On the Optimization of Wake-Up Frequencies in Sensor Networks

Boris Kapchits
On the Optimization of Wake-Up Frequencies in Sensor Networks

Research Thesis

Submitted in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

Boris Kapchits

Submitted to the Senate of
the Technion — Israel Institute of Technology
Adar 5770 Haifa February 2010
This research was conducted under the supervision of Assoc. Prof. Reuven Cohen in the Department of Computer Science.

First and foremost, I would like to express my gratitude to my supervisor Reuven Cohen. It was truly a great honor to be guided by him.

Throughout my graduate studies, I have had the opportunity to become acquainted with diverse individuals; these work relations have greatly assisted me to enhance and expand my scope of knowledge. I would like to express my special thanks to Danny Raz and Itai Dabran. I would also like to extend my appreciation to Israel Cidon, Hanoch Levy and Danny Raz for serving on my thesis committee.

In addition, I would like to thank all the people from the Technion computer science department with whom I have had the pleasure to interact and study. Special thanks are due to my floormate Anya Levin and all of those with whom work was just as great as having fun: Michael Ryabtsev, Benny Applebaum, Nadav Amit, Noam Mori and Konstantin Shagin. It was a true delight to be there with you guys!

Finally, I am indebted to my parents Lena and Mark for their undying support and encouragement; to my sister Ira for making me believe in myself; to my beloved wife Anastasia, for standing beside me every step of the way; and to my sons, Aviel and Itamar, for making me happy.

The generous financial support of the Technion is gratefully acknowledged.
Contents

Abstract 1

Abbreviations and Notations 2

1 Introduction 3

2 Related Work 6

3 An Optimal Wake-Up Scheduling Algorithm for Minimizing Energy Consumption while Limiting Maximum Delay in a Mesh Sensor Network 11
   3.1 Introduction 11
   3.2 Power Consumption Model 12
   3.3 Step 1: Minimizing the total consumed energy 15
   3.4 Step 2: Limiting the Energy Consumed by a Single Node 22
   3.5 Simulation Results 28

4 Energy-Delay Optimization in a Sensor Network with Multiple Gateways 33
   4.1 Introduction 33
   4.2 The MT-OFA Problem 34
   4.3 Constructing an Energy-Efficient Tree 36
      4.3.1 NP-hardness Proof 37
      4.3.2 An Approximation Algorithm for METP 42
   4.4 Simulation Study 43
      4.4.1 Simulation Results for MT-OFA 43
      4.4.2 METP Approximation Simulations 46
      4.4.3 Simulating Efficiency of Multiple Spanning Trees 47
5 Continuous Neighbor Discovery in Asynchronous Sensor Networks
5.1 Introduction ................................................................. 50
5.2 Basic Schemes and Problem Definition ................................. 53
5.3 Estimating the In-Segment Degree of a Hidden Neighbor ............. 56
5.4 An Efficient Continuous Neighbor Discovery Algorithm .......... 64
5.5 Simulation Study ............................................................ 66
   5.5.1 Simulation of the proposed schemes ............................... 66
   5.5.2 Detecting a small segment .......................................... 69

6 Conclusions ........................................................................ 72
List of Figures

3.1 A sensor network (the routing paths to the gateway are marked by thick lines) and a possible packet exchange for delivering data from v to w .......................... 14
3.2 A network with the corresponding T-tree, which is subject to our optimizations . 14
3.3 The considered topologies and the results of Step 1 ........................................... 22
3.4 Algorithm 2 after k – 1 iterations .......................................................... 27
3.5 The energy consumption ratio for a shortest-path routing as a function of the transmission range for different wake-up frequency upper bounds ...................... 29
3.6 The energy consumption ratio for the case where the routing algorithm minimizes the number of internal nodes ................................................................. 30
3.7 Assigned wake-up frequency average and variance as a function of Limit-Factor . 31
3.8 The network lifetime ratio as a function of the transmission range for different values of Limit-Factor ................................................................. 32
4.1 A sensor network with multiple routing trees ............................................. 34
4.2 The proof of Claim 7 .................................................................................. 41
4.3 Approximation algorithm results for multiple trees ..................................... 44
4.4 Average wake-up frequency for different D_i values .................................. 46
4.5 Wake-up assignment to nodes of a tree constructed by Algorithm 5 and a tree constructed by a BFS algorithm ......................................................... 47
4.6 Comparing two τ/D-optimal trees .............................................................. 48
4.7 The effect of multiple trees ......................................................................... 49
5.1 The transmission of HELLO control messages during neighbor discovery state . . 52
5.2 Continuous neighbor discovery vs. initial neighbor discovery in sensor networks 53
5.3 Segments with hidden nodes and links ...................................................... 55
5.4 Definitions for the proof of Theorem 4 .................................................... 61
5.5 $T_N(v)$ as a function of maximum tolerated delay ........................................ 65
5.6 Hidden neighbor detection, for the case of uniform distribution .................. 68
5.7 Hidden neighbors detection with extreme point ........................................... 70
5.8 The hidden neighbor discovery rate for a small detecting segment ............... 71
Abstract

This work addresses the tradeoff between efficient energy consumption and latency for the two most important tasks performed in a sensor network: packet routing and neighbor discovery. For the first task, we present an algorithm for maximizing the lifetime of a sensor network while guaranteeing an upper bound on the end-to-end delay. We prove that the proposed algorithm is optimal, and that it requires simple computing operations that can be implemented by simple devices. To the best of our knowledge, this is the first work to propose a sensor wake-up frequency that depends on the sensor’s location in the routing paths. Using simulations, we show that the proposed algorithm significantly increases the lifetime of the network, while guaranteeing a maximum on the end-to-end delay. This part of the thesis was published in [1].

We then extend this study to address a network with multiple gateways. Here we consider two sub-problems: the problem of constructing efficient routing trees and the problem of wake-up frequency assignment in a network with multiple routing trees. For the first problem we present an optimal algorithm and an approximation algorithm that achieves very close performance but can be more easily implemented. We prove that the second problem is NP-hard and propose a polynomial time approximation algorithm.

For the task of neighbor discovery, we define a new concept, called “ongoing neighbor discovery”. Although in most sensor networks the nodes are static, the node connectivity is subject to changes because of disruptions in wireless connectivity, transmission power changes, or loss of synchronization between neighboring nodes. Hence, even after a sensor is aware of its immediate neighbors, it must continuously maintain its view, a process we call continuous neighbor discovery. In this work we distinguish between neighbor discovery during sensor network initialization and continuous neighbor discovery and focus on the latter task. We view continuous neighbor discovery as a joint task of all the connected sensors. Each sensor employs a simple protocol in a coordinate effort to reduce power consumption without increasing the time required to detect hidden sensors. This part of the thesis was published in [2].
Abbreviations and Notations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MANETs</td>
<td>Mobile Ad-hoc NETworks</td>
</tr>
<tr>
<td>CSMA/CD</td>
<td>Carrier Sense Multiple Access /Collision Detection</td>
</tr>
<tr>
<td>MDS</td>
<td>Minimum Dominating Set</td>
</tr>
<tr>
<td>MCDS</td>
<td>Minimum Connected Dominating Set</td>
</tr>
<tr>
<td>MWCDS</td>
<td>Minimum Weighted Connected Dominating Set</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

A sensor network may contain a huge number of simple sensor nodes that are deployed at some inspected site. In large areas, the sensor network usually has a mesh structure. In this case, some of the sensor nodes act as routers, forwarding messages from one of their neighbors to another.

To minimize idle listening, which is the main source for energy waste [3], the sensors turn their communication hardware on and off. Energy expenditure is commonly assumed to be governed not by the amount of transmitted data but by the time the sensors spend in active mode [4, 5, 6]. This is true, for example, in alarm sensor networks, where the traffic is very low while the latency requirements are high. Thus, there is a clear tradeoff between energy expenditure and the end-to-end delay. The energy expenditure and the end-to-end delay are both governed by the duty cycle of the nodes, which is defined by their wake-up frequency.

In order for two neighboring sensors to communicate, both must be active at the same time. There are two main approaches to scheduling simultaneous wake-up of neighboring sensors: global or local synchronization[7, 8, 9, 10]. Since global synchronization is inefficient and very difficult to achieve in big networks[8, 9, 10], we assume that sensor wake-ups are only synchronized locally. That is, each node selects its wake-up schedule and informs neighboring nodes about its selection. A node that needs to send a packet through a neighbor must wake up and transmit it during the neighbor’s duty cycle. This communication model imposes a clear tradeoff between the delay encountered by a packet routed along the sensor network and the time during which the sensors along the route are in active mode. This tradeoff was studied in several works[11, 12, 13, 6]. Solutions for addressing this trade-off depend, to a large extent, on the specific sensor network model [14], and in particular on the following aspects:

1. The data delivery model, i.e., whether data is delivered continuously by the sensor, or deliv-
ered only after an event of interest.

2. The expected amount of data to be delivered.

3. The routing scheme: whether, for example, a single route is used between each source and the gateway or multiple routes are concurrently employed, or whether routes are selected by the traversed nodes or by the sources (“source routing”).

4. Whether the intermediate nodes process the packets they receive in order to merge similar observations from different sources or forward them as is.

With respect to (1) and (2), if the data is continuously delivered, or if there is a lot of it, an algorithm for determining the wake-up times of each intermediate sensor has to take into account the expected amount of traffic passing through the sensor and the expected receiving times. With respect to (3), if the routing algorithm is flexible enough, it can determine the route to be traversed by each packet according to the expected wake-up times of the intermediate nodes. Finally, with respect to (4), if an intermediate node can process the received data packets, it may choose to delay them until they can be merged with other packets.

In the considered sensor networks model, the sensor nodes are placed randomly over the area of interest and their first step is to detect their immediate neighbors, i.e. the nodes with which they have a direct wireless communication and to establish routes to the gateway. When the amount of traffic generated in a sensor network is permanently high, a sensor needs to invoke no special neighbor discovery protocol during its normal operation. This is because any new node, or a node that has lost connectivity to its neighbors, can listen to the channel for a short time and hear its neighbors. However, for sensor network with low and irregular traffic, a special neighbor discovery protocol should be used.

The rest of the thesis is organized as follows. In Chapter 2 we present related work. In Chapter 3 we present an algorithm for maximizing the lifetime of a sensor network while guaranteeing an upper bound on the end-to-end delay. We prove that the proposed algorithm is optimal, and that it requires simple computing operations that can be implemented by simple devices. We assume that a single route is used from each sensor to the gateway. We present an algorithm for determining the wake-up frequency of the nodes in a sensor network. This algorithm minimizes the energy consumption of the nodes and bounds the maximum delay on the routes from the nodes to the gateway.

In Chapter 4 we extend the results of the previous chapter in two directions. Instead of solving the sensor wake-up scheduling problem for a single routing tree rooted at a single gateway, we
solve it for multiple trees rooted at one or more gateways. The benefits of using multiple trees are clear: (a) faster recovery from gateway failures and (b) load sharing. We also extend the results of Chapter 3 with regard to the routing tree, which is was assumed there to be given in advance. Here, however, the tree building problem is addressed from the perspective of the optimal wake-up frequency assignment. In other words, the algorithm for building the tree takes into account the optimization criterion imposed by the optimal wake-up frequency assignment problem.

In Chapter 5 we expose a new problem in wireless sensor networks, referred to as ongoing neighbor discovery. We argued that continuous neighbor discovery is essential even if the sensor nodes are static. We showed that by having the nodes in a connected segment work together on continuous neighbor discovery, we can guarantee that (a) hidden nodes will be detected with a certain probability $P$ and within a certain time period $T$; and that (b) the energy expended by the segment nodes on the detection of hidden nodes is minimized.

Chapter 6 concludes the work.
Chapter 2

Related Work

While many papers have been written on how to minimize energy consumption in sensor networks, very few have explicitly addressed the trade-off between delay and energy. To the best of our knowledge, our work is the first to propose the assignment of different wake-up frequencies to nodes according to their role in the packet forwarding process. However, the energy-latency tradeoff has been thoroughly studied in sensor networks, as well as in other wireless networks. In this section we present related works, and compare their models and results with ours.

In the S-MAC protocol [11], packet latency caused by periodic sleeping of intermediate nodes is minimized by synchronizing the wake-up schedules of neighboring nodes. The duty cycles of all nodes are equal and predefined. The protocol is not intended to guarantee an upper bound on the end-to-end delay, but to minimize the energy consumption of the nodes.

Another paper [12] proposes to minimize the delay using special scheduling of the nodes’ wake-up periods. This paper extends another work [15] by the same authors, where the nodes are organized in an unidirectional tree. However, in [12] the authors assumption of arbitrary communication patterns renders the problem NP-Complete. The authors propose algorithms that find an optimal solution for specific topologies, such as trees and rings. They also show that their algorithms can be used as heuristics for general graphs.

In [16], the authors address the trade-off between delay and energy in sensor networks from a different viewpoint. They search for an optimal routing path from a source node to the gateway, such that latency is minimized and energy cost is not “too big.” In their network model, sensors randomly switch between sleep and active states. Two alternatives are studied: a centralized global optimization approach and a distributed approach.

Energy efficiency can be achieved in different ways. For example, energy aware routing finds
a routing path while taking into account energy cost and the sensor’s available energy. In [17] the number of hops along the forwarding path is considered, while keeping in mind that transmission between close nodes is more energy efficient, even if the resulting route is longer. The authors use a random network model to show the energy-latency-throughput dependency and to find the optimal transmission power for nodes in an ad-hoc network. As already indicated, we do not address the routing issues in our work. The scheme proposed in [17], as well as many others, can be used for this purpose.

In [13], the tradeoff between energy and latency is investigated using probabilistic computation. The authors consider a network of nodes that switch from passive to active mode independently, but with a predefined frequency. The packets are not forwarded on predefined routes, as they are in our model, but are sent instead to all neighbors in active mode. Therefore, the network density and the duty cycle should be high enough to ensure that each packet will finally reach its destination. A probabilistic analysis finds the portion of time each node is required to be in an active state in order to ensure that the packet is delivered to the gateway on time. This model differs from ours in that our model does not use flooding and assumes that nodes are aware of their neighbor’s duty cycles.

The tradeoff between energy and latency in general wireless networks was also studied in a different context. For example, [18] and [19] investigate this tradeoff when a technique called “modulation scaling” is used. The authors base their work on the observation that, in many coding schemes, the transmission of a packet requires a smaller amount of energy if it lasts longer. They solve the problem of finding an optimal transmission schedule for a node, given that it has to forward a random number of packets whose arrival times follow the Poisson distribution. The optimization criterion is to minimize the overall energy consumption and bound the maximum delay. Two algorithms are proposed: an off-line algorithm that finds an optimal solution, and an on-line algorithm that approximates the optimal solution. This approach is taken further in [20]. The authors deal with a more general setting, assuming that each packet may have a different deadline and number of bits.

In [21], this problem is generalized by considering an aggregation tree with packets routed along the tree to the root. As in [18], the energy cost of a packet transmission is a decreasing convex function of its transmission time. The cost is different for each node because of the different amounts of data to be forwarded. The packet should be delivered to the sink within a limited time period. The authors propose an off-line algorithm for an optimal solution whose running time complexity is unknown, and an approximation algorithm with pseudo-polynomial running time that needs to know the network topology. Although our model is different, the considered prob-
lem is similar to ours. In our work, by making some assumption on the energy-latency dependency, we propose an optimal algorithm with linear complexity.

LEACH protocol[22] presumes that the sensors are able to change their transmitting power in order to build a better communication graph. The proposed solution combines the sensors into local clusters and allows only the cluster heads to contact the gateway. Since a cluster head expends much more energy than any other node, this role is periodically rotated between the cluster nodes.

The tradeoff between energy and latency appears also when data-aggregation techniques are applied. On the one hand, it is better to merge several packets reporting the same event into a single one. Doing so, on the other hand, increases the delay.

In the present work a sensor network with multiple gateways is also studied. Two operation models can be considered for multiple gateway networks. In the first model, each sensor is associated with one gateway, whereas in the second model every node can send data to any gateway. The first model is studied in [23], where a sensor chooses one of the gateways while taking into account the energy efficiency of the routing, energy resources of the intermediate sensors, and load balancing. The same model is also considered in [24], where the authors propose several algorithms for intelligent gateway placement in order to reduce latency and save energy.

The second model is studied in the family of SPIN adaptive protocols [25]. In these protocols a sensor broadcasts a special “advertisement” with a description of the data it has, and an interested gateway returns a request for this information. Another algorithm that addresses sensors communicating with multiple gateways is SAR [26]. This algorithm creates multiple trees, where each root is one hop away from the gateway and most of the nodes belong to multiple trees.

The neighbor discovery problem is also a well known problem discussed in the context of both ad-hoc and sensor networks. In a WiFi network operating in centralized mode, a special node, called an access point, coordinates access to the shared medium. Messages are transmitted only to or from the access point. Therefore, neighbor discovery is the process of having a new node detected by the base station. Since energy consumption is not an issue for the base station, discovering new nodes is rather easy. The base station periodically broadcasts a special control message, referred to in the following to as HELLO\(^1\). A regular node that hears this message can initiate a registration process. The regular node can switch frequencies/channels in order to find the best HELLO message for its needs. Which message is the best might depend on the identity of the broadcasting base station, on security considerations, or on PHY layer quality (signal-to-noise

\(^1\)The various systems and protocols that employ neighbor discovery use different names for their control message, such as BEACON or NEIGHBOR-DISCOVERY. For consistency, throughout this work, we refer to all these control messages as HELLO.
ratio). Issues related to possible collisions of registration messages in such a network are addressed in [27]. Other works try to minimize neighbor discovery time by optimizing the broadcast rate of the HELLO messages [28, 29, 30, 31, 32]. The main differences between neighbor discovery in WiFi and in mesh sensor networks are that in WiFi neighbor discovery is performed only by the central node, which is not concerned of energy consumption. In addition, the hidden nodes are assumed to be able to hear the HELLO messages broadcast by the central node. In contrast, neighbor discovery in sensor networks is performed by all nodes and hidden nodes cannot hear the HELLO messages when they sleep.

In mobile ad-hoc networks (MANETs), nodes usually do not switch to a special sleep state. Therefore, two neighboring nodes can send messages to each other whenever their physical distance allows communication. AODV [33] is a typical routing protocol for MANETs. In AODV, when a node wishes to send a message to another node, it broadcasts a special RREQ (route request) message. This message is then broadcast by every node that hears it for the first time. The same message is used for connectivity management, as part of an established route maintenance. There is no special neighbor discovery protocol besides that.

Minimizing energy consumption is an important target design in Bluetooth [34]. As in WiFi, the process of neighbor discovery in Bluetooth is also asymmetric. A node that wants to be discovered switches to an inquiry scan mode, whereas a node that wants to discover its neighbors enters the inquiry mode. In the inquiry scan mode, the node listens for a certain period on each of the 32 frequencies dedicated to neighbor discovery, while the discovering node passes through these frequencies one by one and broadcasts HELLO in each of them. This process is considered to be energy consuming and slow. A symmetric neighbor discovery scheme for Bluetooth is proposed by [35]. The idea is to allow each node to switch between the inquiry scan mode and the inquiry mode.

