Nearest here means the lowest hop distance because the message transfer delay is computed over multi-hop communications. Note that this could mean a different routing from the grouping phase, where groups were built independently from one another.

Algorithm 4.3 summarizes all steps of SOSP.

Algorithmus 4.3 SOSP algorithm.

Given: a circular sensor field with known radius and transmission ranges of nodes and sinks.
Definitions: initial locations of sinks, S_i, where $i = 1, 2, ..., k$, nodes N_j, where $j = 1, 2, ..., N$.
1. Deployment Phase
   (i) Deploy a random node distribution
2. Grouping Phase
   (i) Place k sinks at CGSCs
   (ii) Create 1-hop neighbors set for S_i by transmitting a signal and whichever node replies to the signal is collected
   (iii) Determine 1-hop neighbors distances from S_i and thus locations by using trilateration with TOA and 3 anchor points
   (vi) Group up to n-hop distance (n-value is obtained by simulation)
3. Sink Location Selection Phase
   For each group, (k sinks represent k groups)
   (i) Determine the fixed candidate locations according to the 1-hop neighbors' locations set from 2(iii)
   (ii) A mobile sink traverses into each candidate location and calculates the maximum worst-case delay
   (iv) Select the best sink, i.e., the one minimizing the maximum worst-case delay
4. Operation Phase
   Upon the selection of the best sink from each group,
   (i) Allow N_j to connect to the nearest (i.e., the shortest hop distance) sink
   (ii) Calculate the maximum worst-case delay

In our algorithm, the only information we need is the field size, the transmission ranges of both sensors nodes and sinks, and the number of nodes. The step following the random node deployment phase is grouping. After assigning each node to a group, we can start with the sink placement phase. The fixed assignment of candidate location sets is calculated based on the location information obtained from the grouping phase. Note that we only use the location information of 1-hop neighbors. After the candidate location sets have been determined, a movable sink traverses from one candidate location to another, computes the maximum worst-case delay, and finally picks the location having the minimum value. We assume that the mobile sink has more than enough energy supply for these operations. The nodes do not consume considerable amounts of energy due to the design of SOSP. Upon achieving the best self-organized sink placement for each group, we can start the operation and then calculate the actual global maximum worst-case delay. In order to obtain a better performance, the nodes are allowed to connect to the nearest sink (i.e.,

the minimum hop distance sink) in SOSP. This approach is helpful for the worst-case delay performance, since the message transfer delay is strongly affected by greater hop distances and aggregate flows toward the sink.

4.5.7. Determination of Distances and Locations

While grouping the 1-hop neighbor nodes, the algorithm first calculates their approximate locations from three main approaches to localization: proximity-based approach, triangulation and trilateration, and scene analysis [61]. We use the triangulation and trilateration approach that exploits geometric properties of a given scenario. Using elementary geometry, distances can be used to derive information about node locations. Trilateration and triangulation methods are based on distances between entities and angles between nodes, respectively. To estimate the locations on a plane, at least three non-collinear anchor points are required. Using the distances (between anchor points and the nodes) and anchor nodes' location, the node positions have to be at the intersection point of the three anchor nodes' transmission ranges. This basic fact is illustrated in Fig. 2. By moving a mobile sink adequately we can obtain the required anchor points for SOSP.

In order to apply the (multi-) lateration method, estimates for the distances from the node to the anchor points are required. This information can be obtained using Time of Arrival (TOA), Time Difference of Arrival (TDOA), Angle of Arrival (AOA) or Received Signal Strength Indicator (RSSI) [61]. In our algorithm, we use TOA as a distance estimation technique. This is also known as the time of flight method. The TOA exploits the conversion of the distance from the transmission time, when the propagation speed is known. Upon receiving a small packet from a sender, a receiver immediately returns the packet to the sender. Assuming the same forward and backward paths, the sender measures the round trip time and estimates the distance from this. The distances obtained by TOA are then used in trilateration to estimate the nodes' locations. We assume that all computations are done by the mobile sink, and thus do not constitute a burden for the sensor nodes.

Figure 4.5.5 illustrates the trilateration process for a node location with 3 anchor points. Knowing the locations of three anchor points (x_i, y_i), we can estimate the distance d_i, where $i = 1, 2, 3$. Using elementary geometry, we can calculate the location of the node position.

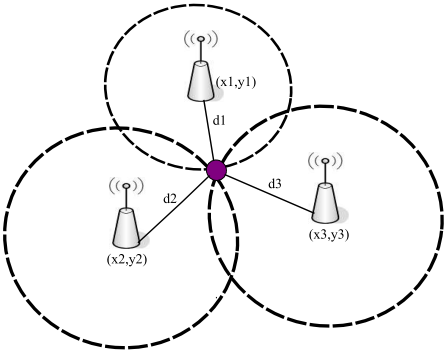


Figure 4.5.5.: Trilateration with 3 anchor points.

4.5.8. Determination of the n-Hop Values

The purpose of calculating the n-hop values is to determine the maximum number of hops necessary for grouping without endangering connectivity of the network. In fact, the determination of the n-hop values is not easily predictable due to several factors on which it depends, such as the number of nodes and sinks, transmission range, and node density. Also, the actual sink location affects the value of n. However, a priori knowledge of the n-hop value helps minimizing energy consumption while keeping balance between groups with respect to the number of nodes. Actually, it is not a fundamental necessity to be balanced, and, in fact, in some configurations it might be better to follow the nearest sink rule for routing in order to minimize the maximum worst-case delay. On the other hand, most often an unbalanced network produces traffic hot spots and diminishes the system performance as well as the lifetime of the WSNs. Therefore, we analyze the n-hop values under varying parameters and determine typical n-hop values for small-scale and large-scale WSNs. Simulative results for n-hop values are shown in the following tables together with their related factors. We analyze n-hop values with sink to node ratios of 1:50 and 1:100. The results are given as the minimum and maximum n-hop values over 10 different random node distributions. Having the same ratio of nodes and sinks, the required n-hop distances are quite stable. Here, we use the same experimental setup as in Section 4.6.1.

Table 4.1.: Table of n-hop values with sink to node ratio of 1:50.

nodes	sinks	txRange	density	(min,max) n-hop
50	1	$16m$	$0.01\frac{nodes}{m^2}$	$(4, 6)$
100	2	$16m$	$0.01\frac{nodes}{m^2}$	$(5, 6)$
200	4	$16m$	$0.01\frac{nodes}{m^2}$	$(6, 7)$
400	8	$16m$	$0.01\frac{nodes}{m^2}$	$(6, 7)$

Table 4.2.: Table of n-hop values with sink to node ratio of 1:100.

nodes	sinks	txRange	density	(min,max) n-hop
100	1	$18m$	$0.01\frac{nodes}{m^2}$	$(6, 8)$
200	2	$18m$	$0.01\frac{nodes}{m^2}$	$(7, 8)$
400	4	$18m$	$0.01\frac{nodes}{m^2}$	$(7, 9)$

4.6. Experimental Setup and Competitors

4.6.1. Experimental Setup

As a framework for the following discussion, the overall experimental setup is given in Algorithm 4.4. Sensor nodes are deployed uniform randomly in a circular network. Depending on the assumption made about the network, the sink placement strategies are selected. We classify two types of sink placement strategies according to the sensor nodes' location. For instance, Monte Carlo can be one of the competitors for GASP under known sensor nodes' positions. To proceed with such a strategy, candidate locations have to be calculated according to Subsection 4.4.1. After selecting a sink placement strategy, the iteration generates the loop body. After placing nodes and sinks, all nodes in the network are connected and checked with respect to their connectivity, e.g., whether the network is fully connected or not. The network is fully connected if any node can communicate to at least one sink. Next, sensor nodes choose their nearest sink and calculate the worst-case delay. The GSP and SOSP strategy run only a single iteration as the sinks are placed at fixed positions though SOSP has higher computation in sink placement phase. For known positions of sensor nodes, the iteration continues G generations until a solution is reached. Finally, the best location which produces the minimum worst-case delay is selected. From a computational perspective GSP and SOSP are of course at advantage over the other strategies.

Algorithmus 4.4 Experimental setup.

1. Deploy uniform random node distribution
2. Sink placement strategy
   i. Known positions of sensor nodes
      a. Calculate candidate locations
   (or)
   ii. Unknown positions of sensor nodes
3. Connect all nodes
4. Check connectivity of network
5. Choose the nearest sink
6. Calculate the maximum delay
7. If 2(ii) Done
8. If 2(i) go back to 2
9. Do G generations
10. Select the locations with minimum worst-case delay

All experiments are based on the following assumptions:

The sensor nodes are deployed with a density of $\frac{1}{100m^2}$ in uniform random node distribution fashion over a circular field shaped network. The transmission range is $16\,m$ for both sensor and sink nodes. The routing algorithm we use is based on Dijkstra's shortest path algorithm: nodes send their data to the sink with the shortest distance in hops (with ties being broken arbitrarily). Under the homogeneous nodes assumption, the token bucket arrival curves and rate-latency service curves are considered for the network calculus operations. In particular, for the service curve we use rate-latency functions that correspond to a duty cycle of $1\,\%$, $5.61\,\%$ and $11.5\,\%$[2] depending on the network size, since for larger networks a duty cycle of $1\,\%$ results in infinite delay bounds. For example, a duty cycle of $1\,\%$ results in a latency of $1.096\,s$ and a forwarding rate of 258 b/s. In each scenario, we analyze 10 different node distributions and take the average of their results to counter random effects; in fact, in none of the comparisons below, $99\,\%$ confidence intervals were even near to overlapping.

4.6.2. Competitors

Optimal Sink Placement (OSP) An exhaustive search of the space of possible combinations of candidate locations for the sinks to be placed,

[2] The values are based on a realistic node model of a Mica2 mote running the TinyOS system (see the CC1000 Radio Stack Manual) [4].

which we call Optimum Sink Placement (OSP) from now on, as it ensures to deliver the sink placement with minimal maximum worst-case delay for a given WSN. In OSP, first we determine the candidate locations as discussed in Section 4.4.1. This will be achieved by sampling all possible regions. After calculating the candidate locations, we enumerate all combinations of candidate locations depending on the number of sinks. In this way, we place the sinks at the optimal combination of candidate locations in order to minimize the maximum worst-case delay. The OSP strategy can (under certain restrictions) guarantee an optimal solution but the serious drawback is that it is very computationally expensive. We will face a combinatorial explosion since the number of candidate locations are super-exponentially increasing for a growing amount of nodes. Nevertheless at the end, it gives an exact, optimal solution. We introduce the OSP strategy not only for the applications which need an optimal sink placement for the minimization of the maximum delay but also to compare the performance of a heuristic GASP strategy. Obviously, the OSP is an upper bound for the GASP strategy.

Monte Carlo Placement (MCP) Monte Carlo-based search, from now on just called Monte Carlo Placement (MCP), where a certain number of purely random sink placements are generated and evaluated with respect to their maximum worst-case delay. Knowing the set of candidate locations by discretizing the search space with given nodes' locations and transmission range, the Monte Carlo strategy places the sinks at random candidate locations. The MCP can be considered as a lower bound on what can be expected from GASP. In comparison with GASP, we always make sure that the MCP evaluates the same number of candidate sink placements. So, the comparison between the two is fair and as the computational effort is absolutely dominated by the sensor network calculus computations for the maximum worst-case delays of a given sink placement, the GASP and MCP have to invest nearly the same computational effort.

Random Sink Placement (RSP) This is not an advisable strategy but it is introduced as a comparison for the other strategies. RSP places sinks randomly so that the results are also arbitrary, even in the same network. Hence, RSP is used as a lower bound to compare with the other strategies. RSP is pure random in a network where candidate locations of sink placement are unknown. The RSP is considered as one of the competitors for the SOSP strategy. Sometimes, RSP produces acceptable results but generally it should not be considered for real-time WSN applications.

4.7. Performance Evaluation of GASP

This section describes the performance evaluation of the GASP strategy. Using the set of candidate locations under the network with known sensor nodes' locations, we compare with an optimal sink placement (OSP) and a Monte Carlo placement (MCP).

The network size is varied from 30 to 500 nodes with a maximum of 7 sinks together with a a duty cycle of $1\,\%$ and $5.61\,\%$. With respect to GASP, we consider population sizes of 40 and 100 and the number of generations varies from 25 to 400, both of these parameters depending again on the network size. For MCP, we generate an according number of random candidate sink placements. For each candidate sink placement we use a total flow analysis [116] to compute the maximum worst-case delay for the whole sensor network. In the next two subsections, we first compare in small-scale networks how far GASP deviates from OSP and then look at its performance in large-scale networks with MCP as a lower bound benchmark.

4.7.1. Small-Scale WSNs: Comparison Between OSP and GASP

In this set of experiments, we analyze the worst-case delay for GASP and OSP for different, but relatively small network sizes of 30, 50, and 100 nodes. The number of candidate locations for the sinks were about 450, 950, and 2200 for the 30-, 50- and 100-node network, respectively. The number of sinks we had to place was restricted to 2 sinks, since for the 100-node network this already meant that the OSP strategy had to evaluate $\binom{2200}{2} = 2418900$ different combinations of sink placements where each evaluation consists of 100 total flow analyses which are not simple operations, resulting in a run-time of several days on a typical PC. For GASP, on the other hand we chose a population size of 40 individuals and the number of generations was set to 200, resulting in only 8000 different sink placements that where evaluated in a few minutes. The worst-case delay results for the best sink placements found by GASP and OSP are shown in Figure 4.7.1.

As can be observed, GASP performs very well in comparison to OSP: for the 30- and 50-node network it actually finds the global optimum, only for the 100 node case the best sink placement found by GASP lies slightly above the one found by OSP ($8.02\,s$ vs. $7.70\,s$). That should be considered a success of the GASP since with a computational effort

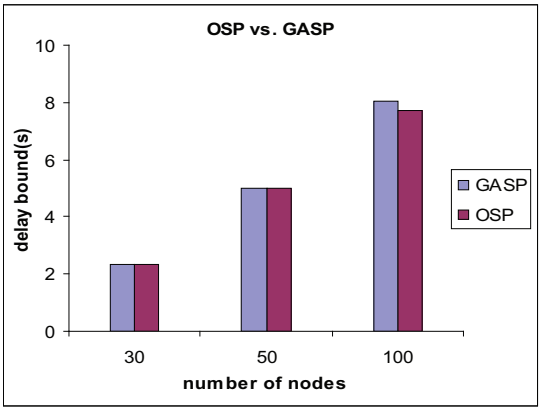


Figure 4.7.1.: The worst-case delay comparison of OSP vs. GASP.

that is several orders of magnitude lower than for OSP, it achieves almost the same quality of solutions. Whether this holds true for larger-scale scenarios is difficult to assess as the OSP is prohibitively computationally expensive. Therefore, in the next subsection, we compare GASP against MCP in larger-scale networks.

4.7.2. Large-Scale WSNs: Comparison Between MCP and GASP

The purpose of this set of experiments is to assess the performance of the GASP strategy in larger WSN scenarios. The question we address is whether GASP constitutes a more intelligent search strategy than a pure random search like MCP. In fact, there would even be the possibility that MCP could outperform GASP. This would be the case if the GA operators were poorly designed and would mislead GASP in areas of the search space that are fruitless. So, in a certain sense the following experiments also validate the design of the GASP operators.

We investigate a 500-node network with up to 7 sinks for 10 different scenarios, i.e., 10 different node distributions. For each of the scenarios, this resulted in approximately 13000 candidate sink locations, so that at maximum the search space becomes as big as $\binom{13000}{7} \approx 1.24 \times 10^{25}$. On the other hand for GASP we used a population size of 100 with 100 generations until termination resulting in 10000 sink placements being evaluated for

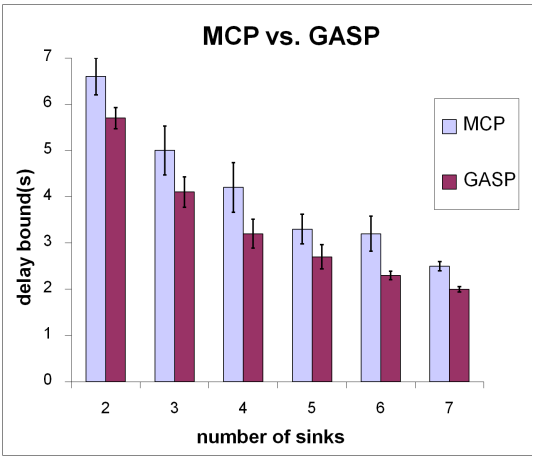


Figure 4.7.2.: The worst-case delay comparison of MCP vs. GASP.

their worst-case delay. As mentioned above, we allow the same amount of evaluations to MCP. In any case, it is clear that this amount of sink placement evaluations constitutes only a tiny fraction of the overall search space.

The results (averaged over the 10 scenarios) of these experiments are shown in Figure 4.7.2. This analysis shows that GASP performs better than MCP. For the GASP strategy, the worst-case delay improves from 5.7 to 4.1 to 3.2 to 2.7 to 2.3 to 2.0 for the 2- to 7-sink scenarios, respectively. For MCP, the worst-case delay improves from 6.6 to 5.0 to 4.2 to 3.3 to 3.2 to 2.5 for the 2- to 7-sink scenarios, respectively. Morevover, at a confidence level of $90\,\%$, the confidence intervals of both strategies do not overlap for the respective number of sinks, which shows the statististical significance of the results. The delay difference between the two strategies is roughly $1\,s$ for most cases. One may think that this is not spectacular, but it should be kept in mind that we are dealing with bounds which usually cannot be improved by factors. Furthermore, we want to emphasize once more that the computational effort for GASP and MCP is approximately the same (with GASP exhibiting a run-time penalty being in the fraction of a percent of the MCP's run-time). This shows that the GA operators do something sensible as the GA search improves on the pure random search of MCP.

4.8. Performance Evaluation of SOSP

We analyze scenarios of 100, 200 and 500 node networks with $2-6$ sinks. For the service curve we use rate-latency functions that correspond to a duty cycle of $1\,\%$ and $11.5\,\%$ depending on the network size. Besides, we use the PMOO network analysis in this experiment.

4.8.1. Performance Comparison of SOSP, GSP, and RSP under Uniform Node Distribution

At first, we evaluate the performance among SOSP, GSP, and RSP strategies. RSP is considered as a baseline for SOSP. The GSP sink placement is used as an initial sink placement in the SOSP algorithm, so the latter always has to outperform GSP. Furthermore, SOSP uses movable sinks for a self-organized network operation and should therefore, have an edge over GSP in terms of delay minimization.

In fact, as expected, SOSP outperforms GSP and RSP strategies as shown in Figure 4.8.1. In a 100 node network, the worst-case delay of RSP has $23.6\,s$, $18.7\,s$, $21.5\,s$, $11.6\,s$, and $13.2\,s$ for the 2- to 6-sink scenarios, respectively. We can see clearly that the delay bounds are very chaotic under a random sink placement. However, the worst-case delay of GSP improves from 14 to 7.7 to 5.8 to 5.1 to $4.4\,s$ for the 2- to 6-sink scenarios, respectively. For SOSP, the worst-case delay improves from 8.5 to 5.7 to 4.5 to 4.3 to $3.5\,s$ for the 2- to 6-sink scenarios, respectively. The delay differences between GSP and SOSP vary from $0.8\,s$ to $5.5\,s$. So, for example, to provide the same delay performance to SOSP with 4 sinks, 6 sinks for the GSP are required. The difference can be seen clearly in RSP vs. SOSP where SOSP is at least 2.7 times and at most 4.8 times better than RSP. In our experiments, we noticed that the more sinks, the smaller the worst-case delay gap between GSP and SOSP strategies.

For 500 nodes, the delay bound of RSP is still quite far from the SOSP strategy. The worst-case delay of RSP has $17.3\,s$, $8.9\,s$, $8.3\,s$, and $6.5\,s$ for the 3- to 6-sink scenarios, respectively. Like the 100 node case, SOSP is at least three times better than RSP. The worst-case delay gaps between the GSP and SOSP strategies are $1.9\,s$, $1.2\,s$, $0.9\,s$ and $0.9\,s$ for 3 to 6 sinks placement, respectively. The reason for having a small delay gap is the node distribution pattern and a higher duty cycle. The higher duty cycle results in a lower latency and a higher forwarding rate, thus resulting in a smaller message transfer delay. Accordingly, the worst-case delay gap becomes smaller. But note again that obtaining the same delay performance as SOSP with 3 sinks would require 6 sinks for GSP.

