DISS. ETH NO. 17801

Energy-Efficient Data Gathering in Sensor Networks

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

PASCAL VON RICKENBACH

MSc ETH CS, ETH Zürich

born 18.05.1978

citizen of
Muotathal SZ

accepted on the recommendation of

Prof. Roger Wattenhofer, examiner
Prof. Magnús M. Halldórsson, co-examiner
Prof. Bhaskar Krishnamachari, co-examiner

2008

Abstract

Wireless sensor networks are applied across a diverse range of application domains. These networks offer the potential to instrument the world at an unprecedented scale, and to bring the pervasive computing vision to fruition. Long-term surveillance of environmental conditions is a prevalent application for wireless sensor networks. By spreading a large number of cheap untethered sensor nodes within an area of interest it is possible to monitor dense temporal and spatial data over an extended period of time enabling scientists to analyze complex interactions within the environment. To enable communication sensor nodes are equipped with a short-ranged radio allowing them to convey their data to an information sink for further processing. Minimizing energy consumption in such systems is of prime importance since the desired operational periods without human interaction range from months to several years.

This dissertation discusses important problem fields in the context of energy-efficient data gathering in sensor networks. In particular, it tackles essential questions arising throughout the whole sensor network life-cycle. Starting with investigations about interference restricting network topologies and the impact of data aggregation on energy consumption, the thesis describes a practical implementation of a low-power network stack that allows long-term continuous data collection. Furthermore, we study energy-efficient management services applicable during node deployment and application reprogramming.

Zusammenfassung

Drahtlose Sensornetze werden in einer Vielzahl unterschiedlicher Szenarien eingesetzt. Diese Netzwerke besitzen das Potential mit der Welt in einem noch nie da gewesenen Massstab zu interagieren und damit die Pervasive Computing Vision zu verwirklichen. Der vorherrschende Anwendungsbereich für Sensornetze liegt dabei in der Überwachung von Umweltbedingungen. Bei diesen Anwendungen werden viele batteriebetriebene Sensorknoten innerhalb des zu beobachtenden Gebiets verteilt, um zeitlich und örtlich hochauflösende physikalische Grössen wie Temperatur, Feuchtigkeit oder seismische Aktivität zu messen. Die so gesammelten Informationen ermöglichen die Erforschung komplexer Zusammenhänge in der Natur. Sensorknoten kommunizieren dabei über Funk miteinander, um gemessene Daten an einen vorgezeichneten Sammelknoten zu übermitteln. Zusammen bilden sie eine dynamische Kommunikationsinfrastruktur, die das ganze zu überwachende Gebiet abdeckt. Ein ökonomischer Umgang mit den stark limitierten Energieressourcen ist in solchen Systemen von höchster Bedeutung, damit eine wartungsfreie Lebensdauer mehrere Jahre erreicht werden kann.

Diese Dissertation befasst sich mit wichtigen Problemfeldern im Bereich der energieeffizienten Datenkommunikation in drahtlosen Sensornetzen. Insbesondere werden Fragen im Zusammenhang mit der Verwaltung solcher Netzwerke aufgegriffen. Neben der Analyse von Netzwerktopologien, die Signalstörungen, oder Interferenz, zwischen den Sensorknoten verringern, untersucht diese Arbeit auch den Einfluss von Datenaggregation auf den Energieverbrauch im Netzwerk. Ausserdem wird eine praktische Implementierung eines energiesparenden Kommunikationsstacks vorgestellt, welcher die kontinuierliche Extraktion relevanter Daten aus dem Netzwerk über beträchtliche Zeitspannen ermöglicht. Des Weiteren werden Methoden zur energieeffizienten Installation und schnellen drahtlosen Neuprogrammierung eines Sensornetzes besprochen.

Acknowledgements

This thesis would not have been possible without the help and support of many people to whom I would like to express my gratitude. First of all, I would like to thank my advisor Roger Wattenhofer for guiding and encouraging me during my studies and for introducing me to the world of academic research. With your vast knowledge, enthusiasm, and manner of scientific thinking, you have always been a great source of inspiration.

I would also like to express my gratitude to my co-examiners Magnús M. Halldórsson and Bhaskar Krishnamachari for their willingness to read through the thesis and serve on my committee board. It has been a great pleasure and honor to receive your positive comments.

Many thanks go to all fellow colleagues of the Distributed Computing Group. First and foremost, I would like to thank Thomas Moscibroda, my faithful companion during many years at ETH. It was great to share the office with you during your time at the Distributed Computing Group. Your keen perception and the fact that both of us are not averse to a hint of sarcasm led to countless exciting discussions about life, the universe, and everything. I am honored to have witnessed your takeoff as a rising star in the networking community. Also, I would like to thank Nicolas Burri, my brother in crime when it comes to sensor network programming. I do not remember how many nights we spent in the office to make the Dozer code run, but each of them was a memorable adventure full of Coke, kebabs, and cold pizza. My sincere gratitude also goes to Aaron Zollinger for guiding me during my way on the interference highway. It was always enlightening to follow your excurses on the peculiarities of the English language, not to speak about how much they improved my writing. I would also like to thank Keno Albrecht, Fabian Kuhn, Regina O'Dell, and Stefan Schmid. Your warm welcome as well as your positive and festive characters contributed a lot to the great working atmosphere in our group.

Furthermore, I would like to thank all the newer members of the Distributed Computing Group—Raphael Eidenbenz, Roland Flury, Olga Gousysevskaia, Michael Kuhn, Christoph Lenzen, Thomas Locher, Remo Meier, Yvonne Anne Pignolet, Johannes Schneider and Philipp Sommer— or providing a warm and inspiring work environment during the final stage of my PhD. I hope you somewhat benefited from my presence in the group, if only because I was a good example of a bad example ;).

I am also deeply indebted to Jan Beutel, Roman Lim, Matthias Woehrle, and Mustafa Yuecel of the Computer Engineering Group. When I started working with real sensor network hardware, your broad experience in this domain as well as your in-depth knowledge of electrical engineering facilitated things quite a bit to me. I will never forget my first lesson“oscilloscope for dummies”.

Moreover, I would like to thank my travel mates Ren Brand, Nicolas Burri, Thomas Moscibroda, Stefan Schmid, Susanna von Arx, and Aaron Zollinger for accompany me through unvorgettable adventures. This thesis might look totally different if I had not experienced a blizzard in Colorado, done backcountry camping on Hawaii, eaten seafood Paella in Barcelona, survived the hot Chinese kitchen in Hong Kong, visited the Vatican in Rom, or surfed in Bali.

Above all, I would like to express my gratitude to my family. I thank my brother Philipp and his wife Brigit for many interesting and distracting discussions and for letting me stay in their apartment while they are abroad. I am forever thankful and indebted to my parents Brigitte and Remy for their love and care. From the very beginning, you have provided me with support and guidance on my way that has now led me to where I stand today. Finally, my deepest thanks belong to girlfriend Myle. Your love and encouragement has been my motivation to work hard throughout the last part of my thesis. I am—and will always be—grateful to you for standing by me through all ups and downs, relaxing and (unfortunately all too often) stressful times.

Contents

1 Introduction **7**

2 Network Structures And Interference **11**
- 2.1 Sender-Based Interference 12
  - 2.1.1 Model 12
  - 2.1.2 Interference in Known Topologies 14
  - 2.1.3 Low-Interference Spanners 17
- 2.2 Receiver-Based Interference 21
  - 2.2.1 Model 23
  - 2.2.2 Interference in Known Topologies 24
  - 2.2.3 One-Dimensional Topologies 25
- 2.3 Interference in Heterogeneous Networks 31
  - 2.3.1 Minimum Membership Set Cover 33
  - 2.3.2 Problem Complexity 34
  - 2.3.3 Approximating the MMSC Problem 35
  - 2.3.4 Practical Networks 41
- 2.4 Related Work 44

3 Gathering Correlated Data **47**
- 3.1 Foreign Coding 49
  - 3.1.1 Correlation Model 49
  - 3.1.2 Algorithm 50
  - 3.1.3 Analysis 52
  - 3.1.4 Distributed Computation 53
- 3.2 Self-Coding 55
  - 3.2.1 Correlation Model 55
  - 3.2.2 Lower Bound 56
  - 3.2.3 Algorithm and Analysis 57
- 3.3 Related Work 58

4 Ultra-Low Power Data Gathering 61
4.1 System Overview 62
4.2 Dozer Implementation 63
4.2.1 Tree Maintenance 64
4.2.2 Scheduler 66
4.2.3 Data Administration 68
4.2.4 Command Management 69
4.3 Experimental Evaluation 70
4.3.1 Hardware and Operation System 70
4.3.2 Small Scale Experiments 71
4.3.3 Office Floor Experiment 73
4.4 Clock Drift Compensation 78
4.4.1 Evaluation 80
4.5 Multiple Sinks 81
4.6 Related Work 83

5 Energy-Efficient Deployment Support 87
5.1 Unstructured Radio Networks 89
5.2 The Deployment Problem 91
5.3 Deployment Algorithms 93
5.3.1 Birthday Algorithm 94
5.3.2 Uniform Algorithm 96
5.3.3 Cluster Algorithm 98
5.3.4 Discussion 107
5.4 Simulations 108
5.5 Related Work 112
5.6 Concluding Remarks 113

6 Differential Application Updates 115
6.1 Overview 117
6.2 Update Mechanism 118
6.2.1 Delta Instructions and Delta File Organization 119
6.2.2 Delta Encoder 120
6.2.3 Delta Decoder 122
6.3 Experimental Evaluation 122
6.4 Related Work 126

7 Conclusion 129

Chapter 1

Introduction

Observation and interpretation of natural phenomena has always been of fundamental importance to gain scientific knowledge and ultimately obtain a better understanding of the natural world. Advances in science and technology are thereby deeply intertwined. The telescope allows the exploration of outer space, satellites help tracking climatic changes, and microscopy enables us to examine molecule structures, thereby expanding what we can perceive and measure.

Recent progress in wireless networking and microelectronics have led to the vision of wireless sensor networks forming a new scientific tool providing dense sensing close to physical phenomena at scales and resolutions that were difficult to obtain before. By spreading large numbers of cheap, untethered sensor nodes we gain a macroscopic view of the region of interest enabling the analysis of complex interactions. This task, known as data gathering or environmental monitoring, consists of two fundamental parts. First, a large number of sensing devices distributed in the area of interest monitor various conditions of their surrounding environment. Second, the sampled data has to be conveyed towards a central authority collecting the entire information of all nodes.

A collection of challenges must be addressed to realize this vision. Individual sensor nodes exhibit inherent resource constraints: They have limited computational power, storage capacity, and communication bandwidth. As one device by itself has only little impact we must aggregate them into sophisticated computation and communication infrastructures to achieve the required temporal and spatial resolution.

The use of wireless communication technology broadens the field of possible applications for sensor networks since their deployments are far less intrusive than tethered solutions. Furthermore, temporary measurements and surveillance of secluded areas without mains supply is enabled. Once deployed, a network is expected to operate for an extended period of time in a robust and autonomous manner. Therefore, the accessible energy resources provided by batteries or energy harvesting techniques are the limiting factor for network lifetime and availability. Consequently, sensor nodes are equipped with low-power radios to minimize energy consumption. As these radios only offer short transmission ranges, multi-hop routing techniques must be applied to extract sensed information from large areas of interest. This requires coordination among individual sensor nodes. Due to the absence of any built-in infrastructure and because nodes are exposed to changing environmental conditions, wireless sensor networks must be robust.

Despite these operational factors, deploying and maintaining nodes must remain inexpensive. This raises the need for additional network management services to avoid costly human interaction during the lifetime of the network. We should be able to survey vital network information such as remaining energy resources at individual nodes or network topology to ensure the required data fidelity or network lifetime. Furthermore, it may be necessary to adapt the applications running on the nodes due to changing sensing demands.

Wireless sensor networks have been deployed in a variety of settings to assist researchers with detailed information. In the domain of environmental monitoring, sensor networks are currently employed to record the ecophysiology of forests [140], gain understanding of the turbulent subgrid-scale physics near the snow-atmosphere interface [131], measure soil moisture tension for irrigation management [16, 20], or to survey glacier displacement [103]. Natural habitat studies use sensor networks to observe seabird colonies [101] or track long term animal migrations [162]. These application examples depend on continuous, periodic data samples at low rates. Sensor sampling intervals of several minutes are sufficient to provide an adequate data resolution. Furthermore, incurred network delay is of minor concern as the collected information is often analyzed offline. On the other end of the spectrum lie applications such as structural monitoring [156, 44, 46, 114] or the observation of volcanic activity [153, 154] which generate data at very high rates. In these systems it critical for the sensor network to sleep as much as possible and wake up quickly upon detection of an "interesting" phase, e.g. a volcanic eruption, to save precious energy and prolong network lifetime. In the industrial sector, sensor networks have the potential to play a significant role in facility management, process monitoring, and security applications. Today, the state of large chemical and refining plants is already assessed by wired sensor systems. The prospect of having them replaced with wireless solutions is bound to have a positive impact on future arising expenses. Furthermore, wireless sensor networks are also expanding into medical, disaster relief, and military sectors.

In this dissertation we will study problems arising in the field of wireless networking thereby always aiming at energy-efficient solutions.[1] The physical

---

[1] Introductions to wireless sensor networks can be found in [78, 163, 161].

and medium access layers, topology control, and routing techniques must all be optimized to comply with the stringent hardware limitations while offering reasonable performance. In the context of topology control, we analyze the trade-off between energy conservation and interference reduction on the one hand and connectivity on the other hand. By allowing nodes to reduce their transmission power levels interference is confined. This in turn lowers node energy consumption by decreasing the number of collisions and consequently packet retransmissions on the media access layer. Dropping communication links however clearly takes place at the cost of network connectivity: If too many edges are abandoned, connecting paths can grow unacceptably long or the network can even become completely disconnected. We investigate this conflict of objectives from a theoretical point of view.

Orthogonal to the question of which communication links should be established, one may also ask which information needs to be transferred what leads to the field of data aggregation and in-network processing. Sensor networks are often deployed to obtain a complete sensing coverage of the area of interest. Therefore, nearby sensor nodes partially monitor the same spatial region. This however results in correlated data at these nodes. Chapter 3 discusses how this data redundancy can be exploited to reduce the amount data traffic in order to extract the desired information from the network.

In Chapter 4, we describe the design and implementation of a low-power network stack for data gathering systems. In contrast to the approach in Chapter 2, energy efficiency is achieved by restricting the uptime of a node's radio component instead of its radio power level. Sensor nodes are able to deactivate their radios most of the time without impeding the overall system performance since all communication follows precise schedules. This reduces costly overhearing and idle listening phases to a minimum. In this context, we also show that the presence of dynamic drift compensation is essential to achieve exact timings and thus energy efficiency in real sensor network deployments.

Chapter 5 and 6 are dedicated to problems arising during node deployment and code updating and have importance beyond the scope of data gathering applications. In Chapter 5, we study the inherent trade-off between energy efficiency and rapidity of event dissemination which is characteristic for wireless sensor networks. As described in Chapter 4 nodes may spend a large portion of their lifetime in an energy-efficient sleep mode during which they can neither receive nor send messages to save energy. On the other hand, the longer nodes stay in sleep mode, the slower will be the reaction time for disseminating an external event. The trade-off is prominently exhibited during the deployment of sensor networks. Namely, if the deployment phase takes long but once all sensor nodes are fully deployed, the network should make the transition to the operational phase as quickly as possible. Once this shift is performed other network management issues arise which need to be addressed to guarantee a long-lasting deployment. One of the most pressing problems is the need for an efficient application reprogramming service. Reasons for updates are manifold ranging from small bug fixes to retasking of the whole sensor network. The scale of deployments and the potential physical inaccessibility of individual nodes ask for a wireless software management scheme. In Chapter 6, we present an efficient code update strategy which utilizes the knowledge of former program versions to distribute mere incremental changes.

Chapter 2

Interference-Aware Network Structures

Since energy is the limiting factor for network lifetime, great efforts have been made to reduce node power consumption in sensor networks and thus extend their lifetime. One of the foremost approaches to achieve substantial energy conservation is by minimizing interference between network nodes. Confining interference lowers energy consumption by reducing the number of collisions and consequently packet retransmissions on the media access layer. The concept of topology control restricts interference by reducing the transmission power levels at the network nodes and cutting off long-range connections in a coordinated way. At the same time transmission power reduction has to proceed in such a way that the resulting topology preserves connectivity[1].

Even though interference reduction has always been one of the main motivations for topology control, most of the previous work addresses the interference issue implicitly by constructing topologies featuring sparseness or low node degree. However, such an implicit notion of interference is not sufficient to reduce interference since message transmission can affect nodes even if they are not direct neighbors of the sending node in the resulting topology. In this chapter we attempt to precisely specify what is understood by interference. From a algorithmic point of view, we are investigating the properties of different topology control algorithms thereby focusing on interference models in graph representations of networks.

---

[1] The term "topology control" sometimes also refers to clustering and the computation of dominating sets. In this chapter we exclusively consider topology control based on transmission power reduction.

2.1 Sender-Based Interference

In contrast to most previous work on the domain of topology control—where the interference issue is seemingly solved by sparseness arguments—, we start out by precisely defining a first concept of interference. This definition of interference is based on the natural question of how many nodes are affected by communication over a certain link. By prohibiting specific network edges, the potential for communication over high-interference links can then be confined.

We will employ this interference definition to formulate the trade-off between energy conservation and network connectivity. In particular we state certain requirements that need to be met by the resulting topology. Among these requirements are connectivity (if two nodes are—possibly indirectly—connected in the given network, they should also be connected in the resulting topology) and the constant-stretch spanner property (the shortest path between any pair of nodes in the resulting topology should be longer at most by a constant factor than the shortest path connecting the same pair of nodes in the given network). After stating such requirements, an optimization problem can be formulated to find the topology meeting the given requirements with minimum interference.

For the requirement that the resulting topology retain connectivity of the given network, we show that most of the currently proposed topology control algorithms—already by having every node connect to its nearest neighbor—commit a substantial mistake: Although certain proposed topologies are guaranteed to have low degree yielding a sparse graph, interference becomes asymptotically incomparable with the interference-minimal topology.