The 802.15.4 standard [36] proposes a rather simple scheme for neighbor discovery. It assumes that every coordinator node issues one special “beacon” message per frame, and a newly deployed node has only to scan the available frequencies for such a message. However, the standard also supports a beaconless mode of operation. Under this mode, a newly deployed node should transmit a beacon request on each available channel. A network coordinator that hears such a request should immediately answer with a beacon of its own. However, this scheme does not supply any bound on the hidden neighbor discovery time.

Neighbor discovery in wireless sensor networks is addressed in [37]. The authors propose a policy for determining the transmission power of every node, in order to guarantee that each node detects at least one of its neighbors using the minimum possible power.
In [28], the authors study the problem of neighbor discovery in static wireless ad hoc networks with directional antennas. At each time slot, a sensor node either transmits HELLO in a random direction, or listens for HELLO messages from other nodes. The goal is to determine the optimal rate of transmission slots and reception slots, and the pattern of transmission directions.

In [30], neighbor discovery is studied for general ad-hoc wireless networks. The authors propose a random HELLO protocol, inspired by ALOHA. Each node can be in one of two states: listening or talking. A node decides randomly when to initiate the transmission of a HELLO message. If its message does not collide with another HELLO, the node is considered to be discovered. The goal is to determine the HELLO transmission frequency, and the duration of the neighbor discovery process.

In [29], the sensor nodes are supposed to determine, for every time slot, whether to transmit HELLO, to listen, or to sleep. The optimal transition rate between the three states is determined using a priori knowledge of the maximum possible number of neighbors.

In [38], the Disco algorithm is proposed for scheduling the wake-up times of two nodes that wish to find each other. This paper proposes that each node would choose a prime number, depending on the required discovery time. Using the Chinese Remainders theorem, it is proved that the wake-up periods of the nodes will overlap within the required time. However, [38] does not discuss the issue of many sensors in the same segment collaborating in order to reduce the energy they expend for discovering hidden nodes. In Section 5.4 we show how the Disco algorithm can be used by our new scheme.

As discussed in Section 5.1, the nodes of the sensor networks spend most of their time in sleep/idle mode, where they cannot receive or transmit messages. Therefore, the ability of a node to discover a new neighbor is limited to periods of time where both are active. In [39], this neighbor discovery model is shown to be similar to the well-known “birthday paradox.” In our work we use the similar analysis, in order to find the probability of a node to get discovered by one of its neighbors.

A novel low power listening (LPL) technique is proposed in [40] to overcome sensor synchronization problems. This technique is implemented by the B-MAC protocol [41]. Using this technique the transmission of a packet is preceded by a special preamble. This preamble is long enough to be discovered by a periodic channel sampling, performed by every node. However, this technique can be hardly used for the initial neighbor discovery, and can not be used for the continues neighbor discovery, because this actually requires the node to stay awake all the time while looking for a new neighbor.
Chapter 3

An Optimal Wake-Up Scheduling Algorithm for Minimizing Energy Consumption while Limiting Maximum Delay in a Mesh Sensor Network

3.1 Introduction

In this chapter we consider a sensor network for alarm events that are generated by the network’s nodes and relayed to the network’s gateway. For example, consider sensor networks for detecting smoke, or locally measured temperature that exceeds some threshold, or water level that exceeds some threshold. The most important property of such networks, with respect to this chapter, is the existence of an upper bound on the time from when an alarm event is generated to the time the gateway must be informed about it. We present an algorithm for maximizing the lifetime of such networks while guaranteeing an upper bound on the end-to-end delay. We prove that the proposed algorithm is optimal, and that it requires simple computing operations that can be implemented by simple devices.

With respect to routing, we assume that a single route is used from each sensor to the gateway. This implies that the routes from the nodes to the gateway define a directed tree, rooted at the sensor network’s gateway. This is a conventional approach in general data networks and sensor networks in particular, where selecting different routes for different flows originated at the same sensor is impractical. We make no assumptions regarding how the routes are selected. In par-
ticular, the schemes proposed in [42, 43] are all applicable. Moreover, in our simulation model (Section 3.5), one of the routing algorithms we consider routes only via nodes with sufficient energy. Following this assumption, each sensor node has a designated neighbor (parent) to which it forwards packets destined for the gateway. These packets are created either by the node itself or by some other, downstream, node. At any given time, the set of wireless links between every node and its preferred neighbor forms a virtual spanning tree. We make no assumption regarding this collection of links, and of course it may change dynamically, e.g., due to topological changes in the network.

The rest of this chapter is organized as follows. In Section 3.2 we describe and justify the power consumption model considered throughout this chapter. Since the proposed algorithm is complex, we present it in two steps. In Section 3.3, we present the basic algorithm, which determines the wake-up time for every sensor such that the total energy consumed by the network is minimized and the end-to-end delay is upper-bounded. However, such an algorithm does not necessarily extend the lifetime of the network because the variance of the assigned frequencies can be too high. This basic algorithm is therefore used in Section 3.4 as a procedure for the ultimate algorithm, which minimizes the total energy consumed by the nodes while ensuring an upper bound on both the maximum delay and on the energy consumed by every individual sensor. Finally, in Section 3.5 we present simulation results for the new algorithm, and show that the proposed algorithm can almost double the lifetime of a sensor network.

### 3.2 Power Consumption Model

In order to minimize power consumption, nodes remain in a sleep mode most of the time, while adhering to the following two simple rules:

(R1) An inner node $v$ in the virtual routing tree, i.e., a node that serves as a parent to at least one of its neighbors, must wake up periodically, in order to receive packets from its children. These children know the times when $v$ switches from sleep to active. The node stays active as long as it receives packets from its neighbors. After a time-out period of not receiving any packet, the node returns to sleep mode.

(R2) Every node also wakes up when its parent wakes up if and only if it needs to forward a packet through the parent to the gateway.

Precise synchronization between neighboring nodes is unnecessary. If the nodes use a CSMA/CA-like MAC protocol in order to send data packets, a packet that is not ACKed due to non-perfect syn-
chronization will be re-transmitted after a short time-out period. This energy expenditure model is formalized as follows. If a node $v$ wakes up $f_v$ times per second, its energy consumption is equal to $f_v \cdot c_v + b_v$ Watt, where

- $c_v$ is the average amount of energy (Watt) consumed during every wake-up period, including charging the registers, listening, receiving packets and transmitting ACKs.
- $b_v$ is a constant amount of energy, expended on monitoring the environment, performing internal calculations, managing the clock, and forwarding packets to the next node along the route to the gateway; we say that $b_v$ is constant for node $v$ because it does not depend on the wake-up frequency $f_v$ of $v$.

The justification for our energy consumption model comes from the major difference between the energy consumed by the sensor radio for transmitting/receiving/listening (i.e., active mode) and for sleeping (i.e., idle mode). For instance, the CC2420 device of Chipcon [44, 45, 46], which is 802.15.4-compatible, requires 52.2mW for transmitting, 56.4mW for receiving or listening, and only 3µW for sleeping. A typical duty cycle of such a sensor is 0.1%. For instance, this is the duty cycle of a sensor that wakes up once every 10 seconds ($f_v = 0.1$) for 10 millisecond. The difference between the energy required for receiving/listening (56.4 mW) and transmitting (52.2 mW) is negligible compared to the difference between receiving/listening/transmitting (52.56 mW) and sleeping (3µW). Hence, while the actual energy consumed by the sensor is 0.001 $\cdot$ 56 + 0.999 $\cdot$ 0.003 + $b_v$ mW, it can be very well approximated by 0.001 $\cdot$ 56 + 0.003 + $b_v$ mW. Thus, by incorporating the fixed part of the energy spent in the idle mode (0.003 mW) into $b_v$, we get that the average consumed energy is indeed $f_v \cdot c_v + b_v$ mW, where $b_v$ does not depend on the wake-up frequency $f_v$ of $v$.

Figure 3.1 depicts the considered communication model for the case where sensor $v$ sends a packet to the gateway $w$ via sensor $u$. Node $v$ wakes up during the duty cycle of node $u$, and sends the packet to this node. The energy it expends for transmitting the packet to $u$ is attributed to its $b_v$ factor, whereas the energy expended by $u$ is attributed to the $f_u \cdot c_u$ factor. Node $u$ wakes up during the duty cycle of $w$ and forwards the packet to $w$. The energy it expends for transmitting the packet is attributed to its $b_u$ factor. Therefore, the end-to-end delay encountered by this packet is the time node $v$ needs to wait until $u$ wakes up, plus the time node $u$ needs to wait until node $w$ wakes up. This time is upper bounded by $1/f_u + 1/f_w$.

Consider the virtual routing tree from the nodes (sensors) to the gateway. Following R1 above, only an inner node $u$ in this tree is assigned a constant wake-up frequency of $f_u$. A leaf $v$ of this tree wakes up only when it needs to send a packet towards the gateway via its parent. Hence, for
such a node, $f_v = 0$ holds. The energy expended by such a node is attributed to its $b_u$ component only. Since we focus on determining the wake-up frequencies of the inner nodes, we shall ignore the leaves of the routing tree. Formally, Let $G = (V, E)$ be a network graph, where $G$ is the set of sensors and the gateway, and $E$ is the set of edges between nodes that can communicate directly. Let $T$ be the routing tree that consists of all the routing paths from the sensors to the gateway without the leaves and their edges towards their parents. For the rest of the chapter we shall address the allocation of wake-up frequencies to the nodes of $T$, assuming that for the other nodes the wake-up frequency is 0. Figure 3.2(a) gives an example for a network with its routing tree. Figure 3.2(b) shows the corresponding $T$-tree, which is subject to our optimizations, for a network with its routing tree.

Following our model, we conclude that energy depletion is not governed by the amount of transmitted data, but by the time the sensors spend in active mode. Therefore, this chapter deals with the trade-off between this frequency and the maximum end-to-end delay in the network. Of course, a sensor node also uses its energy for other tasks, such as computation and sensing. However, it is well known that this energy is marginal compared to that spent in active mode [3].
3.3 Step 1: Minimizing the total consumed energy

Recall our energy expenditure model from Section 3.1, where node \( v \) wakes up \( f_v \) times per second, and its energy consumption is \( f_v \cdot c_v + b_v \). Since \( b_v \) does not depend on the wake-up frequency of \( v \), it is not subject to optimization. Therefore, for the rest of this chapter we consider only the first factor.

**Definition 1** An optimal assignment of wake-up frequencies to nodes is an assignment that guarantees an upper bound \( D \) on the maximum delay while minimizing the total energy spent by the nodes in active mode.

We now formalize this definition as an optimization problem:

**Problem 1** Let \( T \) be a tree that consists of all the paths from the sensors to the gateway without the leaves. Let \( f_v \) and \( c_v \) be the wake-up frequency and the average energy consumption during each wake-up for node \( v \), respectively. Then, we want to minimize:

\[
\sum_{v \in T} f_v \cdot c_v
\]

subject to:

- for every directed path \( v_1 \to \cdots \to v_N \) in \( T \), where \( v_N \) is the gateway,
  \[
  \sum_{v \in \{v_1, \ldots, v_N\}} \frac{1}{f_v} \leq D.
  \]
- for every node \( v \) in \( T \): \( f_v > 0 \).

Note that we can ignore every non-maximal path, which is a sub-path of a longer path, and consider only the paths from the leaves of \( T \) to the gateway.

Although the target function is linear and the problem is convex, the constraints are nonlinear. However, we will make a few observations that allow us to find the optimal solution in linear time. We start by showing that under any optimal solution, the constraint for each maximal directed path is tight.

**Claim 1**

(a) In an optimal assignment of wake-up frequencies to nodes, the delays on the paths from the leaves of \( T \) to its root are equal.
In an optimal assignment, the delay on each path from the leaves to the root is equal to the maximum tolerated delay.

Proof: To prove (a), assume, by contradiction, that we have an optimal wake-up frequency assignment where there are two leaves, \(v_1\) and \(v_2\), such that the maximum delay on path \(P_1\) from \(v_1\) to the root is smaller than the maximum delay on path \(P_2\) from \(v_2\) to the root. By reducing the wake-up frequency of \(v_1\), we can equate the delay on \(P_1\) to the delay on \(P_2\). The new assignment reduces the total energy consumption without affecting the maximum delay, contradicting our assumption. The proof of (b) is similar.

Suppose that by assigning the values \(a_1, a_2, \ldots, a_n\) to the variables \(f_1, f_2, \ldots, f_n\) of Problem 1, we guarantee an upper bound \(D'\) on the delay. Then, by assigning the values \(\alpha a_1, \alpha a_2, \ldots, \alpha a_n\), we guarantee an upper bound of \(\frac{D'}{\alpha}\). Therefore, the problem can be solved in the following way. First, find the minimum delay for which \(\sum f_v c_v = E\) holds. Let this delay be \(D'\). Then, multiply the values assigned to \(f_1, f_2, \ldots, f_n\) by \(\alpha\), where \(\frac{D'}{\alpha}\) is equal to the target maximum delay \(D\).

To solve the former problem, let \(T(v)\) denote the sub-tree rooted at \(v\), \(D(v)\) denote the maximum delay on this sub-tree, and \(E(v)\) denote the amount of energy assigned to \(T(v)\). In addition, let \(e(v)\) denote the amount of energy assigned to node \(v\) itself. Since \(c_v\) denotes the average energy cost of each wake-up of node \(v\), \(e(v) c_v\) is the wake-up frequency of \(v\).

We now present an algorithm that, when given a certain amount of energy \(E\) and a node \(v\), divides this energy among the nodes in \(T(v)\) such that the maximum delay encountered on this sub-tree, \(D(v)\), is minimized. The algorithm starts at the leaves and propagates upstream towards the root in the following way. For a leaf \(v\), the value of \(D(v)\) as a function of \(E(v)\) is simply \(D(v) = c_v / E(v)\). Now, consider a node \(v\) whose children in the tree are \(u_1, u_2, \ldots, u_N\). Suppose that for every \(u_i\) we know the value of \(D(u_i)\) as a function of \(E(u_i)\). From Claim 1(a) we know that in an optimal assignment all the paths from the leaves of \(T(v)\) to \(v\) should have an equal delay. We therefore have the following set of \(N\) equations:

\[
D(u_1) = D(u_2) = \ldots = D(u_N) \quad (3.1)
\]

\[
e(v) + \sum_{i=1}^{N} E(u_i) = E(v). \quad (3.2)
\]

The conditions expressed by these equations are necessary, but not sufficient, since we also have to minimize \(D(v)\). This delay is equal to the delay introduced by \(v\) plus \(D(u_i)\), for any \(1 \leq i \leq N\). Therefore, and without loss of generality, we have \(D(v) = \frac{e(v)}{c_v} + D(u_1)\). To find the value of \(e_v\) that minimizes \(D(v)\) for a given value of \(D(u_1)\), we differentiate \(D(v)\) with respect to \(e(v)\). This
yields the following additional equation:

\[ \frac{d(\frac{c_v}{e(v)} + D(u_1))}{d(e(v))} = 0. \]  \hfill (3.3)

Since the value of \( D(u_i) \) for every \( 2 \leq i \leq N \) is an explicit function of \( E(u_i) \), we get \( N + 1 \) equations with \( N + 1 \) variables: \( e(v), E(u_1), E(u_2), \ldots, E(u_N) \). This set of equations is solved in order to divide \( E(u) \) among \( u_1 \ldots u_N \), while guaranteeing the minimum delay.

When the algorithm finishes, and the equations are solved for each node, including the root \( r \), we know the value of \( D(r) \) as a function of \( E(r) \). In order to solve Problem 1, we set \( D(r) \) to be the maximum tolerated delay and find the minimum overall energy \( E(r) \) that guarantees this maximum.

**Claim 2** The relationship between \( D(v) \) and \( E(v) \) is of the following form:

\[ D(v) = \frac{1}{B_v E(v)}. \]  \hfill (3.4)

\( B_v \) is a constant that depends only on \( c_v \) and on \( c_{u_1} \ldots c_{u_N} \), where \( u_1 \ldots u_N \) are the children of \( v \) in \( T \).

**Proof:** We prove this by induction on the height of the sub-tree of \( v \). If the height is 1, i.e., \( v \) is a leaf, \( D(v) = \frac{c_v}{E(v)} \). Consider now an inner node with \( N \) children, \( u_1, u_2, \ldots, u_N \). By the induction assumption, the delay function for each child of \( v \) is \( D(u_i) = \frac{1}{B_{u_i} E(u_i)} \). Therefore, the system of equations to be solved in order to determine how to divide \( E(v) \) between \( v \) and \( u_1 \ldots u_N \) is:

\[ \frac{1}{B_{u_1} E(u_1)} = \frac{1}{B_{u_i} E(u_i)}, \text{ for every } 2 \leq i \leq N, \]  \hfill (3.5)

\[ e(v) + \sum_{i=1}^{N} E(u_i) = E(v), \]  \hfill (3.6)

\[ \frac{d(\frac{c_v}{e(v)} + \frac{1}{B_{u_1} E(u_1)})}{d(e(v))} = 0. \]  \hfill (3.7)

Substituting Eq. 3.5 into Eq. 3.6 yields

\[ e(v) + \sum_{i=1}^{N} \frac{B_{u_i} E(u_1)}{B_{u_i}} = E(v). \]
Denoting $B = \sum_{i=1}^{N} \frac{B_{u_i}}{B_{u_i}}$, we have

$$E(u_1) = \frac{E(v) - e(v)}{B}. \quad (3.8)$$

Substituting Eq. 3.8 into Eq. 3.7 yields

$$d\left(\frac{c_v}{e(v)} + \frac{B}{(E(v) - e(v))B_{u_1}}\right) = 0 \quad (3.9)$$

or equivalently

$$-\frac{c_v}{e(v)^2} + \frac{B}{(E(v) - e(v))^2 \cdot B_{u_1}} = 0. \quad (3.10)$$

Therefore,

$$B \cdot e(v)^2 - (E(v) - e(v))^2 \cdot B_{u_1} c_v = 0,$$

and

$$(B - B_{u_1} c_v) e(v)^2 + 2B_{u_1} c_v E(v) e(v) - B_{u_1} c_v E(v)^2 = 0.$$

In those cases where $B = B_{u_1} c_v$, we get

$$e(v) = \frac{B_{u_1} c_v E(v)^2}{2B_{u_1} c_v E(v)} = \frac{1}{2} E(v). \quad (3.11)$$

Otherwise,

$$e(v) = E(v) - \frac{B_{u_1} c_v \pm \sqrt{(B_{u_1} c_v)^2 + (B - B_{u_1} c_v) B_{u_1} c_v}}{B - B_{u_1} c_v},$$

thus,

$$e(v) = E(v) - \frac{B_{u_1} c_v \pm \sqrt{B \cdot B_{u_1} c_v}}{B - B_{u_1} c_v}.$$
Substituting the definition of $B$,

$$e(v) = E(v) - B_u c_v \pm \frac{\sum_{i=1}^{N} \frac{B_u}{B_{u_i}} B_{u_i} c_v}{\sum_{i=1}^{N} \frac{1}{B_{u_i}} - B_u c_v},$$

$$e(v) = E(v) - c_v \pm \sqrt{\frac{\sum_{i=1}^{N} c_v B_{u_i}}{\sum_{i=1}^{N} \frac{1}{B_{u_i}} - c_v}}.$$

Since $e(v)$ must be positive and smaller than $E(v)$, the final solution will be:

$$e(v) = E(v) - c_v + \sqrt{\frac{\sum_{i=1}^{N} c_v B_{u_i}}{\sum_{i=1}^{N} \frac{1}{B_{u_i}}}}.$$

We can rewrite the solution such that $e(v) = B'_v E(v)$, where $B'_v$ is a constant number. Therefore $E(u_1) = \frac{E(v) - B'_v E(v)}{B} = B'_v E(v)$ (from Eq. 3.8), and the delay on the sub-tree is indeed

$$\frac{1}{e(v)} + \frac{1}{B_{u_1} E(u_1)} = \frac{1}{B'_v E(v)} + \frac{1}{B_{u_1} B'_v E(v)} = \frac{B_{u_1} B'_u + B'_v}{B'_v B_{u_1} E(v)} = \frac{1}{B_v E(v)},$$

(3.13)

A formal description of the algorithm to be executed is presented further (Algorithm 1). The algorithm consists of two functions: Calculate-Frequency-Division($v$) and Assign-Frequencies($v$, energy). The first function receives a node identifier $v$ and returns the delay $D(v)$ on the sub-tree of $v$ as a function of the amount of energy $E(v)$ assigned to this sub-tree. This function performs a recursive call for each node in $v$’s sub-tree, in order to find its sub-tree delay. Since the energy-delay dependency has a well-defined form $D(v) = \frac{1}{B_v E(v)}$, this function returns only the coefficient $B_v$.