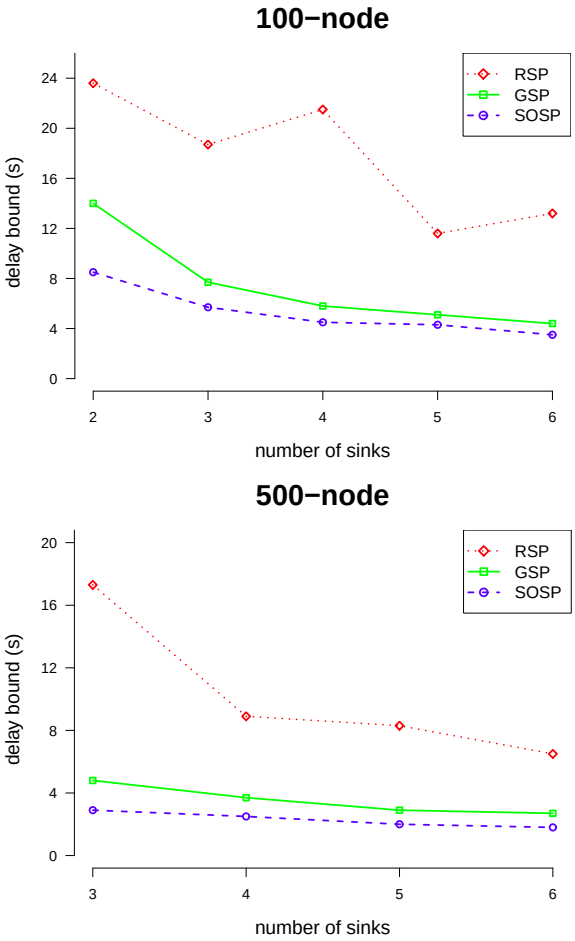


Figure 4.8.1.: The worst-case delay comparison of SOSP, GSP, and RSP.

In fact, all investigated scenarios are based on uniform random node distributions, which is a good "playground" for the GSP strategy. We, therefore, performed another experiment under non-uniform random node distribution in the next section in order to check whether SOSP can outperform GSP strategy more pronouncedly. We leave out RSP for that experiment since it is already clear that RSP under a uniform node distribution is quite far away from SOSP strategy.

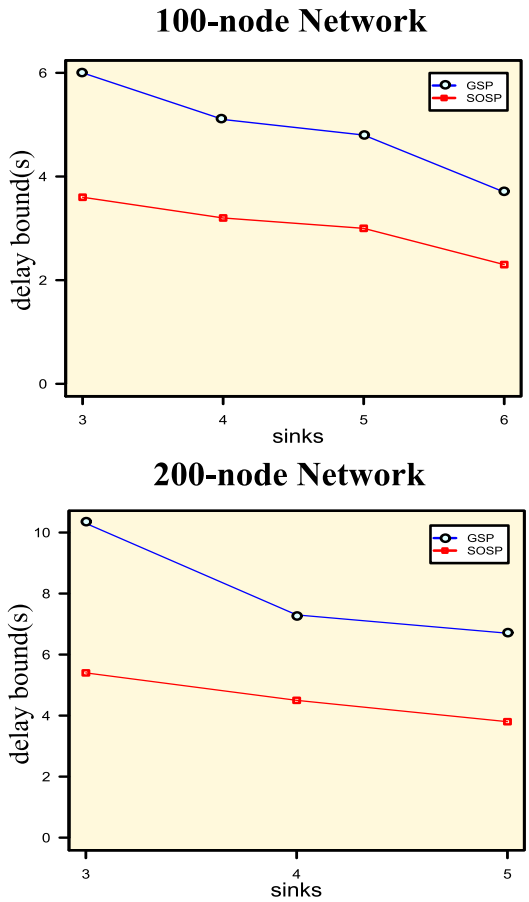


Figure 4.8.2.: SOSP vs. GSP in non-uniform network.

4.8.2. Performance Comparison of SOSP vs. GSP Under Non-Uniform Node Distribution

In the non-uniform random node distribution network, the node density varies from region to region. Nodes are densely deployed in some regions but not in all. The sink placement in GSP is fixed at CGSC and does not depend on the node distribution pattern. Thus, GSP should be expected to produce higher worst-case delays in non-uniform random node distributions. In contrast to GSP, SOSP should perform well in non-uniform

random node distributions. Whatever node distribution pattern is applied, SOSP self-organizes a good sink placement which has a low maximum worst-case delay. In non-uniform node distributions, some candidate locations from regions with lower density may provide good candidates for regions with higher density. This means that candidate locations near regions of higher density can offload the respective sink in those regions to achieve better delay performance. With this flexible movement, SOSP is very robust against different node distribution patterns.

Figure 4.8.2 shows how SOSP outperforms GSP under non-uniform random node distribution. For a fully connected network, the transmission ranges for both sensor nodes and sink have to be increased in non-uniform random node distributions. A $22\,m$ transmission range is used for 100 and 200 node networks with $1\,\%$ duty cycle. We analyze 3 to 6 sinks scenarios for both networks since $1\,\%$ duty cycle results in infinite delay bounds for the 200 node network with 2 sinks.

In the 100 node network, the delay gaps improve from $2.4\,s$, $1.9\,s$, $1.8\,s$, and $1.4\,s$ for 3 to 6 sinks scenarios, respectively. For 200 nodes, the delay gaps improve from $4.9\,s$, $2.8\,s$, $2.9\,s$, and $2.1\,s$ from 3 to 6 sinks scenarios. These results show that SOSP considerably outperforms GSP in non-uniform networks. For example, for both networks SOSP with 3 sinks outperforms GSP even if that one is given 6 sinks.

4.9. Performance Comparison of SOSP vs. GASP

Finally, we evaluate the performance of SOSP and GASP, along with the results for GSP. In particular, we analyze the scenario for a 100 node network with 4 sinks. We restricted the scenario to 100 nodes due to the considerable amount of computations for larger networks with GASP. The comparison is shown in Figure 4.9.1. For the GASP strategy, each evaluation consists of a population size of 80 individuals and the number of generations was set to 100, resulting in 8000 different sink placements. In comparison, SOSP uses about 300 different sink placements for choosing the best location.

The worst-case delay improves from $5.8s$ to $4.5s$ to $4.2s$ for GSP, SOSP, and GASP, respectively. The performance of the SOSP strategy is close to that of the GASP strategy, which can be considered a success.

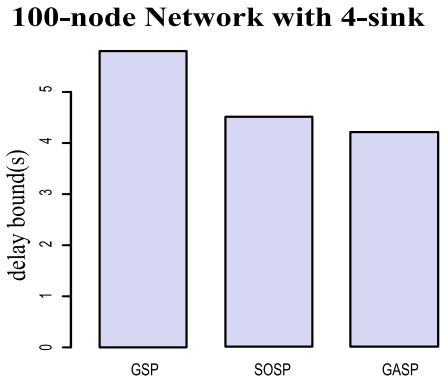


Figure 4.9.1.: The worst-case delay comparison among GSP, SOSP and GASP.

4.10. Tradeoff between Duty Cycle and Number of Sinks

The lifetime of WSNs is investigated by the tradeoff between duty cycle and the number of sinks. For ease of computation, we use the GSP strategy for the tradeoff between duty cycle and number of sinks. The same network parameters as before are used for this experiment. The analysis also determines how many number of sinks should be placed depending on a given number of sensor nodes and the desirable maximum worst-case delay.

We measure WSNs with up to 500 nodes under a varying number of sinks. To be consistent, the same network is used for all scenarios. Figure 4.10.1 shows the tradeoff between duty cycle and number of sinks for 100, 200, and 500 nodes using the GSP strategy. This experiment conducts a performance comparison between a single sink placement with higher duty cycle and multiple sinks with lower duty cycle. The best sink placement is chosen among 10 different node distributions. Each figure shows the delay distribution for a varying duty cycle and number of sinks. Among them, comparable delay distributions for different duty cycles are chosen. In the 100 nodes network, the worst-case delay distribution of $1\,\%$ with 5 sinks is comparable with $2.22\,\%$ with a single sink placement. In the 200 nodes network, $1\,\%$ with 3 sinks can compare with $2.22\,\%$ with 1 sink. Then $2.22\,\%$ with 4 sinks is similar to $5.61\,\%$ with a single sink placement in 500 nodes network. In this comparison, about 80 flows have infinite delay bound. In general, the maximum worst-case delay will be increased if the

network is extended and the duty cycle stays the same. Yet, additional sinks can minimize the maximum worst-case delay, though, of course, incurring extra costs. However, the more sinks with less duty cycle the longer the lifetime of the sensor network.

This illustrates that the lifetime of WSNs is approximately doubled with three to five sinks depending on the network size when compared with a single sink placement with higher duty cycle. On the other hand, it also illustrates the right number of sinks with respect to the number of nodes, for example 6 sinks seem to be required for 100 nodes in order to receive all data within 0 to 5 seconds.

4.11. Discussion and Conclusion

In this chapter, we introduced two sink placement strategies under different assumptions for a time-sensitive WSN such that the worst-case message transfer delay is minimized. For known locations of sensor, we developed a GA-based sink placement. First, we provided a method to discretize the search space for this optimization problem without losing any information and thus without harming the optimality of solutions. This discretization allows to apply classical search techniques. In principle, an exhaustive search can be performed to find the globally optimal sink placement. Yet, as was demonstrated, this is computationally infeasible and, therefore, we developed a GA-based sink placement strategy. The performance evaluation of GASP showed that for small-scale networks, the GASP strategy is very close to the performance of an exhaustive search. For large-scale networks, it was shown that the performance of GASP is much better than a pure random search, thus validating the design of the problem-specific GA operations.

In the second part of this chapter, we developed the SOSP algorithm for unknown sensor node locations. SOSP was inspired by initial sink placement using GSP and discretization within 1-hop neighbors of initial sinks, of which we used the respective advantages. In particular, we put emphasis on the self-organized nature of our sink placement algorithm without using global knowledge as, e.g., the sensor node locations. We consider this key for an application in large-scale WSNs. Most importantly, as we require only information from the 1-hop neighborhood of the initial sink placement (at CGSC), the algorithm, SOSP, should scale up to very large WSNs. From the experimental results, it is clear that SOSP clearly

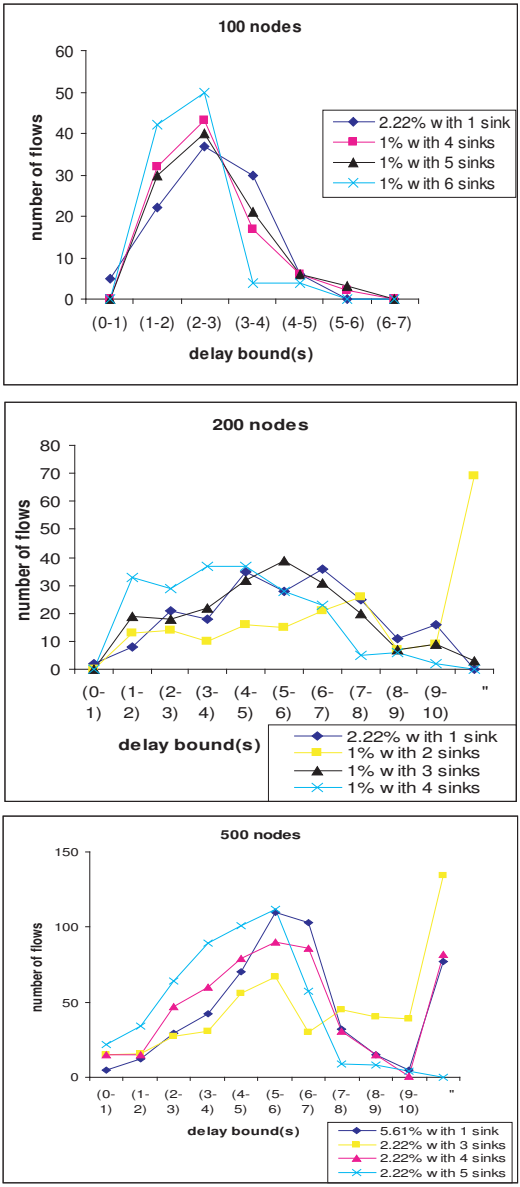


Figure 4.10.1.: Tradeoff between duty cycle and number of sinks for different number of nodes in GSP strategy

outperforms an almost totally scenario-agnostic strategy (GSP) and comes very close to a global, near-optimal strategy (GASP), which however does not scale. Therefore, SOSP strategy is shown to be a promising scalable solution to the sink placement problem under unknown locations of sensors with low computation and communication overhead. Moreover, we show how the lifetime of WSNs depends on the number of sinks with a tradeoff between the duty cycle and the number of sinks.

5. Planning Sink Trajectories in Large-Scale, Time-Sensitive WSNs

5.1. Introduction

Mobility is a mixed blessing for wireless sensor networks (WSNs). On the one hand, the degree of network dynamics induced by mobile nodes or sinks may aggravate the design of networking protocols and distributed algorithms. On the other hand, *controlled mobility* also creates opportunities [47]. By moving the sink throughout the sensor field, the burden of being a direct neighbor of the sink can be shared among all nodes of the network and the network lifetime increases. How to achieve a lifetime prolongation by using mobile sink(s) to collect the data of a WSNs has already been investigated in many works (e.g., [82, 20, 144, 125]). All of these leverage on the effect that the burden of being close to a sink is shared over time among all the nodes in the field. This alleviates the typical hot-spot problem, where nodes near the sink drain their battery much faster than others since they have to relay many data packets for other nodes. However, by using mobile sinks, in general, the information transfer delay from sensors to sinks increases over that of a proper placement of a set of stationary sinks. This is simply due to the fact that there is always a delay-optimal placement of the sinks and if the sinks move away from it the message transfer delay becomes worse. Clearly, this creates a problem for time-sensitive WSN applications. So, using sink mobility, we face a conflict between lifetime maximization and delay bound minimization in large-scale, time-sensitive WSNs. This conflict shows that these two goals have to be carefully traded off against each other when planning the trajectories of multiple mobile sinks.

In this chapter, we first provide a multi-objective optimization problem formulation for planning the trajectories of multiple mobile sinks, called OST (Optimal Sink Trajectory). We remark that already the single objective problem of maximizing network lifetime is known to be NP-hard [81].

Hence, we relax the OST problem by giving it a geometric interpretation, and introduce a heuristic framework which we call orbital sink trajectory. The orbital sink trajectory lends itself to a solution based on the kernel insight that, for a single sink, the problem is reduced to simply finding a minimum enclosing circle, whose circumcenter is the optimal position for the sink to minimize the maximum Euclidean distance. Extending this insight we propose a geometrically principled approach using a polar grid to divide the sensor field into areas of similar size, each of which is the responsibility of a single sink. In each orbit the sinks are moved synchronously (e.g., once a day), following a slow mobility approach [84].

We first formulate the optimal size for 2 orbits and the optimal number of sinks on inner and outer orbit are derived in closed form using geometric arguments. For the case of very large WSNs with many mobile sinks (say hundreds) the n-orbit model generalizes from the 2 orbits trajectory. While we base on this work with respect to giving the problem a geometric interpretation, we remark that the n-orbit case is significantly harder. Most severely, the distribution of K sinks over n orbits leads to a combinatorial explosion of the search space for the values of K that we require in very large WSNs. Similarly, the optimal choice of the number of orbits n as well as the sizing of their radii become very difficult questions. We address these questions with a heuristic framework. This is built on a geometric reduction of the problem, where the two performance characteristics, delay and lifetime, are amalgamated into minimizing the Euclidean distance between nodes and sinks. The intuition behind this is that both, delay and lifetime, benefit from nodes being closer in terms of Euclidean distance to their assigned sinks.

5.1.1. Contributions

The contributions of this chapter are:

- To the best of our knowledge, we are the first to tackle the trajectory planning problem for multiple mobile sinks in very large WSNs under lifetime *and* delay goals.
- We derive a heuristic framework that keeps up its delay and lifetime performance in very large WSNs as long as a constant node to sink ratio is retained. (→Section 5.4 and 5.6)
- For the 2-orbit model of the proposed heuristic framework, we provide a closed form of optimally distributing the sinks and sizing the orbits. (→Subsection 5.5)

- A thorough simulative investigation and comprehensive comparison with alternative approaches inspired by literature is presented. (→Section 5.7)

5.1.2. Outline

This chapter is organized as follows. Section 5.2 provides an overview of related work. Section 5.3 describes the network model, the assumptions on the sinks and nodes, and the original problem formulation for the OST in order to reveal the structure of the problem. Next, the heuristic framework and its derivations are presented in Section 5.4. We introduce an orbital model for the sink trajectories in order to achieve a small distance between nodes and their sinks, as well as a balanced division of the network area into cells based on a polar grid area assignment. A closed form of the optimal sinks distribution and sizing the orbits for 2-orbit model and a generalization to the n-orbit model are provided there in. The performance of the orbital sink trajectory for multiple sinks is evaluated and compared against several alternatives using simulations in Section 5.7. In Section 5.8, we discuss about implementation issues of orbital sink trajectories. Finally we conclude this chapter in Section 5.9.

5.2. Related Work

In literature, a considerable number of works advocate for using a single or multiple mobile sinks [20, 21, 28, 121, 82, 85, 92, 144, 29, 105, 129, 136, 44, 125, 76, 153, 42]. The majority of these deal with single sinks [20, 82, 92, 144, 103, 129, 125, 156, 76] and all of them focus on prolonging lifetimes. The effects on information transfer delay are either completely neglected or simply observed without taking actions to establish delay as an objective of equal importance to lifetime maximization. Clearly, the single mobile sink studies were not conceived with very large WSNs in mind; however, even the works on multiple mobile sinks usually considered a rather low number of sinks and nodes and did not investigate the scalability of the proposed mechanisms. In our work, we target very large WSNs and strive for high lifetime and low delay simultaneously, which sets us apart from the current state-of-the-art. Having said that, many of the previous works have inspired our work and we discuss them now separately:

Sink trajectories can be categorized into random, state-dependent, and predefined. Usage of a random trajectory can, e.g., be found in [28] where mobile sinks perform a random walk and collect the data from the sensors

of their assigned clusters trying to achieve a load balancing and lifetime maximization.

Recently, [20, 129, 92] address state-dependent sink mobility for maximizing the lifetime of WSNs. In their approaches, the sink trajectory is a function of a particular network variable, such as, e.g., the state of nodes' batteries; the sink moves either grid-based [20] or following a straight line [129]. Though the lifetime performance of such trajectories is good, the methods assume knowledge of global and dynamic information for determining the optimal paths and sojourn times, which is a strong assumption in very large WSNs. In addition to prolonging the lifetime of WSN, a mobile sink is useful when real-time traffic with certain end-to-end delay requirement is involved [8]. If traffic congestion happens in the network, most requests for establishing paths for real-time data may be denied or the deadline miss rate of real-time packets may increase significantly. In this case, repositioning of the sink is desirable in order to spread the traffic on additional hops and increase the feasibility of meeting the timeliness requirement. The authors propose to move the sink near to the most heavily loaded node. This approach, however, cannot not minimize the maximum delay.

[44, 127] propose a predefined single sink trajectory independent of any network state such that the sink appears on the same path periodically. Interestingly, [127] considers a predefined trajectory along two concentric circles separated by $2r_{tx}$, where r_{tx} is the transmission range of a node, with the aim of minimizing the total energy consumption. This is very related to our polar grid-based trajectory, yet ours is designed for multiple mobile sinks and takes lifetime as well as delay goals into account. [87] also proposes a geometrically motivated pre-defined trajectory for multiple sinks where the sinks move on the perimeters of a hexagonal tiling. This is shown to be beneficial for lifetime prolongation. Like us they require no a priori knowledge of node locations which is desirable for very large WSNs, yet they do not consider delay performance. In contrast to a periodical movement, the work in [81] proposes a predefined sink trajectory where the sink only appears once at each position along the trajectory. The author studied the improvement of lifetime prolongation by using a joint sink mobility and routing scheme similar to [103]. Most of these studies are concerned with the lifetime prolongation of a WSN, often restricting to the single sink case.

[101] considered the static sink placement, and the approach is very related with our sink location selection in our orbital model. In order to minimize the total energy consumption, in particular, for communication,

the idea is to locate the sink to a place such that the distance to sensor nodes is minimized which actually means finding the minimum enclosing circle. The sink will then be positioned at the center of the circle. In the extended work for two-tierd WSNs [102], the authors introduced special application nodes for the first-tier and the sink position is selected among the application nodes the same way as in the previous approach. In both works, the authors focus on a single sink location problem. We also apply the concept of the minimum enclosing circle problem in selecting sink locations but for multiple sinks. In addition, we intend to minimize both energy consumption and worst-case delay with this concept.

In addition, some prior works for multiple mobile sinks are [136, 21, 92, 87, 42]. In [136], the authors proposed a number of mobile sinks trajectories to optimize the lifetime and delay under a given constrained path. This way, they reduced a large (perhaps infinite) search space of sinks' locations into a smaller subset. However, some applications are not path-constrained in reality. In fact, this constrained path is a special case of the unconstrained case especially if a sink trajectory is a sequence of static sinks' locations. The authors of [21] use a linear programming approach where sinks' trajectories are controlled by the desired metric and are constrained by a set of pre-computed locations. A very similar approach of this work can be found in [92]. [42] also presented sink repositioning for energy efficiency in WSNs with multiple sinks where the best locations are computed by LP programming. In [87], a centralized approach based on mathematical programming to compute the best locations for sinks is presented. In order to apply their approach, the locations of sensors and sinks are assumed to be known a priori.