Furthermore we propose a centralized algorithm that computes an interference-minimal connectivity-preserving topology. For the requirement that the resulting topology be a spanner with a given stretch factor, we present (based on a centralized variant of the algorithm) a distributed local algorithm that computes a provably interference-optimal spanner topology.

2.1.1 Model

We model the sensor network as a graph $G = (V, E)$ consisting of a set of nodes $V \subset \mathbb{R}^2$ in the Euclidean plane and a set of edges $E \subseteq V^2$. Nodes represent sensor nodes, whereas edges stand for links between nodes. In order to prevent already basic communication between directly neighboring nodes from becoming unacceptably cumbersome [120], it is required that a message sent over a link can be acknowledged by sending a corresponding message over the same link in the opposite direction. In other words, only *undirected* (symmetric) edges are considered.

We assume that a node can adjust its transmission power to any value between zero and its maximum power level. The maximum power levels are

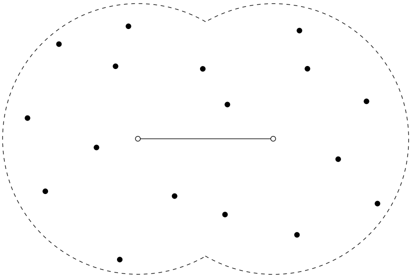

Figure 2.1: Nodes covered by a communication link.

not assumed to be equal for all nodes. An edge (u, v) may exist only if both incident nodes are capable of sending a message over (u, v), in particular if the maximum of the transmission radii of u and v is at least $|uv|$, their Euclidean distance. A pair of nodes u, v is considered connectable in the given network if there exists a path connecting u and v provided that all transmission radii are set to their respective maximum values. The task of a topology control algorithm is then to compute a subgraph of the given network graph with certain properties, reducing the transmission power levels and thereby attempting to lower interference and energy consumption.

With a chosen transmission radius—for instance to reach a node v—a node u affects at least all nodes located within the circle centered at u and with radius $|uv|$. Denoting $D(u, r)$ to be the disk centered at node u with radius r and requiring edge symmetry, we consequently define the *coverage* of an (undirected) edge $e = (u, v)$ to be the cardinality of the set of nodes covered by the disks[2] induced by u and v:

$$\begin{aligned} Cov(e) := \quad & \big|\{w \in V | w \text{ is covered by } D(u, |uv|)\} \\ & \cup \{w \in V | w \text{ is covered by } D(v, |vu|)\}\big|. \end{aligned}$$

In other words, the coverage $Cov(e)$ represents the number of network nodes affected by nodes u and v communicating with their transmission power levels chosen such that they exactly reach each other (cf. Figure 2.1).

The edge level interference defined so far is now extended to a graph interference measure as the maximum coverage occurring in a graph:

Definition 2.1. *The interference of a graph $G = (V, E)$ is defined as*

$$I(G) := \max_{e \in E} Cov(e).$$

---

[2] The results of this section can also be adapted to the case where transmission ranges are not perfect circles centered at the sending nodes. We adhere to this simplified model for clarity of representation.

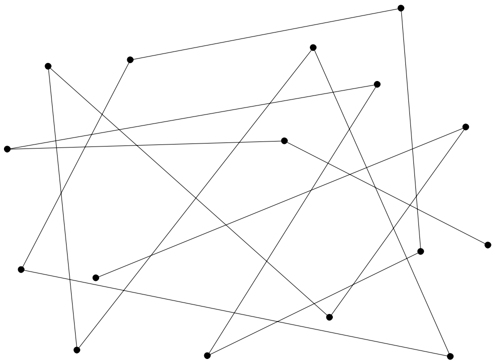

Figure 2.2: Low degree does not guarantee low interference.

Since interference reduction per se would be senseless (if all nodes simply set their transmission power to zero, interference will be reduced to a minimum), the formulation of additional requirements to be met by a resulting topology is necessary. A resulting topology can for instance be required

- to maintain connectivity of the given communication graph (if a pair of nodes is connectable in the given network, it should also be connected in the resulting topology graph),
- to be a spanner with constant stretch of the underlying graph (the shortest path connecting a pair of nodes u, v in the resulting topology is longer by a constant factor only than the shortest path between u and v in the given network), or
- to be planar (no two edges in the resulting graph intersect).

Finding a resulting topology which meets one or a combination of such requirements with minimum interference constitutes an optimization problem.

2.1.2 Interference in Known Topologies

The following basic observation states that—although often maintained—low degree alone does not guarantee low interference. Figure 2.2, for instance, shows a topology graph with degree 2 whose interference is however roughly n, the number of network nodes. A node can interfere with other nodes that are not direct neighbors in the chosen topology graph. Whereas twice the maximum degree of the underlying communication graph of the given network (with all nodes transmitting at full power) is an upper bound for interference, the degree of a resulting topology graph is only a lower bound.

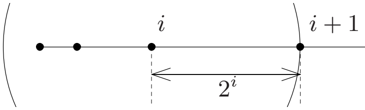

Figure 2.3: Exponential node chain with interference $\Omega(n)$.

There exist instances where also the optimum exhibits interference $\Omega(n)$, for instance a chain of nodes with exponentially growing distances (cf. Figure 2.3, proposed in [105]), whose large interference is caused as a consequence of the requirement that the resulting topology is to be connected. Every node u_i (except for the leftmost) is required to have an incident edge, which covers all nodes left of u_i. Evaluating the interference quality of a topology control algorithm therefore implies that its interference on a given network needs to be compared with the optimum interference topology for the same network.

To the best of our knowledge, all currently known topology control algorithms constructing only symmetric connections (and not accounting for explicit interference) have in common that every node establishes a symmetric connection to at least its nearest neighbor. In the following we show that by including the Nearest Neighbor Forest as a subgraph, the interference of a resulting topology can become incomparably bad with respect to a topology with optimum interference.

Theorem 2.1. *No currently proposed topology control algorithm—required to maintain connectivity of the given network—is guaranteed to yield a nontrivial interference approximation of the optimum solution. In particular, interference of any proposed topology is* $\Omega(n)$ *times larger than the interference of the optimum connected topology, where* n *is the total number of network nodes.*

Proof. Figure 2.4 depicts an extension of the example graph shown in Figure 2.3. In addition to a horizontal exponential node chain, each of these nodes h_i has a corresponding node v_i vertically displaced by a little more than h_i's distance to its left neighbor. Denoting this vertical distance d_i, $d_i > 2^{i-1}$ holds. These additional nodes form a second (diagonal) exponential line. Between two of these diagonal nodes v_{i-1} and v_i, an additional helper node t_i is placed such that $|h_i, t_i| > |h_i, v_i|$.

The Nearest Neighbor Forest for this given network (with the additional assumption that each node's transmission radius can be chosen sufficiently large) is shown in Figure 2.5. Roughly one third of all nodes being part of the horizontally connected exponential chain, interference of any topology containing the Nearest Neighbor Forest amounts to at least $\Omega(n)$. An interference-optimal topology, however, would connect the nodes as depicted in Figure 2.6 with constant interference. $\square$

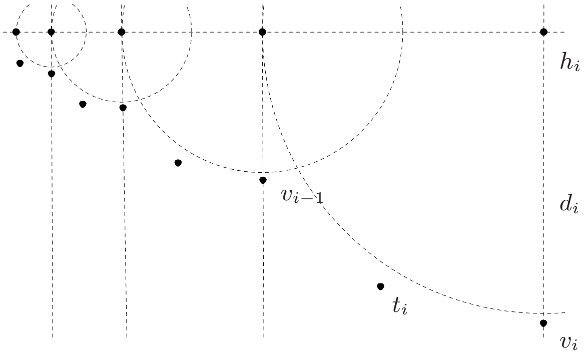

Figure 2.4: Two exponential node chains.

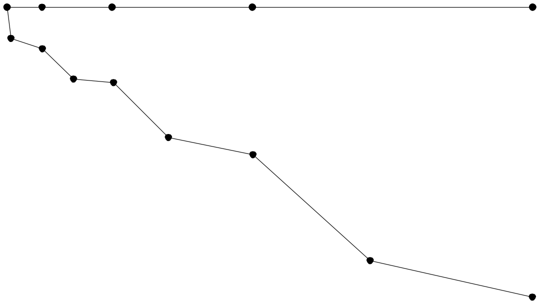

Figure 2.5: The Nearest Neighbor Forest yields interference $\Omega(n)$.

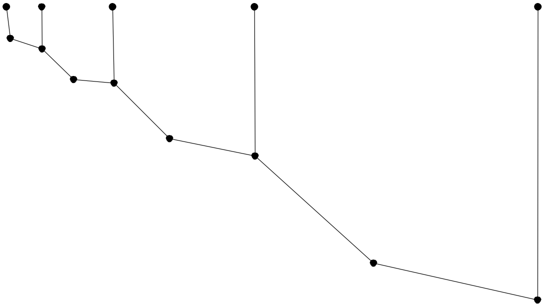

Figure 2.6: Optimal tree with constant interference.

In other words, already by having each node connect to the nearest neighbor, a topology control algorithm makes an irreparable error. Moreover, it commits an asymptotically worst possible error since the interference in any network cannot become larger than n.

As roughly one third of all nodes are part of the horizontal exponential node chain in Figure 2.4, the observation stated in Theorem 2.1 would also hold for an average interference measure, averaging interference over all edges.

2.1.3 Low-Interference Spanners

In this section we present two algorithms that explicitly reduce interference of a given network. They compute an interference-optimal topology with the requirement of constructing a spanner of the given network. Whereas the first spanner algorithm assumes global knowledge of the network, the second can be computed locally. A spanner with stretch factor t can be formally defined as follows:

Definition 2.2 (t-Spanner). *A t-spanner of a graph $G = (V, E)$ is a subgraph $G' = (V, E')$ such that for each pair (u, v) of nodes $c(p^*_{G'}(u, v)) \leq t \cdot c(p^*_G(u, v))$, where $c(p^*_{G'}(u, v))$ and $c(p^*_G(u, v))$ denote the length of the shortest path between u and v in G' and G, respectively.*

In this section we consider Euclidean spanners, that is, the length of a path is defined as the sum of the Euclidean lengths of all its edges. With slight modifications, our results are however extendable to hop spanners, where the length of a path corresponds to the number of its edges.

Algorithm LISE (Algorithm 2.1) is a topology control algorithm that constructs a t-spanner with optimum interference. LISE starts with a graph $G_{LISE} = (V, E_{LISE})$ where E_{LISE} is initially the empty set. It processes all eligible edges of the given network $G = (V, E)$ in descending order of their coverage. For each edge $(u, v) \in E$ not already in E_{LISE}, LISE computes a shortest path from u to v in G_{LISE} provided that the Euclidean length of this path is less than or equal to $t\,|u, v|$. As long as no such path exists, the algorithm keeps inserting all unprocessed eligible edges with minimum coverage into E_{LISE}.

To prove the interference optimality of G_{LISE}, we introduce an additional lemma, which shows that G_{LISE} contains all eligible edges whose coverage is less than $I(G_{LISE})$.

Lemma 2.2. *The graph $G_{LISE} = (V, E_{LISE})$ constructed by LISE from a given network $G = (V, E)$ contains all edges e in E whose coverage $Cov(e)$ is less than $I(G_{LISE})$.*

Proof. We assume for the sake of contradiction that there exists an edge e in E with $Cov(e) < I(G_{LISE})$ which is not contained in E_{LISE}. Consequently,

```
Input: a set of nodes $V$, each $v \in V$ having attributed a maximum transmission radius $r_v^{max}$; a stretch factor $t \geq 1$
 1: $E$ = all eligible edges $(u, v)$ ($r_u^{max} \geq |uv|$ and $r_v^{max} \geq |uv|$)
    // $E$ will contain all unprocessed edges
 2: $E_{LISE} = \emptyset$
 3: $G_{LISE} = (V, E_{LISE})$
 4: while $E \neq \emptyset$ do
 5:    $e = (u, v) \in E$ with maximum coverage
 6:    while $c(p^*(u, v)$ in $G_{LISE}) > t\,|uv|$ do
 7:       $f$ = edge $\in E$ with minimum coverage
 8:       move all edges $\in E$ with coverage $Cov(f)$ to $E_{LISE}$
 9:    end while
10:    $E = E \setminus \{e\}$
11: end while
Output: Graph $G_{LISE}$
```

Algorithm 2.1: Low Interference Spanner Establisher (LISE)

LISE never takes an edge with coverage $Cov(e)$ in line 7, since the algorithm would insert all edges with $Cov(e)$ into E_{LISE} in line 8 instantly (thus also e). There exists however an edge f in E_{LISE} with $Cov(f) = I(G_{LISE})$ eventually taken in line 7. Therefore the inequality $Cov(e) < Cov(f)$ holds. At the time the algorithm takes f in line 7, all edges taken in line 5 must have had coverage greater than or equal to $Cov(f)$, since the maximum of an ordered set can only be greater than or equal to the minimum of the same set. Hence e has never been taken in line 5 and therefore has never been removed from E in line 10. Consequently, e is still in E when f is taken as the edge with minimum coverage in E. Thus it holds that $Cov(f) \leq Cov(e)$ which leads to a contradiction. □

With Lemma 2.2 we are ready to prove that the resulting topology constructed by LISE is an interference-optimal t-spanner.

Theorem 2.3. *The graph $G_{LISE} = (V, E_{LISE})$ constructed by LISE from a given network $G = (V, E)$ is an interference-optimal t-spanner of G.*

Proof. To show that G_{LISE} meets the spanner property, it is sufficient to prove that for each edge $(u, v) \in E$ there exists a path in G_{LISE} with length not greater than $t\,|u, v|$. This holds, since for a shortest path $p^*(u, v)$ in G a path $p'(u, v)$ in G_{LISE} with $|p'| \leq t\,|p|$ can be constructed by substituting each edge on p with the corresponding spanner path in G_{LISE}. For edges in E which also occur in E_{LISE} the spanner property is trivially true. On the other hand an edge (u, v) can only be in E but not in E_{LISE} if a path from u to v in G_{LISE} with length not greater than $t\,|u, v|$ exists (see if-condition in line 6). Thus G_{LISE} is a t-spanner of G.

Interference optimality of LISE can be proved by contradiction. We assume, that G_{LISE} is not an interference-optimal t-spanner. Let $G^* = (V, E^*)$ be an interference-optimal t-spanner for G. Since G_{LISE} is not optimal, it follows that $I(G_{LISE}) > I(G^*)$. Thus all edges in E^* have coverage strictly less than $I(G_{LISE})$. From Lemma 2.2 follows that E^* is a nontrivial subset of E_{LISE}. Let T be the set of edges in E_{LISE} with coverage $I(G_{LISE})$ and $\tilde{G} = (V, \tilde{E})$ the graph with $\tilde{E} = E_{LISE} \setminus T$. $\tilde{G}$ is a t-spanner, since E^* is still a subset of $\tilde{E}$, and $I(\tilde{G}) \leq I(G_{LISE}) - 1$ holds. Because T is eventually inserted into E_{LISE} in line 8, there exists an edge $(u, v) \in E$ that was taken in line 5 and for which no path $p(u, v)$ exists in $\tilde{G}$ with $|p| \leq t\,|u, v|$. Thus $\tilde{G}$ is no t-spanner (and therefore also G^*), which contradicts the assumption that G^* is an interference-optimal t-spanner. □

As regards the running time of LISE, it computes for each edge at most one shortest path. This holds since multiple shortest path computations for the same edge in Line 6 cause at least as many edges to be inserted into E_{LISE} in Line 8 without computing shortest paths for them. Since finding a shortest alternative path for an edge requires $\mathrm{O}\big(n^2\big)$ time and as the network contains at most the same amount of edges, the overall running time of LISE is polynomial in the number of network nodes.

In the following we describe a local algorithm similar to LISE that is executed at all eligible edges of the given network. In reality, algorithm LLISE (Algorithm 2.2) is executed for each edge by one of its incident nodes. The description of LLISE assumes the point of view of an edge $e = (u, v)$. The algorithm consists of three main steps:

1) Collect $(\frac{t}{2})$-neighborhood,
2) compute minimum interference path for e, and
3) inform all edges on that path to remain in the resulting topology.

In the first step, e gains knowledge of its $(\frac{t}{2})$-neighborhood. For a Euclidean spanner, the k-neighborhood of e is defined as all edges that can be reached (or more precisely at least one of their incident nodes) over a path p starting at u or v, respectively, with $c(p) \leq k\,c(e)$. Knowledge of the $(\frac{t}{2})$-neighborhood at all edges can be achieved by local flooding.

During the second step, a minimum-interference path p from u to v with $c(p) \leq t\,c(e)$ is computed. LLISE starts with a graph $G_{LL} = (V, E_{LL})$ consisting of all nodes in the $(\frac{t}{2})$-neighborhood and an initially empty edge set. It inserts edges consecutively into E_{LL}, in non-decreasing order according to their coverage, until a shortest path $p^*(u, v)$ is found in G_{LL} with $c(p^*) \leq t\,c(e)$.

In the third step, e informs all edges on the path found in the second step to remain in the resulting topology. The resulting topology then consists of

```
1: collect $(\frac{t}{2})$-neighborhood $G_N = (V_N, E_N)$ of $G = (V, E)$
2: $E' = \emptyset$
3: $G' = (V_N, E')$
4: repeat
5:    $f$ = edge $\in E_N$ with minimum coverage
6:    move all edges $\in E_N$ with coverage $Cov(f)$ to $E'$
7:    $p$ = shortestPath$(u - v)$ in $G'$
8: until $c(p) \leq t\,|uv|$
9: inform all edges on $p$ to remain in the resulting topology.
   Note: $G_{LL} = (V, E_{LL})$ consists of all edges eventually informed to remain
   in the resulting topology.
```

Algorithm 2.2: LLISE

all edges receiving a corresponding message. In the following we show that it is sufficient for e to limit the search for an interference-optimal path $p(u, v)$ meeting the spanner property to the $(\frac{t}{2})$-neighborhood of e.