In addition, this function computes the partition of the energy assigned to the sub-tree of $v$ between node $v$ and each of its children’s sub-trees: $e(v), E(u_1), \ldots, E(u_N)$. Afterwards, the overall tree energy, $E(root)$, is computed. In order to solve Problem 1, the algorithm sets $D(root)$ to be the maximum tolerated delay, and computes $E(root)$ using the equation $E(root) = \frac{1}{B_{root} D(root)}$. The second function, Assign-Frequencies($v$, energy), performs the actual energy assignment. This function receives a node identifier and the amount of energy available to the sub-tree of this node, and calculates the node’s wake-up frequency using the value of $e(v)$ as found by Calculate-Frequency-Division. It then performs a recursive call for the node’s children, using the values
Algorithm 1 For a given value of maximum delay, this algorithm determines the wake-up frequency for every node such that the overall energy is minimized.

1. Calculate the optimal wake-up frequency assignment by executing function \textit{Calculate-Frequency-Division}(root).

2. Find \( E(root) \)

3. Calculate the precise wake-up frequencies of all the nodes by calling \textit{Assign-Frequencies}(root, E(root)).

Function \textit{Calculate-Frequency-Division}(v)

The function receives a node identifier \( v \). It considers one unit of energy and returns the value of \( B_v \) in Eq. 3.4. This value determines the ratio of \( E(v) \) to \( D(v) \). As a secondary result, the function calculates the energy division between the node and its children sub-trees, \( e(v), E(u_i) \).

\[
\text{if } v \text{ is a leaf} \\
\quad e(v) = 1 \quad \text{/* assign the whole unit of energy */} \\
\quad \text{return } (\frac{1}{c_v}) \quad \text{/* for a leaf: } D(v) = \frac{c_v}{E(v)} \text{ */}
\]

\[
\text{else} \\
\text{for every } u_i \in \text{children}(v) \\
\quad B_{u_i} = \text{Calculate-Frequency-Division}(u_i) \\
\quad \text{if } \sum_{i=1}^{n} \frac{1}{B_{u_i}} = 1 \quad \text{/* this means } B = B_{u_1} */ \\
\quad e(v) = \frac{1}{2} \quad \text{/* from Eq. 3.11 */}
\]

\[
\text{else} \\
\quad e(v) = \frac{-c_v + \sqrt{\sum_{i=1}^{n} \frac{B_{u_i}}{B_{u_1}} - c_v}}{\sum_{i=1}^{n} \frac{B_{u_i}}{B_{u_1}}} \quad \text{/* from Eq. 3.12 */}
\]

\[
E(u_1) = \frac{1-e(v)}{\sum_{i=1}^{n} \frac{B_{u_i}}{B_{u_1}}} \quad \text{/* from Eq. 3.8 */}
\]

\[
\text{for every } u_i \in \text{children}(v) \\
\quad E(u_i) = \frac{B_{u_i}E(u_1)}{B_{u_1}E(u_1) + e(v)} \quad \text{/* from Eq. 3.5 */}
\]

\[
\text{return } \frac{e(v)B_{u_1}E(u_1)}{B_{u_1}E(u_1) + e(v)} \quad \text{/* from Eq. 3.13 */}
\]

Function \textit{Assign-Frequencies}(v, energy)

This function receives a node identifier and the energy assigned to its sub-tree and divides the energy between the node and its children sub-trees:

\[
f(v) = \text{energy} \cdot e(v)
\]
for every \( u_i \in \text{children}(v) \)

Assign Frequencies\((u_i, \text{energy} \cdot E(u_i))\)

\[ \text{Theorem 1} \quad \text{Algorithm 1 finds the optimal energy assignment.} \]

\[ \text{Proof:} \quad \text{Assume, by contradiction, that there exists an optimal assignment } A^*, \text{ which achieves smaller delay than assignment } A \text{ as calculated by Algorithm 1, with the same overall energy. Let } e(v) \text{ and } e^*(v) \text{ denote the wake-up frequency of node } v \text{ under } A \text{ and } A^* \text{ respectively. Let } E(v) \text{ and } E^*(v) \text{ be the overall energy of the sub-tree rooted at } v \text{ under } A \text{ and } A^* \text{ respectively.} \]

Since \( A \neq A^* \), there must exist a node \( u \) such that \( \frac{e(u)}{E(u)} \neq \frac{e^*(u)}{E^*(u)} \). Because for every leaf \( v \), \( \frac{e(v)}{E(v)} = 1 = \frac{e^*(v)}{E^*(v)} \) (all the energy assigned to the sub-tree of a leaf is used only by the leaf), there must exist a node \( u \) such that \( \frac{e(u)}{E(u)} \neq \frac{e^*(u)}{E^*(u)} \), while for all the descendants of \( u \) the two ratios are equal. Let us consider the delay on the sub-trees of the children of \( u \): \( w_1 \ldots w_N \). For each \( w_i \), the delay has the same form as in Eq. 3.4: \( D^*(w_i) = \frac{1}{B_{w_i} \cdot E^*(w_i)} \) under assignment \( A^* \) and \( D(w_i) = \frac{1}{B_{w_i} \cdot E(w_i)} \) under assignment \( A \). Note that the constant \( B_{w_i} \) is the same for both assignments, because it depends on the division of energy between the nodes in \( T(u) \) and not on the actual amount of energy assigned to each node. Recall that \( u \) was chosen such that \( \frac{e(v)}{E(u)} = \frac{e^*(v)}{E^*(u)} \) holds for every descendant \( v \) of \( u \).

According to Claim 1(a), for any two children of \( u \), \( w_i \) and \( w_j \), the delays on their sub-trees are equal: \( D^*(w_i) = D^*(w_j) \). Therefore, the delay on the sub-tree of \( u \) under \( A^* \) is \( D^*(u) = \frac{e^*(u)}{E^*(u)} + \frac{1}{B_{w_i} \cdot E^*(w_i)} \) for every \( i \). Hence, using the equality from Eq. 3.8, we can represent the delay on \( T(u) \) under \( A^* \) as a function of \( e^*(u) \), when \( E^*(u) \) is given as a parameter. But this is exactly the dependency between the delay under \( A \) and the wake-up frequency \( e(u) \) when \( E(u) \) is given as a parameter. As shown in the proof of Claim 2, the delay is minimal for \( e(u) = E(u) \cdot B'_u \). Therefore, when the sub-tree energy is equal to \( E^*(u) \), the delay is minimal if the wake-up frequency of \( u \) is \( e^*(u) = E^*(u) \cdot B'_u = E^*(u) \cdot \frac{e(u)}{E(u)} \). Any other energy division between node \( u \) and its children sub-trees will cause a greater delay. Therefore, assigning node \( u \) this wake-up frequency and adjusting the energies of its children sub-trees respectively will cause the delay on the sub-tree of \( u \) to decrease below \( D^*(u) \), while the overall sub-tree energy remains the same. Consequently, the maximum delay of the whole network can be reduced, in contradiction to the optimality of the assignment \( A^* \).

We have implemented the algorithm above in order to calculate the optimal assignment of 1000 units of energy on different topologies. For simplicity, each node \( v \) is assigned the same
value of $c_v$. Figure 3.3(a) shows a case where the collection of routes forms a full binary tree. As expected, the root receives the highest wake-up frequency and the nodes’ wake-up frequencies decrease exponentially as we go down the tree. The next topology, shown in Figure 3.3(b), is a star with $n$ leaves as the root children. In this case the optimal wake-up frequency assignment requires that the root wake-up frequency be $\sqrt{n}$ times larger than that of the leaves. The third studied topology, in which every inner node has a single child, is presented in Figure 3.3(c). Since in this network there is only one path from the leaves to the root, the optimal energy assignment is to give an equal share to every node. The last topology is a truncated binary tree, presented in Figure 3.3(d). This example reveals that two factors determine the wake-up frequency distribution: the distance of a node to the root and the size of the node’s sub-tree. For an inner node, the size of the subtree is the dominating factor in determining the wake-up frequency distribution ($a > b_2 > c_2 > d_2 > e_2$), while for a leaf node, the distance of the node to the root is the dominating factor ($b_1 < c_1 < d_1 < e_1$).

### 3.4 Step 2: Limiting the Energy Consumed by a Single Node

Network lifetime is not affected only by the overall consumed energy, but also by the way this energy is divided among the network nodes. As shown in the examples of Figure 3.3, the variance of the frequencies assigned by Algorithm 1 might be very high. We now present Step 2 of the algorithm, which addresses this problem. We start with the following definition:

**Definition 2** A $\tau$-optimal assignment of wake-up frequencies to nodes is an assignment that guarantees an upper bound $\tau$ on the energy assigned to every node and an upper bound $D$ on the
maximum delay, while minimizing the total consumed energy.

We now formalize this definition as an optimization problem.

**Problem 2** Let \( T \) be a tree that consists of all the paths from the sensors to the gateway without the leaves. Let \( f_v \) and \( c_v \) be the wake-up frequency and the average energy consumption during each wake-up for node \( v \), respectively. Then, we want to

\[
\text{minimize: } \sum_{v \in T} f_v \cdot c_v
\]

subject to:

\[
\begin{align*}
&\text{• for every directed path } v_1 \rightarrow \cdots \rightarrow v_N \text{ in } T, \text{ where } v_N \text{ is the gateway,} \\
&\quad \sum_{v \in \{v_1, \ldots, v_N\}} \frac{1}{f_v} \leq D. \\
&\text{• for every node } v \text{ in } T: 0 < f_v \cdot c_v \leq \tau.
\end{align*}
\]

The difference from Problem 1 is the second condition, where for every node \( v \), \( f_v \cdot c_v \leq \tau \) must also hold.

In the general case, Problem 2 can be solved, using convex optimization methods, like those described in Chapter 4. We now show how to solve Problem 2 in the special case where the average amount of energy expended by a node \( u \) during each wake-up is equal to or smaller than the average amount of energy expended by its parent \( v \). That is, \( c_u \leq c_v \). We believe that this is a realistic assumption for the following reasons:

- If the traffic generated in the network is sparse, as often is the case in sensor networks for alarm events [47], we can assume that when a node wakes up and stays in active mode for a short period (which is still long enough to determine if there exists a pending packet from a neighboring node), it is able to receive from its downstream neighbors all their awaiting packets. In some busy times, a node will have to extend its active period in order to allow all the packets to be received. However, on the average we could assume that for every two nodes \( v \) and \( u \), \( c_v \approx c_u \).

- If the traffic generated in the network is not sparse, nodes that are located closer to the gateway are expected to be more loaded. Hence, these nodes will have to expand their active periods much more often than their children. Hence, in such networks \( c_u < c_v \) holds if \( v \) is the parent of \( u \).
Claim 3 If node \( v \) is a parent of node \( u \), then under optimal assignment the energy expended by \( v \) is greater than or equal to the energy expended by \( u \) (i.e., \( c_v \cdot f_v \geq c_u \cdot f_u \)).

Proof: Assume the claim does not hold, namely, \( c_v \cdot f_v < c_u \cdot f_u \). By our assumption, \( c_v \geq c_u \). Hence, \( f_v < f_u \). Now, assume that we increase \( f_v \) and decrease \( f_u \) simultaneously, such that \( c_v \cdot f_v + c_u \cdot f_u \) does not change. Clearly, the overall energy expended by the tree’s nodes does not change, while the delay \( D(v) \) on the sub-tree of \( v \) decreases, in contradiction to the optimality of the original assignment.

Claim 4 Let the optimal assignment found by Algorithm 1 be \( A \). Suppose there exists an assignment \( A^\tau \) that achieves the same upper bound on the maximum delay while limiting the energy expended by each sensor to \( \tau \). If under assignment \( A \) the energy expended by a node is greater than \( \tau \), then under assignment \( A^\tau \) the energy expended by this node is exactly \( \tau \).

Proof: Assume the claim does not hold. Namely, there is a non-empty set \( V \) of nodes whose energy consumption under \( A \) is higher than \( \tau \) while their energy consumption under \( A^\tau \) is strictly smaller than \( \tau \).

We first prove that \( V \) cannot contain a leaf node. Suppose that \( v \) is a leaf node in \( V \). Consider the path \( P \) from \( v \) to the root. By Claim 3, the energy expended by all the nodes along this path under \( A \) is higher than \( \tau \). By Claim 1(b), the delay from \( v \) to the root under \( A \) is equal to the maximum tolerated delay \( D \). Hence, under \( A^\tau \), the energy expended by at least one of the nodes along \( P \) is greater than \( \tau \), which yields a contradiction.

Suppose now that \( V \) contains a non-leaf node \( v \). By Claim 1(b), under both \( A \) and \( A^\tau \) the delay from the leaves to the root is equal to the maximum tolerated delay \( D \). However, the delay from node \( v \) to the root under \( A^\tau \) is higher than the delay from node \( v \) to the root under \( A \). This is because, by Claim 3, the energy expended by every ancestor of \( v \) is greater than \( \tau \) under \( A \), but is not greater than \( \tau \) under \( A^\tau \). Therefore, in the sub-tree rooted at \( v \), the delay from the leaves to node \( v \) under \( A^\tau \) must be shorter than the delay from the leaves to node \( v \) under \( A \).

Let \( E^A(w) \) be the energy assigned by \( A \) to the sub-tree of a node \( w \) under \( A \) and \( E^{A^\tau}(w) \) be the energy assigned to this sub-tree by \( A^\tau \). Let \( e^A(w) \) be the energy expended by node \( w \) under \( A \), and \( e^{A^\tau}(w) \) be the energy expended by this node under \( A^\tau \).

We now show that every node in the sub-tree of \( v \) consumes the same amount of the sub-tree’s overall energy under both assignments. If this claim is incorrect, there must exist a non-leaf node \( u \) such that \( \frac{e^A(u)}{E^A(u)} \neq \frac{e^{A^\tau}(u)}{E^{A^\tau}(u)} \), while for any descendant \( w \) of \( u \), \( \frac{e^A(w)}{E^A(w)} = \frac{e^{A^\tau}(w)}{E^{A^\tau}(w)} \) holds. Note that such a \( u \) exists, because, for every leaf \( y \), \( E(y) = e(y) \) under any assignment. Since the wake-up
frequency of every descendant of \( u \) is smaller than \( \tau \), the delay function of every child \( u_i \) of node \( u \) under \( A^\tau \) is 
\[
D^{A^\tau}(u_i) = \frac{1}{B_{u_i}E^{A^\tau}(u_i)}
\]
the same function as under \( A \) with the same constant \( B_{u_i} \). Therefore, the delay function of the sub-tree of \( u \) under \( A^\tau \) is
\[
D^{A^\tau}(u_i) = \frac{1}{e^{A^\tau}(u)} + \frac{1}{B_{u_i}E^{A^\tau}(u_i)}.
\]
The derivation of this function has the same form as in Eq. 3.7, and therefore has only one solution. Hence, the delay function has no local minimum, and if the energy division is not optimal, it can always be improved.

All the nodes in the sub-tree of \( u \) have wake-up frequencies smaller than \( \tau \), as follows from Claim 3. Therefore, if the delay under \( A^\tau \) is 
\[
D^{A^\tau}(u) = \frac{1}{e^{A^\tau}(u)} + \frac{B}{E^{A^\tau}(u)}
\]
as in Eq. 3.9, there must exist an \( \varepsilon \) (positive or negative), such that by defining the wake-up frequency of node \( u \) to be \( e^{A^\tau}(u) + \varepsilon \), and re-computing the energy assigned to every sub-tree of node \( u \)'s children without changing the overall energy of \( u \)'s sub-tree, the delay on the sub-tree of \( u \) is reduced to 
\[
D'(u) = \frac{1}{e^{A^\tau}(u) + \varepsilon} + \frac{B}{E^{A^\tau}(u) - (e^{A^\tau}(u) + \varepsilon)}.
\]
Hence, a smaller maximum delay is achieved with the same sub-tree energy or, optionally, the same maximum delay is achieved with less energy, in contradiction to the fact that \( A^\tau \) is a best energy assignment under the given constraint. We conclude that for every node in the sub-tree of \( v \), its assigned fraction of energy is equal under both assignments.

Since the wake-up frequency of \( v \) under \( A^\tau \) is smaller than under \( A \) (\( e^{A^\tau}(v) < e^A(v) \)), the overall amount of energy consumed by the sub-tree is smaller as well, i.e., \( E^{A^\tau}(v) < E^A(v) \). Therefore, the delay of the sub-tree of \( v \) under \( A^\tau \) is shorter than under \( A \), while the overall amount of energy consumed by the sub-tree is also smaller. This is impossible of course, because the energy division was shown to be the same under both assignments.

The following algorithm is developed, from Claims 3 and 4, in order to find a \( \tau \)-optimal wake-up frequency assignment to the nodes.

Algorithm 2

1. If by assigning the maximum energy \( \tau \) to every node there is a routing path for which the delay is greater than the maximum tolerated delay \( D \), then no solution exists. Otherwise, perform the following steps:

2. Calculate the optimal assignment using Algorithm 1. Recall that this assignment does not guarantee an upper bound on the energy assigned to a single node.

3. For every node \( v \) whose assigned energy is greater than \( \tau \), reduce it to \( \tau \).

4. For every node \( v \) whose assigned energy is equal to \( \tau \) but has a child \( u \) whose assigned energy is smaller than \( \tau \), recalculate the child sub-tree’s energy division using Algorithm...
1 in order to reduce the maximum delay on this sub-tree such that it will be equal to the maximum tolerated delay $D$ minus the maximum delay from the root to $v$.

5. Following step 4, the energy assigned to some nodes may exceed the threshold $\tau$. If there are such nodes, return to step 3.

**Theorem 2** If there exist feasible schedules that meet the constraints on the maximum delay $D$ and the maximum energy assigned to every node $\tau$, Algorithm 2 will find the one with the minimum total energy.