In our work, we tackle the problem of finding good trajectories for multiple mobile sinks such that we keep the maximum message delay low and still achieve a long lifetime. So, delay and energy are traded off against each other. Along similar lines, [129] optimizes this trade-off, too, designing a trajectory for a "data mule" which collects the data from each sensor node directly [121]. Yet, the data mule approach incurs long latencies and is generally not applicable in delay-sensitive WSNs. In [144, 103, 81], the movement of a sink is abstracted as a sequence of a static sink placements assuming that the time scale of sink mobility is much larger than that of data delivery; we also follow this assumption of slow mobility (as it has been coined in [84]) in our work.

5.3. Network Model and Problem Statement

In this section, we first provide our network model along with some basic assumptions and, next, state the problem of planning sink trajectories for multiple mobile sinks as a multi-objective optimization problem. Here, the intention is to shed light on its basic mathematical structure without providing a solution approach yet.

5.3.1. Network Model

Let V be the set of sensor nodes with $|V| = N$; S is the set of sinks with $|S| = K$. For both, N and K we typically assume large scales with N being on the order of thousands and K up to the order of hundreds. The reachability between nodes is modelled as a directed graph, $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = V \cup S$. For all $a, b \in \mathcal{V}$, the edge $(a, b) \in \mathcal{E}$ exists iff a and b are within a disc-based transmission range r_{tx}. The sensor field is assumed to be a circular area with radius R.

5.3.2. The Nodes

The nodes are i.i.d. uniformly distributed over the sensor field. We assume the node density (governed by the parameters R and N) to be high enough to ensure connectivity with high probability (see also Section 5.7). The nodes are homogeneous with respect to their initial energy E and their transmission range r_{tx}. Also, the costs for sending and receiving messages do not differ from node to node. The amount of data produced by each of the nodes is the same and follows the same traffic pattern, e.g., a periodic data generation. We assume that sensors send $L(n)$ data packets in each epoch n. The energy consumption for operations other than receiving or transmitting can be neglected, since for homogenous nodes they consume the same amount of energy. The nodes are stationary and use multi-hop-communication to send their data to their assigned sink. This means the routing topology is actually a forest of sink trees embedded in the reachability graph G. The assignment of nodes to sinks is further discussed in Section 5.4.

5.3.3. The Sink

The sinks are assumed to have no energy constraints. We assume that the sinks' movement is synchronous, i.e., all sinks move at the same time. Further, sink movement takes places on relatively long time-scales (e.g., once a day), much larger than the time-scale of the message transfer delay from sensors to sinks (e.g., on the order of seconds). Therefore, we neglect the time periods when the sinks are actually moving (or being moved) and the sink mobility is abstracted as a sequence of sinks' locations. At each location the sinks stay for an equal amount of time, further on called epoch $n = 0, 1, 2, \ldots$ In particular, we also assume that all data is flushed from the WSN before a sink movement takes place, i.e., there is no data dependency between epochs. We further define the following:

- We define the locations of sink s in epoch n as $l_s(n) \in \mathbb{R}^2$, and by $l(n) \in \mathbb{R}^{2\times K}$ we denote the sinks' placement in epoch n.
- For node to sink assignment, we define $x_{v,s}(n)$ as a binary variable which is set to 1 if node v is allocated to sink s in epoch n and 0 otherwise. Hence, the overall assignment $X(n)$ in epoch n is a binary matrix:

$$X(n) := (x_{v,s}(n))_{v\in V, s\in S} \in \{0,1\}^{N\times K}.$$

- For a certain assignment $X(n)$ we can define a routing as follows:

$$P_{X(n)} := \bigcup_{v\in V, s\in S\,:\, x_{v,s}(n)=1} P_{v,s}$$

  where, $P_{v,s}$ is a path from node v to sink s which is described as the set of edges lying on this path under the assumption of multi-hop communication.

- We call a sequence of triples

$$\big(l(n), X(n), P_{X(n)}\big)_{\,n\in\mathbb{N}} =: \mathcal{S}_n$$

  a strategy.

5.3.4. Optimal Sink Placement: Problem Statement

In this setting, we want to simultaneously achieve a low information transfer delay and a long lifetime. Here, we define the network lifetime as the

time until the first node of the network "dies", i.e., its battery is depleted. For the delay, we consider the worst-case message transfer delay for the whole network.

Based on the above definitions, we formulate the optimization problem of finding sink trajectories for multiple sinks in a WSN with the aim of minimizing the maximum delay and maximizing the network lifetime T of the network:

$$\min_{\mathcal{S}_n} \max_{v \in V,\, n \in \mathbb{N}} D_v(n)$$
$$\text{and} \max_{\mathcal{S}_n} T$$

subject to:

$$\sum_{e \in \delta^-(v)} f_n(e) - \sum_{e \in \delta^+(v)} f_n(e) = L(n) \quad \forall n \in \mathbb{N}, \forall v \in V \tag{5.3.1}$$

$$\sum_{e \in \delta^+(s)} f_n(e) = L(n) \sum_{v \in V} x_{v,s}(n) \quad \forall n \in \mathbb{N}, \forall s \in S \tag{5.3.2}$$

$$\sum_{s \in S} x_{v,s}(n) = 1 \quad \forall n \in \mathbb{N}, \forall v \in V \tag{5.3.3}$$

$$\sum_{n=0}^{T} \left(\sum_{e \in \delta^-(v)} E_{tx}(e, f_n(e)) + \sum_{e \in \delta^+(v)} E_{rcv}(e, f_n(e)) \right) \leq E \quad \forall v \in V \tag{5.3.4}$$

where $\delta^-(v) = \{e \in \mathcal{E} | e = (v, w),\, w \in \mathcal{V}\}$ and $\delta^+(v) = \{e \in \mathcal{E} | e = (w, v),\, w \in \mathcal{V}\}$. The function $f_n : \mathcal{E} \to \mathbb{R}^+$ describes the amount of data sent over an edge in epoch n. Equations 5.3.1 and 5.3.2 are flow balance equations to ensure that no additional data is produced or any data is lost at the nodes. Equation 5.3.3 enforces that a sensor node is assigned to exactly one sink in epoch n. The energy constraint for each node $v \in V$ is defined in Equation 5.3.4; here, the total energy consumption for reception $E_{rcv}(e, f_n(e))$ and transmission $E_{tx}(e, f_n(e))$ up to epoch T, the lifetime of the WSN, must not exceed the initial energy E for any nodes.

The delay function $D_v(n)$ represents the end-to-end delay characteristics for the message transfer from node v to its assigned sink in epoch n. At this point, we still remain abstract about whether, e.g., an average delay over an epoch or the maximum delay experienced is taken. However, later on (in the simulations as presented in Subsection 5.7), based on

sensor network calculus [115], we use a bound on the maximum end-to-end delay to instantiate $D_v(n)$. In any case, the delay function $D_v(n)$ is a very complex function. Similarly, we also remain abstract about the energy functions E_{rcv} and E_{tx}, which are also complex functions, thus aggravating the problem further. To solve this dual-objective problem one basically has to answer three questions in each epoch:

1. **Sink Trajectories**: where should the sinks be positioned?
2. **Sink Selection**: which sink does a node choose to send its data to?
3. **Sink-Tree Routing**: which route is the data sent to the sink?

In this chapter, we focus on the planning of the sink trajectories and "hard-code" the other two questions: for sink selection, each node chooses its nearest sink with respect to Euclidean distance (within the same orbit, more details are given in Section 5.4); for the routing we assume shortest path routing in the reachability graph G, mainly because it is a frequent case. Yet, even under these restrictions, the problem is still a very hard one (even strong reductions of it are NP-hard as mentioned above). The end-to-end delay, as well as the energy consumption, is dependent on the path between nodes and their sink, as well as any other path interfering with this one. Hence there is a dependency structure between the end-to-end delays which is very hard to untangle. In particular, differences in choosing a path for just one node-sink pair, in general, affect multiple end-to-end delays and different level of energy consumption. A last but not least hardness of the problem stems from the two objective functions and their conflicting nature.

5.4. Heuristic Framework

In this section, we present our heuristic framework for planning the sink trajectories in very large WSNs with delay and lifetime goals. Due to its fundamental hardness, we relax the optimal sink trajectory problem, which is basically a graph problem, into a geometric problem. This abstraction is justifiable by the large scale of the WSN as we target it in our work. Basing on the assumption of a large-scale WSN with a more or less uniform node distribution we abstract from nodes as such. For the geometric shape of the sensor field we assume it to be a circle, a somewhat arguable, but often made assumption on this level of abstraction [81]. We briefly come back to a discussion about the circular shape in Section 5.8.

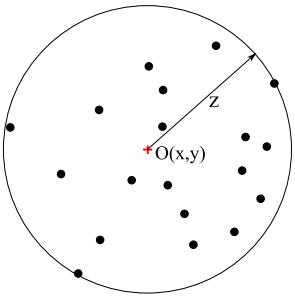


Figure 5.4.1.: An example of a minimum enclosing circle.

Under these abstractions for a geometric problem, the objective of minimizing the maximum delay is reduced to the objective of minimizing the maximum Euclidean distance $d_{v,s}(n) = \|l_s(n) - pos(v)\|_2$ from sink $s \in S$ to node $v \in V$ in epoch n; here, $pos(v)$ refers to the position of sensor node v in the Euclidean space. The rationale behind it is that the delay (mainly governed by the number of hops) needed to reach a sink is proportional to the Euclidean distance from the nodes to their sinks. Somewhat more indirectly, we cater for the lifetime maximization by partitioning the sensor field into areas of similar size (per epoch), each of which is under the responsibility of a single sink. In each epoch, each node is assigned to the sink of its currently corresponding cell. The intuition behind this is that each sink is roughly assigned a similar number of sensors per epoch, thus we can abstract the load assigned to a single sink as the area of its cell. This geometric interpretation of load and delay is instrumental in constructing good sink trajectories, because instead of complex delay and energy functions we can now formulate the problem in terms of the size of cells and Euclidean distances between nodes and sinks, which are considerably simpler measures.

Interestingly, for the single sink case, we remark that by simply substituting the delay function by the Euclidean distance, and neglecting the energy issues, the OST problem becomes a well-known minimum enclosing circle problem [123] (we point out, though, that with K circles the problem remains hard). This problem and its solution by a minimum enclosing circle is illustrated in Figure 5.4.1. The center of such a circle is the optimal placement for a sink in terms of minimizing the maximum distance between sink and sensor nodes. We recur to this basic insight several times further on, when we look for optimal positions of sinks in their respective area.

Our framework to construct sink trajectories $l(n)$ based on solutions to the geometric problem consists of the following steps:

1. We assign areas of similar sizes to the sinks ($\rightarrow$lifetime maximization). In fact, there are different possibilities to achieve this and we discuss them in the following subsection.
2. After that we calculate the optimal placement of the sinks, such that the maximal distance of any point in these areas to its sink is minimized ($\rightarrow$delay minimization).
3. Finally we define the sink trajectory for each sink by specifying its movement to the next position.

5.4.1. The Area Assignment Problem

The area assignment problem is: How to partition a circular network of radius R in order to achieve areas of similar size with respect to a given number of sinks K? A first and exact solution is an equal sectorization which has a nice scalability property in terms of handling an increasing number of sinks K without compromising the equal size of each sector. No matter how large K is, equal sectorization achieves equally sized areas by calculating the center angle of each sector as $\Phi = \frac{2\pi}{K}$. Figure 5.4.2(a) shows an example of equal sectorization for a 14 sinks network. Due to its symmetrical nature, it is sufficient to find a minimum enclosing circle for one of the circular sectors. Although, the equal sectorization achieves beneficial properties like scalability, congruity, and simplicity, the area of each circular sector becomes increasingly narrower for a growing number of sinks K, which results in relatively large maximum distances to a sink. In fact, the maximum distance for a point to its sink in a circular sector is bounded from below by $\frac{R}{2}$. This implies that the delay performance does not improve significantly any more after a certain number of sinks is reached even if more sinks are available.

Therefore, we introduce an alternative way of partitioning the sensor field, which is designed to improve on minimizing the maximum distance for a growing number of sinks K. The resulting partition is usually called a polar grid. The idea is to have multiple concentric circles of radii R_i where $i = 1, 2, ..., n$. Figure 5.4.2(b) illustrates a 2-orbit polar grid-based area assignment for 14-sinks network. In this case, by dividing the circle into two different parts, the maximum distance between any point to its sink can be reduced effectively and the resulting scheme still can achieve a balanced area assignment.

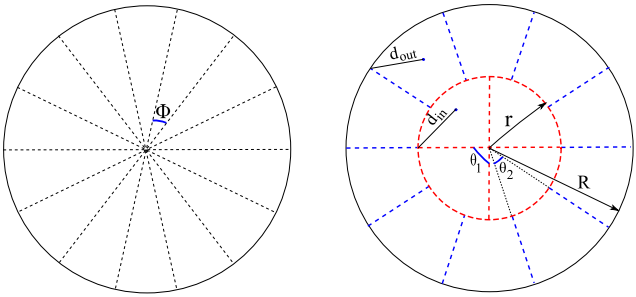


Figure 5.4.2.: Sinks assignment in (a) an equal sectorization, and (b) a polar grid.

The following sections describe the detail of the polar grid-based orbital sink trajectory together with the optimal sink distribution, i.e., how many of the sinks to place in each orbit, and the optimal sizing of the orbits, i.e., the optimal value for the radius of each concentric circle.

5.4.2. Orbital Sink Trajectory

Our heuristic framework is based on an orbital model for the sink trajectories in order to achieve a small distance between nodes and their sinks, as well as a balanced division of the network area into cells. In a nutshell, it works like this: we conceive several circles around the center of the sensor field, with different radii, called orbits; the sinks are placed on these orbits with regular interspaces and revolve around the center, like satellites move around the earth (see Figure 5.4.3). For a more detailed presentation of this n-orbit model, we introduce some definitions and notations (see also Table 5.1):

- We call the innermost orbit the first orbit and the outermost orbit the last or n-th orbit.
- By a sink distribution we refer to how many sinks are placed in each of the orbits; we denote the number of sinks placed in the i-th orbit by K_i.
- The orbits and their sinks divide the network area in a polar grid as illustrated in Figure 5.4.3. The cells within the same orbit have the same shape and size. The sinks are located in the center of their

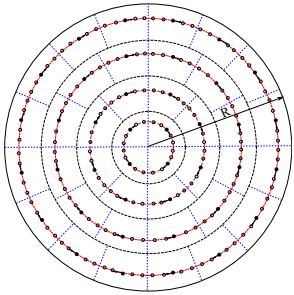


Figure 5.4.3.: The n-orbit model.

cells, such that they minimize the maximal Euclidean distance of any point of this cell to themselves. This center can be calculated by replacing the cell by a trapezoid (or triangle, in the case of the first orbit) sharing the same corner points as the cell and determining the center of the minimal enclosing circle of this trapezoid. A formal proof for the correctness of this intuitive statement can be found in 5.5.

- The polar grid consists of n concentric circles segmenting the sensor field into circular segments as well as ring segments. By R_i ($i \in 1, \ldots, n$) we denote the radii of the concentric circles, where R_1 describes the radius of the circular segments. The ring segments in the i-th orbit have an outer radius of R_i and an inner radius of R_{i-1}. The choice of R_i affects both, the number of nodes in a cell and the maximal distance from any point in the cell to the sink.

- By d_i, we denote the maximal distance of a point within a cell of the i-th orbit to its corresponding sink. Further, by a_i, we denote the area of a cell in the ith orbit.

- To preserve the polar grid structure, after each epoch, the trajectories are constructed by rotating all the sinks by the same angle θ around the center (thus a θ of 0° or 360° would result in no movement at all). More complicated trajectories are conceivable: the angles by which a single sink moves may be different from other sinks, even if they are in the same orbit and could change from epoch to epoch. Such trajectories would, however, not preserve the polar grid structure and be difficult to analyze. Since we are considering very large networks, there might be a practical upper bound on the angle θ the sink can

Table 5.1.: Notations for the orbit model.

R	Radius if the network area.
N	Number of nodes used.
K	Number of sinks used.
L	Leftover sinks.
n	Number of orbits used.
$K_i;\ i \in 1, \ldots, n$	Number of sinks placed in the i-th orbit.
$R_i;\ i \in 1, \ldots, n$	Radius of the i-th circle, constructing the polar grid.
$d_i;\ i \in 1, \ldots, n$	Maximal distance of one point in a cell of the i-th orbit, to the corresponding sink.
$a_i;\ i \in 1, \ldots, n$	The area of one cell in the i-th orbit.
θ	Movement angle of the sinks between epochs.

move between epochs, simply by the limited distance a real mobile sink may move between epochs.

The orbit model is flexible, since one can choose different sink distributions and number of orbits and also form the cells by varying the radii R_i. Through this flexibility we are able to adress different goals like minimizing the overall Euclidean distance from any node to its sink or keeping the cells equally sized. Also the model scales naturally for an increasing number of sinks by simply increasing the number of orbits.

5.5. Optimization of the 2-Orbit Sink Trajectory

We first present the optimal construction of the 2-orbit sink trajectory with respect to the optimal sink distribution and sizing the orbits and will generalize to the n-orbit sink trajectory in the following sections. As shown in Figure 5.4.2(b), sinks are assigned in the inner circle and in the annulus of the outer circle to create a polar grid. The figure illustrates an example of 14 sinks with $K_1 = 4$ and $K_2 = 10$. Let us define d_1 and d_2 as the minimal radii of enclosing circles for the sector and annular segments, respectively, given R_1, K_1 and K_2. Then, the polar grid-based area assignment problem can be formulated as:

$$\min_{0<K_1\leq K} \min_{0\leq R_1\leq R} \max\{d_1, d_2\}. \tag{5.5.1}$$

We calculate d_1 and d_2 from the corresponding minimum enclosing circles. In the following we assume $K_1,\ K_2 \geq 3$ to avoid degenerate cases.

5.5.1. Formulation of d_1 and d_2

There are two types of cells in the polar grid-based assignment scheme: a sector in the inner circle and an annular segment in the annulus of the outer circle. The optimal values of K_1 and K_2 are likely to be unequal in general, which implies two different center angles θ_1 and θ_2 for sector and annular segment, respectively. This is also illustrated in Figure 5.5.1(a) and (b).

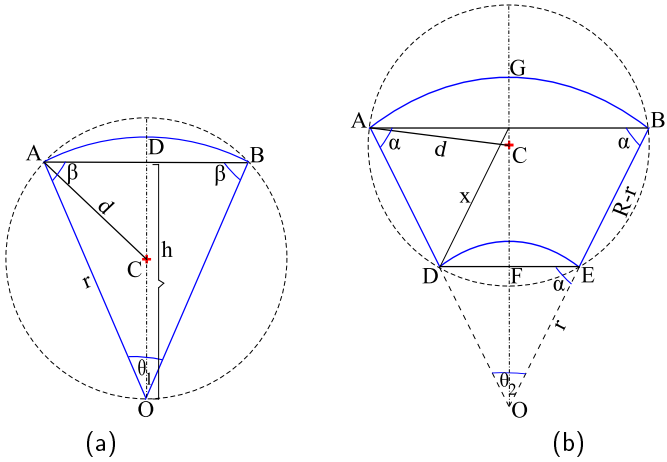
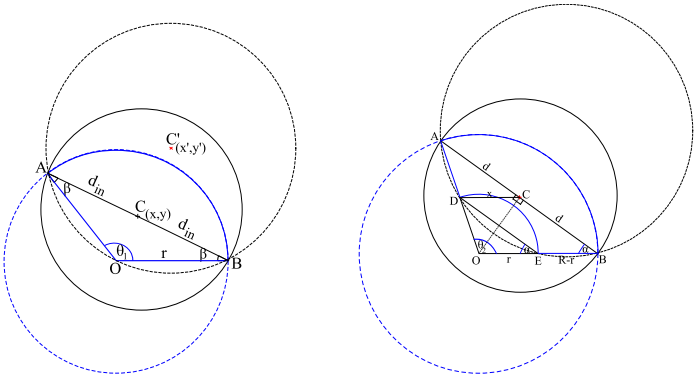


Figure 5.5.1.: Circumscribed circles of polar grid cells: (a) a sector in the inner circle, and (b) an annular segment in the annulus.

We find the minimum enclosing circle and its radius by approximating each polar grid cell by a simpler shape. In particular, we determine the minimum enclosing circles for the isosceles triangle and isosceles trapezoid for the respective polar grid cells. In Figure 5.5.1(a) and (b), the minimum enclosing circles for the isosceles triangle $\triangle ABO$ and the isosceles trapezoid $ABDE$ are depicted, which, in this case, are the circumscribing circles of the triangle and trapezoid, respectively. In the following we denote by h the height in the triangle $\triangle ABO$ and by x the distance between the point E of the trapezoid and the center of the line AB.

The minimal distances d_1 and d_2 are calculated from the respective



(a) Sector Segment (b) Annular Segment

Figure 5.5.2.: Optimal sink placement inside a sector and an annular segment with large $\frac{|AB|}{2}$.

circumscribed circles formulation. Given R_1, R, α and β (see Figure 5.5.1 and 5.5.2), the following equations characterize d_1 and d_2:

$$d_1 = \begin{cases} \frac{R_1}{2\sin\beta} & \text{for } \frac{|AB|}{2} \leq h \\ R_1\cos\beta & \text{for } \frac{|AB|}{2} \geq h \end{cases} \tag{5.5.2}$$

$$d_2 = \begin{cases} \frac{\sqrt{(R-R_1)^2+4R_1R\cos^2\alpha}}{2\sin\alpha} & \text{for } \frac{|AB|}{2} \leq x \\ R\cos\alpha & \text{for } \frac{|AB|}{2} \geq x \end{cases} \tag{5.5.3}$$

Note that for angles $0 \geq \theta_1, \theta_2 \geq \frac{\pi}{2}$ we always have to consider the first cases of Equations (5.5.2) and (5.5.3). The mathematical proofs of Equation 5.5.2 and 5.5.3 are given in Appendix A.