Lemma 2.4. *Given an edge $e = (u, v)$, no path p from u to v with $c(p) \leq t\,c(e)$ contains an edge which is not in the $(\frac{t}{2})$-neighborhood of e.*

Proof. For the sake of contradiction we assume that a path p from u to v with $|p| \leq t\,|e|$ containing at least one edge (w, x) not in the $(\frac{t}{2})$-neighborhood of e. Without loss of generality we further assume that, traversing p from u to v, we visit w before x. Since (w, x) is not in the $(\frac{t}{2})$-neighborhood, by definition, no path from u to w with length less than or equal to $(\frac{t}{2})|e|$ exists (the same holds for any path from v to x). Consequently, the inequality $|p| > t\,|e| + |(w, x)|$ holds, which contradicts the assumption that $|p| \leq t\,|e|$. $\square$

With Lemma 2.4 we are now able to prove that the topology constructed by LLISE is a t-spanner with optimum interference.

Theorem 2.5. *The graph $G_{LL} = (V, E_{LL})$ constructed by LLISE from a given network $G = (V, E)$ is an interference-optimal t-spanner of G.*

Proof. The spanner property of LLISE can be proven similar to the first part of the proof of Theorem 2.3, where LISE is shown to be a t-spanner.

To show interference optimality, it suffices to prove that the spanner path constructed for any edge $e = (u, v) \in G$ by LLISE is interference-optimal, where interference of a path is defined as the maximum interference of an edge on that path. The reason for this is that only edges that lie on one of these paths remain in the resulting topology; non-optimality of G_{LL} would therefore imply non-optimality of at least one of these spanner paths. In the following we look at the algorithm executed by $e = (u, v)$. In line 6 edges in E are consecutively inserted into E', starting with $E' = \emptyset$, until a spanner path p from u to v is found in line 8. Since LLISE inserts the edges into E' in ascending order according to their coverage and p is the first path meeting the spanner property, p is an interference-optimal t-spanner path from u to v in the $(\frac{t}{2})$-neighborhood. From Lemma 2.4 we know that the $(\frac{t}{2})$-neighborhood of e contains all spanner paths from u to v and therefore also the interference-optimal one. Thus it is not possible that LLISE does not see the global interference-optimal t-spanner path due to its local knowledge about G. Consequently, p is the global interference-optimal t-spanner path of e. ☐

In this chapter we have shown that the statement that graph sparseness or small degree implies low interference is misleading. We have then proposed two algorithms that yield low-interference topologies based on the explicit interference model presented in Section 2.1.1. This sender-centric model can however be accused to be somewhat artificial and to poorly represent reality as interference occurs at the intended receiver of a message and not the sender. In the following sections an alternative interference model is discussed defining a receiver-centric concept of interference.

2.2 Receiver-Based Interference

The definition of interference introduced in the previous section is problematic in two respects. First, it is based on the number of nodes affected by communication over a given link. In other words, interference is considered to be an issue at the sender instead of at the receiver, where message collisions actually prevent proper reception. It can therefore be argued that such sender-centric perspective hardly reflects interference as it occurs in real networks.

The second weakness of the model introduced in Section 2.1 is of more technical nature. According to its definition of interference, adding a single node to a given network can dramatically influence the interference measure. In the network depicted in Figure 2.7, addition of the rightmost node to the cluster of roughly homogeneously distributed nodes causes the construction of a communication link covering all nodes in the network; accordingly—merely by introduction of one extra node—the interference value of the represented topology is pushed up from a small constant to the maximum possible value, that is the number of nodes in the network. This behavior contrasts to the intuition that a single additional node also represents only one additional packet source potentially causing collisions.

In contrast to that sender-centric interference definition, we explicitly consider interference at its point of impact, particularly at the receiver in this section. Informally, the definition of interference is no longer based on

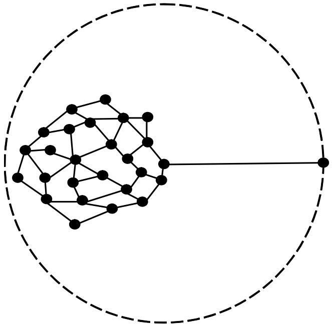

Figure 2.7: In the interference model presented in Section 2.1, addition of a single node increases interference from a small constant to the maximum possible value, the total number of network nodes.

the question of how many nodes are affected by communication over a certain link but by how many other nodes a given network node can be disturbed.

Interestingly, this interference definition not only reflects intuition due to its receiver-centricity. It also results in a robust interference model in terms of measure increase due to the arrival of additional nodes in the network. Particularly, an additional node causes an interference increase of at most one at other nodes of the network. In clear contrast to the sender-centric model from Section 2.1, this corresponds to reality, where one added node contending for the shared medium constitutes only one additional possible collision source for nearby nodes in the network.

As mentioned earlier, interference reduction as an end in itself is meaningless—every node setting its transmission power to a minimum value trivially minimizes interference—without the formulation of additional requirements to be met by the resulting topology. In this section we study the fundamental requirement that the considered topology control algorithms preserve connectivity of the given network.

Similarly as in Section 2.1, we show that for this requirement most of the currently proposed topology control algorithms trying to implicitly reduce interference commit a substantial mistake—even by having every node connect to its nearest neighbor. Based on the intuition that one-dimensional networks already exhibit most of the complexity of finding minimum-interference topologies, we precisely anatomize networks restricted to one dimension—a model also known as the highway model. We first look at a particular network where distances between nodes increase exponentially from left to right. [105] introduces this network as a high-interference example yielding interference $O(\Delta)$, where Δ is the maximum node degree. We show that it is possible to achieve interference $O(\sqrt{\Delta})$ in our model for this network, which matches a lower bound also presented in this section. Based on the insights thereby gained, we then consider general highway instances where nodes can be distributed arbitrarily in one dimension. For the problem of finding a minimum-interference topology while maintaining connectivity, we propose an approximation algorithm with approximation ratio $\mathrm{O}(\sqrt[4]{\Delta})$.

2.2.1 Model

We model the wireless network with the well-known *Unit Disk Graph* (UDG) [25]. In a UDG $G = (V, E)$, there is an edge $\{u, v\} \in E$ iff the Euclidean distance between u and v is at most 1. That is, we assume all nodes to have the same limited transmission ranges. In the following, let Δ refer to the maximum node degree in G.

As in the previous sections, we are only considering undirected (symmetric) edges. Furthermore, we still assume that each node can adjust its transmission power to any value between zero and its maximum transmission power level. Then, the main goal of a topology control algorithm is to compute a low-interference subgraph of the given network graph G that maintains connectivity.

Let N_u denote the set of all neighbors of a node $u \in V$ in the resulting topology. Then, each node u features a value r_u defined as the distance from u to its farthest neighbor. More precisely $r_u = \max_{v \in N_u} |uv|$, where $|uv|$ denotes the Euclidean distance between nodes u and v. Since we assume the nodes to use omnidirectional antennas, $D(u, r_u)$ denotes the disk centered at u with radius r_u covering all nodes that are possibly affected by message transmission of u to one of its neighbors. The transmission radii of the network nodes having been fixed, the definitions of node-level and graph-level interference correspond to the definition in the previous section. In particular, the interference of a node v is defined as the number of other nodes that potentially affect message reception at node v:

Definition 2.3. *Given a graph $G' = (V, E')$, the interference of a node $v \in V$ is defined as*

$$I(v) = |\{u | u \in V \setminus \{v\}, v \in D(u, r_u)\}|.$$

In other words, the interference of a node v represents the number of nodes covering v with their disks induced by their transmission ranges set to a value as to reach their farthest neighbors in G'. Note that although each node is also covered by its own disk, we do not consider this kind of self-interference. The node-level interference defined so far is now extended to a graph interference measure as the maximum interference occurring in a graph:

Definition 2.4. *The interference of a graph $G' = (V, E')$ is defined as*

$$I(G') = \max_{v \in V} I(v).$$

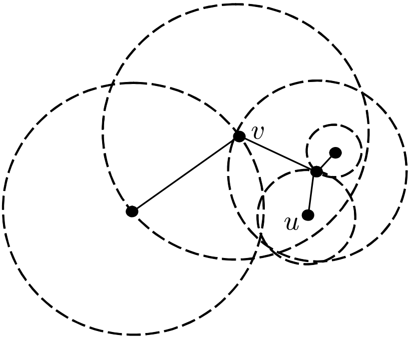

Figure 2.8: A sample topology consisting of five nodes with their corresponding interference radii (dashed circles). Node u experiences interference $I(u) = 2$ since it is covered not only by its direct neighbor but also by node v.

Note that Δ, the maximum node degree of the given unit disk graph $G = (V, E)$, is an upper bound for the interference of any subgraph G' of the given graph since in G each node is directly connected to all potentially interfering nodes. However, in arbitrary subgraphs of G the degree of a node only lower-bounds the interference of that node because a node can be covered by non-neighboring nodes (cf. Figure 2.8).

In this section we study the combinatorial optimization problem of finding a resulting topology which maintains connectivity of the given network with minimum interference. Throughout the section we only consider topologies consisting of a tree for each connected component of the given network since additional edges might unnecessarily increase interference.

2.2.2 Interference in Known Topologies

As motivated earlier, we restrict our considerations to resulting topologies consisting exclusively of symmetric links. To the best of our knowledge, all currently known topology control algorithms (with the exception of the algorithms presented in Section 2.1) constructing only symmetric connections have in common that every node establishes a link to at least its nearest neighbor. With the example configuration also used in Section 2.1, it can be shown that this is already a substantial mistake, as thus interference becomes asymptotically incomparable with the interference-minimal topology, also with this receiver-centric interference definition.

Although the topology control algorithms presented in Section 2.1 do not necessarily include the Nearest Neighbor Forest, it can be shown that those algorithms also perform badly for this receiver-centric interference model.

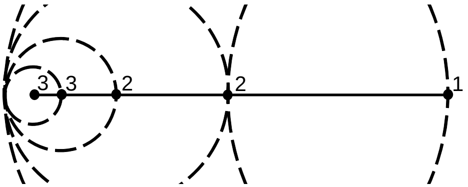

Figure 2.9: Connecting the exponential node chain linearly yields interference $n-2$ at the leftmost node since each node connected to the right covers all nodes to its left. The nodes are labeled according to their experienced interference.

2.2.3 One-Dimensional Topologies

In this section we study interference for the highway model, in which the node distribution is restricted to one dimension. After analyzing an important artificially constructed problem instance, we provide a lower bound for interference of general problem instances in the highway model as well as an asymptotically optimal algorithm matching this bound. Finally, an approximation algorithm is presented.

The Exponential Node Chain

How can n nodes arbitrarily distributed in one dimension connect to each other minimizing interference while maintaining connectivity? In [105], an instance is introduced which seems to yield inherently high interference: The so-called exponential node chain is a one-dimensional graph $G = (V, E)$ where the distance between two consecutive nodes grows exponentially from left to right as depicted in Figure 2.3. The distance between two nodes v_i and v_{i+1} in V is thus 2^i. Throughout the discussion of the exponential node chain, we furthermore assume that the whole node configuration is normalized in a way that the distance between the leftmost and the rightmost node is not greater than 1: Each node can potentially connect to all other nodes in V and therefore $\Delta = n-1$, where $n = |V|$. The nodes are said to be linearly connected if each node—except for the leftmost and the rightmost—maintains an edge to its nearest neighbor to the left and to the right. In other words, node v_i is connected to node v_{i+1} for all $i = 1, \ldots, n-1$ in the resulting topology. In addition to the disks $D(v_i, r_{v_i})$ for each node $v_i \in V$, Figure 2.9 also depicts their interference values $I(v_i)$. Since all disks but the one of the rightmost node cover v_1, interference at the leftmost node is $n-2 \in \Omega(n)$; consequently also interference of the linearly connected exponential node chain is in $\Omega(n)$.

We show in the following that the exponential node chain can be connected in a significantly better way. According to the construction of the exponential node chain, only nodes connecting to at least one node to their right increase v_1's interference. We call such a node a hub and define it as follows:

Definition 2.5. *Given a connected topology for the exponential node chain $G = (V, E)$, a node $v_i \in V$ is defined to be a* hub *in G if and only if there exists an edge (v_i, v_j) with $j > i$.*

The following algorithm $\mathcal{A}_{exp}$ constructs a topology for the exponential node chain G which yields interference $\mathrm{O}(\sqrt{n})$. The algorithm starts with a graph $G_{exp} = (V, E_{exp})$, where V is the set of nodes in the exponential node chain and E_{exp} is initially the empty set. Following the scan-line principle, $\mathcal{A}_{exp}$ processes all nodes in the order of their occurrence from left to right. Initially, the leftmost node is set to be the current hub h. Then, for each node v_i, $\mathcal{A}_{exp}$ inserts an edge (h, v_i) into E_{exp}. This is repeated until $I(G_{exp})$ increases due to the addition of such an edge. Now node v_i becomes the current hub and subsequent nodes are connected to v_i as long as the overall interference $I(G_{exp})$ does not increase. Figure 2.10 depicts the resulting topology if $\mathcal{A}_{exp}$ is applied to the exponential node chain. The exponential node chain is thereby depicted in a logarithmic scale. For clarity of representation, some edges in E_{exp} are drawn as curved arcs. In addition, Figure 2.10 shows the individual interference values at each node.

In the following we show that $\mathcal{A}_{exp}$ reduces interference in the exponential node chain.

Theorem 2.6. *Given the exponential node chain G, applying Algorithm $\mathcal{A}_{exp}$ results in a connected topology with interference $I(G_{exp}) \in \mathrm{O}(\sqrt{n})$.*

Proof. The topology resulting from the application of $\mathcal{A}_{exp}$ shows a clear structure (cf. Figure 2.10). Each hub, not taking into account the first two, is connected to one more node to its right than its predecessor hub to the left. This follows from the fact that if the current topology leads to interference $I(G_{exp}) = I$ immediately after the determination of a new hub, this hub can be connected to $I - 1$ nodes to its right until $I(G_{exp})$ is again increased by one. Therefore the minimum number of nodes n required in an exponential node chain, such that interference $I(G_{exp}) = I$ is obtained, results in

$$n = \sum_{i=1}^{I-1} i + 2 = \frac{1}{2}I^2 - \frac{1}{2}I + 2.$$

By solving for I, with $n \geq 2$, we have

$$I = \left\lfloor \frac{\sqrt{8n - 15} + 1}{2} \right\rfloor \in \mathrm{O}\big(\sqrt{n}\big)\,.$$

□

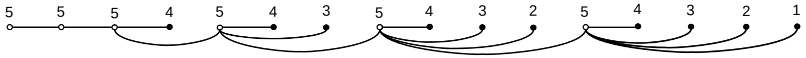

Figure 2.10: The interference of the exponential node chain—shown in a logarithmic scale—is bounded by $O(\sqrt{n})$ by the topology control algorithm $\mathcal{A}_{exp}$. Only hubs (hollow points) interfere with the leftmost node. For clarity of representation, some edges are depicted as curved arcs.

This is an intriguing result since we show in the sequel that $\sqrt{n}$ is a lower bound for the interference of the exponential node chain. We therefore again consider the exponential node chain introduced in Section 2.2.3 with all n nodes located within distance one.

Theorem 2.7. *Given an exponential node chain $G = (V, E)$ with $n = |V|$, $\sqrt{n}$ is a lower bound for the interference $I(G)$.*

Proof. Let H denote the set of hubs (cf. Definition 2.5) in G and S the nodes in $G \setminus H$. In order to prove the theorem, we state two properties for $I(G)$ in the exponential node chain G. First, it holds that $I(G)$ is at least $|H| - 1$, since the leftmost node is interfered with by exactly all hubs except itself (Property 1). On the other hand, $I(G)$ is at least the maximum degree of the resulting topology (Property 2). This holds since a node with maximum degree is covered by at least all disks of its neighboring nodes. We assume for the sake of contradiction that there exists a connected graph that yields interference less than $\sqrt{n}$ for the exponential node chain G. In other words, the degree of any node is required to be at most $\sqrt{n} - 1$, and the number of hubs must not exceed $\sqrt{n}$, including the leftmost node. By the definition of H and S, each node in the graph is either in H or in S and therefore $|H| + |S| = n$ holds. Due to Property 1, it follows that $|H| \leq \sqrt{n}$. Without loss of generality we assume that the hubs are linearly connected among themselves to guarantee connectivity of the graph. Consequently, with Property 2, each hub can connect to at most $\sqrt{n} - 3$ nodes in S (the leftmost and the rightmost hub, respectively, to $\sqrt{n} - 2$). By the definition of a hub, nodes in S are only connected to hubs and not among themselves. Therefore we obtain

$$|S| \leq \sqrt{n}\left(\sqrt{n} - 3\right) + 2.$$

Consequently, $|H| + |S|$ results in $n - 2\sqrt{n} + 2$, which is less than n for $n \geq 2$ and thus leads to a contradiction. $\square$

From Theorems 2.6 and 2.7 it follows that Algorithm $\mathcal{A}_{exp}$ is asymptotically optimal in terms of interference in the exponential node chain.

Arbitrary One-Dimensional Node Distributions

We have considered an important artificially constructed instance in the highway model in the previous section, yielding a lower bound for the interference in arbitrary network graphs. In this subsection we go beyond the study of particular network instances and consider arbitrarily distributed nodes in one dimension.

The question arises if there are instances in the highway model that are asymptotically worse than the exponential node chain, that is, where a minimum-interference topology exceeds $\Omega(\sqrt{\Delta})$. We answer this question in the negative by introducing the $\mathcal{A}_{gen}$ algorithm, which yields interference in $O(\sqrt{\Delta})$ for any given node distribution.

In a first step, the algorithm determines the maximum degree Δ of the given unit disk graph $G = (V, E)$ and partitions "the highway" into segments of unit length 1. Within such a segment, each node can potentially connect to every other node in the segment.