**Proof:** First, we prove that the algorithm stops. If no feasible assignment exists, the algorithm discovers this at step 1, because such an assignment exists if and only if the maximum delay is not larger than $D$ when all the nodes are assigned energy $\tau$. If the algorithm passes step 1, it must stop unless it enters an infinite loop in steps 3–5. However, such an infinite loop is not possible, because after the energy assigned to a specific node is reduced to $\tau$ in step 3, it cannot be changed. Therefore, each time the algorithm returns to step 3, the energy of at least one additional node becomes fixed, and after at most $|V|$ iterations, where $|V|$ is the number of nodes, the algorithm must stop.

Next, we prove that the energy assignment found by the algorithm is feasible, i.e., that both requirements are met. When the algorithm stops, no node is assigned energy greater than $\tau$, thereby meeting the first requirement. The second requirement, namely an upper bound on the maximum delay, is fulfilled due to the correctness of Algorithm 1: every time the energy of a node is reduced in step 3, the energy assignment on every path through this node is recalculated, such that the maximum delay $D$ is guaranteed for this path.

Finally, we show that the energy assignment determined by the algorithm is indeed optimal. We start by proving that after each iteration of step 3 of the algorithm, the set of nodes whose energy is equal to $\tau$ is either empty or it forms a connected sub-tree containing the root. We prove the claim by induction on the iterations. From Claim 3 we know that nodes located closer to the root are assigned greater energy. Therefore, if the energy assigned to a node in step 2 is greater than $\tau$, then all the ascendants of this node towards the root are also assigned energy greater than $\tau$. Hence, the set of nodes whose energy is reduced to $\tau$ during the first iteration of step 3 forms a sub-tree rooted at the root. Let us consider now the $k^{th}$ iteration of step 3. From the induction assumption we know that the nodes whose energy is $\tau$ following the $(k - 1)^{th}$ iteration of step 3 form a sub-tree rooted at the gateway. Let this tree be denoted as $T'$ (see Figure 3.4). For every sub-tree $T_s$ whose energy increases in step 4 of the $(k - 1)^{th}$ iteration, Claim 3 also
Figure 3.4: Algorithm 2 after $k - 1$ iterations

holds. Therefore, if a node $u$ in such a sub-tree is assigned energy greater than or equal to $\tau$, all its ascendants in the same sub-tree $T_s$ are also assigned energy greater than or equal to $\tau$. This implies that if the energy of a node $u$ is reduced in step 3 of the $k^{th}$ iteration, the energy of the sub-tree root $v$ is also reduced, and the same holds for all the nodes on the path from $u$ to $v$ (see Figure 3.4). By the definition of step 4, when the sub-tree root is chosen in step 4 of the $(k - 1)^{th}$ iteration, its parent has energy of $\tau$. By the induction assumption, the parent of a sub-tree root belongs to the sub-tree $T'$, formed by the nodes with energy of $\tau$. Hence, after the $k^{th}$ iteration, all the nodes whose energy is equal to $\tau$ indeed form a connected sub-tree.

We now show that every node $u$ whose assigned energy is $\tau$ will be assigned that energy under any $\tau$-optimal assignment. The proof is by induction on the iteration of step 3, during which node $u$ is assigned $\tau$ energy. After the first iteration, the correctness of the claim follows from Claim 4. Assume that for the first $k - 1$ iterations the claim holds. If the energy of node $u$ is reduced to $\tau$ in the $k^{th}$ iteration, then its energy was recalculated in step 4 of the $(k - 1)^{th}$ iteration. Therefore, $u$ is a member of a sub-tree $T_s$ rooted at node $v$. Since $v$’s parent belongs to $T'$, $v$ was assigned energy greater than $\tau$ in step 4 of the $(k - 1)^{th}$ iteration. By the previous claim, all the ascendants of $v$ are assigned energy of $\tau$ at this moment. Therefore, by the induction assumption, all the ascendants of $v$ are assigned energy of $\tau$ under any $\tau$-optimal assignment. This implies that under any $\tau$-optimal assignment the delay on the sub-tree $T_s$ of $v$ is equal to $D$ minus the delay from $v$ to the root (see Figure 3.4). The latter delay is equal to $\frac{1}{\tau}$ multiplied by the number of nodes along this route. From the correctness of Algorithm 1 we know that in step 4 of the $k^{th}$ iteration,
an optimal assignment for the sub-tree of \( v \) is determined. Therefore, by Claim 4, every node that receives energy greater than \( \tau \) under an optimal assignment will receive energy of \( \tau \) under any \( \tau \)-optimal assignment.

To summarize, when the algorithm stops, every node whose assigned energy is \( \tau \) will have the same energy under any \( \tau \)-optimal assignment, while the energy assigned to each of the other nodes is calculated in the last iteration of step 4 using Algorithm 1, and has already been proved to be optimal.

Algorithm 1 performs two passes along the tree, and for every node the calculations require \( O(1) \) operations. Therefore, the complexity of the algorithm is \( O(2|V|) \), where \( |V| \) is the number of nodes. Algorithm 2, in its naive implementation, requires \( O(|V|^2) \) operations, because the energy division can be recalculated for each node sub-tree. Note however, that as shown in the correctness proof of Algorithm 1, the optimal energy division in a sub-tree is unique and does not depend on the available energy. Since Algorithm 1 is implemented in the first iteration, the optimal energy division for every sub-tree is already determined. This assignment can be used while allocating energy to nodes in step 3 of the algorithm. Therefore, the reallocation process can be performed in \( O(1) \) operations at each node, and the complexity of Algorithm 2 is reduced to \( O(3|V|) \): two passes on the tree in step 2 and an additional pass for the energy reallocation in steps 3, 4 and 5.

3.5 Simulation Results

We compare now the algorithms presented in this chapter with an equal wake-up frequency that does not depend on the location of the nodes. The latter approach is referred to as “equal assignment.” The comparison is conducted by building routing trees on random graph [48, 49, 50]. One thousand nodes, representing the sensors, are randomly placed over a 100X100 grid. The transmission range varies between 15 and 40. For each tested range \( r \), any two nodes whose Euclidean distance is not greater than \( r \) are considered to be connected. If the created graph is not connected, this instance is ignored by the simulator. If it is connected, we select a random node to function as the gateway, and assume that the routing from each sensor to the gateway is performed over the shortest path.

We first compare the overall energy consumption of the network under equal assignment and under Algorithm 2. To this end, we assign an equal wake-up frequency \( e \) to every node and then measure the maximum delay resulting from this assignment. This value is considered as the target maximum delay \( D \). Next, we execute Algorithm 2 with \( D \) as the maximum tolerated
delay and $\tau = e \cdot \text{Limit-Factor}$ as the maximum wake-up frequency assigned to a single node. When \text{Limit-Factor} is 1, this maximum is equal to the wake-up frequency of equal assignment. When \text{Limit-Factor} \to \infty$, there is no limit on the difference between the maximum and minimum assigned wake-up frequencies. Therefore, in this case Algorithm 2 is reduced to Algorithm 1.

Figure 3.5 shows the overall energy ratio between Algorithm 2 and equal assignment for shortest-path routing. We start with the case where \text{Limit-Factor}=1. Despite the fact that the maximum wake-up frequency a single node can be assigned is exactly as in equal assignment, the total consumed energy is reduced by 35%-65%. The reason for this is that under equal assignment the maximum delay from some of the leaves is smaller than $D$, while under our algorithm the maximum delay from all the leaves to the root is equal to $D$. Consequently, leaves closer to the root, as well as nodes on the path from these leaves to the root, can be assigned a smaller wake-up frequency than nodes on the longer paths. If all the leaves had the same depth, the overall energy consumption of the two algorithms would be equal for this \text{Limit-Factor}, but the probability that a random tree will have this property is negligible.

When \text{Limit-Factor} grows, Algorithm 2 has the flexibility to assign higher wake-up frequencies to some of the nodes, thereby approaching the optimal solution of Algorithm 1. Apparently, \text{Limit-Factor}=5 is sufficient for Algorithm 2 to find almost the same wake-up frequency assignment as Algorithm 1. This is evident from the convergence of the curves with \text{Limit-Factor}=5 and \text{Limit-Factor}=50.
The improvement achieved by our algorithm is almost independent of the chosen routing algorithm. To show this, we repeat the experiment, this time considering a routing algorithm that minimizes the number of internal nodes (that is, the number of nodes that have to wake up periodically) [51]. As shown at Figure 3.6, the improvement achieved by our scheme is similar to the improvement in Figure 3.5 for the shortest-path routing.

As shown earlier, in some networks the maximum wake-up frequency (energy) assigned to some of the nodes is much greater than the average wake-up frequency. This may result in a loss of these nodes and possible loss of network connectivity. We performed additional simulations in order to determine how the standard deviation of the assigned wake-up frequency is influenced by the limit imposed on a single node. The transmission range is 15 in one instance and 40 in another. The value of Limit-Factor ranges between 1 to 5. The results are presented in Figure 3.7, where the standard deviation of the wake-up frequency is compared to the average wake-up frequency.

We can see that when Limit-Factor grows, the variance of the assigned wake-up frequencies grows as well, while the average wake-up frequency decreases. The differences in the two graphs can be explained by the different routing tree structures. When the range is equal to 40, the routing tree resembles a star tree. Therefore, a significant portion of its nodes are leaves adjacent to the root. All these nodes are assigned the same low wake-up frequency. They reduce the wake-up frequency average, but increase the wake-up frequency variance.

We now show that the new algorithm significantly extends the sensor network lifetime. For
this study, we use an energy-constrained shortest path routing algorithm that takes into account the energy available at every node, as explained below. In each simulation’s instance, the energy consumed by an inner node during a time unit is set to be equal to the node’s wake-up frequency, while the energy consumed by a leaf node is fixed and small. When the energy of an inner node falls below a threshold, whose value is set to 20% of the initial energy, routing through this node is not allowed any more. Hence, shortest paths are computed without taking this node into account. Consequently, the energy consumed by such a node is minimized. Each simulation instance is stopped when the routing from some sensor to the gateway is not possible any more. The lifetime of the network for this instance is then compared to the case where wake-up frequencies are equal for all nodes. Figure 3.8 shows the ratio between the average lifetime of the network in these two cases, for a 100x100 grid and two Limit-Factor values.

Consider first the upper curve, for Limit-Factor= 3. We can see that for reasonable values of transmission range (< 25), our algorithm increases the network lifetime by 40%-80%. The improvement of our algorithm decreases when the network is dense. This is because in such a case the routes from the sensors to the gateway become shorter. Therefore, the wake-up frequency assigned by our algorithm to the nodes that are close to the gateway is greater, these nodes are likely to run out of energy faster. When the network is not so dense, the routing paths are longer, and the wake-up frequency assigned by our algorithm to these key nodes is smaller. Consequently, routing through these nodes is possible for longer time, and the total network’s lifetime increases.

When we increase the value of Limit-Factor from 3 to 5, the improvement in the network lifetime is smaller. At first glance this result seems to contradict the results shown in the previous
graphs, because we saw there that greater values of Limit-Factor improve the performance of our algorithm. Indeed, when the value of Limit-Factor increases, the total consumed energy is reduced, but the likelihood that some popular nodes will run out of their energy sooner is greater. Therefore, when selecting the value of Limit-Factor to work with, one needs to take into account both the network lifetime and the total consumed energy. If only the latter is important, as is the case when energy cannot be re-charged, the value of Limit-Factor should be close to 1.

Figure 3.8: The network life time ratio as a function of the transmission range for different values of Limit-Factor
Chapter 4

Energy-Delay Optimization in a Sensor Network with Multiple Gateways

4.1 Introduction

In this work we extend the results of Chapter 3 in two directions. Instead of solving the sensor wake-up scheduling problem for a single routing tree rooted at a single gateway, we solve it for multiple trees rooted at one or more gateways. The benefits of using multiple trees are clear: (a) faster recovery from gateway failures and (b) load sharing. In our model there might be multiple trees associated with the same gateway. Figure 4.1 shows an example of an instance of the considered problem. This instance consists of an underlying graph and three routing trees rooted at two different gateways.

We also extend the results of Chapter 3 with regard to the routing tree, which is assumed there to be given in advance. Here, however, the tree building problem is addressed from the perspective of the optimal wake-up frequency assignment. In other words, the algorithm for building the tree takes into account the optimization criterion imposed by the optimal wake-up frequency assignment problem.

Throughout the chapter, we use the same definition of $\tau/D$-optimal wake-up frequency assignment as it presented in Section 3.4. Our energy expenditure model is also similar to the model described in Section 3.2 of the previous chapter. However, since now we assume existence of multiple spanning trees, only a node that is a leave in all the trees is not assigned a wake-up frequency.

The rest of this chapter is organized as follows. In Section 4.2 we study the problem of $\tau/D$-
optimal wake-up frequency assignment in multiple trees and present two algorithms: an optimal one and an approximation. In Section 4.3 we study the problem of constructing an efficient routing tree. We prove that the problem is NP-hard and propose an approximation algorithm. In Section 4.4 we present simulation results.

4.2 The MT-OFA Problem

The MT-OFA problem is to find a $\frac{\tau}{D}$-optimal wake-up frequency assignment for a given network with multiple trees. This problem is formally defined as follows.

**Problem 3 (MT-OFA problem):**

Given a graph $G = (V, E)$, a set of trees $\{T_1, T_2, \ldots, T_n\}$, upper bounds $\{D_1, D_2, \ldots, D_n\}$ on the end-to-end delay in each tree, an upper bound $\tau$ on the energy expended by every node due to (R1), and a function $C : V \rightarrow Z^+$ that defines the energy expended by every node per wake-up, assign to each node $v$ a wake-up frequency $f_v$ such that

1. $\forall v \in V, f_v \cdot c_v \leq \tau$; i.e, the energy expended by every node due to (R1), is upper bounded by $\tau$.  

![Figure 4.1: A sensor network with multiple routing trees](image)
2. For every tree $T_i$ and for every path $P \in T_i$, $\sum_{v \in P} \frac{1}{f_v} \leq D_i$; i.e, the end-to-end delay on each directed path in every tree $T_i$ is bounded by $D_i$.

3. $\sum_{v \in V} f_v \cdot c_v$ is minimized.

The special case of a single gateway and a single routing tree is studied in Chapter 3, where an optimal polynomial time algorithm for this problem is presented. The algorithm performs several iterations over the tree. During each iteration it computes an optimal wake-up frequency assignment for the nodes of some sub-trees while restricting the maximum wake-up frequency. In the rest of this chapter, this algorithm is referred to as the optimal ST-OFA (Single Tree Optimal Frequency Assignment) algorithm.

The polynomial time algorithm presented in Chapter 3 cannot be used for the MT-OFA. In its general form, MT-OFA is a convex optimization problem. To see why, note that the delay through a single node is a convex function of the wake-up frequency. The delay on a path $P$ is a sum of convex functions and, therefore, is a convex function as well. Finally, the target function is linear in $f_v$.

We now reformulate MT-OFA into a conic quadratic optimization problem, which has only linear and convex cone constraints. The reciprocals $\frac{1}{f_v}$ are replaced by new variables $d_v$, which are not part of the target function, and for every $v$ we add a constraint $d_v = \frac{1}{f_v}$. This problem can be solved using the interior point method [52], by traversing the interior values of the problem (unlike the simplex method in which the edges are traversed), until it reaches the maximum. It is shown in [53] that the optimal solution is found after a polynomial number of iterations. The resulting optimal algorithm for MT-OFA is:

**Algorithm 3 (Optimal Algorithm for MT-OFA):**

Use the interior point method to minimize $\sum_{v \in V} f_v \cdot c_v$, subject to:

1. $\forall v \in V, f_v \cdot c_v \leq \tau$; i.e, the energy expended by every node due to (R1) is upper bounded by $\tau$.

2. For every tree $T_i$ and for every path $P \in T_i$, $\sum_{v \in P} d_v \leq D_i$; i.e, the end-to-end delay on each directed path for each tree $T_i$ is bounded by $D_i$.

3. $\forall v \in V, d_v = 1/f_c$.  

4. $\forall v \in V, f_v, d_v \geq 0$.
Algorithm 3 is impractical in many cases because it demands high computational power. Therefore, we also propose an efficient approximation for MT-OFA, which uses the ST-OFA algorithm as a sub-routine.

Algorithm 4 (An approximation algorithm for MT-OFA):

1. Execute the ST-OFA algorithm for each tree $T_i$ and find the wake-up frequency $F_i(v)$ assigned to every node $v$ for each tree $T_i$

2. Assign to each node $v$ a wake-up frequency $F(v) = \max_i F_i(v)$.

Theorem 3 A solution found by Algorithm 4 is feasible because it fulfills the first two constraints in Problem 3. In addition, the total energy expended by the network nodes is at most $n$ times greater than that of the optimal solution, where $n$ is the number of trees.

Proof:

Let $F^*$ be the wake-up frequency vector computed by Algorithm 4, and let $F_1, F_2, \ldots, F_n$ be the wake-up frequency vectors computed by the optimal ST-OFA algorithm for each individual tree. For each vector $F$, let $F(v)$ denote the wake-up frequency assigned to node $v$ and $|F| = \sum_{v \in V} c_v$ the energy expenditure of the network under the wake-up frequency assignment $F$.

For every node $v$ and for every $i = 1, \ldots, n$, $F^*(v) \cdot c_v \leq \tau$. Hence, the first constraint clearly holds. Since the delay on any path $P$ under $F^*$ is not greater than the delay under any other assignment $F_i$, the second constraint holds as well.

To prove the second part, note that $|F^*| = \sum_{v \in V} c_v F^*(v) < \sum_{v \in V} c_v \sum_{1 \leq i \leq n} F_i(v) = \sum_{1 \leq i \leq n} \sum_{v \in V} c_v F_i(v) = \sum_{1 \leq i \leq n} |F_i| \leq n \max_{1 \leq i \leq n} |F_i| \leq n F_{OPT}$, where $F_{OPT}$ is the optimal wake-up frequency assignment.

The time complexity for the first step of Algorithm 4 is $O(n|V|)$ and for the second step it is $O(|V|)$, where $n$ is the number of trees and $|V|$ is the number of nodes in the graph. To implement the algorithm in a distributed environment, each node needs to exchange one short message with each of its neighbors in the tree.

4.3 Constructing an Energy-Efficient Tree

ST-OFA and MT-OFA are defined for an already constructed spanning tree (or set of spanning trees). In this section we address the problem of building energy efficient trees in advance.
4.3.1 NP-hardness Proof

We first formally define the problem and then prove that it is NP-hard even for a single tree.

**Problem 4 (The Minimum Energy Tree Problem (METP)):**
Given a graph $G = \{V, E\}$, a node $\tau$, and a function $C : V \rightarrow \mathbb{Z}^+$ that defines the energy expended by every node during every wake-up, find a spanning tree rooted at $\tau$ for which there exists a $\tau/D$-optimal frequency assignment whose total energy expenditure is minimal. This tree is referred to as the best $\tau/D$-optimal tree.

**Claim 5** Consider the best $\tau/D$-optimal tree in a graph $G = (V, E)$. Let $T$ be this tree without the leaves. Then, $T$ contains no pair of nodes $u$ and $v$ such that $(u, v) \in E$, $(u, v) \notin T$ and either $u$ is a descendant of $v$ in $T$ or vice versa.