5.5.2. Optimal R_1 and Sink Distribution K_1 vs. K_2

Based on the mathematical formulations for d_1 and d_2, we are able to evaluate expression (5.5.1). One sees that for a fixed K_1 and K_2 d_1 is a strictly increasing function in R_1 and d_2 is a decreasing function in R_1. So we have to compute the intersection of the two functions d_1 and d_2, which gives us the optimal value for r, given a combination of K_1 and K_2. The global minimum of d_2 is equal to $R\cos\alpha$ and is achieved at all $R_1 \geq R - 2R\cos^2\alpha$. So to find the intersection of d_1 and d_2 we need to

know, where the function d_1 intersects with the function given in the first case of Equation (5.5.3). The necessary computations for the case where we use the first case of Equation (5.5.2) for d_1 are as follows:

$$\frac{R_1}{2\sin\beta} = \frac{\sqrt{(R-R_1)^2 + 4R_1R\cos^2\alpha}}{2\sin\alpha}$$

$$\Rightarrow \quad r_1, r_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where

$$\begin{aligned} a =& \sin^2\alpha - \sin^2\beta, \\ b =& 2R\sin^2\beta(1 - 2\cos^2\alpha), \\ c =& -R^2\sin^2\beta. \end{aligned}$$

By evaluating $\max\{d_1, d_2\}$ at the minimum of the points $[r_0]_+$, r_1 and r_2 we can find the minimum for this case.

For the case where we use the second case of Equation (5.5.2) we proceed similarly:

$$r_1, r_2 = \frac{-e \pm \sqrt{e^2 - 4df}}{2d}$$

$$r_0 = R - 2R\cos^2\alpha,$$

where

$$\begin{aligned} d =& 4\sin^2\alpha\cos^2\beta - 1, \\ e =& 2R - 4R\cos^2\alpha, \\ f =& -R^2. \end{aligned}$$

Again by evaluating $\max\{d_1, d_2\}$ at the minimum of the points $[r_0]_+$, r_1 and r_2 we can find the minimum of $\max\{d_1, d_2\}$.

For a given K and R, we can now exhaustively search for the optimal values of R_1 trying all possible combinations of K_1 and K_2 (the size of the search space is just $K - 5$ since we assume $K_1,\ K_2 \geq 3$). Among all combinations, we select the best configuration of K_1 and K_2 with respect to the minimum distance of d_1 and d_2 (using the best R_1), thus implementing Equation (5.5.1).

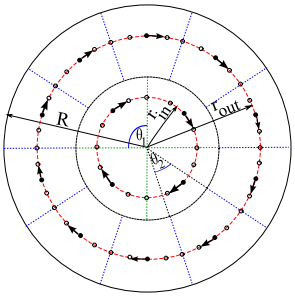


Figure 5.5.3.: An example of 2-orbit sink trajectory for a 14 sinks network.

5.5.3. Designing 2-Orbit Sink Trajectory

Now, we know the optimal points (i.e., the centers of the minimum enclosing circles for sector and annular segments) which produce the optimal d_1 and d_2. Based on these points, we design circular mobile sink trajectories. Let r_{in} and r_{out} denote the distances from the center of the network to the center of the minimum enclosing circles for the sector and annular segment, respectively, as illustrated in Figure 5.5.3.

For $0 \leq \theta_1, \theta_2 \leq \frac{\pi}{2}$,

$$r_{in} = \frac{R_1}{2 \sin \beta} \tag{5.5.4}$$

$$r_{out} = R \sin \alpha. \tag{5.5.5}$$

For $\frac{\pi}{2} \leq \theta_1, \theta_2 \leq \pi$,

$$r_{in} = R_1 \sin \beta \tag{5.5.6}$$

$$r_{out} = \sqrt{\frac{(R - R_1)^2 + 4R_1 R \cos^2 \alpha}{4 \sin^2 \alpha} - R_1^2 \cos^2 \alpha} + R_1 \sin \alpha. \tag{5.5.7}$$

The trajectories of the sinks basically result from rotating the whole polar grid in an attempt to keep both, message transfer delay and load per sink, balanced. Clearly, an interesting parameter is how far we rotate the polar grid, i.e., which step size we use for each sink when going from one epoch to the other. Results concerning this step size and a deeper discussion of its influence are provided in Section 5.7.

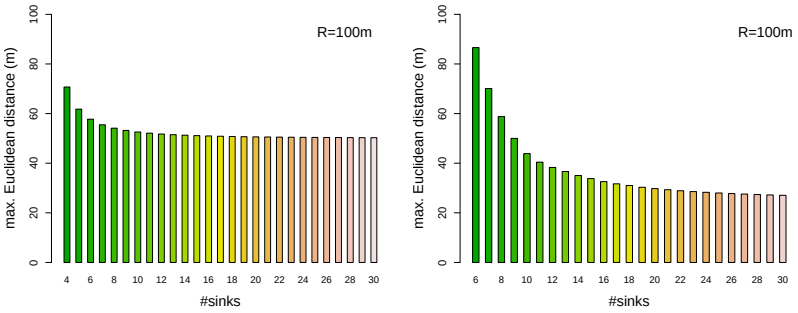


Figure 5.5.4.: The maximum Euclidean distances distribution of (a) an equal sectorization, and (b) a polar grid-based area assignment schemes.

5.5.4. Analytical Evaluation of 2-Orbit Sink Trajectory

Before we delve into a detailed simulative study of our approach, we first analytically compare the equal sectorization and 2-orbit polar grid-based area assignment schemes with each other. Figure 5.5.4(a) and (b) show the maximum distance distributions of an equal sectorization- and a polar grid-based area assignment for $R = 100\,m$ and a varying number of sinks K up to 30. Apparently, a polar grid area assignment effectively reduces the maximum distance as K grows. Note that for $K \leq 8$ the equal sectorization is in fact superior to the polar grid. The reason lies in the restriction of having $K_1,\ K_2 \geq 3$, otherwise the polar grid should always be superior, since equal sectorization can be considered a special case of a polar grid (with $K_2 = 0$ and $R_1 = R$). The results are based on the optimal choice for R_1 and the optimal sink distribution for K_1 and K_2.

We further show the corresponding optimal sink distribution K_1 and K_2 in Figure 5.5.5. Starting from $K = 13$, the value of K_1 is $\lfloor \frac{K}{3} \rfloor$ and consequently the value of K_2 becomes $\lceil \frac{2K}{3} \rceil$. Therefore, the optimal ratio of $\frac{K_1}{K_2}$ becomes $\frac{1}{2}$. In general, the optimal sink distribution is about one third of the sinks for the inner circle and about two-thirds for the annulus. Furthermore, the calculation shows that the optimal R_1 is converging to half of the radius R.

We remark that, in general, the polar grid does not achieve a perfectly equal area assignment. Nevertheless, the differences are not too large

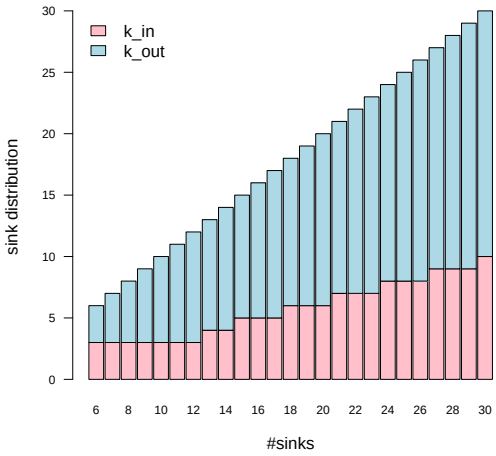


Figure 5.5.5.: The optimal sink distributions (K_1 vs. K_2) for the 2-orbit model.

and as discussed in Section 5.7 the polar grid-based orbital sink trajectory performs favorably with respect to both objectives, lifetime maximization and delay minimization.

5.6. The General n-Orbit Sink Trajectory

While the optimal sink distribution and sizing the orbits is feasible for the 2-orbit model, going to n orbits, however, will be harder to optimize by enumeration as the search space for distributing K sinks over n orbits grows as $\binom{K-n-1}{n-1}$ (allowing orbits to be empty). Similarly, the optimal choice of the number of orbits n as well as the sizing of their radii become very difficult questions. Most severely, the distribution of K sinks over n orbits leads to a combinatorial explosion of the search space for the values of K that we require in very large WSNs. Apart from applying heuristics for that search, one could strive for a closed-form expression over the maximal distances in the n-orbit polar grid to avoid this combinatorial explosion. Hence, a generalization of the 2-orbit model is provided here.

In the following we present two particular orbit models. The first has

the goal of minimizing the maximum Euclidean distance from the nodes to their sink. The second has the goal of keeping the cells equally sized, while reducing the Euclidean distances as much as possible. We speculate to see that equally sized cells are performing better in terms of energy and the strategy which concentrates on minimizing the maximal distance is doing better in terms of delay. Further for values of a certain parameter γ between 1 and 2 we hope to see how the tradeoff between energy and delay develope. Before we delve into the construction of these two orbit models we provide an overview about their construction. In a first step we found, by systematically searching the possible sink distributions and radii R_i, that these follow rather simple rules. In a second step, we search for the number of orbits, which results in the smallest maximal Euclidean distance. Up to this point, however, we handle the sinks, as if we could split them up and place them over several orbits, which is, of course, not possible. Hence in the last step, we distribute the sinks in such a way, that they get close to the formulations found in the first and second steps.

5.6.1. Minimizing the Euclidean Distance

As mentioned above, we derive two types of orbital sink trajectories. The first has the goal of keeping the maximal Euclidean distance small. This goal however is hard to achieve, due to a very complex objective function and a large number of variables. The optimization problem can be formulated as follows:

$$\min_{\{K_i: K_1+\ldots+K_n=K\}} \quad \min_{0\leq R_i\leq R} \quad \{\max_{1\leq i\leq n}\{d_i\}\}$$

with:

$$d_1 = \begin{cases} R_1\cos\beta & \text{if } K_1 = 3 \\ \frac{R_1}{2\sin(\frac{\pi}{2}-\frac{\pi}{K_1})} & \text{if } K_1 > 3 \end{cases}$$

and for $i > 1$:

$$d_i = \begin{cases} R_i\cos\alpha_i & \text{if } R_i\cos\alpha_i \geq x \\ \frac{\sqrt{(R_i-R_{i-1})^2+4R_iR_{i-1}\cos^2(\frac{\pi}{2}-\frac{\pi}{K_i})}}{2\sin(\frac{\pi}{2}-\frac{\pi}{K_i})} & \text{if } R_i\cos\alpha_i < x \end{cases}$$

where α_i, respectively β, is the angle at A in the triangle ΔABO and x is the distance between D and the midpoint on the line between A and B (see Figure 5.5.1). To solve this problem, we have thoroughly explored the respective search spaces systematically, to find the best sink

distribution and radii R_i. Due to the high-dimensional search space and a fine-grained sub-sampling of it this exploration involved a considerable amount of computational effort (several weeks of run-time on high-end PCs). The search does not only consist of the continuous parameters R_i which lie in $[0, R_{i+1}]$ (with $R_n \in [0, R]$), but also one has to consider a combinatorially growing amount of possible sink distributions (assuming orbits can be empty there are $\binom{K-n-1}{n-1}$ distributions).

Denote by K the total number of sinks and by K_i the number of sinks in the i -th orbit, further assume that n is the number of orbits. By sampling we have found that the distribution of sinks, which minimizes the maximal distance in the polar grid, follows roughly the following equations:

$$K_i = iK_1 = K_1 + (i-1)K_1; \qquad K_1 = \frac{2K}{n(n+1)}$$

5.6.2. Using Equal-Sized Areas

So far we have tried to keep the areas similar sized and then minimized the overall maximum distance, resulting in this distance being equal in each cell. In the second type of orbital sink trajectory, we want to keep the cells equally sized and then minimize the overall maximum distance. Here we have a closed form for the radii so sampling can be done much faster. Keeping the notations one sees from the samples that the distribution follows roughly this equations:

$$K_i = K_1 + 2(i-1)K_1; \qquad K_1 = \frac{K}{n^2}$$

n equally sized orbits Let K be the number of sinks which we want to place in n orbits. The radii of the orbits are given by R_i and the number of sinks in the orbits are denoted as K_i where $i = 1, ..., n$, and $R_n = R$. We will denote the sink distribution as a vector, such that $K = K_1 + \ldots + K_n$. The following theorem gives a constructive way, how to choose the radii, such that the cells have equal areas.

Theorem. *5.1: If*

$$R_i^2 = \frac{R_1^2}{K_1} \sum_{k=1}^{i} K_k \quad \forall\ 2 \leq i \leq n$$

$$R_1^2 = \frac{R^2 K_1}{K}$$

then all cells have equal area.

Proof. Let $2 \leq i \leq n$ be arbitrary, then the area of a cell in the k -th orbit is given by:

$$\begin{aligned} A_i =& \frac{\pi(R_i^2 - R_{i-1}^2)}{K_i} = \frac{\pi K_1^{-1} R_1^2 (\sum_{k=1}^{i} K_k - \sum_{k=1}^{i-1} K_k)}{K_i} \\ =& \frac{\pi R_1^2}{K_1} = A_1 \end{aligned}$$

Where A_1 is the area of a cell of the first orbit. Further one sees easily that $R_n = R$. □

To calculate the maximum distance for a given configuration of sinks (and by this the radii of the orbits), we need to find the minimum enclosing circle for a trapezoid or a triangle. Often but not always the minimum enclosing circle coincides with the circumscribing circle. The following theorem distinguishes if this is the case or not.

Theorem. *5.2: Let $ABCD$ be the trapezoid, spanned by the angle θ and the radii R_i and R_{i-1}. Then the radius of the minimum enclosing circle is given by:*

$$d = \begin{cases} \frac{|AB|}{2} & \text{if } \frac{|AB|}{2} \geq x \\ \frac{\sqrt{(R_i - R_{i-1})^2 + 4R_i R_{i-1} \cos^2(\frac{\pi}{2} - \frac{\pi}{K_i})}}{2\sin(\frac{\pi}{2} - \frac{\pi}{K_i})} & \text{if } \frac{|AB|}{2} < x \end{cases}$$

Further for the triangle ABO spanned by angle θ and the radius R_1, the minimum enclosing circle is given by:

$$d = \begin{cases} \frac{\frac{|AB|}{2}}{2} & \text{if } \frac{|AB|}{2} \geq h \\ \frac{r_1}{2\sin(\frac{\pi}{2} - \frac{\pi}{K_1})} & \text{if } \frac{|AB|}{2} < h \end{cases}$$

Here x stands for the distance between the mid point of AB and D h for the height of the triangle ABO , see also the figures 5.5.1(a) and (b).

Proof. We start for the trapezoid: Clearly if the points C and D lie inside the circle in E with radius $\frac{|AB|}{2}$, then also the whole trapezoid lies inside

this circle. If the distance between E and C is larger than $\frac{|AB|}{2}$ we have to choose the radius of the circumscribing radius to enclose the trapezoid. This radius is given by the above formula, as one can easily check. For the triangle the argument works similar: If the height of the triangle is smaller than $\frac{|AB|}{2}$, then the whole triangle lies inside the circle around D with radius $\frac{|AB|}{2}$, else we have again to choose the circumscribing radius.

To distinguish between the two cases, we need to calculate the quantities x and h . By the theorem of Pythagoras we know that

$$x^2 = \mid EF \mid^2 + \mid CF \mid^2 = \mid EF \mid^2 + R_{i-1}^2 \sin(\tfrac{\theta}{2}).$$

The length of $\mid EF \mid$ is further given by

$$\mid EF \mid = R_i \cos(\tfrac{\theta}{2}) - \mid OF \mid = (R_i - R_{i-1}) \cos(\tfrac{\theta}{2}).$$

Note that for only three sinks in an orbit we are always in the case that $\frac{|AB|}{2}$ is the radius of the minimum enclosing circle, in fact for the trapezoid this is established by the inequality:

$$R_{i-1}(-\frac{R_i}{2} + R_{i-1} < R_i(-\frac{R-i}{2} + R_i)$$

leading to:

$$\begin{aligned} x =& \sqrt{\frac{R_i^2}{4} - \frac{1}{2} R_i R_{i-1} + R_{i-1}^2} < \sqrt{\frac{R_i^2}{4} - \frac{R_i^2}{2} + R_i^2} \\ =& R_i \sqrt{\frac{3}{4}} = \frac{\mid AB \mid}{2} \end{aligned}$$

□

However, it is still unclear how to distribute the sinks optimally, such that a low maximal Euclidean distance is achieved. So we still have to deal with the combinatorial explosion of possible sink distributions. Also for this approach we decided to search systematically for the best solution.

5.6.3. Comparing the Two Strategies

The results of these computations for four orbits (for other numbers of orbits the results look similar) can be found in Figure 5.6.1. As seen in Figure 5.6.1 the difference between the two approaches lies mainly in the sink distribution. They follow the rules presented in Table 5.2, the dashed

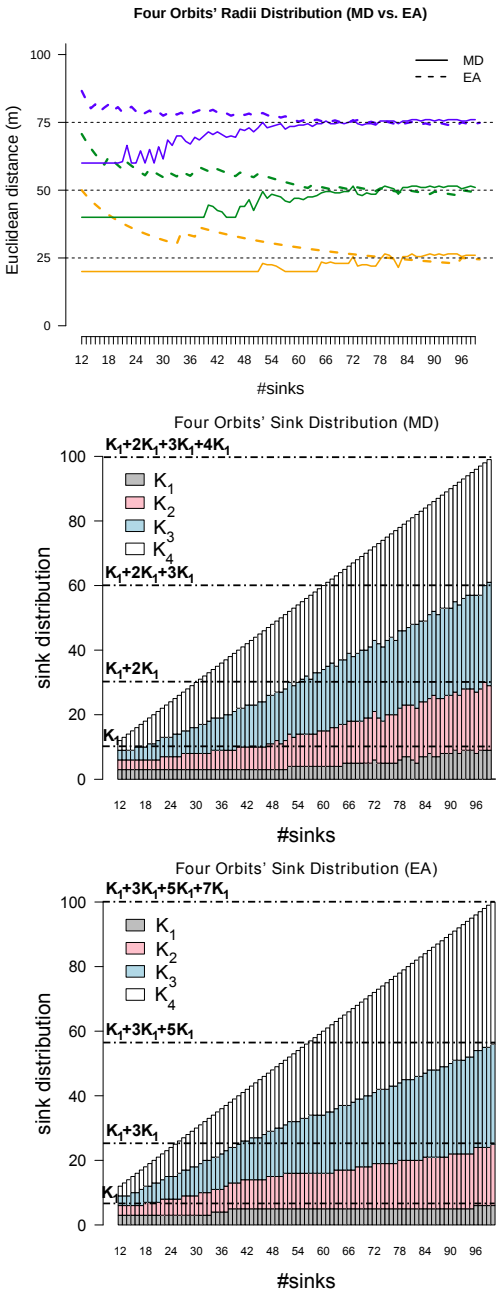


Figure 5.6.1.: Radii and sink distributions for MD and EA.

Table 5.2.: Comparison of MD and EA Methods.

	Minimum Euclidean Distance	Equal Sized Area
Sink distribution	$\frac{K_1}{K_i} = \frac{1}{i}$	$\frac{K_1}{K_i} = \frac{1}{2i-1}$
Orbits' radii	$R_i = i \cdot \frac{R}{n}$	$R_i = i \cdot \frac{R}{n}$
Relationship of K, K_1,and n	$K = K_1 \sum_{i=1}^{n} i$	$K = K_1 \sum_{i=1}^{n} 2i - 1$

lines in the figures represent how the sink distribution, for 100 sinks, would have to look like, if one applies the rule.

Both have a linear increase of sinks in the orbits, but with different rates. One could see the two strategies as special cases of a more general approach which distributes the sinks in the following way:

$$K_i = K_1 + \gamma(i-1)K_1; \qquad K_1 = \frac{2K}{n(n\gamma - \gamma + 2)} \tag{5.6.1}$$

For $\gamma = 1$ this results in the approach of minimizing the maximal distance (further called MD) and for $\gamma = 2$ we get the equally sized area approach (further called EA). (For $\gamma = 0$ one gets an equal distribution of sinks over the orbits.) There are other values for γ imaginable, resulting in hybrid approaches, however, we will not consider other values for γ in this work. For the rest of the investigation we set $R_i = i \cdot \frac{R}{n}$ to make further steps tractable.