In a second step, $\mathcal{A}_{gen}$ considers each segment independently as follows: Starting with the leftmost node of the segment, every $\lceil\sqrt{\Delta}\rceil$-th node (according to their appearance from left to right) becomes a hub. A hub is thereby redefined along the lines of Definition 2.5 as a node that has more than one neighboring node, in contrast to regular nodes, which are connected to exactly one hub. To avoid boundary effects, the rightmost node of each segment is also considered a hub. Then the $\mathcal{A}_{gen}$ algorithm connects the hubs of a segment linearly. That is, each hub, except the leftmost and the rightmost, establishes an edge to its nearest hub to its left and to its right. Two consecutive hubs enclose an interval. $\mathcal{A}_{gen}$ connects all regular nodes in a particular interval to their nearest hub—ties are broken arbitrarily. Figure 2.11 depicts one segment of an example instance after the application of $\mathcal{A}_{gen}$. The nodes within a segment form one connected component.

Finally, $\mathcal{A}_{gen}$ connects every pair of adjacent segments by connecting the rightmost node of the left segment with the leftmost node of the right segment. This yields a connected topology provided that the corresponding unit disk graph is also connected. Note that with this construction, the hubs may have a comparatively high transmission range (smaller than one unit, though). However, the interference range of regular nodes is restricted to their corresponding intervals. This is due to the fact that regular nodes are connected to their nearest hub only, which determines their transmission ranges.

To prove that the resulting topology of $\mathcal{A}_{gen}$ yields $O(\sqrt{\Delta})$ interference, we introduce an additional lemma, which shows that the interference of a node caused by other nodes in the same segment constructed by $\mathcal{A}_{gen}$ is in $O(\sqrt{\Delta})$.

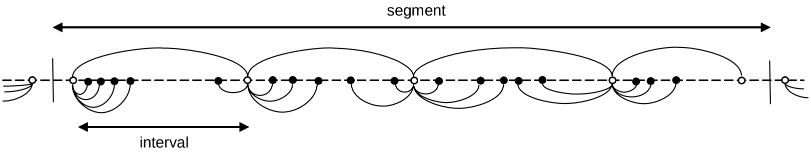

Figure 2.11: $\mathcal{A}_{gen}$ partitions the highway into segments of length 1. In each segment, every $\lceil\sqrt{\Delta}\rceil$-th node becomes a hub (hollow points). While the hubs are connected linearly, each of the remaining nodes in the interval between two hubs is connected to its nearest hub.

Lemma 2.8. *Each node in a segment of the $\mathcal{A}_{gen}$ algorithm experiences at most $O(\sqrt{\Delta})$ interference in the resulting topology of $\mathcal{A}_{gen}$ caused by nodes inside this segment.*

Proof. By definition of a segment, Δ is an upper bound on the number of nodes in the segment. Algorithm $\mathcal{A}_{gen}$ nominates only every $\lceil\sqrt{\Delta}\rceil$-th node a hub. Thus, the number of hubs in σ is upper-bounded by $\frac{\Delta}{\lceil\sqrt{\Delta}\rceil} \in O(\sqrt{\Delta})$. Let hub h_l delimit the interval of a regular node v to the left, and hub h_r to the right, respectively. Furthermore, we can assume without loss of generality that $|h_l, v| < |v, h_r|$. Therefore, $\mathcal{A}_{gen}$ establishes a connection between h_l and v. Because this is the only connection of v it follows that $r_v = |h_l, v|$. Consequently, a regular node only interferes with nodes in the same interval. Since a node v is in at most two intervals—hubs are in two intervals—with at most $\lceil\sqrt{\Delta}\rceil$ nodes, v exhibits interference of at most $O(\sqrt{\Delta})$ regular nodes. Furthermore v is interfered with at most $O(\sqrt{\Delta})$ hubs. □

With this lemma we are ready to prove that the topology constructed by $\mathcal{A}_{gen}$ results in $O(\sqrt{\Delta})$ interference.

Theorem 2.9. *The resulting topology constructed by the $\mathcal{A}_{gen}$ algorithm from a given graph $G = (V, E)$ yields interference $O(\sqrt{\Delta})$.*

Proof. By Lemma 2.8, the interference of a detached segment constructed by $\mathcal{A}_{gen}$ is bounded by $O(\sqrt{\Delta})$. However, interference at node v in segment σ depends also on nodes in the adjacent segments of σ, referred to as σ_l for the segment to the left of σ and σ_r for the segment to the right, respectively. Nodes in other segments do not interfere with v as the length of a segment is chosen according to the maximum transmission range and thus the interference range of a node is limited to two adjacent segments. We know that at most $O(\sqrt{\Delta})$ nodes of σ interfere v. On the other hand, by Lemma 2.8, the rightmost node v' of σ_l is also covered by at most $O(\sqrt{\Delta})$ disks of nodes in σ_l. This implies that at most $O(\sqrt{\Delta})$ nodes of σ_l interfere with v since all nodes interfering with v must also cover v' with their disks. By symmetry, the same holds for segment σ_r. Consequently, $\mathcal{A}_{gen}$ results in interference at most three times the interference of an individual segment at each node, which is in $O(\sqrt{\Delta})$. □

Approximation Algorithm

In contrast to $\mathcal{A}_{gen}$, achieving interference in $O(\sqrt{\Delta})$ for any network instance, this section introduces an algorithm that approximates the optimum for the given network instance. Particularly it yields interference at most a factor in $O(\sqrt[4]{\Delta})$ times the interference value resulting from an interference-minimal connectivity-preserving topology.

Algorithm $\mathcal{A}_{gen}$ is in a sense designed for the *worst-case*. Consider for example an instance where the distances between consecutive nodes are identical. Connecting these nodes linearly, that is, connecting each node to its nearest neighbor in each direction, yields constant interference. Algorithm $\mathcal{A}_{gen}$ however constructs a topology resulting in $O(\sqrt{\Delta})$ interference since a hub connects to one half of the nodes in its corresponding interval for this instance and an interval contains $\lceil\sqrt{\Delta}\rceil$ nodes. Based on this observation, we introduce Algorithm $\mathcal{A}_{apx}$, a hybrid algorithm which detects high interference instances and applies $\mathcal{A}_{gen}$, or otherwise connects the nodes linearly.

In the following, we first present a suitable criterion to identify "high interference" instances. Given a network graph $G = (V, E)$ in the highway model, let the graph $G_{lin} = (V, E_{lin})$ denote the graph where all nodes in V are linearly connected. To cause high interference at a node v in G_{lin}, it is required that many nodes cover v with their corresponding disks. However, with increasing distance to v these nodes need increasing distances to their nearest neighbors in the opposite direction of v to interfere with the latter. This leads to an exponential characteristic of these nodes since the edges in E_{lin} accounting for the interference at v form a fragmented exponential node chain. Consequently, the *critical nodes* of v are defined as follows:

Definition 2.6. *Given a linearly connected graph $G_{lin} = (V, E_{lin})$. The critical node set of a node v is defined as*

$$C_v = \{u | u \neq v; \exists w, |u, w| \geq |u, v| \wedge \{u, w\} \in E_{lin}\}.$$

In other words, the critical nodes of a node v are those nodes interfering with v if the graph G is connected linearly. Based on the previous results in this section we are able to lower-bound the interference of a minimum-interference topology of G as follows.

Lemma 2.10. *Given a graph $G = (V, E)$, let $\gamma = \max_{v \in V} |C_v|$ be the maximum number of critical nodes at any node. A minimum-interference topology for G yields interference in $\Omega(\sqrt{\gamma})$.*

Proof. Let $v \in V$ be the node with maximum interference in G_{lin}. Thus, $|C_v| = \gamma$ as all nodes interfering with v are in C_v. Without loss of generality, we assume that at least half of the nodes in C_v are to the right of v. Let C_v^r be the set of all nodes in C_v to the right of v. We number the nodes $c_i \in C_v^r$ according to their occurrence from left to right. Note that the nodes in C_v^r constitute a *virtual* exponential node chain as the distance to their nearest neighbor to the right must at least double from c_i to c_{i+1}. Therefore, Theorem 2.7 applies directly to the nodes in C_v^r. Due to the fact that $|C_v^r| \geq |C_v|/2$ and together with Theorem 2.7 we obtain $\mathrm{O}(\sqrt{|C_v|})$ as a lower bound for the interference at v. □

Algorithm $\mathcal{A}_{apx}$ makes use of Lemma 2.10 to decide whether the existing instance exhibits inherently high interference. In particular Algorithm $\mathcal{A}_{apx}$ works as follows: $\mathcal{A}_{apx}$ first computes γ. If $\gamma > \sqrt{\Delta}$, $\mathcal{A}_{gen}$ is applied to the graph. Otherwise, if $\gamma \leq \sqrt{\Delta}$, $\mathcal{A}_{apx}$ connects all nodes of the given graph linearly.

Theorem 2.11. *Given a graph G, Algorithm $\mathcal{A}_{apx}$ computes a resulting topology which approximates the optimal interference of G up to a factor in* $\mathrm{O}(\sqrt[4]{\Delta})$.

Proof. We analyze the two possible cases in $\mathcal{A}_{apx}$.

Case $\gamma > \sqrt{\Delta}$: According to Theorem 2.9, $\mathcal{A}_{gen}$ yields interference in $\mathrm{O}(\sqrt{\Delta})$. On the other hand, by Lemma 2.10, a minimum-interference topology produces at least $\Omega(\sqrt{\gamma})$ interference. We therefore obtain an approximation ratio in $\mathrm{O}(\sqrt{\Delta})/\Omega(\sqrt{\gamma}) \in \mathrm{O}(\sqrt[4]{\Delta})$.

Case $\gamma \leq \sqrt{\Delta}$: By Lemma 2.10, the minimum-interference topology results in interference of at least $\Omega(\sqrt{\gamma})$. Connecting G linearly we obtain interference γ by definition. Consequently, the approximation ratio of $\mathcal{A}_{apx}$ is in $\gamma/\Omega(\sqrt{\gamma}) \in \mathrm{O}(\sqrt[4]{\Delta})$. □

2.3 Interference in Heterogeneous Networks

Heterogeneous sensor networks, also known as multi-tier networks, consist of sensor nodes with different capabilities and power requirements. In this section, we assume a two-tier sensor network composed of powerful gateways such as Stargate class nodes [4] and low-power sensor nodes [47]. The gateways are interconnected by an external backbone network; sensor nodes are connected via radio links to gateways. The totality of the gateways forms the infrastructure for distributed applications running on the nodes.

Since communication over wireless links takes place in a shared medium, interference may occur at a node if it is within transmission range of more than one gateway.[3] To prevent such collisions, coordination among the conflicting gateways is required. Commonly this problem is solved by segmenting the available frequency spectrum into channels to be assigned to the gateways in such a way as to prevent interference, in particular such that no two gateways with overlapping transmission range use the same channel.

We assume a different approach to interference reduction. Along the lines of Section 2.2 our analysis is based on the observation that interference effects occurring at a node depend on the number of gateways by whose transmission ranges it is covered. In particular for solutions using frequency division multiplexing as described above, the number of gateways covering a node is a lower bound for the number of channels required to avoid conflicts; a reduction in the required number of channels, in turn, can be exploited to broaden the frequency segments and consequently to increase communication bandwidth. On the other hand, also with systems using code division multiplexing, the coding overhead can be reduced if only a small number of gateways cover a nodes.

The transmission range of a gateway—and consequently the coverage properties of the nodes—depends on its position, obstacles hindering the propagation of electromagnetic waves, such as walls, buildings, or mountains, and the gateway transmission power. Since due to legal or architectural constraints the former two factors are generally difficult to control, we assume a scenario in which the gateway positions are fixed, each gateway can however adjust its transmission power. The problem of minimizing interference then consists in assigning every gateway a transmission power level such that the number of gateways covering any node is minimal (cf. Figure 2.12). At the same time however, it has to be guaranteed that every node is covered by at least one gateway in order to maintain availability of the network.

In Figure 2.12 the area covered by a gateway b transmitting with a given power level is represented by a disk centered at b and having a radius corresponding to the chosen transmission power. Practical measurements however show that this idealization is far from realistic. Not only are mechanical inaccuracies inevitable in the construction of antennas, but more importantly the presence of obstacles to the propagation of electromagnetic signals—such as buildings, mountains, or even weather conditions—can lead to areas covered by signal transmission that hardly resemble disks in practice. These considerations motivate that in order to study the described interference reduction problem we abstract from network node positions and circular transmission areas.

In our analysis we formalize the task of reducing interference as a combinatorial optimization problem. For this purpose we model the transmission range of a gateway having chosen a specific transmission power level as a set

---

[3] Note that a gateway may resolve contention between associated nodes by employing a time division multiple access protocol.

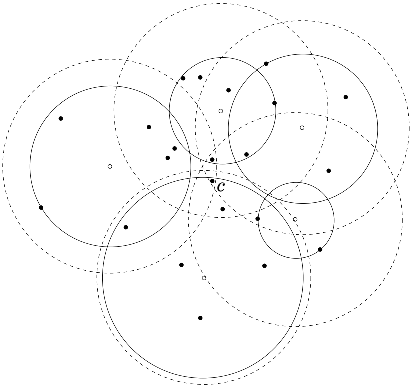

Figure 2.12: If the gateways (hollow points) are assigned identical transmission power levels (dashed circles), node *c* experiences high interference, since it is covered by all gateways. Interference can be reduced by assigning appropriate power values (solid circles), such that all nodes are covered by at most two gateways.

containing exactly all nodes covered thereby. The totality of transmission ranges selectable by all gateways is consequently modeled as a collection of node sets. More formally, this yields the *Minimum Membership Set Cover (MMSC)* problem: Given a set of elements U (modeling nodes) and a collection S of subsets of U (transmission ranges), choose a solution $S' \subseteq S$ such that every element occurs in at least one set in S' (maintain network availability) and that the *membership* $M(u, S')$ of any element u with respect to S' is minimal, where $M(u, S')$ is defined as the number of sets in S' in which u occurs (interference).

Having defined this formalization, we show—by reduction from the related Minimum Set Cover problem—that the MMSC problem is NP-complete and that no polynomial time algorithm exists with approximation ratio less than $\ln n$ unless $NP \subset TIME(n^{\mathrm{O}(\log\log n)})$. We additionally present a probabilistic algorithm based on linear programming relaxation asymptotically matching this lower bound, particularly yielding an approximation ratio in $\mathrm{O}(\log n)$ with high probability. Furthermore we study how the presented algorithm performs on practical network instances.

2.3.1 Minimum Membership Set Cover

As described in the introduction, the problem considered in this chapter is to assign to each gateway a transmission power level such that interference is minimized while all nodes are covered. For our analysis we formalize this problem by introducing a combinatorial optimization problem referred to as *Minimum Membership Set Cover*. In particular, nodes are modeled as elements and the transmission range of a gateway given a certain power level is represented as the set of thereby covered elements. In the following, we first define the membership of an element given a collection of sets:

Definition 2.7 (Membership). *Let U be a finite set of elements and S be a collection of subsets of U. Then the* membership $M(u, S)$ *of an element u is defined as* $|\{T \mid u \in T, T \in S\}|$.

Informally speaking, MMSC is identical to the MSC problem apart from the minimization function. Where MSC minimizes the total number of sets, MMSC tries to minimize element membership. Particularly, MMSC can be defined as follows:

Definition 2.8 (Minimum Membership Set Cover). *Let U be a finite set of elements with $|U| = n$. Furthermore let $S = \{S_1, \ldots, S_m\}$ be a collection of subsets of U such that $\bigcup_{i=1}^{m} S_i = U$. Then* Minimum Membership Set Cover *(MMSC) is the problem of covering all elements in U with a subset $S' \subseteq S$ such that* $\max_{u \in U} M(u, S')$ *is minimal.* [4]

Note that—as motivated in the introduction—the problem statement does not require the collection of subsets S to reflect geometric positions of network nodes. For a given problem instance to be valid, $\bigcup_{i=1}^{m} S_i = U$ is sufficient.

2.3.2 Problem Complexity

In this section we address the complexity of the *Minimum Membership Set Cover* problem. We show that MMSC is *NP*-complete and therefore no polynomial time algorithm exists that solves MMSC unless $P = NP$.

Theorem 2.12. *MMSC is NP-complete.*

Proof. We will prove that MMSC is *NP*-complete by reducing MSC to MMSC. Consider an MSC instance (U, S) consisting of a finite base set of elements U and a collection S of subsets of U. The objective is to choose a subset S' with minimum cardinality from S such that the union of the chosen subsets of U contains all elements in U.

We now define a set $\widetilde{U}$ by adding a new element e to U, construct a new collection of subsets $\widetilde{S}$ by inserting e into all sets in S, and consider $(\widetilde{U}, \widetilde{S})$

---

[4] Besides minimizing the *maximal* membership value over all elements, also minimization of the *average* membership value can be considered a reasonable characterization of the interference reduction problem. The fact however that—given a solution S'—the sum of all membership values equals the sum of the cardinalities of the sets in S' shows that this min-average variant is identical to the Weighted Set Cover [24] problem with the set weights corresponding to their cardinalities.

as an instance of MMSC. Since element e is in every set in $\widetilde{S}$, it follows that e is an element with maximum membership in the solution S' of MMSC. Moreover, the membership of e in S' is equal to the number of sets in the solution. Therefore MMSC minimizes the number of sets in the solution by minimizing the membership of e. Consequently we obtain the solution for MSC of the instance (U, S) by solving MMSC for the instance $(\widetilde{U}, \widetilde{S})$ and extracting element e from all sets in the solution.

We have shown a reduction from MSC to MMSC, and therefore the latter is *NP*-hard. Since solutions for the decision problem of MMSC are verifiable in polynomial time, it is in *NP*, and consequently the MMSC decision problem is also *NP*-complete. $\square$

Now that we have proved MMSC to be *NP*-complete and therefore not to be optimally computable within polynomial time unless $P = NP$, the question arises, how closely MMSC can be approximated by a polynomial time algorithm. This is partly answered with the following lower bound.

Theorem 2.13. *There exists no polynomial time approximation algorithm for MMSC with an approximation ratio less than* $(1 - \mathrm{o}(1)) \ln n$ *unless* $NP \subset TIME(n^{\mathrm{O}(\log \log n)})$.