**Proof:** Assume by contradiction that such a pair of nodes exists. Without loss of generality, let $u$ be a descendant of $v$. Let $T'$ be the same as $T$, except that the sub-tree of $u$ is directly connected to $v$. That is, we remove from $T$ the edge between $u$ and its parent and add an edge $(u, v)$. We now show that a $\tau/D$-optimal frequency assignment for $T'$ is better than a $\tau/D$-optimal frequency assignment for $T$. Consider a $\tau/D$-optimal frequency assignment for $T$. If we use the same assignment for $T'$, the delay from each node in the sub-tree of $u$ to the root in $T'$ is smaller than in $T$ because the path from $u$ to the root in $T'$ is a sub-path of the corresponding route in $T$. Hence, we can decrease the wake-up frequency and therefore the energy expended by some of the nodes in the sub-tree of $u$ without increasing the maximum latency beyond $D$. Thus, $T'$ is a better $\tau/D$-optimal spanning tree than $T$, in contradiction to our assumption. \[\blacksquare\]

**Claim 6** The overall energy expenditure of a tree whose structure without the leaves is a star under a $\tau/D$-optimal wake-up frequency assignment is $(\sqrt{c} + \sqrt{\sum_{i=1}^{n} c_i})^2$ if $\tau$ is infinite and $D = 1$. Here, $c$ and $c_1, \ldots, c_n$ are the amounts of energy expended per wake-up by $v$ and $v_1, \ldots, v_n$ respectively.

**Proof:** Note that in the following analysis we ignore the constant components $b_v$, since their values are not affected by the tree structure. Let $v$ be the root and $v_1, \ldots, v_n$ be the leaves of the inner tree. The energy expended per time unit for $v$ is $c$ and for $v_1, \ldots, v_n$ it is $c_1, \ldots, c_n$. Following this notation, the wake-up frequency of $v$ is $c/e$, and the wake-up frequency of every other node $v_i$ is $e_i/c_i$. In order to solve the ST-OFA problem, the following set of $n + 1$ equations
has to be solved:

\[
E = e + e_1 + e_2 + \ldots + e_n
\]

\[
\frac{c_1}{e_1} = \frac{c_2}{e_2} = \ldots = \frac{c_n}{e_n}
\]

\[
\frac{\delta (e + e_1)}{\delta e} = 0.
\]

In the first equation \(E\) is the total energy expended by all the nodes in the tree. The next \(n - 1\) equations dictate that the delay on the paths from the leaves to the root are equal and the last equation dictates that this delay is minimal, because, as shown in Chapter 3, minimizing the delay for a given \(E\) is equivalent to minimizing the energy expenditure for a given delay.

From this set of equations we get that for every \(i, 1 \leq i \leq n\), \(e_i = (e_1/c_1) \cdot c_i\).

Hence,

\[
E = e + \sum_{i=1}^{n} \frac{e_1}{c_i} c_i = e + \frac{e_1}{c_1} \sum_{i=1}^{n} c_i.
\]

Therefore,

\[
e_1 = \frac{E - e}{\sum_{i=1}^{n} c_i/c_1}
\]

and

\[
\frac{\delta E}{\delta e} + \frac{c_1}{\sum_{i=1}^{n} c_i} \frac{\delta e}{\sum_{i=1}^{n} c_i} = \frac{\delta e}{\delta e} + \frac{\sum_{i=1}^{n} c_i}{E - e} = \delta e - \frac{c}{e^2} + \frac{\sum_{i=1}^{n} c_i}{(E - e)^2} = 0.
\]
We then get that

\[-c(E - e)^2 + e^2 \sum_{i=1}^{n} c_i = \]
\[-eE^2 + 2Ece - ce^2 + e^2 \sum_{i=1}^{n} c_i = \]
\[(\sum_{i=1}^{n} c_i - c)e^2 + 2Ece - cE^2 = 0\]

and

\[e = \frac{-2Ec \pm \sqrt{4E^2c^2 + 4E^2c(\sum_{i=1}^{n} c_i - c)}}{2 \sum_{i=1}^{n} c_i - c} = \frac{-2Ec \pm 2E \sqrt{c^2 + c(\sum_{i=1}^{n} c_i - c)}}{2 \sum_{i=1}^{n} c_i - c} = E \frac{-c \pm \sqrt{c^2 + c(\sum_{i=1}^{n} c_i - c^2)}}{\sum_{i=1}^{n} c_i - c} = E \frac{-c \pm \sqrt{c(\sum_{i=1}^{n} c_i)}}{\sum_{i=1}^{n} c_i - c} = E \frac{-c \pm \sqrt{c \sum_{i=1}^{n} c_i}}{\sum_{i=1}^{n} c_i - c}.\]
Therefore, the delay on the tree is

\[
\frac{c}{e} + \frac{c_1}{e_1} = \frac{\sum_{i=1}^{n} c_i - c}{E(-c + \sqrt{c\sum_{i=1}^{n} c_i})} + \frac{\sum_{i=1}^{n} c_i}{E - E^{-c + \sqrt{c\sum_{i=1}^{n} c_i}}} \\
= \frac{1}{E} \left( \frac{\sum_{i=1}^{n} c_i - c}{-c + \sqrt{c\sum_{i=1}^{n} c_i}} + \frac{\sum_{i=1}^{n} c_i}{1 - 1 - \frac{e}{c}\sqrt{c\sum_{i=1}^{n} c_i}} \right) \\
= \frac{1}{E} \left( \frac{\sum_{i=1}^{n} c_i - c}{\sqrt{c}(\sqrt{\sum_{i=1}^{n} c_i} - \sqrt{c})} + \frac{\sum_{i=1}^{n} c_i}{\sqrt{\sum_{i=1}^{n} c_i}(\sqrt{\sum_{i=1}^{n} c_i} - \sqrt{c})} \right) \\
= \frac{1}{E} \left( \frac{\sqrt{c}(\sum_{i=1}^{n} c_i - c)\sqrt{\sum_{i=1}^{n} c_i}}{\sqrt{\sum_{i=1}^{n} c_i}(\sqrt{\sum_{i=1}^{n} c_i} - \sqrt{c})} + \frac{\sum_{i=1}^{n} c_i}{\sqrt{\sum_{i=1}^{n} c_i}(\sqrt{\sum_{i=1}^{n} c_i} - \sqrt{c})} \right) \\
= \frac{1}{E} \left( \frac{\sqrt{c}(\sum_{i=1}^{n} c_i - c)\sqrt{\sum_{i=1}^{n} c_i} + \sum_{i=1}^{n} c_i(\sum_{i=1}^{n} c_i - c)}{\sqrt{\sum_{i=1}^{n} c_i}(\sqrt{\sum_{i=1}^{n} c_i} - \sqrt{c})} \right) \\
\right) \\
= \frac{1}{E} \left( \frac{\sqrt{c}(\sum_{i=1}^{n} c_i - c)(\sqrt{\sum_{i=1}^{n} c_i} + \sum_{i=1}^{n} c_i)}{\sqrt{\sum_{i=1}^{n} c_i}(\sqrt{\sum_{i=1}^{n} c_i} - \sqrt{c})} \right) \\
= \frac{1}{E} \left( \frac{\sqrt{c} + \sqrt{\sum_{i=1}^{n} c_i}}{\sqrt{\sum_{i=1}^{n} c_i}} \right) \\
= \sqrt{c} + \sqrt{\sum_{i=1}^{n} c_i}.
\]

Taking \( D \) to be equal to 1, we have

\[
E = \sqrt{c} + \sqrt{\sum_{i=1}^{n} c_i}.
\]

Note that for the case where every \( c_v = 1 \), the overall energy expenditure of the tree is \( E = 1 + 2\sqrt{n-1} + n - 1 = n + 2\sqrt{n-1} \).

Claim 7 \( \text{METP} \) is \( \text{NP-hard} \) even in the special case where the energy expended by each node is equal to \( f_v \), namely, \( \forall v \in V, c_v = 1 \).
Proof: We present a reduction from the well-known NP-hard minimum dominating set (MDS) problem [54] to METP. The input for the MDS problem is an undirected graph \( G = (V, E) \) and the goal is to find a minimum subset of nodes \( V' \) such that every node \( v \in V \) is either in \( V' \) or has a neighbor in \( V' \).

Let \( G = (V, E) \), where \( V = \{v_1, v_2, \ldots, v_n\} \) is an instance for the MDS problem. We build a new graph \( G' = (V', E') \) such that every node \( v \in V \) appears in \( G' \) twice: as \( v \) and as \( v' \). In addition, \( V' \) contains a new node \( r \). The set \( E' \) of edges is built such that \( E' = \{ (r, v') \} \cup \{ (v, v') \} \cup \{ (v', u) | (v, u) \in E \} \). Figure 4.2 shows an example of this transformation. We use \( G' \) as an instance for METP.

We now prove that a solution for METP on \( G' \) can be transformed into a solution for MDS on \( G \). Let \( T \) be the sub-tree of the inner nodes in the solution for METP. By Claim 5, each node that has an edge to \( r \) in \( G' \) must be connected directly to \( r \) in \( T \). Therefore, every \( v' \) is either not in \( T \) or is directly connected to \( r \). For the same reason, no \( v \) node is part of \( T \), which implies that \( T \) is a star.

By Claim 6, under a \( \tau/D \)-optimal wake-up frequency assignment of an ST-OFA algorithm, the energy expended by \( r \) is \( \sqrt{|T| - 1} \) and the overall energy expenditure is \( |T| + 2\sqrt{|T|} - 1 \). The latter expression gets its minimum for the smallest possible \( |T| \). Therefore, for the optimal spanning tree, a node \( v' \) belongs to \( T \) iff \( v \) is part of the minimum dominating set of \( G \). For example, the \( \tau/D \)-optimal spanning tree for the graph in Figure 4.2(b) consists of two nodes, \( r \) and \( d' \), which have the same wake-up frequency of 2, while the MDS of the graph in Figure 4.2(a) consists of node \( d \).

\( \square \)
4.3.2 An Approximation Algorithm for METP

We now identify the best and the worst \( \tau/D \)-optimal trees for ST-OFA. This will help us to develop an approximation algorithm for METP.

Claim 8 For a given set of nodes that should build the inner tree \( T \), the tree with the best \( \tau/D \)-optimal wake-up frequency assignment has a star topology.

Proof: Follows from Claim 5.

Claim 9 For a given set of nodes that should build the inner tree \( T \), the tree with the worst \( \tau/D \)-optimal wake-up frequency assignment is a cascade.

Proof: Let \( F \) be the wake-up frequency assignment to the nodes of the cascade tree and \( P \) the path from the leaf to the root in \( T \). Consider a non-cascade tree \( T' \) built over the same graph using the same nodes. We assign a wake-up frequency to the nodes of \( T' \) according to \( F \). Since any path \( P' \) in \( T' \) is a sub-path of \( P \), the overall delay on \( P' \) is smaller than the maximum tolerated delay \( D \). Hence, the wake-up frequency assigned according to \( F \) to \( T' \) can be improved, thereby making \( T' \) a better \( \tau/D \)-optimal tree. We conclude that any non-cascade tree, consisting of the same set of nodes, is a better \( \tau/D \)-optimal tree than \( T \), which implies that \( T \) is the worst \( \tau/D \)-optimal tree.

Algorithm 5 (an approximation for METP):

Let the input for the METP problem be a graph \( G \) and a function \( C : V \rightarrow \mathbb{Z}^+ \) that defines the energy expended by every node during each wake-up. Let \( G \) and \( C \) be also the input of the minimum-weighted connected dominating set (MWCDS) problem, where \( C \) in this case is used as a weight function. Let \( V' \) be a solution for the MWCDS problem. We create a directed spanning tree in \( G \) whose inner nodes are \( V' \). If several trees can be created, we pick one of them arbitrarily.

Claim 10 The selected tree \( T \) is an \( n' \)-approximation for METP, where \( n' \) is the size of a minimum-weighted connected dominating set \( V' \).

Proof:

Let us find a lower bound on the energy expenditure of a spanning tree in \( G \). By Claim 6, for any tree \( T' \) the smallest energy expenditure is achieved if the inner tree of \( T' \) is a star. In this case, the energy expenditure is at least \( \sum_{v \in T'_{\text{inner}}} c_v \). We seek to minimize this sum by careful selection of the inner tree nodes. Since the inner tree is a connected dominating set, the minimum
of $\sum_{v \in T_{inner}} c_v$ is obtained for a minimum-weighted dominating set $V'$. That is, $\sum_{v \in V'} c_v$ is a lower bound on the energy expended by any tree. By reducing the value of $\tau$, the tree total energy expenditure can only increase. Hence, $\sum_{v \in V'} c_v$ is a lower bound on the energy expenditure for any $\tau$.

Although the actual energy consumption of $T$ depends on the graph topology, we can still find an upper bound on the energy expended by its nodes. By Claim 9, the worst case occurs when the inner tree is a cascade tree. We now present a feasible wake-up frequency assignment, assuming that $\tau$ is infinite. We choose the wake-up frequencies so that the energy expended by all the nodes is equal, i.e., $c_v f_v = e$ for every $v \in G$. In order to find the value of $e$, note that the delay imposed by each node is equal to $c_v/e$. Hence, the delay on the path from the inner tree leaf to the root is $n' \sum_{v \in T_{inner}} \frac{c_v}{e} = D$, where $n'$ is the path length (i.e., the number of nodes in the inner tree of $T$). Taking $D = 1$, we get that $e = \sum_{v \in T_{inner}} c_v$ and the overall energy expended by this tree is $n' \sum_{v \in T_{inner}} c_v$.

Next, we show that the above expression is an upper bound on the total energy expenditure for any value of $\tau$. Obviously, for certain values of $\tau$ this solution is not feasible because $e > \tau$. However, in this case no feasible solution for the cascade-type inner tree exists, and any feasible solution on any other tree has a total energy expenditure smaller than $n' \sum_{v \in T_{inner}} c_v$.

To summarize, we obtained an upper bound on the energy expenditure of a spanning tree $T$ constructed from $V'$. This upper bound is $n'$ times greater than the lower bound obtained for any spanning tree. Note that in the proof we implicitly assumed that $r \in V'$. However, since $r$ must be part of the inner tree both in the optimal and the approximated solutions, we can take its weight to be 0. Consequently, $r$ can be added to any solution without affecting the energy expenditure.

The minimum-weighted connected dominating set (MWCDS) problem is a generalization of the minimum connected dominating set (MCDS) problem, and is therefore NP-hard [54]. In the general case, MWCDS has an approximation algorithm with factor $O(\log n)$ [55], where $n$ is the number of nodes. However, for the unit disk graphs, which are also known as geometric graphs, there exist algorithms with a constant approximation factor [56].

4.4 Simulation Study

4.4.1 Simulation Results for MT-OFA

To study the actual performance of the algorithm presented in Section 4.2 for MT-OFA, we use the simulation model presented in earlier works [48, 49, 50] as follows. We randomly place sensors
We start with comparing the performance of several wake-up assignment algorithms for a given tree. The trees are built using a BFS algorithm. However, we obtained similar results when we used other algorithms for building the tree. The number of trees varies between 2 and 10. We find the optimal solution by Algorithm 3 and compare it to the approximation proposed by Algorithm 4. Recall that the energy expenditure of a node \( v \) is \( f_v c_v + b_v \), where \( c_v \) is the energy expended during every wake-up period. However, the nodes are assumed to have a similar per wake-up energy expenditure, namely, \( c_v = 1 \) holds for every \( v \). The value of \( \tau \) is taken to be infinite and the upper bound \( D \) on the end-to-end delay is 100 time units for every tree.

In order to take advantage of available tools for finding an optimal solution using Algorithm 3, another relaxation is made to the MT-OFA problem: the constraint \( d_v f_v = 1 \) is replaced by \( d_v f_v \geq 1 \). Since we solve a minimization problem where all the coefficients in the target function are positive, the function receives its minimum only when all of the \( f_v \) values receive their minimum. This happens when \( d_v \cdot f_v \) is exactly 1. Hence, these two constraints are equivalent. In its new form, the problem represents a conic constraint optimization problem. The target function is linear and the first set of constraints is linear as well. The second set of constraints represents a rotated quadratic cone. To calculate the optimal solution, we use the MOSEK tool [57].

Figure 4.3: Approximation algorithm results for multiple trees
For the implementation of Algorithm 4, the wake-up frequency for each node is calculated for every tree using the optimal ST-OFA algorithm. Then, each node is assigned the maximum frequency value. Finally, we compare the ratio of the overall energy expenditure of Algorithm 4 and the overall energy expenditure of the optimal solution obtained by Algorithm 3.

Figure 4.3 presents the simulation results for 2-10 spanning trees in networks with 200 and 2000 nodes. The graphs show the performance ratio as a function of the transmission range. For each transmission range and number of trees, the simulation is conducted 50 independent times on randomly created instances. Then, the average value is calculated and plotted. We can see that for every number of trees, the actual performance of Algorithm 4 is much better than the worst-case theoretical performance. For 200 nodes (Figure 4.3(a)), this ratio ranges between 1.04 for 2 trees and 1.19 for 10 trees, while the theoretical ratio ranges between 2 and 10. For 2000 nodes (Figure 4.3(b)), the performance of the approximation algorithm is even better. Thus, we believe that the approximation algorithm introduces an excellent combination of good performance and low complexity.

We can see that the relative performance of Algorithm 4 improves when the transmission range increases, because when it does, more and more nodes of each tree are leaves, and the percent of inner nodes in the tree decreases. In this case, most of the nodes are either not part of any inner tree or belong to no more than one of the inner trees. In the former case the nodes have a 0 wake-up frequency under both algorithms. In the latter case the nodes are assigned the same (non-zero) wake-up frequency under both algorithms.

In Figure 4.4 we study the effect of assigning a different upper bound $D$ on the end-to-end delay to five different trees. For some of the trees $D = 100$ while for the others $D = 500$. The upper curve shows the average node wake-up frequency when all the trees have $D = 100$. The second curve from the top shows the case when 4 trees have $D = 100$ and one has $D = 500$, the third when 3 trees have $D = 100$, the fourth when 4 trees have $D = 100$, and the fifth when 5 trees have $D = 100$. As expected, it can be seen from the upper curve that the average wake-up frequency of the first tree is approximately 5 times greater than that of the last one. We also observe that the gaps between the upper curves are smaller than those of the lower curves. This can be explained by the fact that the inner trees are not completely disjoint, but have a small number of common nodes. These nodes are assigned the maximum wake-up frequency among all 5 assignments.
4.4.2 METP Approximation Simulations

To study the efficiency of the tree construction algorithm, we use similar settings to those described in Section 4.4.1. We build a sensor network by randomly placing 1,000 nodes in a 100x100 grid. The transmission range varies between 15% and 50% of the whole grid. For each range $r$, any two nodes whose Euclidean distance is not greater than $r$ are considered to be connected. If the created graph is not connected, this instance is ignored. If it is connected, we select a random node to function as the gateway. The nodes are assumed to have similar per wake-up energy expenditure: namely, for every node $v$, $c_v = 1$. The value of $\tau$ is taken to be infinite and the upper bound $D$ on the end-to-end delay is 100 time units.

We compare a BFS tree to Algorithm 5. Because we take $c_v = 1$ for every $v$, we can approximate METP using a minimum connected dominating set (MCDS) rather than a minimum-weighted connected dominating set. An MCDS is constructed using the algorithm presented in [58].

For each simulation instance we perform the following steps:

1. Build two spanning trees: one by Algorithm 5 and another by a BFS algorithm.
2. Invoke the ST-OFA algorithm on each tree in order to assign optimal wake-up frequencies to the nodes.