5.6.4. Choosing the Right Number of Orbits

Clearly, choosing the right number of orbits is an important factor. In the smaller scale setting, we found significant gains when going from a single orbit to a 2-orbit trajectory (see Section 5.5). Hence, in very large-scale WSNs as we target in this work, we have to find out which number of orbits is optimal. For this purpose, we compute for different number of sinks the optimal number of orbits by checking through all numbers of orbits from 1 up to 100 for both, MD and EA. How this computation was performed for a given number of sinks is shown in Algorithm 5.1. The algorithm takes as inputs, the number of sinks and the value of γ and outputs the number of orbits, which results in the smallest maximal Euclidean distance between any point and its allocated sink. The alert reader may notice that the algorithm takes only the first and last orbit into account for calculating the maximal Euclidean distance. This is due to the following theorem:

Algorithmus 5.1 Computing the optimal number of orbits.

Inputs: The number of sinks K, the network radius R, and the parameter $\gamma = \begin{cases} 1 & \text{for } MD \\ 2 & \text{for } EA \end{cases}$.

for $n = 1$ to 100

1. Compute $K_1 = \frac{2K}{n(\gamma n - \gamma + 2)}$
2. **if** $(K_1 \geq 3)$

2.1. Compute $d_1 = \begin{cases} \frac{R}{2n\sin(\frac{\pi}{2} - \frac{\pi}{K_1})} & \text{for } K_1 > 3 \\ \frac{R}{n}\cos(\frac{\pi}{2} - \frac{\pi}{K_1}) & \text{for } K_1 = 3 \end{cases}$

2.2. Compute

$d_n = \frac{R}{n}\frac{1}{2\sin(\frac{\pi}{2} - \frac{\pi}{\gamma n - \gamma + 2})}\sqrt{1 + 4(n-1)n\cos^2(\frac{\pi}{2} - \frac{\pi}{\gamma n - \gamma + 2})}$

2.3. Compute $D^n_{max} = max\{d_1, d_n\}$

end

return orbit j such that $D^j_{max} = \min_{1 \leq i \leq n} D^i_{max}$

Theorem. *5.3: Let K and n be such that $3n(n\gamma - \gamma + 2) \leq 2K$, then d_i is increasing in i for all $i \geq 2$.*

Proof. (Sketch, a complete proof can be found in Appendix B) d_i can be given by:

$$d_i = \frac{R}{n}\Big(\frac{1}{4\sin^2(\frac{\pi}{2} - \frac{\pi}{(\gamma i - \gamma + 1)K_1})} + (i-1)i\cot^2(\frac{\pi}{2} - \frac{\pi}{(\gamma i - \gamma + 1)K_1})\Big)^{1/2}.$$

Since d_i is positive, it is sufficient to show that d_i^2 has a positive derivative, after factoring out $\cos(\frac{\pi}{2} - \frac{\pi}{(\gamma i - \gamma + 1)K_1})\sin(\frac{\pi}{2} - \frac{\pi}{(\gamma i - \gamma + 1)K_1}) \geq 0$, its numerator (the denominator is positive) can be given by

$$(2i-1)\sin\left(\pi - \frac{2\pi}{(\gamma i - \gamma + 1)K_1}\right) - \frac{\pi\gamma}{(\gamma i - \gamma + 1)^2 K_1}\left(4(i^2 - i)\sin(\frac{\pi}{2} - \frac{\pi}{(\gamma i - \gamma + 1)K_1}) + 1\right).$$

using a Taylor-Expansion around π for the first sum and using that $\sin(x) \leq 1$ for all x, we know that the above expression is larger than zero, if

$$\frac{2\pi^3}{3i^2 K_1^3} \leq 1$$

which is true for all $i \geq 2$ and $K_1 \geq 3$, which is the case by our assumptions on K and n. □

Table 5.3.: The optimal number of orbits for MD and EA.

MD(#sinks)	EA(#sinks)	#orbits
3-8	3-11	1
9-18	12-29	2
19-35	30-59	3
36-59	60-98	4
60-90	99-146	5
91-127	147-200	6
128-170	-	7
171-200	-	8

The condition of the lemma is not restrictive, as, for sink distributions according to Table 5.2, it translates to having at least three sinks per orbit. Having two or less sinks in one orbit would mean to place them in the center of the whole sensor field as this minmizes the Euclidean distance, effectively wasting a complete orbit. Hence, excluding such cases does not influence the best selection of orbits. The optimal number of orbits for different number of sinks (up to 200) and for the different approaches are displayed in Table 5.3.

5.6.5. Distribution of Leftover Sinks

In our rules for the sink distribution we treat the sinks as real numbers. Of course, they are integral and thus we simply use the following sink distribution:

$K_1 = \left\lfloor \frac{2K}{n(\gamma n - \gamma + 2)} \right\rfloor ; \qquad K_i = \lfloor K_1 + \gamma(i-1)K_1 \rfloor$

To run the experiments we choose different distributions of the sinks by the above formula for differing γ. However the fraction $\frac{2K}{n(n\gamma-\gamma+2)}$ is not an integer for every n. In this case we consider $\lfloor \frac{2K}{n(n\gamma-\gamma+2)} \rfloor$ and get an distribution which consists of $K' < K$ sinks. The remaining $K - K'$ have to be distributed in some fashion which approximates the real values of K_i . Let $L = K - \sum_{i=1}^{n} K_i \neq 0$ be the number of leftover sinks. We deal with these leftover sinks simply by distributing them over the orbits, in a greedy fashion, according to the goal of the respective approach. In the MD case, we place one sink at a time in the orbit which currently exhibits the maximal Euclidean distance, whereas in the EA case we place the sink in the orbit which contains the cells with the largest area. Algorithm 5.2 illustrates this procedure.

Algorithmus 5.2 Handling of leftover nodes for MD and EA.

Inputs: The number of sinks K, the number of orbit n, the network radius R and the parameter $\gamma = \begin{cases} 1 & \text{for } MD \\ 2 & \text{for } EA \end{cases}$.

Begin:

1. Compute $K_1 = \left\lfloor \frac{2K}{n(\gamma n - \gamma + 2)} \right\rfloor$ and
   $K_i = \lfloor K_1 + \gamma(i-1)K_1 \rfloor$ for $i = 2n$ to 100
2. Compute the leftover nodes $L = K - \sum_{i=1}^{n} K_i$
3. **while**$(L \neq 0)$ **do**
   3.1. **if** $(\gamma = 1)$
      3.1.1. Find orbit j such that $d_j = \max_{1 \leq i \leq n} d_i$
      3.1.2. K_j++ and L- -
      end
   3.2. **if** $(\gamma = 2)$
      3.2.1. Find orbit j such that $a_j = \max_{1 \leq i \leq n} a_i$
      3.2.2. K_j++ and L- -
      end

end

return K_i where $i = 1, \ldots, n$ such that $K = \sum_{i=1}^{n} K_i$

5.7. Performance Evaluation

In discrete-event simulations, we evaluate the delay and the lifetime performance of our heuristic framework by comparing it to three different mobile sink trajectories and two static sink placement strategies. We use the DISCO network calculator and MICAz-based energy model as discussed in Chapter 2.

5.7.1. Competitors

We have realized different competitors to compare our heuristic with. Unfortunately, the field of multiple mobile sink for very large WSNs is barely tapped so it was hard to find direct competitors. To create competitors we generalized ideas from other (smaller scale) proposals ([82], [53], [92]). The competitors are briefly described in the following; some of them are illustrated in Figure 5.7.1.

Random Walk Initially, sinks are placed uniformly random in the sensor field. At the start of each epoch, the sinks randomly choose a direction and step size (ensuring, however, that they do not leave the sensor field). We use this competitor as a baseline and also because it has been discussed in literature [28].

Outer Periphery [82] remarks that, in the single sink case, a trajectory along the periphery of the network optimizes the lifetime by balancing the load distribution. We generalize this concept by moving each of our sinks along the cell peripheries, where the cells are formed according to the MD approach.

Following the Energy (FE) In this strategy, the sinks are placed randomly over the network area for the first epoch. For the following epochs, the K sensor nodes with the highest residual energy left are identified and the sinks move near to them. We use this one only as a competitor for lifetime, as its delay peformance is very bad. It is a simple representative of the group of state-aware trajectories (e.g. [92]).

K-Center Heuristic [53] presents a polynomial 2-approximation for the NP-hard K-center optimization problem. The competitiveness of the algorithm is illustrated by the result of [56] which shows that if there exists an δ-approximation with $\delta < 2$ this results in $NP = P$. The authors use their algorithm on a fully connected weighted graph, nevertheless the idea can be carried over to our graph. This is a competitor only for the worst-case delay, as it performs badly with respect to lifetime due to being static. It serves as a representative for algorithms based on graph-theoretic abstractions and is expected to perform very well for delay due to its nice theoretical properties.

Static MD This takes the same sink distribution as generated by our MD heuristic, but the sinks are not moving. Instead we run the MD strategy for a whole set of possible positions and choose the one, which has minimal delay. This obviously bad lifetime competitor is included to show both, how the lifetime of the network is increased by mobility as well as its negative effect on delay.

5.7.2. Experimental Setup

Using discrete-event simulations, we evaluate the worst-case delay and the lifetime performance of our heuristic framework. In the experiments, nodes are uniformly distributed over a circular field with radius R. The respective network radii are chosen such that always a node density of $\frac{1}{100\,m^2}$ is achieved. A $20\,m$ disc-based transmission range is used under a shortest path routing for the sink trees. Token-bucket arrival curves and rate-latency service curves are considered for SNC operations. In particular,

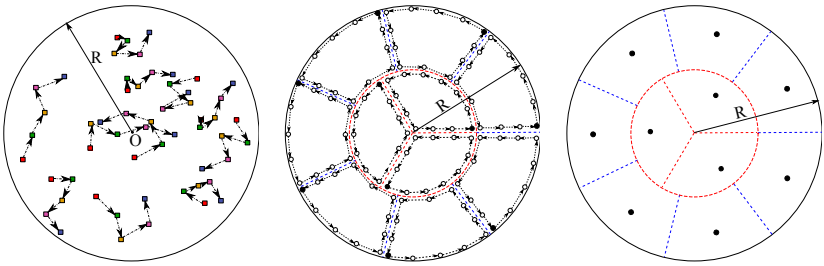


Figure 5.7.1.: Competitors: (a) a random walk, (b) an outer periphery trajectory, and (c) a static MD.

for the service curve we use a rate-latency function that corresponds to a duty cycle of $1\,\%$ and it takes $5\,ms$ time on duty with a $500\,ms$ cycle length which results in a latency of $0.495\,s$[1]. The corresponding forwarding rate becomes $2500\,bps$. Initially, the nodes are set to an initial battery level of $0.1\,joule$. Packets of $100\,bytes$ length are sent to the corresponding sinks. Apart from static sinks, all others move synchronously to their next position between epochs. The MD and EA methods use a movement angle of $\theta = 10°$. To compute the energy consumption (Equation 2.6.3), we use the following data based on [126, 7]. The current consumption is $8.5\,mA$ with $-25\,dBm$ for distances up to $12.5\,m$, and $9.9\,mA$ for distances between $12.5\,m$ and $23\,m$ with $-20\,dBm$. For receiving a data packet, a $1\,\%$ duty cycle is considered with a current of $19.7\,mA$. A constant voltage of $3\,V$ is used. A transmission data rate of $250\,Kbps$ is used, which takes $t_{tx} = 3.2\,ms$ for a $100\,byte$ packet.

We analyze the following three scenarios: 1500 nodes with 15 sinks, 5000 nodes with 50 sinks, and 10000 nodes with 100 sinks. So, we keep a constant node to sink ratio of 100 nodes/sink. For each scenario, we analyze the energy consumption per epoch, the lifetime and the worst-case delay. For all experiments, we performed 10 replications and present the average results from these. For the large majority of results, we obtained non-overlapping 95% confidence intervals, so we do not show these in most of the graphs for reasons of legibility. The static MD and the K-center heuristic are static sink placements so that we compute the lifetime based on the overall number of packets transmitted and translate it into an equivalent number of epochs (using the results from the other methods).

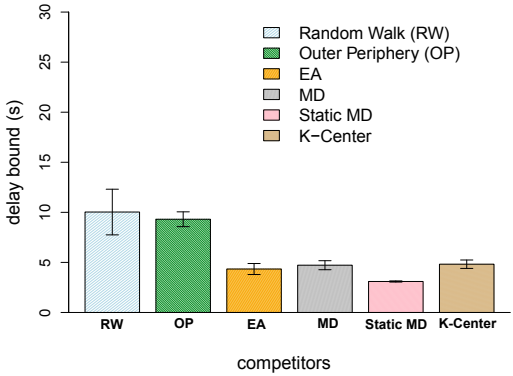
[1]The values are calculated based on the TinyOS files CC2420AckLpl.h and CC2420AckLplP.nc.

5.7.3. Worst-Case Delay Evaluation

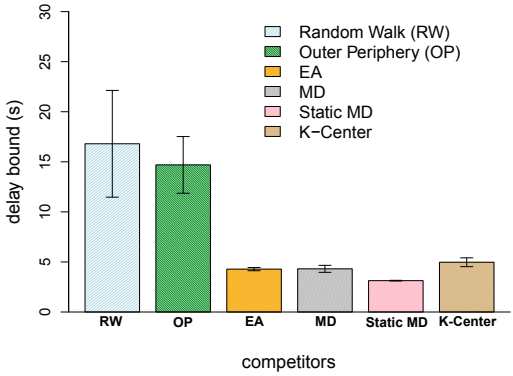
Figure 5.7.2 compares the delay performance of the four mobile sinks and two static sinks strategies. In all scenarios, the best delay performance is achieved by the static MD, closely followed by our mobile sinks strategies EA and MD. As already discussed in Section 5.1, there is a price to pay for the prolonged lifetime by mobile sinks in terms of delay, yet as we see here that price is rather low. EA and MD perform almost equally well with a slight advantage for EA. More importantly, both of them achieve roughly the same delay performance across the different scenarios and are thus scalable with respect to delay. For the outer periphery and the random walk, the assessment is very different: their delay performance is much worse and also the delay increases with growing network size, so they do not scale well with respect to delay. Somewhat surprisingly, the K-center heuristic, which requires a high computational effort and centralized information, is not doing particularly well and is actually slightly outperformed by the mobile trajectories EA and MD, which indicates again that their delay performance is very good.

5.7.4. Lifetime Evaluation

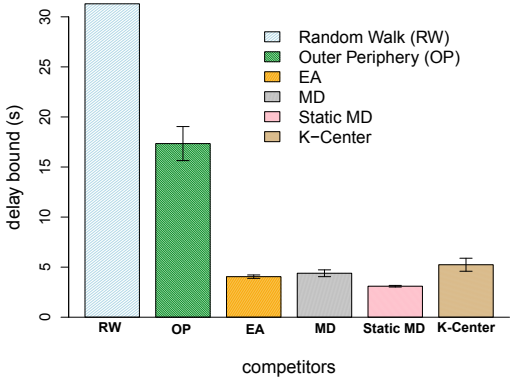
The simulation results for the lifetime performance of the competitors are shown in Figure 5.7.3. The graph shows the total energy consumption in the sensor field over the number of epochs, so the lengths of the lines indicates the lifetime performance of the respective method. Looking over all scenarios, MD turns out to be the clear winner with respect to lifetime. EA basically achieves the same lifetime in the 1500-nodes scenario, but cannot keep up with MD in the larger scenarios. All other competitors perform rather poorly: the random walk is a complete failure with a lifetime of 1.5 epochs in the largest scenario; the FE strategy also performs very bad and does not fulfil the hopes one could have in a state-aware trajectory (admittedly, it is a simple strategy and more sophisticated state-aware trajectories could be doing better); the outer periphery strategy is a little bit better, but at the expense of a high overall energy consumption. Interestingly, the static MD does not perform too badly, it outperforms FE and the random walk, which shows that trajectory planning must be done with care otherwise one could do even worse than a good static strategy. On the other hand, we can see very clearly the lifetime prolongation effect of using mobile sinks when comparing static MD with the MD sink trajectory: for example, in the 1500-nodes scenario MD achieves 10.8 epochs whereas static MD achieves only 4.6 epochs.



(a)

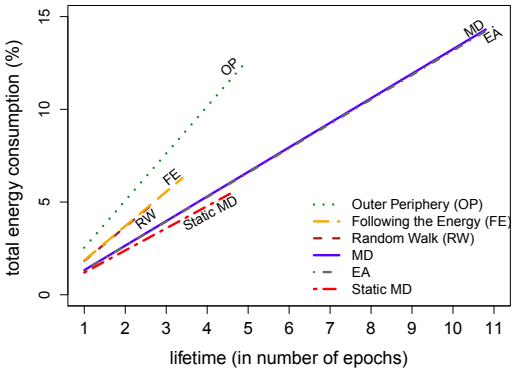


(b)

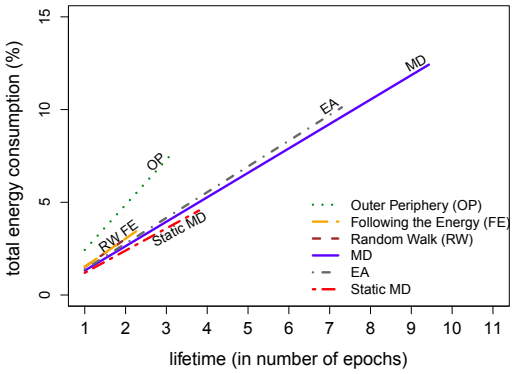


(c)

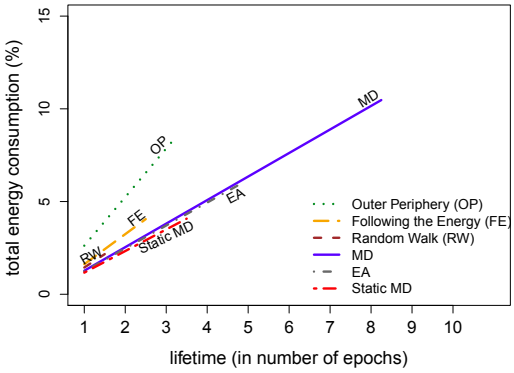
Figure 5.7.2.: Delay bound comparison: (a) 1500 nodes and 15 sinks, (b) 5000 nodes and 50 sinks, and (c) 10000 nodes and 100 sinks.



(a)



(b)



(c)

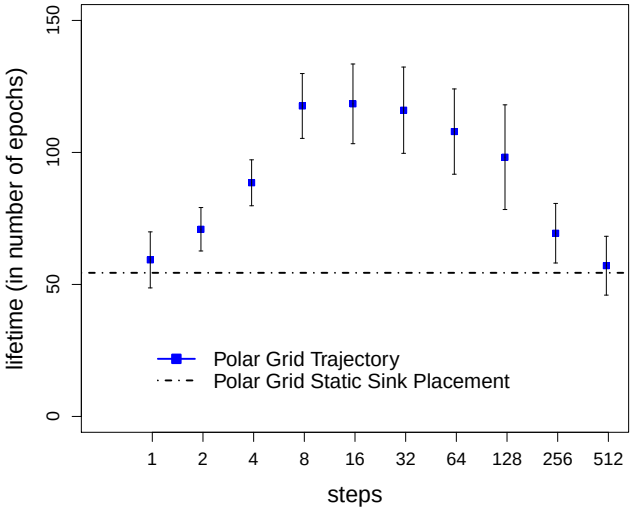
Figure 5.7.3.: Lifetime comparison: (a) 1500 nodes and 15 sinks, (b) 5000 nodes and 50 sinks, and (c) 10000 nodes and 100 sinks.

5.7.5. Varying Step Sizes

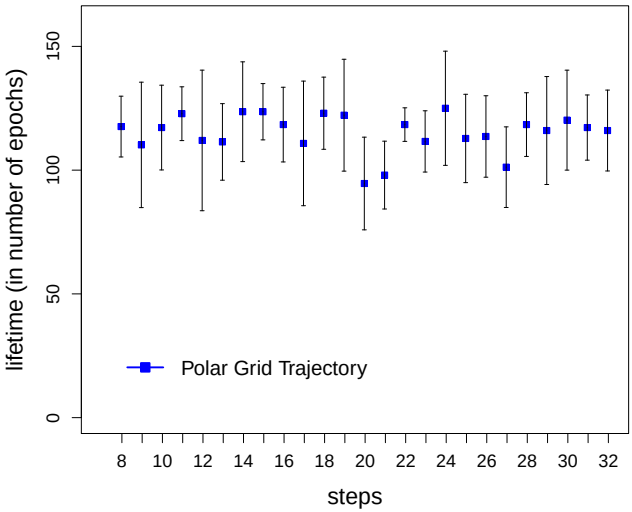
From the previous experiments, we can clearly see that the orbital sink trajectory is a promising heuristic for minimizing the worst-case delay and maximizing the lifetime of large-scale WSNs. In this experiment, we now investigate the effect of varying the step size of the orbital sink trajectory. In particular, we investigate the effect of varying step sizes under the 2-orbit model. Figure 5.7.4(a) and (b) show the lifetimes of the polar grid trajectory for different step sizes in a 200 node network with 10 sinks (here (b) provides a zoom-in for an interesting range of (a)). The interpretation of the x-axis is as follows: based on the center angle of an annular segment $\theta_2 = \frac{2\pi}{K_2}$, the different step sizes are computed as $\frac{\theta_2}{p}$, where p represents the value displayed on the x-axis; this means the x-axis runs from large step sizes to very small ones. More specifically, the optimal value of K_2 in this experiment is 7 (out of 10 sinks) and thus $\theta_2 = \frac{2\pi}{7}$ and the step size is varied by letting $p = 2^k$ for $k = 0, ..., 9$.