Proof. The reduction from MSC to MMSC in the proof of Theorem 2.12 is approximation-preserving, that is, it implies that any lower bound for MSC also holds for MMSC. In [37] it is shown that $\ln n$ is a lower bound for the approximation ratio of MSC unless $NP \subset TIME(n^{\mathrm{O}(\log \log n)})$. Thus, $\ln n$ is also a lower bound for the approximation ratio of MMSC. $\square$

2.3.3 Approximating the MMSC Problem

In the previous section a lower bound of $\ln n$ for the approximability of the MMSC problem by means of polynomial time approximation algorithms has been established. In this section we show how to obtain a $\mathrm{O}(\log n)$-approximation with high probability[5] using LP relaxation techniques.

LP Formulation of MMSC

We first derive the integer linear program which describes the MMSC problem and then formulate the linear program that relaxes the integrality constraints.

Let $S' \subseteq S$ denote a subset of the collection S. To each $S_i \in S$ we assign a variable $x_i \in \{0, 1\}$ such that $x_i = 1 \Leftrightarrow S_i \in S'$. For S' to be a set cover, it is required that for each element $u_i \in U$, at least one set S_j with $u_i \in S_j$ is in S'. Therefore, S' is a set cover of U if and only if for all $i = 1, ..., n$ it holds that $\sum_{S_j : u_i \in S_j} x_j \geq 1$. For S' to be minimal in the number of sets

---

[5] Throughout the chapter, an event E occurring "with high probability" stands for $Pr[E] = 1 - \mathrm{O}(\frac{1}{n})$.

that cover a particular element, we need a second set of constraints. Let z be the maximum membership over all elements caused by the sets in S'. Then for all $i = 1, ..., n$ it follows that $\sum_{S_j : u_i \in S_j} x_j \leq z$. The MMSC problem can consequently be formulated as the integer program $\mathrm{IP}_{\mathrm{MMSC}}$:

$$\begin{aligned}
&\text{minimize } z \\
&\text{subject to} \quad \sum_{S_j : u_i \in S_j} x_j \geq 1 \qquad i = 1, ..., n \\
&\qquad\qquad\quad\; \sum_{S_j : u_i \in S_j} x_j \leq z \qquad i = 1, ..., n \\
&\qquad\qquad\qquad\quad x_j \in \{0, 1\} \qquad j = 1, ..., m
\end{aligned}$$

By relaxing the constraints $x_j \in \{0, 1\}$ to $x'_j \geq 0$, we obtain the following linear program $\mathrm{LP}_{\mathrm{MMSC}}$:

$$\begin{aligned}
&\text{minimize } z \\
&\text{subject to} \quad \sum_{S_j : u_i \in S_j} x'_j \geq 1 \qquad i = 1, ..., n \\
&\qquad\qquad\quad\; \sum_{S_j : u_i \in S_j} x'_j \leq z \qquad i = 1, ..., n \\
&\qquad\qquad\qquad\quad x'_j \geq 0 \qquad j = 1, ..., m
\end{aligned}$$

The integer program $\mathrm{IP}_{\mathrm{MMSC}}$ yields the optimal solution z^* for an MMSC problem. The derived linear program LP_{MMSC} therefore obtains a fractional solution z' with $z' \leq z^*$, since we allow the variables x'_j to be in [0,1].

Algorithm and Analysis

We will now present a $\mathrm{O}(\log n)$-approximation algorithm, referred to as $\mathcal{A}_{\mathrm{MMSC}}$, for the MMSC problem. Given an MMSC instance (U, S), the algorithm first solves the linear program $\mathrm{LP}_{\mathrm{MMSC}}$ corresponding to (U, S). In a second step, $\mathcal{A}_{\mathrm{MMSC}}$ performs randomized rounding (see [121]) on a feasible solution vector $\underline{x}'$ for $\mathrm{LP}_{\mathrm{MMSC}}$, to derive a vector $\underline{x}$ with $x_i \in \{0, 1\}$. Finally it is ensured that $\underline{x}$ is a feasible solution for $\mathrm{IP}_{\mathrm{MMSC}}$ and consequently a set cover.

For the analysis of $\mathcal{A}_{\mathrm{MMSC}}$ the following two mathematical facts are required. Their proofs are omitted and can be found in mathematical textbooks.

```
Algorithm $\mathcal{A}_{\text{MMSC}}$
Input: an MMSC instance $(U, S)$
1: compute solution vector $\underline{x}'$ to the linear program $\text{LP}_{\text{MMSC}}$ corresponding to $(U, S)$
2: $p_i := \min\{1,\ x'_i \cdot \log n\}$
3: $x_i := \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{otherwise} \end{cases}$
4: for all $u_i \in U$ do
5:     if $\sum_{S_j : u_i \in S_j} x_j = 0$ then
6:         set $x_j = 1$ for any $j$ such that $u_i \in S_j$
7:     end if
8: end for
Output: MMSC solution $S'$ corresponding to $\underline{x}$
```

Fact 2.1. **(Means Inequality)** *Let* $\mathcal{A} \subset \mathbb{R}^+$ *be a set of positive real numbers. The product of the values in* $\mathcal{A}$ *can be upper-bounded by replacing each factor with the arithmetic mean of the elements of* $\mathcal{A}$*:*

$$\prod_{x \in \mathcal{A}} x \ \leq \ \left(\frac{\sum_{x \in \mathcal{A}} x}{|\mathcal{A}|} \right)^{|\mathcal{A}|} .$$

Fact 2.2. *For all* n*,* t*, such that* $n \geq 1$ *and* $|t| \leq n$*,*

$$e^t \left(1 - \frac{t^2}{n}\right) \leq \left(1 + \frac{t}{n}\right)^n \ \leq \ e^t .$$

We prove $\mathcal{A}_{\text{MMSC}}$ to be a $\mathrm{O}(\log n)$-approximation algorithm for IP_{MMSC} in several steps. We first show that the membership of an element in U after the randomized rounding step of $\mathcal{A}_{\text{MMSC}}$ is bounded with high probability.

Lemma 2.14. *The membership of an element* u_i *after Line 3 of* $\mathcal{A}_{\text{MMSC}}$ *is at most* $2e \log n \cdot z^*$ *with high probability.*

Proof. The optimal solution of LP_{MMSC} leads to fractional values x'_j and does not admit a straightforward choice of the sets S_j. Using randomized rounding, $\mathcal{A}_{\text{MMSC}}$ converts the fractional solution to an integral solution S'. In Line 3, a set S_j is chosen to be in S' with probability $x'_j \cdot \log n$. Thus, the expected membership of an element u_i is

$$E[M(u_i, S')] = \sum_{S_j : u_i \in S_j} x'_j \cdot \log n \leq \log n \cdot z' . \tag{2.1}$$

The last inequality follows directly from the second set of constraints of LP_{MMSC}. Since $z' \leq z^*$, it follows that the expected membership for u_i

is at most $\log n \cdot z^*$. Now we need to ensure that, with high probability, u_i is not covered too often. Since randomized rounding can be modeled as Poisson trials, we are able to use a Chernoff bound [112]. Let Y_i be a random variable denoting the membership of u_i with expected value $\mu = E[M(u_i, S')]$. Applying the Chernoff bound we derive

$$Pr\left[Y_i \geq (1+\delta)\,\mu\right] < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}.$$

Choosing $\delta \geq 2e - 1$, the right hand side of the inequality simplifies to

$$\left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu} \leq \left(\frac{e^{\delta}}{(2e)^{(1+\delta)}}\right)^{\mu} < \left(\frac{e^{\delta}}{(2e)^{\delta}}\right)^{\mu} = 2^{-\delta\mu}. \tag{2.2}$$

Since the above Chernoff bound corresponds to the upper tail of the probability distribution of Y_i and as μ is at most $\log n \cdot z^*$, it follows that

$$Pr\left[Y_i \geq (1+\delta)\log n \cdot z^*\right] \leq Pr\left[Y_i \geq (1+\delta)\,\mu\right].$$

But for this inequality to hold, only $(1+\delta)\mu \leq c \log n \cdot z^*$ for some constant c is required. Thus, by setting $(1+\delta)\mu = c\log n \cdot z^*$ and using Inequality (2.1), we obtain

$$\delta\mu \quad \geq \quad (c-1)\log n \cdot z^*. \tag{2.3}$$

Using Inequalities (2.2) and (2.3) we can then bound the probability that the membership of u_i is greater than $c \log n \cdot z^*$ as follows:

$$Pr\left[Y_i \geq c\log n \cdot z^*\right] < 2^{-\delta\mu} \leq 2^{-(c-1)\log n \cdot z^*} = \frac{1}{n^{(c-1)z^*}}.$$

To compute c, we again consider the equation $(1+\delta)\mu = c \log n \cdot z^*$. Solving for δ, we derive

$$\delta = \frac{c\log n \cdot z^*}{\mu} - 1.$$

As a requirement for Inequality (2.2) we demand δ to be greater or equal to $2e - 1$. Furthermore, the right hand side of the inequality is minimal if μ is maximal. Thus, using Inequality (2.1) we obtain

$$\frac{c\log n \cdot z^*}{\log n \cdot z^*} - 1 \quad \geq \quad 2e - 1$$

or $c \geq 2e$. Taking everything together and using $z^* \geq 1$ it follows that

$$Pr\left[Y_i \geq 2e\log n \cdot z^*\right] < \frac{1}{n^{(2e-1)z^*}} \in \mathrm{O}\left(\frac{1}{n^4}\right).$$

□

Now we are ready to show that after randomized rounding all elements have membership at most $2e\log n \cdot z^*$ with high probability.

Lemma 2.15. *The membership of all elements in* U *after Line 3 of* $\mathcal{A}_{\mathrm{MMSC}}$ *is at most* $2e\log n \cdot z^*$ *with high probability.*

Proof. Let E_i be the event that the membership of element u_i after Line 3 of $\mathcal{A}_{\mathrm{MMSC}}$ is greater than $2e\log n \cdot z^*$. Then, the probability that the membership for all elements in U is less than $2e\log n \cdot z^*$ equals

$$Pr[\bigwedge_{i=1}^{n} \overline{E_i}\,].$$

We know from Lemma 2.14 that the probability $\Pr[E_i]$ is less than $1/n^{(2e-1)z^*}$. Since the events are clearly not independent, we cannot apply the product rule. However, it was shown in [136] that

$$Pr[\bigwedge_{i=1}^{n} \overline{E_i}\,] \geq \prod_{i=1}^{n} Pr[\,\overline{E_i}\,]. \tag{2.4}$$

We can make use of this bound, since $\mathrm{IP}_{\mathrm{MMSC}}$ features the *positive correlation* property assumed in [136]. Consequently, setting $\alpha = (2e-1)z^*$ and using Inequality (2.4), it follows that

$$\begin{aligned} Pr[\bigwedge_{i=1}^{n} \overline{E_i}\,] &\geq \left(1-\frac{1}{n^\alpha}\right)^n \geq \left(1-\frac{1}{n^\alpha}\right)^{\frac{n^\alpha-1}{n^{\alpha-1}-\frac{1}{n}}} \\ &\geq e^{-\frac{1}{n^{\alpha-1}-\frac{1}{n}}} > 1-\frac{1}{n^{\alpha-1}-\frac{1}{n}}. \end{aligned}$$

For the third inequality we use Fact 2.2 with $t=-1$, which leads to the inequality

$$e^{-1} \leq (1-1/n)^{n-1}.$$

The last inequality is derived through Taylor series expansion of the left hand term. Consequently, using $\alpha = (2e-1)z^*$ and $z^* \geq 1$ we obtain

$$Pr[\bigwedge_{i=1}^{n} \overline{E_i}\,] = 1 - \mathrm{O}\left(\frac{1}{n^3}\right).$$

□

Since $\mathcal{A}_{\mathrm{MMSC}}$ uses randomized rounding, we do not always derive a feasible solution for $\mathrm{IP}_{\mathrm{MMSC}}$ after Line 3 of the algorithm. That is, there exist elements in U that are not covered by a set in S'. But we can show in the following lemma that each single element is covered with high probability.

Lemma 2.16. *After Line 3 of $\mathcal{A}_{\text{MMSC}}$, an element u_i in U is covered with high probability.*

Proof. For convenience we define C_i to be the set $\{S_j \mid u_i \in S_j\}$. From LP_{MMSC} we know that $\sum_{S_j \in C_i} x'_j \geq 1$. Thus, it follows that

$$\sum_{S_j \in C_i} p_j \geq \log n. \tag{2.5}$$

Let q_i be the probability that an element u_i is contained in none of the sets in S' obtained by randomized rounding, that is, $q_i = Pr\,[M(u_i, S') = 0]$. Consequently, we have

$$\begin{aligned} q_i &= \prod_{S_j \in C_i} (1 - p_j) \leq \left(1 - \frac{\sum_{S_j \in C_i} p_j}{|C_i|}\right)^{|C_i|} \\ &\leq e^{-\sum_{S_j \in C_i} p_j} \leq e^{-\log n} = \frac{1}{n}. \end{aligned}$$

The first inequality follows from Fact 2.1, the second inequality follows from Fact 2.2, and the third step is derived from Inequality (2.5). □

In Lines 4 to 8 of $\mathcal{A}_{\text{MMSC}}$ it is ensured that the final solution S' is a set cover. This is achieved by consecutively including sets in S', until all elements are covered. In the following we show that the additional maximum membership increase caused thereby is bounded with high probability.

Lemma 2.17. *In Lines 4 to 8 of $\mathcal{A}_{\text{MMSC}}$, the maximum membership in U is increased by at most $\mathrm{O}(\log n)$ with high probability.*

Proof. To bound the number of sets added in the considered part of the algorithm we again employ a Chernoff bound. Let Z be a random variable denoting the number of uncovered elements after Line 3 of $\mathcal{A}_{\text{MMSC}}$. From Lemma 2.16 we know that an element is uncovered after randomized rounding with probability less than $1/n$. Then, the expected value μ for Z is less than 1. Using a similar analysis as in Lemma 2.14, we obtain

$$Pr\,[Z \geq c] < 2^{-c+1},$$

where $c \geq 2e$ is required. Setting $c = \log n + 2e$, it follows that

$$Pr\,[Z \geq \log n + 2e] < \frac{2}{n \cdot 4^e} \in \mathrm{O}\left(\frac{1}{n}\right).$$

The proof is concluded by the observation that each additional set added in the second step of $\mathcal{A}_{\text{MMSC}}$ increases the maximum membership in U by at most one. Since only $\mathrm{O}(\log n)$ elements have to be covered with high probability and as it is sufficient to add one set per element, the lemma follows.[6] $\square$

Now we are ready to prove that $\mathcal{A}_{\mathrm{MMSC}}$ yields a $\mathrm{O}(\log n)$-approximation for $\mathrm{IP}_{\mathrm{MMSC}}$ and consequently also for MMSC.

Theorem 2.18. *Given an MMSC instance consisting of m sets and n elements, $\mathcal{A}_{\mathrm{MMSC}}$ computes a $\mathrm{O}(\log n)$-approximation with high probability. The running time of $\mathcal{A}_{\mathrm{MMSC}}$ is polynomial in $m \cdot n$.*

Proof. The approximation factor in the theorem directly follows from Lemmas 2.15 and 2.17. The running time result is a consequence to the existence of algorithms solving linear programs in time polynomial in the program size [71] and to the fact that $\mathrm{LP}_{\mathrm{MMSC}}$ can be described using -1, 0, and 1 as coefficients only. $\square$

Alternative Algorithm

In an alternative version of the algorithm, the values $\underline{x}'$ obtained by solving $\mathrm{LP}_{\mathrm{MMSC}}$ can be directly employed as probabilities for randomized rounding (without the additional factor of $\log n$). In this case randomized rounding is repeated for all sets containing elements not yet covered until resulting in a set cover. With similar arguments as for $\mathcal{A}_{\mathrm{MMSC}}$, it can be shown that this modified algorithm achieves the same approximation factor and that it terminates after repeating randomized rounding at most $\log n$ times, both with high probability.

2.3.4 Practical Networks

Whereas the previous section showed that $\mathcal{A}_{\mathrm{MMSC}}$ approximates the optimal solution up to a factor in $\mathrm{O}(\log n)$, this section discusses practical networks. In particular, the algorithms $\mathcal{A}_{\mathrm{MMSC}}$ and $\widetilde{\mathcal{A}}_{\mathrm{MMSC}}$—the alternative algorithm described in Section 2.3.3—are considered. Since the approximation performance of algorithms is studied, we denote by the *membership of a solution* the minimization function value—that is the maximum membership over all nodes—of the corresponding MMSC solution.

The studied algorithms were executed on instances generated by placing gateways and nodes randomly according to a uniform distribution on a square field with side length 5 units. Adaptable transmission power values were modeled by attributing to each gateway circles with radii 0.25, 0.5, 0.75, and

---

[6] Since in the above Chernoff bound μ is at most a constant, a more careful analysis would yield that the maximum membership in U is increased—with high probability—by $\mathrm{O}(\log n/\log\log n)$ only. This improvement has however no impact on the main result of this chapter.

1 unit; each such circle then contributes one set containing all covered nodes to the problem instance thereafter presented to the algorithms.

As shown in the previous section, the approximation factor of the algorithms depends on the number of nodes. For this reason the simulations were carried out over a range of node densities. Since the membership value obtained by solving LP_{MMSC} lies below the optimal solution and therefore the gap between the algorithm result and the solution of the linear program is an upper bound for the obtained approximation ratio, the LP_{MMSC} result z' is also considered.