Figure 4.5(a) shows the average wake-up frequency of the nodes as a function of the transmission range for the two spanning trees. It turns out that the average wake-up frequency assigned to
a tree built by Algorithm 5 is half of the value assigned to a BFS tree. This indicates that even if an optimal algorithm is used for wake-up frequency assignment, it is crucial to optimize the tree structure.

Figure 4.5(b) shows the number of inner nodes in both trees. It is evident that the relation between the inner tree sizes is similar to the relation between the average wake-up frequency assignment. This indicates that an efficient tree is indeed one whose set of inner nodes is minimal.

Finally, carry out combined simulations. In one instance, multiple trees are created on a random graph using Algorithm 5, and wake-up frequencies are assigned using Algorithm 4. In another instance, the trees are constructed using a BFS algorithm, and the wake-up frequencies are again assigned using Algorithm 4. Figure 4.6(a) shows the simulation results for the BFS tree and Figure 4.6(b) for the tree constructed by Algorithm 5. It is evident that for every number of trees, Algorithm 5 significantly reduces the average wake-up frequency of the nodes.

4.4.3 Simulating Efficiency of Multiple Spanning Trees

The main advantage of using multiple trees is improved reliability. In a single routing tree, neighboring nodes that discover an event usually follow very similar paths to the gateway. Hence, a single failure may disconnect all the reporting nodes from the gateway. However, if several routing trees are used, and each node randomly chooses a gateway, at least some of the messages are likely to reach a one of the chosen gateways even if a failure occurs. In Figure 4.7(a) we compare
this aspect of reliability for a single and for multiple gateways. In this simulation study, one or more trees are constructed on a random graph. We simulate the worst case, which is the failure of a node adjacent to a gateway. These nodes are likely to have more descendants than other nodes. We randomly choose one of the trees, denoted by $T$, whose root is the gateway $g$. We then randomly choose a node $v$ adjacent to $g$, and remove all $v$’s edges from the graph. Consequently, all the descendants of $v$ in $T$ are disconnected from $g$. If multiple trees are used, each of these disconnected nodes is likely to have a path to another gateway. Figure 4.7(a) shows the percent of nodes that have no gateway connectivity for 1, 2 and 3 trees. We can see that for 2 routing trees the number of disconnected nodes is dramatically reduced. For 3 trees this number is often 1, which indicates that only the failed node loses connectivity.

Creating and maintaining multiple trees is more energy demanding than a single tree. However, the various algorithms proposed in this chapter reduce this extra energy cost. This is especially true for distant gateways, because energy expenditure in the vicinity of the gateways is greater. This is clearly shown in Figure 4.7(b), where we simulate the network lifetime for 1-5 spanning trees to which wake-up frequencies are assigned using Algorithm 4. Each node is equipped with some initial amount of energy, which allows the node to be active for some fixed time duration. After every time unit, the energy of each node is reduced according to the node’s wake-up frequency. A node whose energy runs below 20% of its initial value can no longer act as an inner node. Therefore, new trees are built, for which every inner node has at least 30% of its initial energy. If a spanning tree cannot be rebuilt, the network is considered “dead.”

Figure 4.6: Comparing two $\tau/D$-optimal trees
Figure 4.7: The effect of multiple trees

Figure 4.7(b) shows the ratio between the average lifetime of a network with a single gateway and networks with 2-5 gateways, as a function of the transmission range. The extra reliability provided by multiple trees reduces the network lifetime by 35% for 2 trees and by 57% for 5 trees.
Chapter 5

Continuous Neighbor Discovery in Asynchronous Sensor Networks

5.1 Introduction

Despite the static nature of the sensor nodes, after the network has been established its connectivity is still subject to changes. In particular, even after a sensor node is aware of its immediate neighbors, it must continuously look for new ones in order to accommodate the following situations:

1. Loss of local synchronization due to accumulated clock drifts.

2. Disruption of wireless connectivity between adjacent nodes by a temporary event, such as a passing car or animal, a dust storm, rain or fog. When these effects disappear, the hidden nodes must be rediscovered.

3. The ongoing addition of new nodes, in some networks to compensate for nodes which have ceased to function because their energy has been exhausted (so-called dead nodes).

4. The increase in transmission power of some nodes, in some networks, in response to certain events, such as loss of connectivity with neighboring nodes or detection of important local happening.

For these reasons, detecting new links and nodes in sensor networks must be considered as an ongoing process. In the following discussion we distinguish between the detection of new links and nodes during initialization and their detection during normal operation. The former will
be referred to as initial neighbor discovery whereas the latter will be referred to as continuous neighbor discovery. While previous works [28, 37, 39] address initial neighbor discovery and continuous neighbor discovery as similar tasks, to be performed by the same protocol, we claim that they should be addressed by different protocols for the following reasons:

- Initial neighbor discovery is usually performed when the sensor has no clue about the structure of its immediate surroundings. In particular, the sensor cannot communicate with the gateway, and is therefore very limited in performing its tasks. Hence, in this state for the sake of quicker detection, more extensive energy use is justified. It is very important to detect the immediate surroundings as soon as possible in order to establish a path to the gateway and to contribute to the operation of the network. In contrast, continuous neighbor discovery is performed when the sensor is already operational. This is a long-term process whose optimization is crucial for increasing the network life time.

- When the sensor performs continuous neighbor discovery, it is already aware of most of its immediate neighbors. It can therefore perform continuous neighbor discovery together with these neighbors in order to consume less energy. In contrast, initial neighbor discovery is an individual task, that must be executed by each sensor separately.

We now show, by means of an example, why an initial neighbor discovery protocol is inefficient for continuous neighbor discovery. Figure 5.1 presents a simple protocol. In this figure we assume that node $u$ is in the initial neighbor discovery state, where its main task is to search for new neighbors. To this end, it periodically wakes up, at random times, and transmits a bunch of HELLO messages (the bunch size in the figure is 3). In the figure we see that the first 5 bunches of HELLO messages are transmitted when node $v$ is sleeping, and therefore they cannot be received by $v$. The 6th bunch is transmitted when $v$ is in active mode. Therefore, $v$ is likely to receive at least one message of the 6th bunch, to which it responds with HELLO-ACK. From this time, the two nodes view each other as neighbors, and they maintain this relationship using periodic HELLO messages.

If a hidden node has duty cycle of 1%, we can assume that the discovering node $v$ “hits” $u$ when $u$ is awake with probability of 1%. In this case, by the rules of geometric distribution, the discovery demands, in average, 100 bunches of HELLO messages. Hence, in order to guarantee the average discovery time of 10 seconds, $u$ has to wake up every 0.1 second. Even if every wake-up lasts only 10 msec, it gives us the duty cycle of 10 %, thereby expending a lot of its energy on finding its neighbors. Working with such a duty cycle might be reasonable only when node $u$
is added to the network, i.e., in the neighbor discovery state, but not as an ongoing algorithm for continuous neighbor discovery.

To summarize, a node in the initial neighbor discovery state should not use the same neighbor discovery algorithms used by a node in the continuous neighbor discovery state. In the initial neighbor discovery state a node has no information about its surroundings, and therefore must remain active for a relatively long time in order to detect new neighbors. In contrast, a node in the continuous neighbor discovery state should use a more efficient algorithm. This algorithm is the subject of our study.

We distinguish between initial and continuous neighbor discovery in sensor network. Figure 5.2 summarizes this idea. When node $u$ is initialized, it performs initial neighbor discovery. After a certain time period in the initial neighbor discovery state, during which the node is expected, with high probability, to find most of its neighbors, the node moves to the continuous neighbor discovery state. The main idea behind the continuous neighbor discovery scheme proposed in this chapter is that the task of finding a new node is divided among all the nodes in its vicinity.

Throughout this chapter, a node in the initial neighbor discovery state is referred to as a hidden node. A node in the continuous neighbor discovery state is referred to as a segment node.

The rest of this chapter is organized as follows. Section 5.2 presents our basic scheme and problem definition. The core of this chapter is Section 5.3, which presents three approaches for estimating the in-segment degree of a hidden neighbor and analyzes their accuracy. Section 5.4 presents our continuous neighbor discovery protocol, which is based on our findings in Section 5.3.
Continuous neighbor discovery vs. initial neighbor discovery in sensor networks

Section 5.5 presents simulation results that demonstrate the efficiency of the proposed protocol. It also discusses issues related to the detection of two small segments by each other.

5.2 Basic Schemes and Problem Definition

In the following discussion, two nodes are said to be neighboring nodes if they have direct wireless connectivity. We assume that all nodes have the same transmission range, which means that connectivity is always bidirectional. For our analysis we also assume that the network is a unit disk graph; namely, any pair of nodes that are within the transmission range of each other are neighboring nodes. Two nodes are said to be directly connected if they have discovered each other and they are aware of the wake up times of each other. Two nodes are said to be connected, if there is a path of directly connected nodes between them. A set of connected nodes is referred to as a segment. Consider a pair of neighboring nodes that belong to the same segment but are not aware that they have direct wireless connectivity. See, for example, nodes $a$ and $c$ in Figure 5.3(a). These two nodes can learn about their hidden wireless link using the following simple scheme.

Scheme 1 (detecting a hidden link inside a segment) One of the segment nodes issues a special SYNC message to all segment members, asking them to wake up and periodically broadcast a bunch of HELLO messages. This SYNC message is distributed over the already known wireless links of the segment. Thus, it is guaranteed to be received by every node in the segment. By having all the nodes wake up “almost at the same time” for a short period, we can ensure that all the wireless links between the segment’s members will be detected with minimum energy cost.

This scheme needs to be involved only when a new node is discovered by one of the segment nodes. The discovering node will also be the node, that triggers the protocol.

53
One may argue that Scheme 1 actually synchronizes the segment’s nodes, which is known to be inefficient [8, 9, 10]. However, this global synchronization is not used for regular message exchange. Rather, it is infrequently invoked, e.g., once a minute, and it therefore imposes very small energy overhead. For an illustration of a simple way to implement Scheme 1, suppose that every node wakes up once a second in order to receive messages from its in-segment neighbors. Suppose also that the node stays active for about 10 milliseconds, thereby having a duty cycle of 0.1%. In this case, the SYNC message can reach every node in the segment within at most $D$ seconds, where $D$ is the distance between the segment leader and the farthest node. The SYNC message carries a WAKE-UP-TIME field, which is initialized to $D$ and decremented by $t_{(v,u)}$ by every node $v$ that transmits the SYNC, where $t_{(v,u)}$ is the interval between the time $v$ receives the SYNC and the time it transmits it to its in-segment neighbor $u$. Since this scheme is not frequently involved, we can allow nodes to remain active for a relatively long period of time, compensating for possible synchronization inaccuracy.

Scheme 1 allows $u$ and $v$ to discover each other if they belong to the same connected segment. However, as discussed in Section 5.1, in order for two neighbors not yet connected to the same segment to detect each other, each node should also execute the following scheme:

**Scheme 2 (detecting a hidden link outside a segment)** Node $u$ wakes up randomly, every $T(u)$ seconds on the average, for a fixed period of time $H$. During this time it broadcasts several HELLO messages, and listens for possible HELLO-ACK messages sent by new neighbors. The value of $T(u)$ is as follows:

- $T(u) = T_1$, if node $u$ is in the initial neighbor discovery state of Figure 5.2.
- $T(u) = T_N(u)$, if node $u$ is in the continuous neighbor discovery state of Figure 5.2, where $T_N(u)$ is computed according to the scheme presented in Section 5.3.

A random wake-up approach is used to minimize the possibility of repeating collisions between the HELLO messages of nodes in the same segment. Theoretically, another scheme may be used, where segment nodes coordinate their wake-up periods to prevent collisions and speed up the discover of hidden node. However, finding an efficient time division is equivalent to the well known node coloring problem, which is NP-hard and also cannot be well approximated even if the graph is known to every node (which is not our case). In any case, since that the time period during which every node wakes up is very small, and the beacon transmission time is even smaller, the probability for two neighboring nodes active at the same time is practically 0. In the
rare case of collisions, CSMA/CD can be used to schedule retransmissions. Note also, that under the “wake-up synchronization” scheme there exists a very undesired scenario, where a hidden node chooses to wake-up at some slot, which is not covered by its neighbors wake-ups. In this case the hidden neighbor will be never discovered, unless the wake-up coordination is rescheduled, for some reason.

By Scheme 1, the discovery of an individual node by any node in a segment leads to the discovery of this node by all of its neighbors that are part of this segment. Therefore, discovering a node that is not yet in the segment can be considered a joint task of all the neighbors of this node in the segment. As an example, consider Figure 5.3(a), which shows a segment $S$ and a hidden node $u$. In this figure, a dashed line indicates a hidden wireless link, namely, a link between two nodes that have not yet discovered each other. A thick solid line indicates a known wireless link. After execution of Scheme 1, all hidden links in $S$ are detected (see Figure 5.3(b)). The links connecting nodes in $S$ to $u$ are not detected because $u$ does not belong to the segment. Node $u$ has 4 hidden links to nodes in $S$. Hence, we say that the degree of $u$ in $S$ is $\deg_S(u) = 4$. When $u$ is discovered by one of its four neighbors in $S$, it will also be discovered by the rest of its neighbors in $S$ as soon as Scheme 1 is reinvoked. Consider one of the four segment members that are within range of $u$, node $v$ say. Although it may know about the segment members within its own transmission range, it does not know how many segment neighbors participate in discovering $u$.

In the next section we consider several methods that allow $v$ to estimate the value of $\deg_S(u)$ for a hidden node $u$, and compare their accuracy and applicability.
5.3 Estimating the In-Segment Degree of a Hidden Neighbor

As already explained, we consider the discovery of hidden neighbors as a common task to be performed by all segment nodes. To determine the discovery load to be imposed on every segment node, we need to estimate the number of in-segment nodes that are neighbors of every hidden node. That is the in-segment degree of the hidden neighbor, denoted by $\deg_S(u)$. In this section we present methods that allow node $v$ in the continuous neighbor discovery state to estimate the number $\deg_S(u)$ of in-segment neighbors of its hidden neighbor $u$. Node $u$ is assumed not to be connected to the segment yet, and it is in the initial neighbor discovery state. Three methods are presented:

1. Node $v$ measures the average in-segment degree of the segment’s nodes, and uses this number as an estimate of the in-segment degree of $u$. The average in-segment degree of the segment’s nodes can be calculated by the segment leader. To this end, it gets from every node in the segment a message indicating the in-segment degree of the sending node, which is known due to Scheme 1. We assume that the segment size is big enough for the received value to be considered equal to the expected number of neighbors of every node.

2. Node $v$ discovers, using Scheme 1, the number of its in-segment neighbors, $\deg_S(v)$, and views this number as an estimate of $\deg_S(u)$. This approach is expected to yield better results than the previous one when the degrees of neighboring nodes are strongly correlated.

3. Node $v$ uses the average in-segment degree of its segment’s nodes and its own in-segment degree $\deg_S(v)$ to estimate the number of node $u$’s neighbors. This approach is expected to yield the best results if the correlation between the in-segment degrees of neighboring nodes is known. A special case is when the in-segment nodes are uniformly distributed.

The in-segment degree of $v$ and $u$ depends on how the various nodes are distributed in the network. Let $X$ be a random variable that indicates the degree $\deg_S(v)$ of $v$ in the segment $S$. Let $Y$ be a random variable that indicates the degree $\deg_S(u)$ of $u$ in $S$. Note that $u$ itself is not aware of the value of $Y$. Let $Y'$ be the value of $Y$ estimated by $v$. Clearly, we want $Y'$ to be as close as possible to $Y$. In what follows we analyze the three methods considered above and compare their accuracy and applicability. Since the in-segment degree of both the segment node ($v$) and the non-segment node ($u$) may have different values for different segment nodes, we use the mean square error measure ($MSE$) to decide how good the estimate is. The $MSE$ is defined as $E((Y - Y')^2)$. Since $v$ and $u$ are two random nodes in the same graph, we can claim that $X$ and
have the same distribution. Let us denote the correlation between \(X\) and \(Y\), \(\text{corr}(X, Y)\), by \(C\). We assume that the node’s average degree is small compared to the network size.

Let us denote the average graph degree by \(\mu\). Clearly, \(E(X) = E(Y) = \mu\). Then, for the first method, the following holds:

\[
MSE_1 = E((Y - Y')^2) = E((Y - \mu)^2) = \text{Var}(Y).
\] (5.1)

For the second method, we have \(Y' = X\). Hence,

\[
MSE_2 = E((Y - Y')^2) = E((Y - X)^2) = \sum_x \sum_y (y - x)^2 P(X = x, Y = y) = \sum_x \sum_y (y^2 - 2xy + x^2) P(X = x, Y = y) = E(X^2) + E(Y^2) - 2E(XY).
\] (5.2)

Now, using the definition of random variables correlation and the fact that \(\text{Var}(X) = \text{Var}(Y)\)

\[
\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\text{Var}(X)}
\]

and using the definition of covariance

\[
\]

Hence

\[
E(XY) = \text{cov}(X, Y) + E(X)E(Y) = \text{corr}(X, Y) \text{Var}(X) + E(X)E(Y) = C \text{Var}(X) + E(X)E(Y).
\] (5.3)
Substituting into Eq. 5.2 and keeping in mind that $X$ and $Y$ have the same distribution

\[\text{MSE}_2 = E(X^2) + E(Y^2) - 2C \text{Var}(X) + E(X)E(Y)\]

\[= E(X^2) + E(X^2) - 2C \text{Var}(X) - 2E(X)E(X)\]

\[= 2E(X^2) - 2E(X)^2 - 2C \text{Var}(X)\]

\[= 2 \text{Var}(X) - 2C \text{Var}(X)\]

\[= (2 - 2C) \text{Var}(X).\]

The third estimation approach is actually a linear prediction problem, namely a problem of finding two constants $\beta$ and $\gamma$ such that $Y' = \beta X + \gamma$ and $E((Y - Y')^2)$ is minimized. By differentiating the $\text{MSE}$ with respect to $\gamma$, we get

\[
\frac{\delta \text{MSE}}{\delta (\gamma)} = \frac{\delta (E((Y' - Y)^2) = E((\beta X + \gamma - Y)^2))}{\delta (\gamma)}
\]

\[= 2\gamma + 2\beta \mu - 2\mu.
\]

Equating the expression result to 0 yields

\[2\gamma + 2\beta \mu - 2\mu = 0\]

\[\gamma = \mu - \beta \mu.\]  \hspace{1cm} (5.4)

In a similar way, differentiating the $\text{MSE}$ expression with respect to $\beta$ yields that

\[
\frac{\delta \text{MSE}}{\delta (\beta)} = 2\beta E(X^2) + 2\gamma \mu - 2E(XY).
\]

We can replace $\gamma$ with the value of $\gamma$ from Eq. 5.4 and get:

\[
\frac{\delta \text{MSE}}{\delta (\beta)} = 2\beta E(X^2) + 2\mu - 2E(XY)
\]

\[= 2\beta E(X^2) + 2(\mu - \beta \mu) \mu - 2E(XY)
\]

\[= 2\beta E(X^2) + 2\mu^2 - 2\beta \mu^2 - 2E(XY).
\]
Therefore, the value of $\beta$ that brings $MSE$ to its minimum is equal to

$$
\beta = \frac{\mu^2 - E(XY)}{\mu^2 - E(X^2)} = \frac{\text{cov}(X,Y)}{\text{Var}(X)}.
$$