From this experiment, we can see that the step size has a significant effect in prolonging the lifetime. In particular, it is neither good to move too much nor too little, but there is a step size that optimizes the lifetime. For comparison, we also show the performance of a static polar grid-based sink placement, which basically provides the baseline lifetime performance. Hence, this shows another time that sink mobility pays off, but most if the trajectory is designed carefully (in fact, random walk and equal sectorization performed worse than the static polar grid). A zoom-in for the interesting range of p between 8 and 32, where the optimum step size lies for this experiment, is shown in Figure 5.7.4(b). As can be observed, the lifetime behavior is rather chaotic in this range, which hints at the difficulty of obtaining a closed form for the optimal step size under the polar grid.

Varying step size does not differ too much for the issue of minimizing the maximum delay as presented in Figure 5.7.5. The figure shows the delay bound of 500 nodes with 10 sinks network under three scenarios: (1) step size equivalent to the center angle 5 degree, (2) step size equivalent to the center angle 12 degree, and (3) step size equivalent to the center angle 16 degree. As shown in Figure 5.7.5, the delay performance is not sensitive under varying step size and the result remains the same for higher or lower step sizes.



(a)



(b)

Figure 5.7.4.: Lifetime comparisons under varying step sizes in a 200 nodes with 10 sinks network.

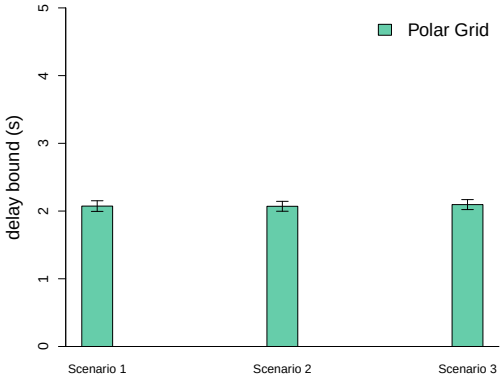


Figure 5.7.5.: Delay bounds under three scenarios of different step sizes in 500 nodes with 10 sinks network.

5.7.6. Lifetime vs. Delay and Scalability

In this subsection, we wrap up the previous results by particularly looking at the combined lifetime vs. delay performance as presented in Figure 5.7.6. The x-axis represents the delay and the y-axis shows the corresponding lifetime performance in terms of the number of epochs. The shape and color of the symbols represents the different strategies and the size of the symbols encodes the scale of the experiment, i.e., the large symbols represent the experiments with 100 sinks, while the medium-sized and small symbols represent the experiments with 50 and 15 sinks, respectively. By following the path from small to large symbols one can see, how the strategies scale for larger WSNs. Clearly, the goal must be to stay within the upper left quadrant of this graph. Only MD achieves this goal, EA has a problem with respect to lifteime scalability. All other competitors do not really offer good lifetime-delay tradeoffs and are at best good in one of them.

One may even become suspicious about MD for its scalability, because as can be observed in Figure 5.7.6, there is a certain degradation with respect to lifetime for it, too. However, the lifetime definition that we use here (when the first node dies) somewhat looses its usefulness with an increasing number of nodes, as it becomes more and more likely that some single node is in an unfortunate position where its battery is drained much quicker than for others. Therefore, we provide some more information on the "death"

process of the nodes in the field when we continue network operation after the first node died in Figure 5.7.7 (again the size of the symbols represents the scale of the scenario). In particular, when we redefine lifetime as the time until which 10% of the nodes have died then we see that MD scales very well, i.e., it achieves almost the same lifetime in all three scenarios. In comparison, EA still does not scale that well, though arguably it also benefits from this redefinition of the lifetime.

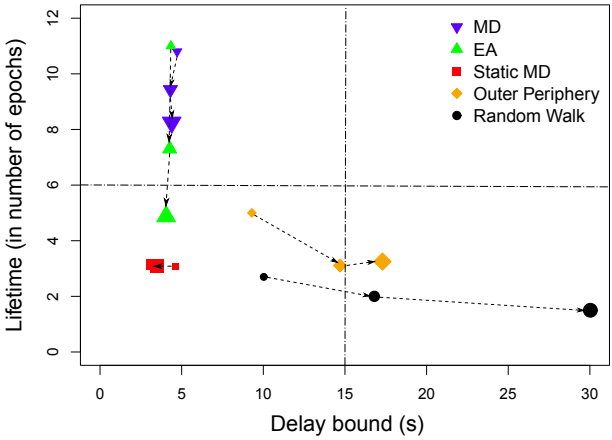


Figure 5.7.6.: Lifetime-delay tradeoff among competitors.

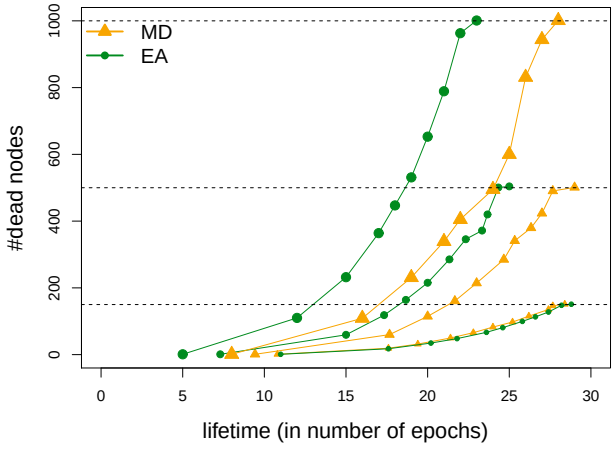


Figure 5.7.7.: MD/EA lifetime performance under a different definition.

5.8. Discussion

In this section, we want to discuss some relaxations of the assumptions in our framework and how the geometric interpretation can still be helpful, thereby pointing out directions for future work:

- what if the shape of the network field is not a circle?

While a circular shape network seems a strong assumption in our orbital sink trajectory, a generalization of a circular shape is feasible for other extreme shapes such as a very long rectangle. In this case, smaller distortions (linear transformations) of the circular shape would result in ellipsoid shapes (probably with segments of unequal size within one orbit), which can still be dealt with a similarly distorted orbit model (mapping the position of the sinks by the same transformation to the new sensor field). A change of the underlying distance norm would result in completely different shapes, we could think of a squared network area as a circle under the maximum norm $||\cdot||_\infty$ (similarly one could change to the $||\cdot||_1$-norm to handle a rhombus-shaped sensor field).

- what if the nodes are not uniformly distributed?

In some networks, there might be clusters of high density and regions with low density. Here more sinks are needed in the clusters, while the sparse areas can be handled by less sinks, this could be achieved by altering the distances the sinks move between the epochs in our orbit model. Slowing the sinks down, when reaching the clusters would result in accumulating sinks in that region and speeding them up again, when leaving the clusters, moves them fast through the sparse areas.

- what if the nodes are non-homogenerous?

The assumption of node homogeneity may be relaxed by going to a three-dimensional geometric interpretation of the original problem where the third dimension could capture, e.g., nodes with (initially) higher battery levels. Clearly, the problem will not become simpler, but based on the good experience we made with the geometric interpretation of the underlying problem, we believe that this could be a winning strategy also for such advanced problem settings.

- what if the exact orbital sink trajectory is not feasible in the sensor field?

In fact, the sinks' candidate locations along the orbits are not always feasible in real world applications due to environmental disturbance and obstacles. Under these situations, one may think the sinks' locations as one of the nodes' locations near the predefined sink's location. Nevertheless, the performance differences are not too large since the routing topology will not change dramatically. Non-concentric, but still periodic (following a closed circuit) strategies are imaginable, for example a star shaped trajectory. As a generalization of the concentric class of strategies one may hope for further improvement under the assumption of a successful optimization. In fact, we have experimented with a specific (unoptimized) star-shaped trajectory, yet it was inferior to the polar grid trajectory.

- what if the sinks' movement is not synchronized? How to make the sinks move?

We defined the orbital sink trajectory following a slow mobility approach (e.g., sinks move once a week). In general, sinks are not energy- and resource-constrained. Under these conditions, synchronization should always be feasible by utlizing a synchronization protocol.

- How to achieve the shortest path routing from the node to the designated sink in each epoch?

Unter a discrete movement of sink (i.e., once a week), the shortest path routing can be achieved easily. Since the trajectory is predefined, the sinks' locations are already determined during the design phase. Therefore, each node knows the location of the sink at a given time and can compute the respective shortest path to its nearest sink. The routing path can be computed after the deployment phase but before the operation phase of the application. In fact, a more sophisticated but efficient routing can also be applied in the orbital sink trajectory.

5.9. Conclusion

In this chapter, we have proposed a flexible heuristic framework to design the trajectories for multiple mobile sinks such that a good tradeoff between a high network lifetime and a low information transfer delay can be achieved in very large sensor networks. Due to its fundamental hardness, we resorted to a geometric interpretation of the problem for which we introduced and optimized orbital sink trajectory. The framework uses an n-orbit model which is based on a geometric rationale that, in large sensor networks, cell areas and Euclidean distances between nodes and sinks are good proxy measures for lifetime and delay. Two instances of the framework are derived: one which focuses on the minimization of the maximal Euclidean distances (MD), and one which targets to equalize the area assignment and takes distance minimization as a secondary goal (EA). Both are compared with several competitors in detailed discrete-event simulations and show very good lifetime-delay tradeoffs. Especially, the MD strategy shows a very scalable behavior for its lifetime and delay performance when the number of nodes becomes large. In particular, in contrast to all other methods it keeps up the delay and lifetime values of smaller scenarios when scaled to larger scenarios under a constant node-to sink ratio. More abstractly, we believe to have provided strong evidence that the orbital sink trajectories provide for a natural scalability to very large sensor networks, if designed carefully.

6. Summary and Outlook

Recently, large-scale and time-sensitive WSN applications are more and more demanding and, thus, scalability and delay bound minimization become important design criteria. As long as sensor nodes are dependent on the limited battery resource, energy-efficient designs and algorithms remain essential for lifetime maximization. Hence, we developed design algorithms for controlling the information transfer delay and the total network lifetime. The design algorithms are concerned with multiple static and mobile sinks. By simulation, we showed that placing the sinks and planning the trajectory carefully really pay off.

In chapter 2, we surveyed related design problems in WSNs. A realistic energy model and worst-case delay analysis methods are introduced.

In chapter 3, we studied the performance of polygon-based node placement designs for large-scale WSNs under exact and disturbed placement. Then we introduced and carefully evaluated several node placement strategies for WSNs.

In chapter 4, large-scale and time-sensitive WSNs are designed using multiple sinks. The goal is to find an optimal sink placement. Due to the hardness of the original problem, heuristic approaches are taken into account. Under different assumptions, we introduced a GA-based and a self-organized sink placement for large-scale and time-sensitive WSNs.

In chapter 5, the mobile sink approach is taken into account. We introduced the orbital sink trajectory to simultaneously achieve delay and lifetime goals. The main idea of orbital sink trajectory is that both, delay and lifetime, benefit from nodes being closer in terms of Euclidean distance to their assigned sinks.

6.1. Summary of Contributions

In summary, the following are major contributions of the dissertation:

- To the best of our knowledge, we are the first to tackle the sink placement problem and the trajectory planning problem for multiple sinks in very large WSNs under lifetime *and* delay goals.

- As static sink placements, we developed a near optimal heuristic sink placement called genetic algorithm sink placement (GASP) and a self-organized sink placement (SOSP) algorithm with lower computation and communication overhead for large-scale and time-sensitive WSNs. How to find the set of candidate locations for sink placement is also contributed by discretizing the original continuous search space into a finite search space based on the concept of regions of indifference.

- As mobile sink trajectory, we derive a heuristic framework based on an orbital model that keeps up its delay and lifetime performance in very large WSNs as long as a constant node to sink ratio is retained. For the 2-orbit model of the proposed heuristic framework, we provide a closed form of optimal distributing the sinks and sizing the orbits.

- As node placement strategies, we especially studied semi-regular tiling-based node placements and investigate their performance on coverage, energy consumption, and worst-case delay. We also contribute the concept of k-coverage map to check all possible coverage areas which will be useful to analyze the relative frequency of exactly k-covered points.

- By simulation, the proposed designs and algorithms are thoroughly investigated and compared with alternative approaches inspired by literature.

6.2. Outlook

There is a growing interest from WSNs towards cyber physical systems (CPSs) for various purposes. As CPSs are basically closed-loop systems many WSN applications will have to operate under stringent timing requirements. If this requirement is not achieved, such time-sensitive applications become unreliable and unusable. The design algorithms for controlling the information transfer delay proposed in this dissertation are basically well suited to CPSs. CPSs, however, is composed of different physical devices embedded with sensors which can be integrated into networks or stand-alone devices. It makes more challenges in design problems due to complex interactions among different physical appliances, continuous dynamic network topology, ambiguities in all aspects of cyber and physical systems, etc. Hence, a possible extension of this dissertation is controlling the information transfer delay under the heterogeneity of WSNs designs towards CPSs.

A. Proofs of Equation 5.5.2 and 5.5.3

The following proofs show the correctness of Equations (5.5.2) and (5.5.3).

Lemma. *A.1: For $\frac{|AB|}{2} \leq h$, the center of the circumscribed circle of the isosceles triangle $\triangle ABO$ minimizes the maximum distance of the sector ABO (refer Figure 5.5.1(a)).*

Proof: Denote the center of the circumscribing circle by C, its radius by d and the circle itself by C_d. Since $\frac{|AB|}{2} \leq h$ the minimum enclosing circle is C_d. Since $\triangle ABO$ is a subset of the sector, this means that the minimum enclosing circle for the sector has at least radius d. Hence it is sufficient to show that the sector lies inside the circle C_d. Again from $\frac{|AB|}{2} \leq h$ we know that C lies inside the triangle, so $d \leq R_1$. Obviously $C \neq A, B$ and by this even $d < R_1$ holds. Denote now by AB_O the arc between A and B with its center in O. It is sufficient to show that any point lying on this arc has distance not bigger than d to C. Denote by D the intersection between the arc AB_O and the line through O and C (see Figure 5.5.1(a)). Then by the triangle-inequality (for the triangle $\triangle ACO$) holds:

$$2d \geq R_1 = |OD| = |OC| + |CD| = d + |CD|$$

leading to:

$$d \geq |CD|$$

Suppose now there would exist a point D' on AB_O with $d < |CD'|$. This would only be possible if there exists a point D'' on the arc AB_O which also lies on C_d. Together with A and B this point would be a third intersection point between the circles C_d and O_r, leading to the equality of the two circles, especially to $d = R_1$, which is a contradiction to the already established inequality $d < R_1$. Hence all points of the arc, and by this the whole sector, lie inside the circle C_d .

Lemma. *A.2: For $\frac{|AB|}{2} \leq x$, the center of the circumscribed circle of the isosceles trapezoid $ABDE$ minimizes the maximum distance of the annular segment $ABDE$ (refer Figure 5.5.1(b)).*

Proof: As in the previous proof we know, by the assumption that $\frac{|AB|}{2} \leq x$, that the circumscribing circle C_d of the trapezoid is the minimum enclosing circle of the trapezoid and its center lies in the trapezoid. We proceed in the same way as in the previous proof, however here it is not as easy to see that the radius d of the minimum enclosing circle of the trapezoid is smaller than R. For that we denote by F the intersection of the angle bisector of θ_2 with the line $|DE|$ (see Figure 5.5.1(b)). The triangle $\triangle AFO$ has its largest angle at F, which is for $R_1 < R$ larger than $\frac{\pi}{2}$ hence:

$$R = |OA| > |AF|$$

A similar argument leads to $|AF| > |DF| = |EF|$, hence we can find an enclosing circle for the trapezoid around F with radius $|AF| < R$ and by this the minimum enclosing circle also has a radius d smaller than R. Now denote again by AB_O the arc between A and B with center in O and by G the intersection between this arc and the line through O and C, where C denotes the center of the minimum enclosing circle. Then again by the triangle inequality:

$$|OC| + |CA| = |OC| + d \geq R = |OC| + |CG|$$

hence:

$$d \geq |CG|$$

By the same contradiction as in the previous proof, one can show that the complete arc AB_O lies inside the minimum enclosing circle of the trapezoid.

Lemma. *A.3: For $\frac{|AB|}{2} \geq h$, the line segment AB of the triangle $\triangle ABO$ is the diameter of the minimum enclosing circle (refer Figure 5.5.2(a)).*

Proof: In the triangle $\triangle ACO$ we have at C a right angle, hence: $d < R_1$. Denote again by D the intersection of the arc AB_O and the line through O and C, then by the triangle inequality we have:

$$|OD| = |OC| + |CD| \leq |OC| + |CA|$$

Knowing that $|CD| \leq |CA| = d$ we can construct the same contradiction as in the previous proofs to see that the whole arc AB_O lies inside C_d. By this we know that C_d is an enclosing circle. Since $|AB| = 2d$ we also know that any enclosing circle has at least radius d, thus C_d is a minimum enclosing circle.

Lemma. *A.4: For $\frac{|AB|}{2} \geq x$, the line segment AB of the trapezoid $ABDE$ is the diameter of the minimum enclosing circle (refer Figure 5.5.2(b)).*

Proof: The proof works like the previous one replacing R_1 by R and G taking the role of D.

B. Proof of Theorem 5.3

Lemma. *B.1: Let K and n be such that $3n(n\gamma - \gamma + 2) \leq 2K$, then d_i is increasing in i for all $i \geq 2$.*

Proof. Let K sinks be given and let $n \geq 1$ such that $3(n\gamma-\gamma+2)n \leq 2K$. Then define as before:

$$R_i = i\frac{R}{n}$$

For the sink distribution we have to think about, what happens, when the total number of sinks is not such a multiple of K_1 that Equation (5.6.1) is fullfilled. We start by defining

$$K_1 = \left\lfloor \frac{2K}{n(\gamma n - \gamma + 2)} \right\rfloor \quad \text{and} \quad K_i = \lfloor K_1 + \gamma(i-1)K_1 \rfloor$$

and denote the rest of the sinks by L :

$$L = K - \sum_{i=1}^{n} K_i$$

□

Lemma. *B.2: The chosen distribution fulfills:*

$$\sum_i K_i = K$$

Proof. We know that:

$$\sum_i K_i = \sum_i iK_1 + l_i = L + \sum_i iK_1 = K$$

□

Lemma. *B.3: It holds $K_1 \geq 3$*

Proof. Since $3(\gamma n - \gamma + 2)n \leq 2K$ we have $\frac{2K}{n(\gamma n-\gamma+2)} \geq 3$, from which follows $\lfloor \frac{2K}{n(\gamma n-\gamma+2)} \rfloor \geq 3$. □

For calculating the maximal distance $d(n)$ in the polar grid, we would need to calculate the distances in the orbits $d_i(n)$. To calculate these distances we need to know, if we are considering "short" trapezoids or "long" trapezoids. The following theorem shows, that we have to consider only "short" trapezoids in all orbits, which makes the upfollowing calculations much easier.

Theorem. *B.1: For all orbits holds that the resulting trapezoids are short, i.e.*

$$\frac{|AB|}{2} \leq x \tag{B.0.1}$$

Proof. Fix some i and assume first that $L = 0$, then

$$\frac{|AB|}{2} = i\frac{R}{n}\sin(\frac{\pi}{(\gamma i - \gamma + 1)K_1})$$

and

$$x^2 = R_i^2 \sin^2(\frac{\pi}{(\gamma i - \gamma + 1)K_1}) - 2R_iR_{i-1}\sin^2(\frac{\pi}{(\gamma i - \gamma + 1)K_1}) + R_{i-1}^2$$

Hence (B.0.1) is equivalent to the condition that:

$$2R_iR_{i-1}\sin^2(\frac{\pi}{(\gamma i - \gamma + 1)K_1}) \leq R_{i-1}^2$$

Which is, by inserting values for the radii, equivalent to the condition:

$$\begin{aligned} i - 1 &\geq 2i\sin^2(\frac{\pi}{(\gamma i - \gamma + 1)K_1}) \\ 1 &\leq i(1 - 2\sin^2(\frac{\pi}{(\gamma i - \gamma + 1)K_1}) = i\cos(\frac{2\pi}{(\gamma i - \gamma + 1)K_1}) \end{aligned}$$

This is fulfilled for all $i \geq 2$ and all $K_1 \geq 3$, $\gamma \in [1, 2]$. To give the maximal distance in the whole polar grid, it suffices to calculate the maximal distance in the n -th orbit, if $L = 0$: □

Theorem. *B.2:* $d_i(n)$ *is an increasing function in* $i \geq 2$ *for all* $\gamma \in [1, 2]$.