For a gateway density of 2 gateways per unit disk, Figure 2.13(a) shows the mean membership values over 200 networks—for each simulated node density—for the results computed by $\mathcal{A}_{\text{MMSC}}$, $\widetilde{\mathcal{A}}_{\text{MMSC}}$, and the values obtained by solving LP_{MMSC}. The results depict that for this relatively low base-station density all measured values are comparable and increase with growing node density. In contrast, for a higher base-station density of 5 gateways per unit disk (cf. Figure 2.13(b)), a gap opens between the $\mathcal{A}_{\text{MMSC}}$ and LP_{MMSC} results. Whereas the ratio between these two result series—as mentioned before, an upper bound for the approximation ratio—rises sharply for low node densities, its increase diminishes for higher node densities, which corresponds to the $\text{O}(\log n)$ approximation factor described in the theoretical analysis. Additionally, it can be observed that $\widetilde{\mathcal{A}}_{\text{MMSC}}$ performs significantly better than $\mathcal{A}_{\text{MMSC}}$. The reason for this effect lies in the fact that $\mathcal{A}_{\text{MMSC}}$ multiplies the $\underline{x}'$ values resulting from LP_{MMSC} with the factor $\log n$ to obtain the probabilities employed for randomized rounding, whereas this multiplication is not performed by $\widetilde{\mathcal{A}}_{\text{MMSC}}$. The approximation gap becomes even wider for higher gateway densities, such as 10 gateways per unit disk (Figure 2.13(c)). Our simulations showed however that beyond this gateway density no significant changes in the membership results can be observed.

The increasing gap between the simulated algorithms and the LP_{MMSC} solution with growing gateway density can be explained by the following observation: For low gateway densities—where problem instances contain a small number of sets—a relatively large number of nodes are covered by only one set, which consequently will have to be chosen in both the LP_{MMSC} and the algorithm solutions; for high gateway densities, in contrast, the solution weights $\underline{x}'$ computed by LP_{MMSC} can be distributed more evenly among the relatively high number of available sets, and the potential of "committing an error" during randomized rounding increases.

In summary, the simulations show that the considered algorithms approximate the optimum solution well on practical networks. Comparing $\mathcal{A}_{\text{MMSC}}$ and $\widetilde{\mathcal{A}}_{\text{MMSC}}$, it can be observed that, in practice, the latter algorithm performs even better than the former.

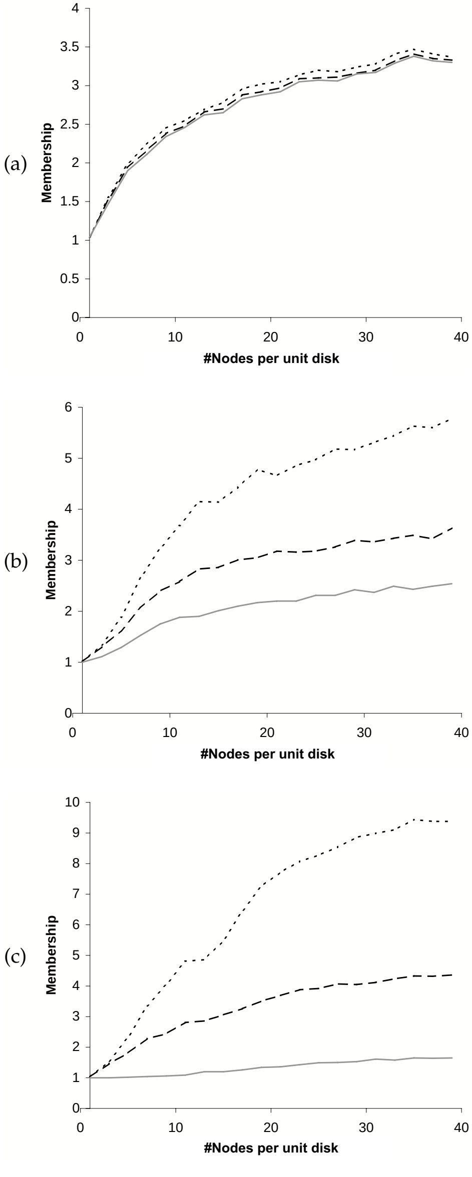

Figure 2.13: Mean values of the membership results obtained by $\mathcal{A}_{\text{MMSC}}$ (dotted), $\widetilde{\mathcal{A}}_{\text{MMSC}}$ (dashed), and the LP_{MMSC} solution with 2 (a), 5 (b), and 10 (c) gateways per unit disk.

2.4 Related Work

The assumption that nodes are distributed randomly in the plane according to a uniform probability distribution formed the basis of pioneering work in the field of topology control in ad hoc networks [55, 138].

Later proposals adopted constructions originally studied in computational geometry, such as the Delaunay Triangulation [57], the minimum spanning tree [124], the Relative Neighborhood Graph [14], or the Gabriel Graph [129]. Most of these contributions mainly considered energy-efficiency of paths preserved by the resulting topology, whereas others exploited the planarity property of the proposed constructions for geographic routing [77, 15, 84].

The Delaunay Triangulation and the minimum spanning tree not being computable locally and thus not being practicable, a next generation of topology control algorithms emphasized locality. The CBTC algorithm [151] was the first construction to simultaneously focus on several desired properties, in particular being an energy spanner with bounded degree. This process of developing local algorithms featuring more and more properties was continued, partly based on CBTC, partly based on local versions of classic geometric constructions such as the Delaunay Triangulation [92] or the minimum spanning tree [90]. Among the most recent such results are a locally computable planar constant-stretch distance (and energy) spanner with constant-bounded node degree [149] or a construction with similar properties additionally having low overall energy consumption [93]. Other approaches try to build on minimal assumptions about the capabilities of nodes and signal propagation characteristics [152] or takes up the average-graph perspective of early work in the field; [12] for instance shows that the simple algorithm choosing the k nearest neighbors works surprisingly well in such graphs. Yet another thread of research considered topology control in three dimensions employing existing approaches such as the Delaunay Triangulation [43] or techniques based on CBTC [6, 148] to generate connectivity preserving topologies.

A different aspect of topology control is considered by algorithms trying to form clusters of nodes. Most of these proposals are based on (connected) dominating sets [41, 42, 66, 3, 58, 2, 83, 80] and focus on locality and provable properties. Cluster-based constructions are commonly regarded as a variant of topology control in the sense that energy-consuming tasks can be shared among the members of a cluster.

Topology control having so far mainly been of interest to theoreticians, first promising steps are being made towards exploiting the benefit of such techniques also in practical networks [73, 26, 18, 33]. A more detailed overview of topology control techniques in general can be found in [130].

As mentioned earlier, reducing interference—and its energy-saving effects on the medium access layer—is one of the main goals of topology control besides direct energy conservation as a consequence of transmission power restriction. However, all the above topology control algorithms at the most implicitly try to reduce interference. Where interference is mentioned as an issue at all, it is maintained to be confined at a low level as a consequence of sparseness or low degree of the resulting topology graph.

To the best of our knowledge, the authors of [105] are the first to define an explicit notion of interference. Based on this interference model between edges, a time-step routing model and a concept of congestion is introduced. It is shown that there are inevitable trade-offs between congestion, power consumption and dilation (or hop-distance). For some node sets, congestion and energy are even shown to be incompatible.

The static interference models—in the sense that they are defined independent of current network traffic—, as introduced in this chapter, formed the basis for continuing research [107, 109, 67, 48, 11]. For the receiver-based inference model presented in Section 2.2, our approach was advanced in [48], leading to a structure with interference in $O(\sqrt{\Delta})$ in two-dimensional node distributions without restriction to the highway model. Their results rely on computational geometric tools such as local neighbor graphs, ϵ-nets, and quad-tree decomposition. Moreover, there exists a simpler randomized algorithm computing low-interference topologies for such networks.

The interference issue as a scheduling problem over time—in a sense taking up the approach from [105]—was later again studied in [110]. In contrast to the work presented in this chapter, [110] models interference with the physical Signal-to-Interference-plus-Noise Ratio and defines the concept of scheduling complexity for the connectivity of wireless networks. [111] extends this concept to arbitrary given network topologies and demonstrates the existence of a relation between this scheduling complexity and the receiver-centric interference model defined in Section 2.2.

The problem of minimizing interference in multi-tiered systems addressed in Section 2.3 is related to studies of interference issues in cellular networks in the context of frequency division multiplexing. There, the available network frequency spectrum is divided into narrow channels assigned to cells in a way to avoid interference conflicts. In particular two types of conflicts can occur, adjacent cells using the same channel (co-channel interference) and insufficient frequency distance between channels used within the same cell (adjacent channel interference). Maximizing the reuse of channels respecting these conflicts is generally studied by means of the combinatorial problem of conflict graph coloring using a minimum number of colors. The settings in which this problem is considered are numerous and include hexagon graphs, geometric intersection graphs (such as unit disk graphs), and planar graphs, but also (non-geometric) general graphs. In addition both static and dynamic (or on-line) approaches are studied [113]. The fact that channel separation constraints can depend on the distance of cells in the conflict graph is studied by means of graph labeling [64]. The problem of frequency assignment is tackled in a different way in [36] exploiting the observation that in every region of an area covered by the communication network it is sufficient that exactly one gateway with a unique channel can be heard. As mentioned, all these studied models try to avoid interference conflicts occurring when using frequency division multiplexing. In contrast, the problem described in Section 2.3 assumes a different approach in aiming at interference reduction by having the gateways choose suitable transmission power levels.

The problem of reducing interference is formalized in a combinatorial optimization problem named Minimum Membership Set Cover. As suggested by its name, at first sight its formulation resembles closely the long-known and well-studied Minimum Set Cover (MSC) problem, where the number of sets chosen to cover the given elements is to be minimized [68]. That the MMSC and the MSC problems are however of different nature can be concluded from the following observation: For any MSC instance consisting of n elements, a greedy algorithm approximates the optimal solution with an approximation ratio at most $H(n) \leq \ln n + 1$ [68], which has later been shown to be tight up to lower order terms unless $NP \subset TIME(n^{\mathcal{O}(\log \log n)})$ [37, 98]. For the MMSC problem in contrast, there exist instances where the same greedy algorithm fails to achieve any nontrivial approximation of the optimal solution.

In the context of network traffic congestion, [96] considered a problem similar to our analysis of the MMSC problem in that linear program relaxation was employed to minimize a maximum value.

Chapter 3

Gathering Correlated Data

Dependent on the node density of a deployed sensor network, different sensor nodes partially monitor the same spatial region. Thus, the nodes may observe the same physical phenomenon leading to correlation in their acquired data. To account for this circumstance and to save energy, data should be processed on its way from the information source to the sink. This technique is commonly referred to as (in-network) data aggregation. Thereby, a sensor node uses a so-called aggregation function to encode the available data before forwarding it to the sink. Several coding strategies were proposed in recent research that can be classified as follows. On the one hand, there exist the so-called multi-input coding strategies [45, 159, 116] where a node waits until information from several other nodes is available before it performs data aggregation. On the other hand, there also exist single-input coding strategies [27] where the encoding of a node's information only depends on the information of one other node.

In this chapter, we only consider conditional coding where data from one node can be compressed in the presence of data from other nodes. We therefore only refer to conditionally encoded data as encoded data and speak about raw data otherwise.[1] In particular, we focus on single-input coding strategies because they feature several advantages compared to multi-input coding strategies. The most important one is certainly the ability to apply single-input coding also in asynchronous networks where no timing assumptions can be made. Using multi-input coding on the other hand, certain timing assumptions have to be made since packets cannot be delayed for an indefinite time at intermediate nodes while waiting for belated information [135]—data freshness at the sink should not suffer too much from data aggregation. We distinguish two classes of single-input coding, namely self-coding and foreign coding. Using self-coding, data is only allowed to be encoded

---

[1] Even though, a node may also apply an encoding scheme to its measured data in the absence of side information.

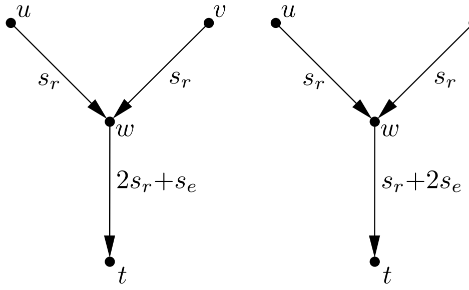

Figure 3.1: Simple network example. Raw data has size s_r while encoded data is of size s_e, with $s_e < s_r$. On the left hand side self-coding is applied while foreign coding is used on the right. For this example foreign coding is more energy efficient than self-coding.

at the producing node and only in the presence of side information from at least one other node. With foreign coding in contrast, a node is only able to encode raw data originating at another node as it is routed towards the sink using its own data.

Figure 3.1 depicts a simple network example consisting of three sensor nodes (u, v, and w) that want to propagate their raw data of size s_r to a sink t. Communication links only exist between nodes u and w, v and w, as well as w and t, respectively. Therefore, packets from u and v have to be relayed at w to reach the sink t. If a node is able to encode data due to side information the data size reduces to s_e, with $s_e < s_r$. The configuration on the left depicts the usage of self-coding. Since u and v do not have any side information they both send a packet of s_r bits towards t. Because of side information from u, and v, respectively, node w is able to encode its data such that the corresponding packet has size s_e and therefore $2s_r + s_e$ bits have to be sent over the link (w, t). On the right hand side of Figure 3.1 the same network is shown if foreign coding is applied. As with self-coding, nodes u and v send their packets of size s_r to w. However, w encodes the raw data of u and v using its own data before it relays this coded data along with its own raw data towards the sink t. Thus, only $s_r + 2s_e$ bits are sent over (w, t). Using $s_e < s_r$, foreign coding transmits less bits over (w, t) than self-coding and is thus more energy efficient for this configuration.

In case of self-coding, [27] shows that already for a very restrictive model where raw data is of size s_r and a node can encode its data to use only s_e bits, with $s_e < s_r$, in the presence of any side information, the problem of finding a minimum-energy data gathering tree is NP-complete. To the best of our knowledge, we are the first that provide an approximation algorithm for this problem with approximation ratio $2(1+\sqrt{2})$. The algorithm is based on the *shallow light tree* (SLT) introduced in [5, 10, 74] that unifies the properties of the *minimum spanning tree* (MST) and the *shortest path tree* (SPT). Considering foreign coding, we introduce an algorithm resulting in a minimum-energy data gathering topology under the assumption that the topology of the network and the correlation structure of the nodes are known. The algorithm obtains an optimal solution for the data gathering problem by reducing it to the problem of finding an MST in a directed graph which is known to be computable in polynomial time.

We assume the same network model as in Section 2.1; the network is represented as a graph $G = (V, E)$. Furthermore, let $t \in V$ denote a particular node called sink at which data from all nodes in V should be gathered. We refer to the process of gathering information from all nodes as a round. Therefore, in each round data from each node in V is sent to t for further processing. The weight $w(e)$ for an edge $e = (v_i, v_j) \in E$ is defined to be the cost of transmitting one bit of information from node v_i to node v_j, or vice versa. That is, we use an energy metric for the graph G. However, we do not consider a specific radio model such as the popular first order radio model presented in [51], since we want our results to be independent from the applied radio model. Thus, the radio model is abstracted in the edge weights of the graph G.

3.1 Foreign Coding

In this section we first introduce a model for the data correlation in a network. Based on this model an algorithm is presented that solves the minimum-energy data gathering problem. We then propose a distributed version of the algorithm which works in a slightly more restrictive model.

3.1.1 Correlation Model

Since sensor networks are often used to sense real world phenomena, each sensor node continuously produces information as it monitors its vicinity. Thus, we assume that each node $v_i \in V$ generates one data packet p_i of size s_i bits per round that describes the measured information sample. Note that data packets from different nodes need not have equal size.

Distributed sensor data is often correlated and it is therefore often possible to perform in-network aggregation. Data aggregation can potentially take place at any intermediate node as a data packet is routed towards the sink node. However, once a packet is encoded at a node it is not possible to alter the packet again; hence, recoding is not possible. In other words a node v_i can use its data to encode packets containing correlated data that are routed through v_i on their paths to the sink node t. A packet from node v_i that is encoded at a node v_j is denoted by p_i^j; its corresponding size is s_i^j.

The compression rate depends on the data correlation between the involved nodes v_i and v_j, denoted by the correlation coefficient $\rho_{ij} = 1 - s_i^j / s_i$. Encoding at a node v_i only depends on data collected by v_i and not on other data also routed through v_i—recording is not possible. However, it must be guaranteed that encoding does not result in cyclic dependencies that cannot be resolved while decoding at the sink t. Such an encoding strategy does not depend on timing assumptions in the encoding nodes, and therefore it is also applicable to asynchronous networks.

The *minimum-energy data gathering* problem for a given graph $G = (V, E)$ is then defined as follows. Find a routing scheme and a coding scheme to deliver data packets from all nodes in V to a sink node $t \in V$, such that the overall energy consumption is minimal. Let $f(e)$ with $e \in E$ denote the number of bits transmitted over edge e. Thus, we aim at minimizing the following cost function:

$$C(G) = \sum_{e \in E} w(e) f(e). \tag{3.1}$$

3.1.2 Algorithm

In the following we present the *minimum-energy gathering algorithm* (MEGA). The resulting topology of the algorithm is a superposition of two tree constructions. A directed minimum spanning tree of a directed graph whose edges are specified later in this section is used to determine the encoding nodes for all data packets. Once a packet is encoded it is routed on a shortest path towards the sink to save energy. Given a graph $G = (V, E)$ and a sink node $t \in V$, MEGA computes a *shortest path tree* (SPT) of G rooted at t (e.g. using Dijkstra's algorithm [30]). Since the weight of an edge in E corresponds to the energy spent to transmit one bit of information from one incident node to the other, the SPT comprises an energy-minimal path from each node in V to the sink.