Since we know the correlation $\text{corr}(X,Y)$ and $X$ and $Y$ have similar distribution,

$$
\text{cov}(X,Y) = \text{corr}(X,Y) \text{ Var}(X)
= C \times \text{Var}(X).
$$

We conclude that $\beta = \frac{C \times \text{Var}(X)}{\text{Var}(X)} = C$ and $\gamma = \mu - \beta \mu = (1 - C)\mu$ are the values that minimize $MSE$. The exact value of $MSE_3$ is

$$
MSE_3 = E((Y' - Y)^2)
= E((CX + (1 - C)\mu - Y)^2)
= E(C^2 X^2 + 2C(1 - C)X\mu - 2CXY - 2(1 - C)\mu Y + (1 - C)^2\mu^2 + Y^2)
= C^2 E(X^2) + 2C(1 - C)E(X)\mu - 2CE(XY) - 2(1 - C)\mu E(Y) + (1 - C)^2\mu^2 + E(Y^2)
= C^2 E(X^2) + E(Y^2) + (2C - C^2 - 1)\mu^2 - 2CE(XY)
$$

Using the fact that $X$ and $Y$ have the same distribution and substituting the value of $E(XY)$ from Eq. 5.3

$$
MSE_3 = (C^2 + 1)E(X^2) + (2C - C^2 - 1)\mu^2
- 2C(C \times \text{Var}(X) + \mu^2)
= (C^2 + 1)E(X^2) - (C^2 + 1)\mu^2 - 2C^2 \text{Var}(X)
= (C^2 + 1)(E(X^2) - \mu^2) - 2C^2 \text{Var}(X)
= (C^2 + 1) \text{Var}(X) - 2C^2 \text{Var}(X)
= (1 - C^2) \text{Var}(X)
$$

Hence, we have the following accuracy for the three estimation approaches:

1. $\text{Var}(X)$
2. $(2 - 2C) \text{Var}(X)$
3. $(1 - C^2) \text{Var}(X)$
Clearly, this results depend on two parameters, the correlation between neighboring nodes degrees and the node degree variance. Let us now analyze one specific case, a network with uniformly distributed nodes. Let us find the correlation between the neighboring nodes degrees. From the definition,

\[
\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\text{Var}(X)} = \frac{E(XY) - \mu^2}{\text{Var}(X)} \quad (5.6)
\]

Now, note that

\[
E(XY) = \sum_{y} \sum_{x} xyP(X = x, Y = y) = \sum_{y} \sum_{x} xyP(X = x|Y = y)P(Y = y) = \sum_{y} [yP(Y = y) \sum_{x} xP(X = x|Y = y)] = \sum_{y} [yP(Y = y)E(X|Y = y)]. \quad (5.7)
\]

We now show how to find \(E(X|Y = y)\), namely, the expected number of neighbors of \(v\) given that the number of neighbors of \(u\) is known and equal to \(y\). The set of neighbors of \(v\) can be divided into two subsets: subset \(A\) includes neighbors of \(v\) that are also neighbors of \(u\); subset \(B\) includes neighbors of \(v\) that are not neighbors of \(u\). In the same way, the set of neighbors of \(u\) can be divided into two sets: the same subset \(A\), and subset \(B'\) of neighbors of \(u\) that are not neighbors of \(v\). Theorem 4 shows the relationship between the neighbors of \(v\) and the neighbors of \(u\):

**Theorem 4** Let \(u, v\) and \(w\) be nodes in a geometric graph with the same transmission range, where nodes are distributed uniformly. If \(u\) is a neighbor of \(v\) and \(v\) is a neighbor of \(w\), then the probability that \(u\) is also a neighbor of \(w\) is \(\mathcal{P} = 1 - \frac{3}{4\pi\sqrt{3}} \approx 0.586503\).

*Proof:* Consider Figure 5.4. Node \(w\) is a neighbor of \(u\) only if it resides in the area marked as \(S_3\). To find the probability that this is indeed the case, we have to find the ratio between \(S_3\) and the unit circle area. Now, note that

\[
S_1 = r \sin \frac{\theta}{2} r \cos \frac{\theta}{2}; S_2 = \pi r^2 \frac{\theta}{2\pi}; S_3 = 2(S_2 - S_1).
\]
Figure 5.4: Definitions for the proof of Theorem 4

Since $\cos \frac{\alpha}{2} = \frac{x}{2r}$, where $x$ is the distance between $u$ and $v$, then $\frac{\alpha}{2} = \arccos \frac{x}{2r}$ holds. Hence, we can write:

$$S_1 = \frac{x}{2} r \sqrt{1 - \frac{x^2}{4r^2}}; \quad S_2 = \frac{r^2}{2} 2 \arccos \frac{x}{2r}.$$

Thus, $S_3 = 2r^2 \arccos \frac{x}{2r} - xr \sqrt{1 - \frac{x^2}{4r^2}}$. A neighbor of $v$ is also a neighbor of $u$ only if it lies inside $S_3$. The probability for this is $\frac{S_3}{\pi r^2}$. Denote the probability for such an event as $P_x$, where $x$ is the distance between $u$ and $v$. Hence,

$$P_x = \frac{2}{\pi} \frac{x}{2r} \arccos \frac{x}{2r} - \frac{x}{\pi r} \sqrt{1 - \frac{x^2}{4r^2}}.$$

In order to find the probability $P$ that $w$ is a neighbor of $u$, we should consider all possible values of $x$, from 0 to $r$, while taking into account the density function:

$$P = \frac{1}{\pi r^2} \int_{x=0}^{r} 2\pi x P_x \, dx$$

$$= \frac{2}{r^2} \int_{x=0}^{r} x \left( \frac{2}{\pi} \arccos \frac{x}{2r} - \frac{x}{\pi r} \sqrt{1 - \frac{x^2}{4r^2}} \right) \, dx$$

$$= \frac{2}{r^2} \left( \int_{x=0}^{r} \frac{2}{\pi} x \arccos \frac{d}{2r} \, dx \right) - \frac{2\pi}{r^2} \left( \int_{x=0}^{r} \frac{1}{\pi r} x^2 \sqrt{1 - \frac{x^2}{4r^2}} \, dx \right)$$

$$= \frac{2}{r^2} \frac{2}{\pi} \left[ -\frac{1}{2} x^{\frac{3}{2}} \sqrt{1 - \frac{x^2}{4r^2}} + \frac{1}{2} x^2 \arccos \frac{x}{2r} \right]_0^r + \frac{2}{r^2} \frac{2}{\pi} \left[ r^2 \arcsin \frac{x}{2r} \right]_0^r$$

$$- \frac{2}{r^2} \frac{1}{\pi r} \left[ \frac{1}{8} x (-2x^2 + x^2) \sqrt{4 - \frac{x^2}{4r^2}} \right]_0^r - \frac{2}{r^2} \frac{1}{\pi r} \left[ r^3 \arcsin \frac{x}{2r} \right]_0^r.$$
Substituting the integration limits yields:

\[
P = \frac{4}{\pi r^2} \left[ -\frac{r^2}{2} \sqrt{1 - \frac{1}{4} + \frac{r^2}{2}} \arccos \frac{1}{2} + \frac{r^2}{2} \arcsin \frac{1}{2} \right] - \frac{2}{r^2} \left( \frac{1}{8} (-r^2) \sqrt{3} + \frac{r^3}{6} \arcsin \frac{1}{2} \right)
\]

\[
= \frac{4}{\pi r^2} \left[ -\frac{1}{2} r^2 \sqrt{\frac{3}{4} + \frac{1}{2} \frac{2\pi}{3} + \frac{r^2}{6}} \right] - \frac{2}{r^2} \left( \frac{1}{8} r^2 \sqrt{3} + \frac{r^3 \pi}{6} \right)
\]

\[
= 2 \left( \left[ -\frac{1}{2\pi\sqrt{3}} + \frac{1}{3} \right] - \left[ -\frac{1}{8\pi\sqrt{3}} + \frac{1}{6} \right] \right)
\]

\[
= \frac{1}{\pi \sqrt{3}} + \frac{4}{3} + \frac{1}{4\pi \sqrt{3}} - \frac{1}{3}
\]

\[
= 1 - \frac{3}{4\pi \sqrt{3}} \approx 0.586503.
\]

Following Theorem 4, we conclude that (a) the expected size of subset \( A \) is equal to the number of neighbors of \( u \) multiplied by \( P \), namely \( E(|A|) = P \deg_S(u) \); (b) the expected size of subset \( B \) is equal to the average graph degree multiplied by \((1 - P)\), where \((1 - P)\) is the part of the area covered by \( v \) but not by \( u \), as follows from Theorem 4. Since the degree of \( v \) is \(|A| + |B|\), we get

\[
E(X|Y = y) = P y + (1 - P) \mu.
\]

Substituting this into Eq. 5.7 yields:

\[
E(XY) = \sum_y y P(Y = y)(P y + (1 - P) \mu)
\]

\[
= \sum_y P y^2 P(Y = y) + (1 - P) \mu \sum_y y P(Y = y)
\]

\[
= P E(Y^2) + (1 - P) \mu^2.
\]

62
Therefore, substituting into Eq. 5.6 we get

\[
\text{corr}(X, Y) = \frac{E(XY) - \mu^2}{\text{Var}(X)} = \frac{\mathcal{P}E(Y^2) - \mathcal{P}\mu^2 + \mu^2 - \mu^2}{\text{Var}(X)} = \frac{\mathcal{P}\text{Var}(Y)}{\text{Var}(X)} = \frac{\mathcal{P}\text{Var}(X)}{\text{Var}(X)} = \mathcal{P}
\]

Hence, if we assume uniform distribution of nodes, the three estimation approaches have the following accuracy.

1. \(\text{Var}(X)\)

2. \(\approx 0.84 \text{Var}(X)\)

3. \(\approx 0.66 \text{Var}(X)\)

We see that the third approach yields the best (smaller) MSE. However, note that this approach requires some global knowledge of the network topology, while the second approach requires only local knowledge.

Following our network model, the node degree has the binomial distribution, where the probability of success \(p\) is the probability that a node \(v\) will fall in the transmission range of the node \(u\). This probability is the relation between the area covered by a node \(v\) and the area occupied by the whole network. For this kind of distribution, the variance is known to be \(np(1-p)\), where \(n\) is the number of nodes. Note that the \(np\) is the expected node’s degree, \(\mu\).

And summarizing the discussion, it can be seen, that for greater values of correlation the second and the third schemes become even more accurate, and as the correlation approaches 1, the estimation error approaches 0. For example, in a sensor network which is build to have denser and sparser areas, the correlation between node degrees is greater than in the uniformly distributed network, hence we expect that the second and the third schemes are even more accurate in this case. For smaller correlation the first estimation scheme is more accurate than the second and similar to the third, but, thinking of any realistic network, which have large number of nodes, the correlation between the neighboring node degrees can not be less than this of the uniform distributed nodes. It follows from the geometrical properties of the network. The only network
that may have smaller correlation is the network where the node degree has the same order as the number of network nodes. In this case, for example, if we have a node with large degree, we would expect its neighbors to have lower degree, just because there are no nodes left for them.

5.4 An Efficient Continuous Neighbor Discovery Algorithm

Suppose that node $u$ is in initial neighbor discovery state, where it wakes up every $T_I$ seconds for a period of time equal to $H$, and broadcasts HELLO messages. Suppose that the nodes of segment $S$ should discover $u$ within a time period $T$ with probability $P$. Finally, suppose that each node $v$ in the segment $S$ is in continuous neighbor discovery state, where it wakes up every $T_N(v)$ seconds for a period of time equal to $H$, and broadcasts HELLO messages.

We assume that in order to discover each other, nodes $u$ and $v$ should have an active period that overlaps by at least a portion $\delta$, $0 < \delta < 1$, of their size $H$. Thus, if node $u$ wakes up at time $t$ for a period of $H$, node $v$ should wake up between $t - H(1 - \delta)$ and $t + H(1 - \delta)$. The length of this valid time interval is $2H(1 - \delta)$. Since the average time interval between two wake-up periods of $v$ is $T_N(v)$, the probability that $u$ and $v$ discover each other during a specific HELLO interval of $u$ is $\frac{2H(1-\delta)}{T_N(v)}$.

Let $n$ be the number of in-segment neighbors of $u$. When $u$ wakes up and sends HELLO messages, the probability that at least one of its $n$ neighbors is awake during a sufficiently long time interval is $1 - (1 - \frac{2H(1-\delta)}{T_N(v)})^n$.

For the sake of our analysis, consider a division of the time axis of $u$ into time slots of length $H$. The probability that $u$ is awake in a given time slot is $\frac{H}{T_I}$, and the probability that $u$ is discovered during this time slot is $P_1 = \frac{H}{T_I}(1 - (1 - \frac{2H(1-\delta)}{T_N(v)})^n)$. Denote by $D$ the value of $\frac{T}{T_I}$. Then, the probability that $u$ is discovered within at most $D$ slots is $P_2 = 1 - (1 - P_1)^D$. Therefore, we seek the value of $T_N(v)$ that satisfies the following equation:

$$1 - (1 - P_1)^D \geq P$$

which can also be stated as

$$P_1 \geq 1 - \frac{P}{\sqrt{1 - P}}.$$ 

Since $P_1 = \frac{H}{T_I}(1 - (1 - \frac{2H(1-\delta)}{T_N(v)})^n)$, we get

$$\frac{H}{T_I}(1 - (1 - \frac{2H(1-\delta)}{T_N(v)})^n) \geq 1 - \frac{P}{\sqrt{1 - P}},$$

64
Figure 5.5: $T_N(v)$ as a function of maximum tolerated delay

$$T_N(v) \leq \frac{2H(1 - \delta)}{1 - \sqrt{1 - \frac{P}{H}(1 - \sqrt{1 - P})}}.$$

Since $v$ does not know the exact value of $n$, it can estimate it using the schemes presented in Section 5.3.

In Figure 5.5 we present two graphs that show the dependency between $T$ and $T_N(v)$. We assume that a hidden node wakes up once every 100$H$ time units on the average, and that $T_I = 100$, $H = 1$, and $\delta = 0.5$. In Figure 5.5(a) the estimated value of $n$ is 10. The curves present the value of $T_N(v)$ as a function of the desired discovery time $T$ for 3 different values of $P$: 0.5, 0.8 and 0.95. In Figure 5.5(b) $P$ is set to 0.8 and $n$ varies between 5 and 50. Again, $T_N(v)$ is calculated as a function of the desired discovery time. As expected, the nodes have to work harder to achieve greater discovery rate in less time, while the increase in the density of in-segment nodes allows to choose greater $T_N(v)$. In both graphs the dependency between $T_N(v)$ and $T$ is almost linear and, as we can see in Figure 5.5(b), the slope of the curves is almost linear in the value of $n$ as well. This means that, for simpler implementation, a node $v$ can use linear approximation to compute the value of $T_N(v)$. Another interesting observation can be made, regarding the accuracy of the achieved neighbor discovery probability. Indeed, the discovery time actually depends on the overall number of HELLO messages transmitted in the vicinity of a hidden node. This number depends on the degree of this node, as estimated by the segment nodes. In dense networks the estimation degree errors of different node’s neighbors cancel each other. Since the frequency of
HELLO messages transmitted by a segment node depends linearly on the estimation result, we can also claim that the overall frequency of HELLO messages is very close to the “correct” one, namely, the estimation errors of the various nodes cancel each other. This result can be seen in the various graphs of Section 5.5.

If we need to enforce some average hidden neighbor discovery delay but also to achieve some upper bound on this delay, another wake-up scheme, inspired by the Disco algorithm [38], may replace the random wake-up scheme, proposed in this section. Recall that the Disco algorithm, in its simplified version, proposes each node to pick a number. Each node $v$ that has chosen a number $p$ should wake up every $p$ slots. Hence, assuming that a pair of neighboring nodes that wish to discover each other has co-prime numbers $p_v$ and $p_u$, these nodes will discover each other within at post $p vp_u$ slots. In our case we can simplify the task in the following way. Each node in the initial neighbor discovery state chooses some predefined prime number, and each node in the continuous neighbor discovery state chooses a number according to the required discovery time and the estimated in-segment degree of its hidden neighbor. This way, the number chosen by a hidden node and the number chosen by its in-segment neighbor are always co-prime. The in-segment node should perform the following calculation. If a hidden node wakes up every $T_I$ slots and it has $k$ in-segment neighbors, each wakes-up every $T_N$ slots, the active periods overlap within at most $T_IT_N$ slots and, assuming uniform distribution of the wake-up slots, in average the hidden node is discovered within $(T_IT_N)/(2k)$ slots. Hence, in order to achieve the average discovery time of $D$ the in-segment node $v$ should choose the wake up period $T_N = 2Dk/T_I$. However, note that (a) this scheme is less flexible than the first one, (b) although the average discovery time is constant, the upper bound on the discovery time grows in dense networks and (c) if the wake-up periods of the in-segment nodes become synchronized for some reason, the variance of the neighbor discovery time may grow significantly.

5.5 Simulation Study

5.5.1 Simulation of the proposed schemes

In this section we present a simulation study for the schemes presented in this chapter. Our goal is to simulate a large sensor network, with nodes distributed randomly and uniformly over the area of interest. We assume that the nodes have an equal and constant transmission range. Communication is always bi-directional. We also assume that most of the nodes already discovered each other before our algorithm is implemented.
Our simulation model consists of 2,000 sensor nodes, randomly placed over a 10,000 x 10,000 grid. The transmission range is set to $r$ units. Any two nodes whose Euclidean distance is not greater than $r$ are considered to have wireless connectivity. A portion of the nodes are randomly selected to be hidden. These nodes are uniformly distributed in the considered area. We set the algorithm parameters such that every hidden node will be detected with probability $P$ within a predetermined period of time $T$. For the study reported in this section, $r$ is chosen to be 300, and the detection probability ranges between 0.3 and 0.7, and the target detection time is 100 time units.

The hidden nodes are assumed to be in the initial neighbor discovery state, where they are supposed to wake up randomly, every $T_I$ time units on the average, and to exchange HELLO messages with other nodes during a period of $H$ time units. A non-hidden node $v$ is assumed to be in the continuous neighbor discovery state, where it wakes up randomly, every $T_N(v)$ time units on the average for a period of $H$ time units, in order to discover hidden nodes. For the study reported in this chapter, $T_I = 20$, $H = 1$ and $\delta = 0.5$. When a node is detected, it joins the segment and immediately learns about its in-segment neighbors. A hidden node that detects another hidden node remains in the initial neighbor discovery state and does not establish a wireless link.

Our simulations reveal that when the hidden nodes are uniformly distributed around the grid, our three algorithms yield very similar results. For short transmission range, the main reason is the small node’s degree variance, which makes the three algorithms similar. For longer transmission ranges, the main factor is the high average degree. In this case the estimation errors are likely to cancel each other. Therefore, in most of the graphs we show only the results of one algorithm (Algorithm 3).

Figure 5.6(a) shows the ratio of hidden nodes to the total number of nodes as a function of time. The initial ratio is 0.05. We can see that after 100 time units, this ratio decreases to 0.035 for $P = 0.3$, to 0.025 for $P = 0.5$, and to 0.015 for $P = 0.7$. After 200 time units, the ratio of the hidden nodes are about 0.025, 0.012 and 0.005 for the desired detection probabilities of 0.3, 0.5 and 0.7 appropriately.