Proof. We know that:

$$d_i(n) = \frac{R(1+4(i-1)i\cos^2(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1}))^{\frac{1}{2}}}{2n\sin(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})}$$
$$= \frac{R}{n}(\frac{1}{4\sin^2(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})}+(i-1)i\cot^2(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1}))^{\frac{1}{2}}$$

Since $d_i(n)$ is positive, it is sufficient to show that $(d_i(n))^2$ has positive derivative, which is given by:

$$D_i(d_i^2(n)) = \frac{-\cos(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})\sin(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})(\frac{\gamma\pi}{(\gamma i-\gamma+1)^2K_1})}{2\sin^4(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})}$$
$$+\frac{(4i-2)\cos^2(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})\sin^2(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})}{2\sin^4(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})}$$
$$-\frac{4(i^2-i)\cos(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})\sin^2(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})(\frac{\gamma\pi}{(\gamma i-\gamma+1)^2K_1})}{2\sin^4(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})}$$

Since the denominator is larger zero for all $i \geq 2$, $K_1 \geq 3$ and $\gamma \in [1,2]$ we can concentrate on the numerator. Note that we can factor out:

$$\cos(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})\sin(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1}) = \frac{1}{2}\sin(\pi-\frac{2\pi}{(\gamma i-\gamma+1)K_1}) \geq 0$$

Hence we have to prove:

$$(2i-1)\sin(\pi-\frac{2\pi}{(\gamma i-\gamma+1)K_1}) \geq \frac{\gamma\pi}{(\gamma i-\gamma+1)^2K_1}(4(i^2-i)\sin(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})+1) \quad \text{(B.0.2)}$$

We start with the left side of (B.0.2). Using Taylor-Expansion around π we have:

$$(2i-1)\sin(\pi-\frac{2\pi}{(\gamma i-\gamma+1)K_1}) \geq (2i-1)\left(\frac{2\pi}{(\gamma i-\gamma+1)K_1}-(\frac{2\pi}{(\gamma i-\gamma+1)K_1})^4\frac{1}{4!}\right)$$

the right side of (B.0.2) will be also treated by a Taylor-Expansion around π :

$$\frac{\gamma\pi}{(\gamma i-\gamma+1)^2K_1}(4(i^2-i)\sin(\frac{\pi}{2}-\frac{\pi}{(\gamma i-\gamma+1)K_1})+1)$$
$$\leq \frac{\gamma\pi}{(\gamma i-\gamma+1)^2K_1}(4(i^2-i)(\frac{\pi}{(\gamma i-\gamma+1)K_1}+\frac{\pi^3}{3!(\gamma i-\gamma+1)^3K_1^3})+1)$$

Comparing these two expressions we need to show:

$$(2i-1)(2(\gamma i-\gamma+1)-\frac{\pi^3}{3(\gamma i-\gamma+1)^2K_1^3})$$
$$\geq 4\gamma(i^2-i)(\frac{\pi}{(\gamma i-\gamma+1)K_1}+\frac{\pi^3}{3!(\gamma i-\gamma+1)^3K_1^3})+\gamma$$

Next we will eliminate the parameter γ by noting that $\gamma \in [1,2]$ and hence $(\gamma i - \gamma + 1) \in [i, 2i-1]$. The expressions (3) and (4) can hence be bounded and we can simplify the inequality further:

$$(2i-1)(2i - \frac{\pi^3}{3i^2K_1^3} \geq 8(i^2-i)(\frac{\pi}{iK_1} + \frac{\pi^3}{3!i^3K_1^3}) + 2$$

Multiplying this by $3i^2K_1^3$ the inequality can be rewritten into a multinomial in K_1 and i :

$$12i^4K_1^3 - i^3(6K_1^3 + 24\pi K_1^2) + i^2(24\pi K_1^2 6K_1^3) - 6i\pi^3 + 5\pi^3 \geq 0$$

With standard methods of analysis one can show, that this inequality is fulfilled for each $K_1 \geq 3$ and $i \geq 2$ (see also the next part). Hence (B.0.2) is fulfilled for all $\gamma \in [1,2]$ and by this the derivative $D_i(d_i^2(n))$ is positive.

Then, d_i can be given by:

$$d_i = \frac{R}{n}\Big(\frac{1}{4\sin^2(\frac{\pi}{2} - \frac{\pi}{(\gamma i-\gamma+1)K_1})} + (i-1)i\cot^2(\frac{\pi}{2} - \frac{\pi}{(\gamma i-\gamma+1)K_1})\Big)^{1/2}.$$

Since d_i is positive, it is sufficient to show that d_i^2 has a positive derivative, after factoring out $\cos(\frac{\pi}{2} - \frac{\pi}{(\gamma i-\gamma+1)K_1})\sin(\frac{\pi}{2} - \frac{\pi}{(\gamma i-\gamma+1)K_1}) \geq 0$, its numerator (the denominator is positive) can be given by

$$(2i-1)\sin\left(\pi - \tfrac{2\pi}{(\gamma i-\gamma+1)K_1}\right) - \tfrac{\pi\gamma}{(\gamma i-\gamma+1)^2K_1}\left(4(i^2-i)\sin(\tfrac{\pi}{2} - \tfrac{\pi}{(\gamma i-\gamma+1)K_1}) + 1\right).$$

using a Taylor-Expansion around π for the first sum and using that $\sin(x) \leq 1$ for all x, we know that the above expression is larger than zero, if

$$\frac{2\pi^3}{3i^2K_1^3} \leq 1$$

which is true for all $i \geq 2$ and $K_1 \geq 3$, which is the case by our assumptions on K and n. □

C. The k-Center Heuristic

The k -Center heuristic of Hochbaum and Shmoys will be presented and modified, such that it fits our needs. The k -Center heuristic is at its core a bisection search for the optimal value for the paramter "mid". The algorithm chooses with the help of this parameter a set of sinks, if this set has less than K or just K elements, the parameter "mid" must be decreased (because we can assume to achieve a better maximal distance, if we can place more sinks), if the set is larger than K we have to increase "mid", since we are using too much sinks. The original algorithm works on a fully connected, edge-weighted graph, satisfying the triangle-inequality. Hochbaum and Shmoys algorithm is a 2-approximation, which is best possible, in the sense that finding a δ -approximation wit polynomial runtime and $\delta < 2$ leads to $NP = P$. Before we explain the algorithm, we need some notations. We talk of $G = (V, E)$ being a complete graph with edge weights w the edges are sorted by their weight, this means:

$$w(e_i) \leq w(e_j) \quad \forall\, i < j \leq m = |E|$$

The graph is stored in adjacency-list-form. This means for each vertex v the adjacent vertices are listed in increasing edge weight order. We need two more notations $G_i = (V, E_i)$, where $E_i = \{e_1, \ldots, e_i\}$ and $ADJ_i(x)$ which is the adjacency list of x in G_i . Next we will present the algorithm as it can be found in the paper of Hochbaum and Shmoys:

To adapt this algorithm to our purpose we made a few changes. At first our sensor network is not fully connected and on the other side links have no weights. we solve these two problems by using the euclidean distances between the nodes as link weight and assume the network to be fully connected. Further we are not operating an a complete list of edges, instead each node has its own list, which again contains the neighbours of the node in the order of increasing edge-weights. for this we denote by $n(v) = (x_{v,1}, x_{v,2}, \ldots, x_{v,N})$ the vector of neighbours of v and by $n_i(v) = (x_{v,1}, x_{v,2}, \ldots, x_{v,i})$ the first i neighbours of v . Watch out that

$n_i(v) \neq ADJ_i(v)$, in the first vector we have pruned the list of v to i neighbours. In the second vector we have pruned the complete set of edges to E_i and then take all neighbours of v which are left. A second change to the original algorithm is, that we are not deleting the neighbours of the neighbours of v from the set T . Instead we are just deleting the neighbours of v , which leads to less coordination between the nodes. The algorithm looks then like Algorithm C.2:

Algorithmus C.1 The k-center heuristic.

```
Begin:
  low:= 1;
  high:= m;
  if k =| V |
   S = V;
  end
  while high > low + 1 do
        mid:= ⌊high + low/2⌋
        S := ∅;
        T := V;
        while ∃x ∈ T do
           S := S ∪ {x};
           for v ∈ ADJ_mid(x) do
              T := T − ADJ_mid(v) − {v};
           end
        end
        if | S |⩽ k
         high := mid;
         S' := S;
        end
        if | S |> k
         low := mid;
        end
  end
end
```

Algorithmus C.2 The k-center heuristic for WSN.

Begin:
  low:= 1;
  high:= N;
  if $k =\mid V \mid$
   $S = V$;
  end
  while $high > low + 1$ **do**
    mid:= $\lfloor high + \frac{low}{2} \rfloor$
    $S := \emptyset$;
    $T := V$;
    while $\exists x \in T$ **do**
      $S := S \cup \{x\}$;
      $T := T - n_{mid}(x)$;
        First we have to convert the vector to a set at this line.
        The set is built simply by collecting all entries of the vector.
    end
    if $\mid S \mid \leqslant k$
     $high := mid$;
     $S' := S$;
    end
    if $\mid S \mid > k$
     $low := mid$;
    end
  end
end

If the algorithm outputs a set of sinks which has less than K elements, we place the difference of sinks randomly over the network. Note that as a result of this sink placement, we can bound the maximal euclidean distance from any node to the nearest sink by $\max_{v \in V}\{x_{v,mid}\}$.

Bibliography

[1] Chipcon CC2420 Datasheet: 2.4 GHz IEEE 802.15.4/ ZigBee-ready RF Transceiver.

[2] <http://www.btnode.ethz.ch>.

[3] <http://www.tinyos.net>.

[4] <http://www.tinyos.net/tinyos-1.x/doc/mica2radio/CC1000.html>.

[5] <http://www.xbow.com>.

[6] The 29 Palms Experiment: Tracking Vehicles with a UAV-delivered Sensor Network. <http://www.tinyos.millennium.berkeley.edu.29Palms.htm>.

[7] Crossbow Technology INC. MPR-MIB Users Mannual, June 2007.

[8] K. Akkaya and M. Younis. Relocation of Gateway for Enhanced Timeliness in Wireless Sensor Networks. In *Proc. of the IEEE Workshop on Energy-Efficient Wireless Communications and Networks (EWCN)*, April 2004.

[9] K. Akkaya and M. Younis. COLA: A Coverage and Latency Aware Actor Placement for Wireless Sensor and Actor Networks. In *Proc. IEEE Vehicular Technology Conference (VTC-Fall'06)*, September 2006.

[10] I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater Acoustic Sensor Networks: Research Challenges. *Ad Hoc Networks*, 3(3):257–279, 2005.

[11] S. M. N. Alam and Z. J. Haas. Coverage and Connectivity in Three-Dimensional Networks. In *Proc. ACM MobiCom*, pages 346–357, New York, NY, USA, 2006. ACM.

[12] C. Alippi and C. Galperti. An adaptive system for optimal solar energy harvesting in wireless sensor network nodes. *IEEE Transactions on Circuits and Systems*, 55(6):1742 –1750, july 2008.

[13] G. D. Bacco, T. Melodia, and F. Cuomo. A MAC Protocol for Delay-Bounded Applications in Wireless Sensor Networks. In *Proc. Med-Hoc-Net*, 2004.

[14] X. Bai, S. Kumar, and D. Xuan. Deploying Wireless Sensor to Achieve Both Coverage and Connectivity. In *Proc. ACM MobiHoc*, 2006.

[15] X. Bai, D. Xuan, Z. Yun, T. H. Lai, and W. Jia. Complete Optimal Deployment Patterns for Full-Coverage and k-Connectivity (k<=6) Wireless Sensor Networks. In *Proc. ACM MobiHoc*, May 2008.

[16] X. Bai, Z. Yun, D. Xuan, W. Jia, and W. Zhao. Pattern Mutation in Wireless Sensor Deployment. In *Proc. IEEE INFOCOM*, 2010.

[17] X. Bai, Z. Yun, D. Xuan, T. H. Lai, and W. Jia. Deploying Four-Connectivity and Full-Coverage Wireless Sensor Networks. In *Proc. IEEE INFOCOM*, 2008.

[18] P. Balister and S. Kumar. Random vs. Deterministic Deployment of Sensors in the Presence of Failures and Placement Errors. In *Proc. IEEE INFOCOM*, 2009.

[19] N. Bartolini, T. Calamoneri, T. F. La Porta, and S. Silvestri. Autonomous Deployment of Heterogeneous Mobile Sensors. *IEEE Transactions on Mobile Computing*, june 2011.

[20] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z. M. Wang. Controlled Sink Mobility for Prolonging Wireless Sensor Networks Lifetime. *Wireless Sensor Network, Springer Netherlands*, 14(6):831–858, December 2008.

[21] S. Basagni, A. Carosi, and C. Petrioli. Heuristics for Lifetime Maximization in Wireless Sensor Networks with Multiple Mobile Sinks. In *Proc. IEEE ICC*, June 2009.

[22] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. MOS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms. In *ACM Kluwer Mobile Networks and Applications Journal, Special Issue on Wireless Sensor Networks*, 2005.

[23] A. Bogdanov, E. Maneva, and S. Riesenfeld. Power-Aware Base Station Positioning for Sensor Networks.

[24] A. Boukerche, X. Fei, and R. B. Araujo. A Coverage Preserving and Fault Tolerant Based Scheme for Irregular Sensing Range in Wireless Sensor Networks. In *Proc. IEEE GLOBECOM*, San Francisco, CA, November 2006.

[25] J. Bruck, J. Gao, and A. Jiang. MAP: Medical Axis Based Geometric Routing in Sensor Networks. In *Proc. ACM MobiCom*, 2005.

[26] D. Brunelli, L. Benini, C. Moser, and L. Thiele. An efficient solar energy harvester for wireless sensor nodes. In *DATE*, pages 104–109, 2008.

[27] T. M. Cavalier, W. Conner, E. Castillo, and S. I. Brown. A Heuristic Algorithm for Minimax Sensor Locations in the Plane. In *Europian Journal of Operational Research*, 2007.

[28] I. Chatzigiannakis, A.Kinalis, S. Nikoletseas, and J. Rolim. Fast and Energy Efficient Sensor Data Collection by Multiple Mobile Sinks. In *Proc. ACM MobiWac*, pages 25–32, New York, NY, USA, 2007.

[29] C. Chen, J. Ma, and K. Yu. Designing Energy-Efficient Wireless Sensor Networks with Mobile Sinks. In *Proc. ACM SenSys*, Boulder, Colorado, USA., October 2006.

[30] X. Cheng, D. Z. Du, L. Wang, and B. Xu. Relay Sensor Placement in Wireless Sensor Networks. *Wireless Networking Journal*, 14(3):347–355, 2008.

[31] I. D. Choi, J. S. Park, K. H. Lee, and B. K. Oh. The Three Dimensional Node Deployment for Sensor Network. In *Proc. of the Advanced Software Engineering and Its Applications*, pages 271–275, Washington, DC, USA, 2008. IEEE Computer Society.

[32] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Suluja. Sensor Deployment Strategy for Target Detection. In *Proc. WSNA'02: The 1st ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, Georgia, September 2002.

[33] K. Dasgupta, M. Kukreja, and K. Kalpakis. Topology-Aware Placement and Role Assignment for Energy-Efficient Information Gathering in Sensor Networks. In *Proc. IEEE ISCC*, Kemer-Antalya, Turkey, June 2003.

[34] S. S. Dhillon and K. Chakrabarty. Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks. In *Proc. IEEE WCNC*, New Orleans, LA, March 2003.

[35] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proc. IEEE LCN*, 2004.

[36] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. 2005.

[37] A. Duttagupta, A. Bishnu, and I. Sengupta. Maximal Breach in WSNs: Geometric Characterization and Algorithm. In *Proc. AlgoSensors: International Workshop on Theoretical and Algorithmic Aspects of Sensor Ad Hoc Wireless and Peer-to-Peer Networks*, 2008.

[38] A. Efrat, S. Peled, and J. Mitchel. Approximation Algorithms for Two Optimal Location Problems in Sensor Networks. In *Broadband Networks*, 2005.

[39] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-Network Aggregation Techniques for Wireless Sensor Networks: A Survey. In *IEEE Wireless Communication*, 2007.

[40] M. Fidler and G. Einhoff. Routing in Turn-Prohibition Based Feed-Forward Networks. pages 1168–1179, 2004.

[41] R. L. Francis and J. A. White. *Facility Layout and Location: An Analytical Approach*. Prentice-Hall, Englewood Cliffs, N.J, 1974.

[42] S. R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy Efficient Schemes for Wireless Sensor Networks with Multiple Mobile Base Stations. In *Proc. IEEE GLOBECOM*, December 2003.

[43] S. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. *Proc. ACM SIGMOBILE Mobile Computing and Communications Review*, 5:11–25, October 2001.

[44] S. Gao, H. Zhang, and S. Das. Efficient Data Collection in Wireless Sensor Networks With Path-constrained Mobile Sinks. In *Proc. IEEE WoWMoM*, October 2009.

[45] A. Gogu, D. Nace, A. Dilo, and N. Meratnia. Optimization Problems in Wireless Sensor Networks. In *Proc. International Conference on Complex, Intelligent and Software Intensive Systems*, 2011.

[46] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA, 1989.

[47] D. K. Goldenberg, J. Lin, A. S. Morse, B. E. Rosen, and Y. R. Yang. Towards Mobility as a Network Control Primitive. In *Proc. ACM MobiHoc*, pages 163–174. ACM Press, 2004.

[48] H. H. Gonzalez-Banos and J. C. Latombe. A Randomized Art-Gallery Algorithm for Sensor Placement. In *Proc. of the 17th ACM Symposium on Computational Geometry (SoCG)*, jun 2001.

[49] R. Govindan, C. Intanagonwiwat, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. ACM/IEEE MobiCom*, 2000.

[50] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes for Large Scale Sensor Networks. In *Proc. of the 9th Annual International Conference on Mobile Computing and Networking*, pages 81–95, 2003.

[51] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *Proc. ACM/IEEE MobiCom*, 1999.

[52] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proc. of the 33rd Annual Hawaii International Conference on System Sciences*, volume 2. IEEE, January 2000.

[53] D. S. Hochbaum and D. B. Shmoys. A Best Possible Heuristic for the k-Center Problem. *Mathematics of Operations Research*, 10(3):180–184, May 1985.

[54] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[55] Y. T. Hou, Y. Shi, H. D. Sherali, and S. F. Midkiff. Prolonging Sensor Network Lifetime with Energy Provisioning and Relay Node Placement. In *Proc. 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2005.

[56] W-L. Hsu and G. L. Nemhauser. Easy and Hard Bottleneck Location Problems. *Discrete Applied Mathematics*, 1:209–215, 1979.

[57] C. F. Huang and Y. C. Tseng. The Coverage Problem in a Wireless Sensor Network. In *Proc. ACM MobiCom*, San Diego, CA, 2003.

[58] M. Ishizuka and M. Aida. Performance Study of node Placement in Sensor Networks. In *Proc. ICDCS'04: The 24th International Conference on Distributed Computing Systems Workshops*, Tokyo, Japan, March 2004.

[59] R. Iyengar, K. Kar, and S. Banerjee. Low-coordination Topology for Redundancy in Sensor Networks. In *Proc. ACM MobiHoc*, 2005.

[60] K. Kar and S. Banerjee. Node Placement for Connected Coverage in Sensor Networks. In *Proc. WiOpt'03: The Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[61] H. Karl and A. Wittig. *Protocols and Architectures for Wireless Sensor Networks*. Wiley, 2005.

[62] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. ACM MobiCom*, 2000.

[63] A.-M. Kermarrec and G. Tan. Greedy Geographic Routing in Large-Scale Sensor Networks: A Minimum Network Decomposition Approach. In *Proc. ACM MobiHoc*, 2010.

[64] R. Kershner. The Number of Circles Covering a Set. *American Journal of Mathematics*, 1939.

[65] H. Kim, Y. Seok, N. Choi, Y. Choi, and T. Kwon. Optimal Multi-sink Positioning and Energy-efficient Routing in Wireless Sensor Networks. In *Lecture Notes in Computer Science(LNCS)*, volume 3391, pages 264–274, Tucson, AZ, USA, December 2005.

[66] D. Kotz, C. Newport, and C. Elliott. The Mistaken Axioms of Wireless Network Research. Technical report, 2003.

[67] A. Koubaa, M. Alves, and E. Tovar. Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks (RTSS'06). In *Proc. IEEE Real-Time Systems Symposium (RTSS'06)*, Rio de Janeiro, Brazil, 2006.

[68] A. Kuehn and M. Hamburger. A Heuristic Program for Locating Warehouses. *Management Science*, 1963.

[69] S. Kumar, T. H. Lai, and J. Balogh. On k-Coverage in a Mostly Sleeping Sensor Network. In *Proc. ACM MobiCom*, pages 144–158, New York, NY, USA, 2004. ACM.

[70] J.-Y. LeBoudec. Application of Network Calculus to guaranteed sevice networks. In *IEEE Transactions on Information Theory*, 1998.

[71] J.-Y. LeBoudec and P. Thiran. *Network Calculus - A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.

[72] M. Lee and V. W. S. Wong. An Energy-aware Spanning Tree Algorithm for Data Aggregation in Wireless Sensor Networks. In *Proc. IEEE PacRrim*, 2005.