In a second step, the algorithm computes for each node v_i a corresponding node v_j that encodes the packet p_i using its own data. Since cyclic dependencies in the encoding must not occur to guarantee decoding this results in a so-called *coding tree*. In order compute this coding tree, we make use of an algorithm solving the *directed minimum spanning tree* (DMST) problem (also known as the *minimum weight arborescence* problem). Consider a directed graph $G = (V, E)$ and a weight $w(e)$ associated with each edge e. The problem is to find a rooted directed spanning tree such that the sum of $w(e)$ over all edges e in the tree is minimized provided that all nodes are reachable from the root. Chu and Liu[23], Edmonds [34], and Bock [13] have independently proposed efficient algorithms for finding the MST given a directed graph. Tarjan [139] shows an efficient implementation for the problem (see also [19]). Edmonds algorithm is also described in [87]. Furthermore, a distributed algorithm is given in [61].

```
MEGA
Input: Graph $G = (V, E)$ and sink $s \in V$
$T_{\mathrm{SP}}$ = shortest path tree in $G$ rooted at $t$
$\widetilde{G} = (V, \widetilde{E})$ = complete directed graph
for all $(v_i, v_j) \in \widetilde{E}$ do
    $w'(v_i, v_j) = s_i\,(w(v_i, v_j) + w(v_j, t)(1 - \rho_{ij}))$
end for
$T'$ = DMST on $\widetilde{G}^T$
$T = (V, E_T) = T'^T$
for all $v_i \in V$ do
    consider $v_j$ such that $(v_i, v_j) \in E_T$
    set $v_j$ as encoding relay for $p_i$
end for
Output: Minimum-energy data gathering topology for $G$
```

In the following we propose the directed graph on which MEGA computes the DMST with one of the above-mentioned algorithms. First, MEGA builds a complete directed graph $\widetilde{G} = (V, \widetilde{E})$. The weight $\widetilde{w}(e)$ for a directed edge $e = (v_i, v_j)$ in $\widetilde{E}$ is defined as

$$\widetilde{w}(e) = s_i\,(\sigma(v_i, v_j) + \sigma(v_j, t)(1 - \rho_{ij}))\,, \tag{3.2}$$

whereas $\sigma(v_i, v_j)$ denotes the weight of a shortest path from v_i to v_j in G, that is, the sum of the edge weights on that path. The weight of an edge in $\widetilde{G}$ therefore stands for the total energy consumption required to route a data packet p_i to the sink using node v_j as an encoding relay. This also depends on the correlation coefficient of the involved nodes. The DMST is by definition a minimum-weight directed tree with edges directed off the root node (e.g. the sink t) but we aim at a directed tree with edges pointing towards t. Therefore, MEGA does not apply a DMST algorithm to $\widetilde{G}$ but to the transposed graph[2] $\widetilde{G}^T$. Then, the edges in $\widetilde{G}$ corresponding to the ones in the DMST of $\widetilde{G}^T$ form a tree that defines the encoding relays for a nodes in V. The resulting topology of MEGA comprises for each node v_i all edges on a shortest path from v_i to its encoding relay v_j found by the above described DMST construction and all edges on the path from v_j to t on the SPT. If the data is pairwise independent for all nodes, the resulting topology of MEGA is the SPT—this is the minimum-energy data gathering topology for uncorrelated data.

On the left hand side of Figure 3.2 an example graph G is depicted. The information from all nodes in G should be gathered at the sink node t. To

---

[2] The transpose of a directed graph $G = (V, E)$ is $G^T = (V, E')$ with $(v_i, v_j) \in E'$ if and only if $(v_j, v_i) \in E$. That is, the direction of each edge in G is reversed in the transposed graph.

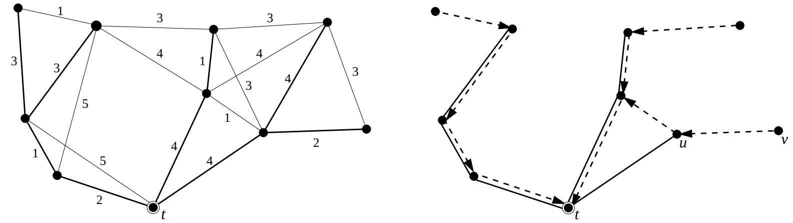

Figure 3.2: The left figure depicts an example graph G. All nodes try to send their data to the sink node t. Each edge e in the given graph G is labeled according to the energy required to send one bit of data over e. The SPT rooted at t is indicated by bold edges. On the right hand side the resulting minimum-energy data gathering topology obtained by MEGA is shown. The coding tree (dashed arrows) determines for each node its corresponding encoding node. Encoded data is sent on the SPT (solid lines) towards the sink t.

simplify matters it is assumed that all nodes send a packet of equal size s_r. For each edge e in G the edge weight $w(e)$ corresponding to the energy consumption of sending one bit of data over e is depicted in Figure 3.2. Data correlation among nodes v_i and v_j is defined to be inverse proportional to their Euclidean distance. That is, $\rho_{ij} = 1/(1+d(v_i, v_j))$. Furthermore, a path loss exponent of 2 is assumed and thus $d(v_i, v_j) = \sqrt{w(e)}$, with $e = (v_i, v_j)$. Algorithm MEGA first computes a SPT rooted at t—bold lines on the left of Figure 3.2. The coding tree established by MEGA is depicted with dashed arrows on the right hand side of Figure 3.2. The encoding relay of each node is thereby determined by its outgoing arrow. The algorithm routes packets along the SPT towards the sink once they are encoded. Thus, data packets at a node are not always sent to the same neighboring node. For example, at node u the packet received from node v is first encoded and then forwarded directly to the sink t, whereas u's packet is encoded at an intermediate node to circumvent the costly edge (u, t).

The running time of the algorithm is $\mathrm{O}\big(n^3\big)$ since solving the all-pair shortest path problem on G takes $\mathrm{O}\big(n^3\big)$ time and the running time for the computation of the DMST on $\widetilde{G}^T$ is $\mathrm{O}\big(n^2\big)$.

3.1.3 Analysis

To show that MEGA is optimal, we first establish some properties of an optimal data gathering topology. In an optimal solution for a graph G, each packet p_i is routed along a distinct path from node v_i to the sink t since multiple paths would unnecessary increase the total energy consumption. The packet p_i is encoded at no more than one node v_j on its path towards the sink. Note that it is also possible that p_i is sent to t without any encoding. In this case t is considered to be the encoding relay for node v_i. Along the lines of MEGA we can therefore establish a directed graph $\widetilde{G}_{opt} = (V, \widetilde{E}_{opt})$ for the optimal solution where each node v_i in V apart from t has exactly one outgoing edge in $\widetilde{E}_{opt}$ pointing towards the encoding relay of the packet p_i. It follows that $|\widetilde{E}_{opt}| = n - 1$. To guarantee the successful decoding of all data at the sink node it is required that the encoding does not lead to cyclic dependencies. By the construction rules of the directed graph $\widetilde{G}_{opt}$, such cyclic dependencies would be reflected in cycles in $\widetilde{G}_{opt}$ and therefore $\widetilde{G}_{opt}$ must be cycle-free. Consequently, the above elaborated properties of the directed graph $\widetilde{G}_{opt}$, namely

- $|\widetilde{E}_{opt}| = n - 1$,
- every node v_i, $v_i \in V \setminus \{t\}$, has an edge leading out from it,
- and $\widetilde{G}_{opt}$ does not contain cycles,

characterize a directed tree pointing into node t. Thus, we derive the following theorem:

Theorem 3.1. *Given a graph $G = (V, E)$ and a sink $t \in V$, algorithm MEGA computes a minimum-energy data gathering topology for G.*

Proof. The path for a packet p_i in the optimal solution can be divided into two parts. First, p_i is routed on a path from v_i to its encoding node v_j and in a second step the encoded packet p_i^j is routed from v_j to the sink t. In an optimal topology both sub-paths are minimum-energy paths—and thus shortest paths in G—in order to minimize the overall cost function $C(G)$ defined in Equation (3.1). In Equation (3.1) the total energy consumption is computed by summing up the load of each edge in E times its corresponding weight. Another way to compute the total energy consumption is to charge each node v_i the energy the packet p_i spends during $p_i's$ way to the sink. For each node v_i and its corresponding encoding relay v_j the costs are summing up to $\widetilde{w}(v_i, v_j)$ as defined in Equation (3.2). Consequently, the total energy consumption is defined to be the sum of all edge weights in $\widetilde{G}_{opt}$. MEGA computes exactly these costs for all possible encoding relays of a node v_i and assigns them to the corresponding edges in $\widetilde{G}$. Using a DMST algorithm on the transposed graph, a directed tree pointing into t is obtained that minimizes the sum of all edge weights. Since the optimal solution G_{opt} also corresponds to a directed spanning tree in $\widetilde{G}_{opt}$, MEGA minimizes $C(G)$. $\square$

3.1.4 Distributed Computation

So far, the proposed centralized algorithm MEGA requires total knowledge of the correlation among all nodes and the topology of the network. In this

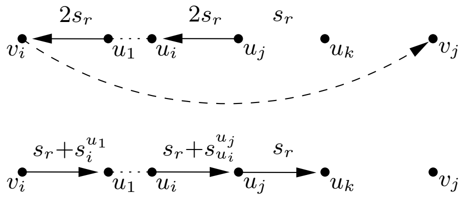

Figure 3.3: Configuration of the graph $\widetilde{G}$ if the encoding relay v_j of node v_i is more than one hop away from v_i (top) and a configuration using only one-hop relays that results in less energy consumption (bottom).

section we consider the well studied *Unit Disk Graph* (UDG) model [25] where all nodes have the same limited transmission radii. Additionally, we restrict the raw-data packets of all nodes to have equal size s_r. Data correlation in sensor networks is often assumed to be regional. In the following, the data correlation between two nodes v_i and v_j is therefore modeled to be inverse proportional to their Euclidean distance $d(v_i, v_j)$.

Using the distributed algorithm described in [61] to compute the DMST, MEGA can be implemented in a distributed way. In a setup phase, the sink t starts building a SPT rooted at t using e.g. Dijkstra's algorithm. Thus, each node in V is able to determine the energy consumption to send t one bit of information. Then, each node v_i in the graph G broadcasts a sample packet p_i. Upon receipt of a packet p_j from a neighboring node v_j, v_i encodes p_i using p_j to compute the correlation coefficient between the two neighbors. Additionally, v_i can determine the energy cost of a transmission to v_j by using a Received Signal Strength Indicator (RSSI) [125]. Node v_i establishes an directed edge (v_i, v_j) whose weight is set according to Equation (3.2). The graph $\widetilde{G}$ then only consists of edges between direct neighbors in G. To guarantee that MEGA still results in an optimal topology we have to show that in an optimal solution each node and its corresponding encoding relay are only one hop away from each other. That is, they are neighbors in the graph G.

Assume for the sake of contradiction that the encoding relay v_j for a node v_i in a minimum-energy data gathering topology G_{opt} is more than one hop away from v_i. Then the packet p_i is routed along a path $p(v_i, v_j) = (v_i, u_1 \ldots u_i, u_j, u_k \ldots v_j)$ to its encoding relay v_j. Since v_i and v_j are not adjacent in G and G is a UDG, it follows that $d(v_i, v_j) > d(v_i, u_1)$ and consequently $\rho_{v_i,v_j} < \rho_{v_i,u_1}$—data correlation is inverse proportional to the Euclidean distance. It follows that $s_i^j > s_i^{u_1}$. Thus if (u_1, v_i) is not in $\widetilde{G}_{opt}$, that is v_i is not the encoding relay for u_1, we choose u_1 to be the encoding node of v_i and obtain a topology with less energy dissipation which contradicts the assumption. If (u_1, v_i) is in $\widetilde{G}_{opt}$ we are in a configuration as it is depicted in Figure 3.3 at the top. Node v_i has an edge to v_j in $\widetilde{G}_{opt}$ (dashed arrow) and each node on $p(v_i, v_j)$ up to u_j has an edge to its predecessor on the path. Since $\widetilde{G}_{opt}$ is cycle-free, there is at least one node on the path (u_k in Figure 3.3) that does not point to its predecessor. In Figure 3.3 all edges on the path are labeled according to their load caused by all raw data packets from nodes on the path. However, by changing the edges in $\widetilde{G}_{opt}$ subject to the configuration at the bottom of Figure 3.3—and thus also the encoding relays— and due to the assumption that all raw-data packets have equal size s_r edge (u_i, u_j) has a load of at most $s_r + s_{u_i}^{u_j}$ since the packet p_{u_i} is sent to u_j and the encoded packet $p_{u_i}^{u_j}$ possibly back to u_i on its way to the sink. The same holds for all other edges on the path for which the corresponding edge in $\widetilde{G}_{opt}$ is reversed. Since $s_{u_i}^{u_j} < s_r$ for all direct neighbors in G, it follows that we can decrease the energy consumption of G_{opt} by applying the transformation shown in Figure 3.3 which leads to a contradiction. It is therefore adequate to restrict the directed graph $\widetilde{G}$ to comprise only edges connecting neighboring nodes in G in order to obtain an optimal topology. This consequently allows for the distributed computation of the minimum-energy data gathering topology of G.

3.2 Self-Coding

In this section we first determine a lower bound for the energy consumption of an optimal data gathering topology using self-coding. Then, an algorithm is presented that approximates an optimal topology up to a constant factor.

3.2.1 Correlation Model

In this section we consider the problem of constructing efficient data gathering trees in the model of explicit communication introduced in [27]. In this model, nodes are only able to encode their own raw data in the presence of other raw data routed through them.[3] Thus, the reduction in data size at a node v_i is due to the direct availability of side information locally at v_i. If no side information is available at node v_i, the packet size of p_i is s_r bits. However, if raw data is routed on their way to the sink t through v_i, the node can encode its data such that the size of p_i reduces to s_e bits with $s_e < s_r$. That is, different from the correlation model in Section 3.1 the correlation between data is uniform and therefore $\rho_{ij} = 1 - s_e/s_r$ for all $v_i, v_j \in V$ with $i \neq j$. Consequently, if a node encodes its data using some other data, the encoded data will have exactly s_e bits. It is shown in [27] that

---

[3] In contrast to the model introduced in Section 3.1 where the inverse restriction is assumed.

for this restricted correlation model the problem of finding minimum-energy data gathering trees is NP-complete, by applying a reduction to set cover. Moreover, [27] proposes a heuristic based on a combination of a shortest path tree augmented by travelling salesperson paths.

3.2.2 Lower Bound

Given a graph $G = (V, E)$ and a sink node $t \in V$, we present an approximation algorithm that guarantees a data gathering tree for which the cost $C(G)$ defined in Equation (3.1) is only a constant factor higher than the cost of an optimal topology. We first give a lower bound on the cost, that is, the energy consumption of the optimal topology.

Lemma 3.2 (Lower Bound). *The cost of an optimal topology* c_{opt} *is bounded from below by* $c_{opt} \geq max(s_e \cdot c_{SSP}, s_r \cdot c_{MST})$*, where* c_{SSP} *is the sum of the costs of all the shortest paths to the sink* t*, and* c_{MST} *is the cost of the minimum spanning tree of all nodes in* V*.*

Proof. Nodes in the graph can either send their raw data directly to the sink, or use the raw data of other nodes to encode their data, and then send their data to the sink. Let B be the set of nodes sending their raw data to the sink t without encoding. Let the nodes who encode their data using the raw data of node $u \in B$ be the set S_u. The set B and the sets S_u for all $u \in B$ form a partition over all nodes in V, that is: $V = B \cup \sum_{u \in B} S_u$.

After deciding how V will be partitioned, the optimal algorithm will use the shortest paths (SP) to deliver the encoded data of all nodes in $V \setminus B$ to the sink since this minimizes the total energy consumption. Therefore, nodes in S_u need to encode their data using the raw data of node u, u being a node in set B. On the other hand, the sink t needs to decode the encoded data of nodes in S_u; to do so, t also requires the raw data of u. The optimal topology to distribute the raw data of u is given by the Steiner tree (ST) where the nodes in S_u, node u, and t are terminal nodes. Summing up, the cost of the optimal topology is therefore

$$c_{opt} = \sum_{u \in B} \left(s_r \cdot \mathrm{ST}(S_u, u, t) + \sum_{v \in S_u} s_e \cdot \mathrm{SP}(v, t) \right).$$

We can lower-bound this equation in two ways. By definition $\mathrm{SP}(v_i, v_j) = \mathrm{ST}(v_i, v_j)$ and any additional terminal node in the Steiner tree increases the cost of the tree. Furthermore, since $s_e < s_r$ it follows that

$$\begin{aligned}
c_{opt} &= \sum_{u \in B} \left(s_r \cdot \mathrm{ST}(S_u, u, t) + \sum_{v \in S_u} s_e \cdot \mathrm{SP}(v, t) \right) \\
&\geq \sum_{u \in B} s_e \cdot \mathrm{SP}(u, t) + \sum_{u \in B} \sum_{v \in S_u} s_e \cdot \mathrm{SP}(v, t) \\
&= \sum_{u \in V} s_e \cdot \mathrm{SP}(u, t) = s_e \cdot c_{SSP}.
\end{aligned}$$

On the other hand, all nodes in B send and all nodes in $V \setminus B$ receive at least one packet containing raw data. Thus, raw data is distributed at least on a spanning tree. Since the minimum spanning tree (MST) is the cheapest possible spanning tree, the cost of the optimal algorithm is also bounded from below by the cost of the MST, used to transmit raw data. The lemma follows immediately. ☐

3.2.3 Algorithm and Analysis

In the following we present the *low energy gathering algorithm* (LEGA), an approximation algorithm that is optimal up to a constant factor. The algorithm is based on the *shallow light tree* (SLT), a spanning tree that approximates both the MST and the SPT for a given node (e.g. the sink). The SLT was introduced in [5, 10]. Given a graph $G = (V, E)$ and a positive number γ, the SLT of G has two properties:[4]

- Its total cost is at most $1 + \sqrt{2}\gamma$ times the cost of the MST of the graph G;

- The distance on the SLT between any node in V and the sink is at most $1 + \sqrt{2}/\gamma$ times the shortest path from that node to the sink.

The algorithm contains the follows steps: First, the SLT spanning tree is computed, the sink node t being the root of the SLT. Then, t broadcasts its packet p_t to all its one-hop neighbors in the SLT. When node v_i receives a packet p_j consisting of raw data of a neighboring node v_j, v_i encodes its locally measured data p_i using p_j, and transmits the packet p_i^j to the sink t on the path given by the SLT. Then node v_i broadcasts its packet p_i to all its one-hop neighbors but v_j; in other words, each node receives the raw data from its parent in the SLT. The sink t has its own data p_t available locally (or it can use data of one of its first-hop neighbors), and thus can perform recursive decoding of the gathered data, based on the encoded data that it receives from all other nodes in V.