The next simulation use a bit different setup. In this case, to emphasis the effect of the algorithm, we choose to start the run with the hidden nodes rate of 0.5. Figure 5.6(b) shows the change in the average frequency of HELLO intervals of the in-segment nodes, as a function of time, for the same three values of $P$. We can see that for the smaller value of $P$ (the lower curve), the frequency is almost 75% lower than the frequency for the larger value of $P$. We can also see that for a given value of $P$, the average frequency of HELLO periods decreases with the time. This is because the segment grows, and more nodes participate in the discovery process. This change
Figure 5.6: Hidden neighbor detection, for the case of uniform distribution

exists also in the case of the initial hidden node ratio of 0.05, as in the previous simulation, but it can hardly be observed due to the small change in the segment size during the simulation.

Another interesting case is when the hidden nodes are distributed non-uniformly in the area. To simulate this case, we randomly select some points as “dead areas,” and assume that the probability of a node to be hidden increases when its distance to one of these points decreases. The rationale here is that bad weather, dust storms, or other environmental conditions may adversely affect wireless connectivity in some areas more than in others. Unlike the uniform distribution case, here we do see differences between the three estimation algorithms presented in Section 5.3.

Figure 5.7(a) shows the percent of hidden nodes as a function of time for the three estimation algorithms and $P = 0.5$. First, we can see that the results here are somewhat similar to those depicted by the middle curve ($P=0.5$) in Figure 5.6(a). We can also see that the third algorithm results are somewhere between the results of the first and the second algorithms. This is not surprising, since the third estimation scheme is a linear combination of the first two. The first algorithm is less efficient because it assigns the same value of $T_N(v)$ to all the nodes in the segment. Thus, nodes that are located close to the “dead areas” invest less energy than needed in the discovery process, whereas nodes that are located far from this area invest more energy than needed. In contrast, the second estimation algorithm underestimates the number of in-segment neighbors of the nodes that are located closer to the dead zone, than the estimating node. Hence, these in-segment nodes work harder than necessary to detect their hidden neighbors. The last algorithm gives the best estimate in this case, and it therefore yields the most efficient topology management scheme.

In Figure 5.7(b) we show the ratio of hidden nodes after $T$ for networks with different trans-
mission ranges, and hence with different node average degrees. The purpose of this graph is to reveal the flexibility of our protocol and its ability to adjust the wake-up frequency to the network density. We show this by comparing our protocol to a trivial protocol that does not take the network density into account. It turns out that the trivial protocol is too aggressive in dense networks and too unaggressive in sparse networks. Recall that the goal of our protocol is not to discover nodes as fast as possible, but to save as much energy as we can while imposing an upper bound on the discovery time. In light of this goal, it turns out that our protocol has a better performance not only when it finds the hidden nodes more rapidly, but also in the left part of the graph, where it finds the hidden nodes less rapidly. The important point is that its discovery rate is fixed, and so its overall expended energy.

The simulation starts with 5% hidden nodes, and each node in Init is configured with $P = 0.5$. For all transmission ranges, our protocol indeed guarantees that after $T$ time units the number of hidden nodes will decrease by about a half to 2.5%. For the trivial protocol all the nodes have the same wake-up frequency. The actual values depend on the wake-up frequency of the nodes and are unimportant. The interesting point is the curve behavior as a function of the transmission range. The trivial protocol discovers half of the hidden nodes only when the transmission range is $\approx 0.06$. When the transmission range is shorter, the trivial protocol discovers a smaller fraction of the hidden nodes. For instance, for a range of 0.03, the ratio of hidden nodes is reduced from 0.05 to 0.04. When the transmission range is greater than 0.06, the trivial protocol discovers more nodes during a time period of $T$. But this is, of course, at the expense of much more energy than needed by our protocol. We conclude that our algorithm can self-adjust to invest the minimum energy needed to guarantee the required discovery rate, whereas the trivial algorithm cannot.

### 5.5.2 Detecting a small segment

So far we have assumed that the detecting node belongs to a big cluster to which the detected node joins. However, it is still possible that two nodes in the Init state of Figure 5.2 discover each other. In such a case each of the two nodes can either stay in the Init state or switch to the Normal state. It is more efficient to switch to the Normal state because the overhead of detecting more neighbors is shared by all of the segment nodes. However, some of the assumptions made in Section 5.3 are not valid when the detecting node is not a part of a big segment: the assumption that the expected in-segment degree of the segment’s nodes is equal to the expected in-segment degree of the hidden node, and the (implicit) assumption that the segment’s size is significantly bigger than the expected node’s degree. For example, when a single node $v$ in a small segment of size two detects another
node, the expected in-segment degree of the nodes in the detecting segment is 1. In contrast, by Theorem 4, the expected degree of the hidden neighbor of \( v \), is about 1.58.

Figure 5.8(a) shows simulation results for hidden neighbors discovery rate for a small detecting segment. The simulation setup is similar to what described in Section 5.5. The hidden neighbor degree is estimated using Method 2, as it is described in Section 5.3, but the other two methods present similar results. From this graph we can learn that the discovery rate is higher than required, namely that the in-segment nodes send HELLO messages too frequently, which indicated that the hidden neighbor degree is indeed underestimated.

To cope with this problem the estimation result of the “big segment algorithm” is multiplied by 0.585, which is the probability found in Theorem 4 and increased by one. This is motivated by the following rationale. For a small segment, with a high probability all nodes are neighbors of each other. Hence, the estimating node may consider only its immediate neighbors as possible neighbors of a hidden node. For example, consider a node \( v \) in a segment of two nodes. Its hidden neighbor \( u \) has at least one neighbor, \( v \), and with probability of 0.585 a neighbor of \( v \) is also a neighbor of \( u \). The same would hold for a segment of three nodes. However, as the segment becomes bigger, this rationale does not hold, because the probability that a hidden node \( u \) has neighbors that are not neighbors of \( v \) becomes higher. Figure 5.8(b) shows that the heuristic indeed performs well for a segment of 2 nodes, the discovery rate before the adjustment is about 0.6. The adjustment brings the discovery rate to 0.47, which is much closer to the required 0.5. For a segment of 3 nodes, the use of the heuristic changes the discovery rate from 0.55 to 0.47.

Figure 5.7: Hidden neighbors detection with extreme point

(a) Decrease in hidden node ratio

(b) Our protocol compared to a trivial protocol that does not adjust wake-up frequency
which represents no significant improve to the accuracy, and finally, for the bigger segment, the adjustment is inefficient.

However both with and without the presented heuristic, a hidden node in the vicinity of a small segment is discovered almost within the required probability.
Chapter 6

Conclusions

In this thesis we presented new schemes for saving energy resources in sensor networks. We presented an algorithm for determining the wake-up frequency of the nodes in a sensor network. This algorithm minimizes the energy consumption of the nodes and bounds the maximum delay on the routes from the nodes to the gateway. We simulated the algorithm over random sensor networks with different topologies and studied its impact on network energy consumption. This study revealed that the algorithm reduces the total energy consumption by 60-70% compared to energy consumption under equal assignment. When the proposed algorithm is used along with an energy-aware routing algorithm, the network lifetime was shown to increase by 40%-80%. Future work is needed in order to find closed formulas for the relationship between the upper bound $\tau$ on the frequency assigned to every node and the network lifetime. This relationship depends on several important factors such as network topology, the algorithm for building the routing tree $T$ and the algorithm for deciding when a node with limited energy should be excluded from the routing paths.

We extended the previous case and studied the problem of energy efficient routing in a sensor network with multiple gateways. Due to the complexity of this problem, we divided it into two sub-problems: the problem of constructing efficient routing trees (METP) and the problem of wake-up frequency assignment in a network with multiple routing trees (MT-OFA).

We showed that MT-OFA can be formulated as a conic quadratic optimization problem that has only linear and convex cone constraints. Thus, it can be solved using the interior point method. However, this algorithm is impractical in many cases because it demands high computational power. Therefore, we also proposed an efficient approximation that uses the optimal ST-OFA algorithm as a sub-routine. For METP, we proved that it is NP-hard, and proposed an approximation
algorithm that builds a minimum-weighted connected dominating set.

Our simulations showed that the actual performance of the MT-OFA approximation is very close to the optimal. Moreover, when the spanning tree is constructed using the METP approximation algorithm, the wake-up frequencies assigned to the nodes decreases significantly in comparison to the BFS algorithm. Finally, our simulation results show that multiple routing spanning trees significantly improve network reliability.

We also defined a new concept in wireless sensor networks, referred to as ongoing continuous neighbor discovery. We argued that continuous neighbor discovery is essential even if the sensor nodes are static. We showed that by having the nodes in a connected segment work together on continuous neighbor discovery, we can guarantee that (a) hidden nodes will be detected with a certain probability $P$ and within a certain time period $T$; and that (b) the energy expended by the segment nodes on the detection of hidden nodes is minimized.

We showed that our scheme works well if every node connected to a segment estimates the in-segment degree of its potential hidden neighbors. To this end, we proposed three estimation algorithms and analyzed their mean square errors. We then presented an continuous neighbor discovery algorithm that determines the frequency with which every node enters the HELLO period. Using simulations, we analyzed several aspects of our algorithms. We showed that when the hidden nodes are uniformly distributed in the area, the simplest estimation algorithm is good enough. When the hidden nodes are concentrated around some dead areas, the third algorithm, which requires every node to take into account not only its own degree, but also the average degree of all the nodes in the segment, was shown to be the best.
Bibliography


[34] J. Haartsen. *Bluetooth Baseband Specification v. 1.0*.


[57] The MOSEK optimization software.

הלירוי לשわずת כehr פעלית:

אלגוריתמים ליזלי שוכנים וליזלי שוכנים עצים של התMotor לשיזן להראות קיים בשיעור שיאו מאכזב את קורט ישראל, לועות את, עול האלגוריתמים של קוקטי כל השאנטר פארציצא האנרגיה, כי היא מוצרلعبית קרובות.

אלגוריתמים האראו יתי ל莉ית ליחנה שלחנה פועל לב, מביל הלחקים מידה עם השכנים. לועות את, האלגוריתמים היו צמוד לעколо השחתוןMcC של מתור את רוב השכנים ויכולה לשiktig פעלית.

ואנו מציעים מנגנון המאפרץ לתחון ושחתון עד שייך אחדר לולאת בורה מחירה את השאר השכנים. המגנון מתבסס על מנגנון אחדר, המאפרץפארט הצרחה של almוהון לולאת המקור לולאת המקור מורת של ה藫 ולגרגא תכל הלולאה. המגנון שפיתחנו מתיחות אל מישנה הגיון של שק חנה והלא çח שמהתחנו לולאת איש סופי, לולאת הלולאה庄园 מסויית ל莉ית ליחנה תדשה, יש ליקח בשובות את מסרפ השכنين הפונטיאלי של התמונה התלת הנמצאים בותר הקולקטורים. מסרפר זה איו דו歐 ולק אוג מיום שלושה ישראל פליקתון.

הביאה:

• לכל התנהות שולט על מסרפ השכنين הממוצע בקולקטורVELO על מסרפ השכنين בקולקטור

• לכל התנהות שולט על מסרפ השכنين של הקולקטור VEL על מסרפ השכنين בקולקטור של

• לכל התנהות שולטת בתיורלי יייא לי שטי השטיון הקודמות על מנט להעריך את

• מסרפ השכنين בקולקטור של התנהות הנשתטר.

לנتكون אית לבין שלוחת השטיון והשקפת תמונות בנסוחות ברווח השנייה (Vsq error) התשובה המתקלת היה פונקציה של הקורלציה בין מסרפר השכنين של התנהות איזה ממוצע שנקשר את ת sla קורלציה בין אלגוריתמים את מוציאים את הקורלציה בין מסרפר השכنين ובמידה של פיזור אחיד לשיזן הרשע. במקורה זה השתיית השנייה הרמה ובד ביאור. בלבל האבה אוג מוצא אלגוריתמים הקובע לכל התנהות בראות קרב ממוצע עליה לישון את הורישה HELLO על מנט הולכיון שישס תמור י砝לה בהפוסברים רazıיך להקפיים והזון.

ומנייה השתיית הנשתטר מוטוריית בתדריתידועה מראשה.

1. (NP-complete)
2. minimum connected dominating set
3. minimum weighted connected dominating set
4. maximum clique
5. maximum independent set
6. maximum matching
7. maximum flow
8. minimum cut
9. maximum cut
10. maximum matching
11. maximum flow
12. minimum cut
13. maximum matching
14. maximum flow
15. minimum cut
16. maximum matching
17. maximum flow
18. minimum cut
19. maximum matching
20. maximum flow
21. minimum cut
22. maximum matching
23. maximum flow
24. minimum cut
25. maximum matching
26. maximum flow
27. minimum cut
28. maximum matching
29. maximum flow
30. minimum cut
31. maximum matching
32. maximum flow
33. minimum cut
34. maximum matching
35. maximum flow
36. minimum cut
37. maximum matching
38. maximum flow
39. minimum cut
40. maximum matching
41. maximum flow
42. minimum cut
43. maximum matching
44. maximum flow
45. minimum cut
46. maximum matching
47. maximum flow
48. minimum cut
49. maximum matching
50. maximum flow
51. minimum cut
52. maximum matching
53. maximum flow
54. minimum cut
55. maximum matching
56. maximum flow
57. minimum cut
58. maximum matching
59. maximum flow
60. minimum cut
ישנן פעילות של תחנות ולהחליפה ההודעות, על שיתוף לרחוב vidéי. ישנו העבודה המетесь סכומי גולו, של הת合わ, המ שזור ושלי התמונה עגברה חידי لماב. פעייל. במכרה הזד לתחנה שלו, זכי התמונה, искус וקשא ומאזים בלמות ברחות שבבח מספר על היא התונה שיאני ימצאת בטיוו הגדורה והתצלמה או התשינה. tema שדח科尔 ב再造ט היממה. בעובדה זה אנימ בהמודחכול עליה מודל של כל התحجم בחרת לעצמי. בשתי תחנות תוכלנה להחליפה הודעותمه שדורש מכל התחנות לעבור יחדיו למצב, עבודות המציעות סנכרון גלובלי של תחנות. בארק זה קל להחליפה הודעות בין התחנות. פעיל כמו, שבהן מספר רב של תחנות שאינן נמצאות בטווח השידור והקליטה של השפה, וכמו התוצרות באמור ע翯ית ימיית עכ שיתוף. אניairy, התמונה הגרשה לשדרה. למגוון הת畢竟ות של התמונה הנבאה במשטח היגוות. כמו, איז בדר לתחנה ממסר התודה ציריך לשגר חמש חנות, אם כי התנה מתעוררה ומיידעת על כל שכניה. המרבי היא.Time. על ידי הקטנת קצב התצרות התוצרות של הת.same רשתות עם התتحرור לשחיתות, זומן לתוך למור. צייר האורזב, או הדבר الكرח בגדבת החשיבה של התודה лучше. כיוון שבדר לשיון תוספים על די זני תוצרות, יש להימש את קצב התצרות שחריר כדי להגיא התוצרות במילוי פונקציה של הת التداول התוצרות של תחנת השיחה של התודה ברשת. הזהל הארושא, ובטור להHdrגיה. לאגרחיה, קורסים, המימן של התוצרותและความוול של כל חותםращ农机 לאٿת ממוסר,pritper תום, ובהשוואת של התוצרות ה猛地 למור. ירי התערבות אופטיות מב Showing את התمم, ובהמתגר על פתויו, אופטיות. עיבור התמי עיסי של הים. הסופי של דכר האלוטרכים מ ViewBag אל שורש עיז ובכר מיסים את הקצאת קצב התצרות.

בארץ האלוטרכים על תסרוקות תוע של ושומע הסצר, שככל שהשחתה קרובת יותר, מושך התצרות עם결 של יוניסוף, איה הזקן קצב התצרות בן גוות. שלג תзыва.出炉, התמונה של ג'רנש שתחברת סכלי פועל, מסר זהธรรม. BUILD מתחזק של מוסלמי יוניסוף, פועלות יורה מביתות ציריך הארגון, גנוך. קיים, ואתפוש מססר בונים כל החרת שלג' בין למסב כלי התמונה בסיבית של התמונה יידע. ציור את כל הארגונים שלג' אינן ימיות لمיש החכמה ממוסר. דחי למוסר המחזק, וסמכו האלוטרכים באמצעות הבגרולה תידור התצרות של כל התמונה לתמונה. כלומר, האלוטרכים

- עינה תוצרות תחתות כשוicie ימי
- קצב התצרות של כל תחתות תוספ על ידי 2
- תפ סכימי האגרגציה של הרשת מתועברת.
TexCoord

 shalt be chosen so that the network can decide on its own when it can transition from the active state to the sleeping state and vice versa. The first type of sensor allows the sensors to differentiate between the two types of sensors that are active while the communication routes are built, so that a report can reach them at any time. The second type of sensors, who must enter an active state with a defined and known state, therefore must wake up, and only serve as a relay station and transmit the reports that they are not used. Only when there is a report to be transmitted, they can be considered as a transmission station with a higher power than the sensors that are not used as a relay station, which is much higher than that of the sensors that are not used as a relay station.
הכרת תודה

ארושה כל, ברצוני ל абיע את הערכתי והודותי לכל האנשים שהúsqueda והחפща בפקולטה לממדה למדם. תודה לי高频 고ון להיות תלמידו.

ברצוני להודות לדני רז ואני הים את הבכורה של תחומי התחום ואת הודותיות הערכתי לישראל צידון, חותם לי高频 고ון ושלבים על הידע והיכולות של המדונה.

בנוסף, הבכורה של דני רז, אני מודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי וד니 רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודה לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודם לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודם לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודם לחרותם של חנוך לוי ודני רז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודם לחרותם של חנוך לוי ודני��ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני רז ומודם לחרותם של חנוך לוי ודני��ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני��ז ומודם לחרותם של חנוך לוי ודני��ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני��ז ומודם לחרותם של חנוך לוי ודני��ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני��ז ומודם לחרותם של חנוך לוי ודני��ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דני��ז ומודם לחרותם של חנוך לוי ודני��ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דניに戸ז ומודם לחרותם של חנוך לוי ודניに戸ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דניに戸ז ומודם לחרותם של חנוך לוי ודניに戸ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דניに戸ז ומודם לחרותם של חנוך לוי ודניに戸ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דניに戸ז ומודם לחרותם של חנוך לוי ודניに戸ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דניに戸ז ומודם לחרותם של חנוך לוי ודניに戸ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דניに戸ז ומודם לחרותם של חנוך לוי ודניに戸ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דניに戸ז ומודם לחרותם של חנוך לוי ודניに戸ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה גם להביע את הודותיות עם דניに戸ז ומודם LCHARותם של חנוך LCHARי ודני��ז על הודותיות ואת הודותיות של וועדות הבכורה.

אני רוצה also veutתיה של זה, ואתהaban BOARD thúרי.
אופטימיזציה של הקצאת קצבי התוויה
ברשתות סנסורים

חיבる על מחקר

לשם مليוי חלקי של הדרישות לקרבל התואר
דוקטור לפילוסופיה

בוריס קפצ

הוגש לסנט הטכניון – מכון טכנולוגי לישראל
פברואר 2010
חיפה
ער"ש 2010
אופטימיזציה של הקצאת קצבי תשתירות

ברשמת סנסורים

בורייס קפיצ'ין