[73] M. Leoncini, G. Resta, and P. Santi. Analysis of a Wireless Sensor Dropping Problem in Wide-Area Environmental Monitoring. In *Proc. IEEE IPSN*, apr 2007.

[74] J. Li, Y. Bai, H. Ji, and D. Qian. POWER: Planning and Deployment Platform for Wireless Sensor Networks. In *Proc. GCCW'06: The Fifth International Conference on Grid and Cooperative Computing Workshops*, 2006.

[75] X.-Y. Li, J.-J. Wan, and O. Frieder. Coverage in Wireless Ad Hoc Sensor Networks. *IEEE Transactions on Computing*, 2003.

[76] W. Liang, J. Luo, and X. Xu. Prolonging Network Lifetime via a Controlled Mobile Sink in Wireless Sensor Networks. In *Proc. IEEE GLOBECOM*, 2010.

[77] S. Lindsey and C. S. Raghavendra. PEGASIS: Power Efficient GAthering in Sensor Information Systems. In *Proc. IEEE Aerospace Conference*, 2002.

[78] S. Lindsey, C. S. Raghavendra, and K. Sivalingam. Data Gathering in Sensor Networks Using the Energy-Delay Metric. In *Proc. IPDPS Workshop on Issues in Wireless Networks and Mobile Computing*, 2001.

[79] Y. Liu, G. Zhou, J. Zhao, G. Dai, X.Y. Li, M. Gu, H. Ma, L. Mo, Y. He, J. Wang, M. Li, K. Liu, W. Dong, and W. Xi. Long-Term Large-Scale Sensing in the Forest: Recent Advances and Future Directions of GreenOrbs. *Frontiers of Computer Science in China*, 4, 2010.

[80] H. Long, Y. Liu, Y. Wang, R .P. Dick, and H. Yang. Battery Allocation for Wireless Sensor Network Lifetime Maximization Under Cost Constraints. In *Proc. IEEE/ACM ICCAD*, 2009.

[81] J. Luo. *Mobility in Wireless Networks: Friend or Foe - Network Design and Control in the Age of Mobile Computing*. PhD thesis, School of Computer and Communication Sciences, EPFL, Switzerland, 2006.

[82] J. Luo and J.-P. Hubaux. Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks. In *Proc. IEEE INFOCOM*, March 2005.

[83] J. Luo and J.-P. Hubaux. Mobility to Improve the Lifetime of Wireless Sensor Networks: A Theoretical Framework. In *Proc. IEEE/ACM DCOSS-MSWSN Workshop*, 2006.

[84] J. Luo and L. Xiang. Prolong The Lifetime of Wireless Sensor Networks Through Mobility: A General Optimization Framework. *Theoretical Aspects of Distributed Computing in Sensor Networks*, pages 553 –588, June 2010.

[85] M. Ma and Y. Yang. Data Gathering in Wireless Sensor Networks with Mobile Collectors. In *Proc. IEEE IPDPS*, April 2008.

[86] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Stream. In *Proc. ACM SIGMOD*, 2005.

[87] M. Marta and M. Cardei. Improved Sensor Network Lifetime with Multiple Mobile Sinks. *Pervasive and Mobile Computing*, pages 542–555, 2009.

[88] K. Mechitov, W. Kim, G. Agha, and T. Nagayama. High Frequency Distributed Sensing for Structure Monitoring. In *Proc. EmNetS-II: The 2nd IEEE Workshop on Embedded Networked Sensors*, jun 2004.

[89] S. Megerian, F. Koushanfar, G. Qu, G. Veltri, and M. Potkonjak. Exposure in Wireless Sensor Networks: Theory and Pratical Solutions. *Wireless Networks*, 2002.

[90] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage Problems in Wireless Ad Hoc Sensor Networks. In *Proc. IEEE INFOCOM*, 2001.

[91] L. Mo, Y. He, Y. Liu, J.Z. Zhao, S. Tang, X. Y. Li, and G. Dai. Canopy Closure Estimates with GreenOrbs: Substainable Sensing in the Forest. In *Proc. ACM SenSys*, 2009.

[92] M. Mudigonda, T. Kanipakam, A. Dutko, M. Bathula, N. Sridhar, S. Seetharaman, and J.O. Hallstrom. A Mobility Management Framework for Optimizing the Trajectory of a Mobile Base-station. In *Proc. EWSN*, 2011.

[93] A. Nasipuri and K. Li. A Directionality Based Location Discovery Scheme for Wireless Sensor Networks. In *Proc. ACM WSNA*, sept 2002.

[94] S. Nath, P. B. Gibbons, Z. R. Anderson, and S. Seshan. Synopsis Diffusion for Robust Aggregation in Sensor Networks. In *Proc. ACM SenSys*, 2004.

[95] H. A. Nguyen, A. Forster, D. Puccinelli, and S. Giordano. Sensor Node Lifetime: An Experimental Study. In *Proc. IEEE PERCOM Workshop*, 2011.

[96] N. Nulusu, J. Heidemann, and D. Estrin. GPS-Less Low Cost Outdoor Localization For Very Small Devices. *IEEE Personal Communications Magazine*, pages 28–34, 2000.

[97] S. Olariu and I. Stojmenovic. Design Guidelines for Maximizing Lifetime and Avoiding Energy Holes in Sensor Networks with Uniform Distribution and Uniform Reporting. In *Proc. IEEE INFOCOM*, 2006.

[98] J. O'Rourke. Art Gallery Theorems and Algorithms. *Oxford University Press*, 1987.

[99] E. I. Oyman and C. Ersoy. Multiple Sink Network Design Problem in Large Scale Wireless Networks. In *Proc. IEEE ICC*, June 2004.

[100] J. Paek, K. Chintalapudi, J. Cafferey, R. Govindan, and S. Marsi. A Wireless Sensor Network for Structural Health Monitoring: Performance and Experience. may 2005.

[101] J. Pan. Locating Base-Stations for Video Sensor Network. In *Proc. IEEE Vehicular Technology Conference (VTC-Fall'03)*, October 2003.

[102] J. Pan, L. Cai, Y. T. Hou, Y. Shi, and S. X. Shen. Optimal Base Station Locations in Two-tired Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 4:458–473, 2005.

[103] I. Papadimitriou and L. Georgiadis. Maximum Lifetime Routing to Mobile Sink in Wireless Sensor Networks. In *Proc. SoftCOM*, September 2005.

[104] S. Poduri, S. Pattem, B. Krishnamachari, and G. S. Sukhatme. Sensor Network Configuration and the Curse of Dimensionality. In *Proc. of The Third IEEE Workshop on Embedded Networked Sensors*, Cambridge, MA, 2006.

[105] W. Y. Poe, M. Beck, and J. B. Schmitt. Planning the Trajectories of Multiple Mobile Sinks in Large-Scale, Time-Sensitive WSNs. In *Proc. IEEE DCOSS*, June 2011.

[106] W. Y. Poe and J. B. Schmitt. Placing Multiple Sinks in Time-Sensitive Wireless Sensor Networks using a Genetic Algorithm. In *Proc. MMB'08: 14th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems*, Dortmund, Germany, March 2008. GI/ITG.

[107] L. Qui, R. Chandra, K. Jain, and M. Mahdian. Optimizing the Placement of Integration Points in Multi-Hop Wireless Networks. In *Proc. ICNP*, 2004.

[108] A. M. Reddy, V. AVU. P. Kumar, D. Janakiram, and G. A. Kumar. Operating Systems for Wireless Sensor Networks: A Survey. Technical report, 2007.

[109] M. Ringwald and K. Romer. Deployment of Sensor Networks: Problems and Passive Inspection. In *Proc. of Fifth International Workshop on Intelligent Solutions in Embedded Systems*, 2007.

[110] V. Rodoplu and T. H. Meng. Minimum Energy Mobile Wireless Networks. In *Proc. IEEE ICC*, 1998.

[111] V. Rodoplu and T. H. Meng. Ad Hoc On-Demand Distance Vector (aodv) Routing. In *Proc. MCSA Workshop*, 1999.

[112] K. Roemer and F. Mattern. The Design Space of Wireless Sensor Networks. *IEEE Wireless Communications*.

[113] Gowrishankar. S, T. G. Basavaraju, Manjaiah D. H., and S. K. Sarkar. Issues in Wireless Sensor Networks. In *Proc. World Congress on Engineering*, 2008.

[114] P. Santi and D. M. Blough. The Critical Transmitting Range for Connectivity in Sparse Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 2003.

[115] J. B. Schmitt and U. Roedig. Sensor Network Calculus - A Framework for Worst Case Analysis. In *Proc. DCOSS*, June 2005.

[116] J. B. Schmitt and F. A. Zdarsky. The DISCO Network Calculator - A Toolbox for Worst Case Analysis. In *Proc. ACM VALUETOOLS'06: The First International Conference on Performance Evaluation Methodologies and Tools*, November 2006.

[117] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay Bounds Under Arbitrary Aggregate Multiplexing: When Network Calculus Leaves You in the Lurch... In *Proc. IEEE INFOCOM)*, April 2008.

[118] J. B. Schmitt, F. A. Zdarsky, and U. Roedig. Sensor Network Calculus with Multiple Sinks. In *Proceedings of the Performance Control in Wireless Sensor Networks Workshop at the 2006 IFIP Networking Conference*, May 2006.

[119] J. B. Schmitt, F. A. Zdarsky, and L. Thiele. A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing. In *Proc. IEEE Real-Time Systems Symposium (RTSS'07)*, Tucson, AZ, USA, December 2007.

[120] W.K.G. Seah, Zhi Ang Eu, and Hwee-Pink Tan. Wireless sensor networks powered by ambient energy harvesting (wsn-heap) - survey and challenges. In *The 1st International Conference on Wireless VITAE 2009*, pages 1 –5, may 2009.

[121] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. In *Proc. IEEE Workshop SNPA*, pages 30–41, 2003.

[122] S. Shakkottai, R. Srikant, and N. Shroff. Unreliable Sensor Grids: Coverage, Connectivity and Diameter. In *Proc. IEEE INFOCOM*, volume 2, pages 1073–1083, New York, NY, USA, March 2003. ACM.

[123] M. I. Shamos and D. Hoey. Closest-Point Problems. *16th Annual Symposium on Foundations of Computer Science, FOCS*, pages 151–162, 1975.

[124] H. She, Z. Lu, A. Jantsch, L-R. Zheng, and D. Zhou. Traffic Splitting with Network Calculus for Mesh Sensor Networks. In *Proc. Future Generation Communication and Networking (FGCN'07)*, Washington, DC, USA, 2007.

[125] Y. Shi and Y. T. Hou. Theoretical Results on Base Station Movement Problem for Sensor Network. In *Proc. IEEE INFOCOM*, Phoenix, AZ,USA, 2008.

[126] R. Shokri, P. Papadimitratos, M. Poturalski, and J. P. Hubaux. A Low-Cost Method to Thwart Relay Attacks in Wireless Sensor Networks. Project Report IC-71, Security and Cooperation in Wireless Networks, Doctoral School of the I and C School of EPFL, 2007.

[127] A. A. Somasundara, A. Kansal, D. D. Jea, D. Estrin, and M. B. Srivastava. Controllably Mobile Infrastructure for Low Energy Embedded Networks. *IEEE Transactions on Mobile Computing*, 5:958–973, 2006.

[128] K-F. Ssu, C-H. Ou, and H. C. Jiau. Localization With Mobile Anchor Points in Wireless Sensor Networks. In *IEEE Transactions on Vehicular Technology*, May 2005.

[129] R. Sugihara and R. K. Gupta. Optimizing Energy-Latency Trade-off in Sensor Networks with Controlled Mobility. In *Proc. IEEE INFOCOM*, pages 2566–2570, Rio de Janeiro, Brazil, April 2009.

[130] P. Suriyachai, U. Roedig, and A. Scott. Implementation of a Deterministic Wireless Sensor Network. In *Proc. EWSN'08: The 5th IEEE European Workshop on Wireless Sensor Networks*, Bologna, Italy, January.

[131] G. Takahara, K. Xu, and H. Hassanein. How Resilient is Grid-based WSN Coverage to Deployment Errors? In *Proc. IEEE WCNC*, 2007.

[132] G. Tan, M. Bertier, and A.-M. Kermarrec. Convex Partition of Sensor Networks and Its Use in Virtual Coordinate Geographic Routing. In *Proc. IEEE INFOCOM*, 2009.

[133] G. Tan, M. Bertier, and A.-M. Kermarrec. Visibility-Graph-based Shortest-Path Geographic Routing in Sensor Networks. In *Proc. IEEE INFOCOM*, 2009.

[134] G. Tan, S.A. Jarvis, and A.-M. Kermarrec. Connectivity- Guaranteed and Obstacle-Adaptive Deployment Schemes for Mobile Sensor Networks. *IEEE Transactions on Mobile Computing*, june 2009.

[135] J. Tang, B. Hao, and A. Sen. Relay Node Placement in Large-Scale Wireless Sensor Networks. *Computer Communication Journal, Special Issue on Wireless Sensor Neworks*, 29:490–501, 2006.

[136] S. Tang, J. Yuan, X.Y. Li, Y. Liu, G. H. Chen, M. Gu, J.Z. Zhao, and G. Dai. DAWN: Energy Efficient Data Aggregation in WSN with Mobile Sinks. In *Proc. IWQoS*, June 2010.

[137] V. V. Vazirani. Approximation Algorithms. *Springer-Verlag*, 2001.

[138] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak. Minimal and Maximal Exposure Path Algorithms for Wireless Embedded Sensor Networks. In *Proc. ACM SenSys'03*, 2003.

[139] Z. Vincze, K. Fodor, R. Vida, and A. Vidacs. Electrostatic Modelling of Multiple Mobile Sinks in Wireless Sensor Networks. In *Proc. IFIP'06: The Performance Control in Wireless Sensor Networks Workshop*, May 2006.

[140] Z. Vincze, R. Vida, and A. Vidacs. Deploying Multiple Sinks in Multi-hop Wireless Sensor Networks. In *Proc. IEEE Pervasive Services*, July 2007.

[141] P. Wan and C. Yi. Coverage by Randomly Deployed Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 13:2658–2669, 2006.

[142] G. Wang, G. Cao, and T. La Porta. Movement-Assisted Sensor Deployment. In *Proc. IEEE INFOCOM*, Hong Kong, March 2004.

[143] G. Wang, G. Cao, T. La Porta, and W. Zhang. Sensor Relocation in Mobile Sensor Networks. In *Proc. IEEE INFOCOM*, Miami, FL, March 2005.

[144] Z. M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli. Exploiting Sink Mobility for Maximizing Sensor Networks Lifetime. In *Proc. of the 38th Hawaii International Conference on System Sciences*, Big Island, Hawaii, January 2005.

[145] M. K. Watfa, M. Moubarak, and A. Kashani. Operating System Design in Future Wireless Sensor Networks. In *Journal of Networks*, 2010.

[146] J. Wu and S. Yang. SMART: A Scan-Based Movement Assisted Sensor Deployment Method in Wireless Sensor Networks. In *Proc. IEEE INFOCOM*, Miami, FL, March 2005.

[147] X. Wu and G. Chen. Dual-Sink: Using Mobile and Static Sinks for Lifetime Improvement in Wireless Sensor Networks. In *Proc. ICCCN'07: The 16th International Conference on Computer Communications and Networks*, August 2007.

[148] X. Wu, G. Chen, and S. K. Das. Avoiding Energy Holes in Wireless Sensor Networks with Nonuniform Node Distribution. *IEEE Transactions on Parallel and Distributed Systems*, 2008.

[149] Y. Wu and Y. Li. Construction Algorithms for k-Connected m-Dominating Sets in WSN. In *Proc. ACM MobiHoc*, 2008.

[150] K. Xu, H. Hassanein, and G. Takahara. Relay Node Deployment Strategies in Heterogeneous Wireless Sensor Networks: Multiple-Hop Communication Case. In *Proc. IEEE SECON*, September 2005.

[151] R. Xu and D. Wunsch II. Survey of Clustering Algorithms. In *IEEE Transactions on Neural Networks*, may 2005.

[152] M. Younis and K. Akkaya. Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey. *Ad Hoc Networks Journal*, 6(4):621–655, 2008.

[153] M. Younis, M. Bangad, and K. Akkaya. Base-Station Repositioning for Optimized Performance of Sensor Networks. In *Proc. IEEE Vehicular Technology Conference (VTC-Fall'03)*, October 2003.

[154] W. Youssef and M. Younis. Intelligent Gateway Placement for Reduced Data Latency in Wireless Sensor Networks. In *Proc. IEEE ICC*, June 2007.

[155] Z. Yuanyuan, X. Jia, and H. Yanxiang. Energy Efficient Distributed Connected Dominating Sets Construction in WSN. In *Proc. ACM IWCMC*, 2006.

[156] Y. Yun and Y. Xia. Maximizing the Lifetime of Wireless Sensor Networks with Mobile Sink in Delay-Tolerant Applications. In *IEEE Transactions on Mobile Computing*, volume 9, pages 1308–1318, 2010.

[157] H. Zhang and J. C. Hou. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. In *Proc. AlgoSensors: International Workshop on Theoretical and Algorithmic Aspects of Sensor Ad Hoc Wireless and Peer-to-Peer Networks*, 2004.

[158] B. Zhao and M. C. Valenti. Practical Relay Networks: A Generalization of Hybrid-ARQ. *IEEE Journal of Selected Areas in Communications*, 2005.

[159] G. Zhou, T. He, S. Krishamurthy, and J. A. Stankovic. Impact of Radio Irregularity on Wireless Sensor Networks. In *Proc. ACM MobiSYS*, 2004.

[160] J. Zhu, S. Chen, B. Bensaou, and K. Hung. Tradeoff between Lifetime and Rate Allocation Problems in WSN: A Cross Layer Approach. In *Proc. IEEE INFOCOM*, 2007.

List of Author's Publications

1. W. Y. Poe and J. B. Schmitt. Minimizing the Maximum Delay in Wireless Sensor Networks by Intelligent Sink Placements. Technical Report 362/07, University of Kaiserslautern, Germany, July 2007.

2. W. Y. Poe and J. B. Schmitt. Placing Multiple Sinks in Time-Sensitive Wireless Sensor Networks using a Genetic Algorithm. In Proc. MMB'08: 14th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems, Dortmund, Germany, March 2008. GI/ITG.

3. W. Y. Poe and J. B. Schmitt. Self-Organized Sink Placement in Large-Scale Wireless Sensor Networks. In Proc. IEEE MASCOTS, September 2009.

4. W. Y. Poe and J. B. Schmitt. Sink Placement without Location Information in Large-Scale Wireless Sensor Networks. In Proc. AINTEC'09: The 5th Asian Internet Engineering Conference, Bangkok, Thailand, November 2009.

5. W. Y. Poe and J. B. Schmitt. Node Deployment in Large Wireless Sensor Networks: Coverage, Energy Consumption, and Worst-Case Delay. In Proc. AINTEC'09: The 5th Asian Internet Engineering Conference, Bangkok, Thailand, November 2009.

6. W. Y. Poe, M. Beck, and J. B. Schmitt. Planning the Trajectories of Multiple Mobile Sinks in Large-Scale, Time-Sensitive WSNs. Tech. Report 381/11, University of Kaiserslautern, Germany, February 2011.

7. W. Y. Poe, M. Beck, and J. B. Schmitt. Planning the Trajectories of Multiple Mobile Sinks in Large-Scale, Time-Sensitive WSNs. In Proc. IEEE DCOSS, Barcelona, Spain, June 2011.

8. W. Y. Poe, M. Beck, and J. B. Schmitt. Achieving High Lifetime and Low Delay in Very Large Sensors Networks using Mobile Sinks. Tech. Report 385/11, University of Kaiserslautern, Germany, July 2011.

9. W. Y. Poe, M. Beck, and J. B. Schmitt. Achieving High Lifetime and Low Delay in Very Large Sensors Networks using Mobile Sinks. In Proc. IEEE DCOSS, Hangzhou, China, May 2012.

Curriculum Vitae

Personal Information

Name:	Wint Yi Poe
Place of Birth:	Okpho, Myanmar
Nationality:	Myanmar

Education

2006–2012	**Ph.D.**, in **Computer Science (Dr.-Ing.)**, Distributed Computer Systems Lab, TU Kaiserslautern, Germany
2003–2004	**M.Eng.**, in **Information Technology**, Yangon Technological University, Yangon, Myanmar
1998–2002	**B.Eng.**, in **Information Technology**, Mandalay Technological University, Mandalay, Myanmar
1985–1997	**Basic Education High School**, Okpho, Myanmar

Professional Experience

8.2006– present	**Scientific Staff Member**, Distributed Computer Systems Lab, Technical University Kaiserslautern, Germany
04.2004–07.2006	**Lecturer**, Department of Technical and Vocational Education, Yangon, Myanmar

Fellowships

12.2009– present	Doctoral scholarship from Technical University Kaiserslautern
08.2009–11.2009	STIBET scholarship from DAAD
08.2006–07.2009	Fellow of Gottlieb Daimler and Karl Benz Foundation

Internship

06.2002–02.2003	Vocational training instructors course by JICA (Information and Computer Engineering) at Polytechnic University in Hashimoto, Tokyo, Japan