---

[4] For more details on the construction of the shallow light tree (SLT) we refer to [74].

Theorem 3.3. *LEGA achieves a* $2(1+\sqrt{2})$*-approximation of the optimal data gathering topology.*

Proof. The total cost of LEGA is given by

$$c_{LEGA} = s_r \cdot c_{SLT} + \sum_{v_i \in V} s_e \cdot |path_{SLT}(v_i, t)|.$$

The first term follows from the fact that each node sends its raw data to all its children in the SLT. The second term corresponds to the sum of the shortest paths from all nodes in V to the sink node t in the SLT. Using the SLT properties and setting $\gamma = 1$ we obtain

$$\begin{aligned} c_{LEGA} &= s_r \cdot (1+\sqrt{2})c_{MST} + s_e \cdot (1+\sqrt{2})c_{SSP} \\ &\leq 2(1+\sqrt{2})c_{opt}. \end{aligned}$$

The second equation follows directly from Lemma 3.2. ☐

Since [74] provides an algorithm constructing the SLT of a graph G that runs in $\mathrm{O}(m + n\log n)$ time, the running time of LEGA is also $\mathrm{O}(m + n\log n)$.

3.3 Related Work

During the design of data gathering protocols for sensor networks, researcher have identified the importance of data aggregation to improve energy efficiency [63, 79]. In [63] Directed Diffusion is proposed, a protocol in which sensors create gradients of information in their respective neighborhoods. The sink node requests data by broadcasting interests. If interests fit gradients, paths of information flow are formed and data is aggregated on the way to reduce communication costs. The key idea in [51] is to reduce the number of nodes communicating directly with the sink by forming randomized clusters. Each cluster-head encodes data arriving from nodes in its cluster, and sends an aggregated packet to the sink. However, the main drawback of the protocol in [51] is the requirement that all nodes must be able to directly communicate with the sink.

In [45] the problem of data gathering is addressed by using concave, non-decreasing cost functions to model the aggregation function applied at intermediate nodes. The authors propose a hierarchical matching algorithm resulting in a aggregation tree that simultaneously approximates all such cost functions up to a logarithmic factor. However, in their model only the number of nodes providing data to an aggregating node decides on its aggregation performance regardless of the correlation among the available data. That is, the impact of data correlation is not explicitly considered. This too simplistic assumption that an intermediate node can aggregate multiple incoming packets into a single outgoing packet irrespective of their correlation is also assumed in other work [70, 94, 95]. In contrast, a distance based correlation model is employed in [116]. The authors propose a static clustering scheme leading to energy efficient data gathering topologies for a wide range of spacial correlations.

Based on signal processing techniques, [22] and [27] tackle the problem of minimum-energy data gathering by applying Slepian-Wolf coding. In their model the correlation among nodes is known *a priori* and is used to optimize the rate allocation of a distributed compression algorithm which obviates the need to exchange data between sensor nodes to learn their correlation structure.

The work which is most closely related to the problem we consider in this chapter is the one involving the concept of self-coding [27]. The authors prove that already for a restricted model, where nodes are only allowed to encode their own data in the presence of side information, the problem of finding minimum-energy data gathering trees is NP-complete, by applying a reduction to set cover. Moreover, in [27] a heuristic based on a combination of a shortest path tree augmented by travelling salesperson paths is proposed. We continue their work by establishing a strict classification of coding strategies. Furthermore, we provide an approximation algorithm in case of self-coding and an optimal one for the foreign coding strategy.

Chapter 4

Ultra-Low Power Data Gathering

In Chapter 2 and 3, we attempted to lower the power consumption of a sensor network by means of confining interference or avoiding the transmission of redundant information. In this chapter we investigate another strategy to conserve energy which is orthogonal to the aforementioned approaches. It is based on the observation that the energy wastage of the radio, even in idle listening, is three orders of magnitude higher than a node's power drain in sleep mode. As a consequence, we try to turn on the radio only if a data transfer is pending. This requirement is hard to fulfill since multi-hop routing techniques must be applied to be able to transmit data from all nodes in a possibly large spatial area to the data sink. Energy-efficient data exchange is a nontrivial task in single-hop networks but becomes even harder if routing over multiple hops is required. Sensor nodes are no longer able to schedule their transmissions strictly according to their own demands since they also need to activate their radios to relay data messages from other nodes towards the sink.

We focus on data gathering applications producing continuous, low-rate data traffic throughout the rest of this chapter. Examples thereof include precision agriculture [16], glacier displacement measurements [103], natural habitat monitoring [101], or microclimatic observations [140]. All of these applications generate periodic data samples at low rates resulting in light traffic load and thus low bandwidth requirements. We propose Dozer, an ultra-low power network stack tailored for this category of applications. It incorporates a MAC layer, topology control, and a routing protocol. We refrained from integrating existing low-power solutions for any of these subsystems since it is our strong belief that only a perfectly orchestrated network stack is able to achieve minimum power consumption and therefore maximize network lifetime. The primary goal of Dozer is to reduce idle listening and overhearing. In theory, a TDMA-based MAC protocol constructing a global schedule to determine exact send and receive times for each node would solve the problem of overhearing and idle listening. In a real-world setting clock drifts and a frequently changing external conditions render plain TDMA costly since maintaining an accurate schedule is a highly complex and energy consuming task. Dozer takes these facts into account. It builds a data gathering tree on top of the underlying network topology and provides nodes with precise wake-up schedules for all communication, relying only on local synchronization. Furthermore, it addresses the problem of temporary network partition and energy efficient tree adaptation in case of local link failures. Despite these additional considerations Dozer attains low radio duty cycles in both single-hop and multi-hop networks and thus achieves high energy efficiency. The protocol was implemented using TinyOS. Its performance was evaluated using an indoor deployment consisting of 40 TinyNode [47] sensor nodes. Using a sampling period of two minutes Dozer achieved an average duty cycle of less than 0.2% on all nodes. Given two off-the-shelf alkaline batteries with a capacity of 2000 mAh each and ignoring power consumption of application specific sensor equipment, as well as battery self discharge, Dozer is able to operate the network for a lifetime of approximately 5 years. As a consequence, system lifetime is determined by the self-discharge rate of modern batteries.

4.1 System Overview

The Dozer system is intended to meet common demands of environmental monitoring applications. It enables reliable data transfer, has self-stabilizing properties—and is thus robust to changes in the environment—, and it is optimized for long system lifetime. Network latency and flexibility towards dynamic bandwidth demands are considered to be of less importance.

In order to forward data to the base station Dozer establishes a tree structure on top of the physical network. This guarantees that information from any node is conveyed on a loop-free path to the data sink which constitutes the root of the tree. Each node fills two independent roles in tree maintenance. On the one hand, it acts as a parent for directly connected nodes one level deeper down the tree. On the other hand, it is a child of exactly one node one level higher in the tree. Data is transferred to the sink using a TDMA protocol. However Dozer does not construct one global schedule for the whole network but splits it up at each node. Consequently, each node has two independent schedules; one in its role as a parent and one for its child role. As a parent a node decides when each of its children is allowed to upload data. Vice versa, in its role as a child it receives an update slot from its own parent. Thus, Dozer only constructs single-hop schedules and does not rely on any global synchronization. Each round of a parent's TDMA schedule is initiated by the transmission of a beacon message. Simplified, beacons are the heart beat of the Dozer system. In its child role, a node

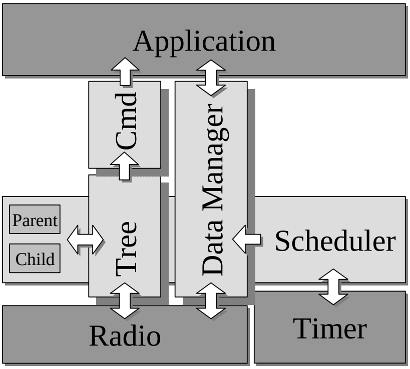

Figure 4.1: Architecture of the Dozer system represented by the light gray boxes. Arrows indicate the command flow between the different modules.

synchronizes on the received parent beacon. However, it does not adjust its internal clock but calculates the outset of its upload slot in relation to the last beacon reception time.

Dozer does not make use of a traditional MAC protocol. In fact, the system does not try to prevent nodes from sending at the same time. Collisions are explicitly accepted. However, using randomization Dozer ensures that two schedules drift apart quickly in case of a collision. This scheme is advantageous as a message receiver always exactly knows when the corresponding sender is going to start its transmission. This greatly prolongs network lifetime since nodes are able to maximize their time in energy-efficient sleep mode. Facing collisions data transmissions in Dozer are always explicitly acknowledged.

As network conditions change over time so does the network topology. Consequently, the data gathering tree cannot be stable in the long run. To reduce increased message delay in case of link failures, each node maintains a list of additional potential parents. Choosing a candidate from this list a new connection can be established with little overhead.

4.2 Dozer Implementation

The high-level overview in Section 4.1 outlines that Dozer handles several interwoven tasks in parallel. More precisely, the system can be subdivided into four logical components. Figure 4.1 depicts the individual components and shows how they interact with each other. In the following the function of each component is discussed in detail.

4.2.1 Tree Maintenance

The *Tree Maintenance* module coordinates a node's integration in the data gathering tree of Dozer and guarantees constant connectivity. Furthermore in case of network failures it sets the node in a energy efficient suspend mode.

Connection Setup

It is essential for every node to be part of Dozer's data gathering tree. Nodes without connectivity are unable to provide data to the base station and are thus of no use. Upon waking up, a node tries to join the tree as quickly as possible in the so-called bootstrap phase. Since it does not yet have any conception of its neighborhood, it starts listening for beacon messages of nearby nodes. Beacon messages are periodically sent by already connected nodes at the beginning of their TDMA schedule to enable the integration of disconnected nodes. After scanning for the full length of a TDMA round each received beacon message is analyzed and the corresponding node is ranked according to a rating function. The function's current implementation considers a node's distance to the sink as well as its load—the number of direct children—in this computation. Both of these values are part of the beacon message and are thus readily available. To minimize tree depth, distance has a higher weight than load in the computation. The node now tries to connect to the highest rated neighbor and the gathered information about all other overheard potential parents is stored.[1]

The actual connection setup is initiated after the transmission of the next beacon of the selected neighbor (see Figure 4.2). After sending its beacon the potential parent stays in receive mode for a short amount of time. Within this contention window it accepts incoming connection requests. The child uses a simple back-off mechanism to determine when to send its connection request message. This contention phase is needed since multiple nodes may want to establish a connection with this parent at the same time. On receiving a connection request message the parent assigns the new child a slot in its TDMA schedule and returns this informations by means of a handshake message. In the current implementation a node only accepts one new child per beacon interval. This restriction serves as a simple form of load balancing. A node failing to connect to a specific neighbor may first try to join the tree at another node with similar rating before retrying on the same parent.

Since listening for the whole length of the contention window after each beacon transmission is expensive, in Dozer an activation mechanism precedes the actual connection setup. As depicted in Figure 4.2 the child transmits an *activation frame* immediately after receiving the potential parent's beacon

---

[1] Other possible parent selection strategies may also rely on randomization or link quality measurements. Some of these mechanisms were first suggested in [40] and later analyzed in [164].

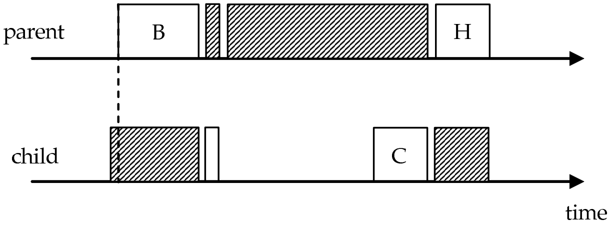

Figure 4.2: Connection setup – The parent node sends a beacon (B). Upon beacon reception the child sends a busy tone to activate the contention window. The child then transmits its connection request (C). A handshake (H) serves as an acknowledgment. Shaded areas denote the times a node is actually listening.

message. On the other hand, the parent switches to receive mode and polls the channel right after sending its beacon. Only if the received radio signal strength (RSSI) indicates channel activity the contention phase is activated. If multiple nodes want to connect to the parent in the same round their activation frames collide. However, this imposes no problem since the parent does not try to detect a specific pattern and the sensed RSSI still clearly indicates activity on the medium.

Connection Recovery

Wireless links are fragile to changes in the environment and must be expected to break at any time. Unstable weather conditions or temporary obstacles in the area of interest can have a negative impact on the network stability. Dozer incorporates functionality to confront this problem.

A connection to the current parent breaks if multiple consecutive data transfers fail: The parent is declared unreachable. To replace it with little overhead, the orphaned node queries its stored list of potential parent for a well suited substitute and tries to establish a new connection. In case of success this procedure costs a reasonably small amount of energy. However, if no replacement can be found in this list the node falls back to bootstrap mode (see Section 4.2.1) and has to conduct a costly scan to detect new potential parents. To guarantee the availability of reasonably up-to-date information about its stored potential parents, a node periodically listens for their beacons. This refresh is cheap since future beacon transmission times of a node can be predetermined accurately based on the point in time of its last overheard beacon. This calculation is performed by the Scheduler component described in Section 4.2.2. Additionally, to learn about the existence of new, potentially well suited parents, a random listen mechanism is applied. Unrelated to their two schedules nodes periodically overhear the channel for beacons of yet unknown nodes. To keep the incurred overhead low, these scans must only be executed infrequently.

Suspend Mode

If a node is not connected to a parent and also cannot hear any beacons—even when listening for a full beacon interval—it assumes the network to be down or out of reach. Constant channel surveillance in this situation would result in high power consumption; a node's lifetime would decrease to a couple of days. To circumvent such energy wastage Dozer features a special suspend mode. Along the line of low power listening [119] the node periodically samples the channel for activity and remains in sleep mode for the remainder of the time. However, unlike low power listening this mode does not ensure reception of all messages. Energy efficiency and quick reactivation in case of channel usage can be balanced depending on the demands of the application running on top of Dozer. Frequent channel polling results in higher power drain but rapid connection establishment on network availability. On the other hand, longer intervals between scans lead to improved energy efficiency but possibly delayed reintegration of suspended nodes.

4.2.2 Scheduler

The energy efficiency of the Dozer system mostly stems from the *Scheduler* module. By providing the Tree Maintenance and Data Manager modules with precise timings it allows efficient radio usage.

Communication between a parent and its children is coordinated by a TDMA protocol. That is, all transmissions happen at exactly predetermined moments in time. For the exchange of a message neither sender nor receiver have to spend energy beyond what is required to transmit or receive the actual data. In particular nodes do not have to waste energy on overhearing the channel for pending transmissions. A global TDMA scheme however is expensive since it demands the existence of a network-wide time synchronization mechanism. To circumvent this burden Dozer only aligns one hop neighbors in the data gathering tree. As all nodes are simultaneously parent and child they all have to maintain two schedules; one provided by their parent and one self-determined as a reference for their children. In this setting it is complex to synchronize the internal clocks of a parent and its children. Only by means of global time synchronization it would be possible for each node to service both schedules with only one clock.

While in theory wake-up times can be calculated perfectly at both, parent and children, clock drift has to be considered in real-world applications. The current generation of sensor nodes is usually equipped with an electronic oscillator exhibiting a skew of 30-50 parts per million (ppm) at room temperature. Thermal differences between sender and receiver lead to significant, additional skew.

The self-determined TDMA schedule of a node, in the following also denoted as parent schedule, is of fixed length and divided into equal time slots. Upon connection of a new child the Tree Maintenance module requests a free slot from the Scheduler. This slot is henceforth marked as occupied and reserved for the new child. The assignment outlasts the end of the schedule and is only released if the corresponding child disconnects. That is, each child owns the same time slot in every iteration of the schedule. As a consequence, the total number of slots of the TDMA schedule confines the maximum number of children a node is able to manage.

After connection establishment between parent and child, the personal slot number of the child in its parent's schedule is known at both nodes. They can thus compute the start of this slot relative to the beginning of the schedule. For each slot of its schedule the parent checks if it is occupied and listens for incoming data if necessary. Analogously, at the child the Scheduler triggers the Data Manager component at the start of its upload slot to permit a timely transfer of potentially queued data messages.

As mentioned above the protocol does not provide for any direct clock synchronization. Instead, at the outset of a new round of the schedule the Tree Maintenance module is triggered to send a beacon message. This beacon is received by all children and timestamped according to their local clocks. Since both parent and children share the knowledge about the time of the beacon transmission this moment in time serves as an anchor point for implicit clock synchronization. No adjustments of system clocks are required but only this timestamp needs to be stored for further timing calculations. For the remainder of this round of the TDMA schedule all events are computed in relation to this timestamp. The transmission time of the next beacon is also determined according to this value. As a positive side effect, clock drift accumulation over multiple rounds of the schedule is prevented. Furthermore, the complexity for handling both independent schedules diminishes since only two values, used as offsets for the internal clock need to be stored.

Without a global schedule, collisions between the transmissions of neighboring nodes that are not part of the same schedule can no longer be excluded. Other systems facing the same problem (e.g. [53]) apply secondary MAC protocols such as CSMA/CA to resolve it. However, since bandwidth demands in the considered scenario are low, collisions happen infrequently. Dozer thus refrains from handling them actively. In the long run the costs for retransmissions are cheaper than the costs it would take to prevent them. But regarding collisions there exist an additional problem which needs to be tackled. Collisions may indicate the undesired alignment of two independent schedules. If this is the case collisions will recur in subsequent rounds of the schedule, without intervention. To counter this threat, Dozer extends the length of a TDMA round by a randomly chosen time span—also referred to as jitter. The parent draws a new random number for each round of the schedule which is then added to the round's length. It can be shown that a linear relation between the maximum transmission time per slot and a reasonable upper bound for this random offset exists. Dozer uses a bound of seven times the time needed to flush the local message buffer (c.f. in Section 4.2.3). With this value, in case of a collision between two unsynchronized transmissions, the chance for a second consecutive collision is less than 50% in expectation. For any realistic scenario this implies that a maximum jitter of less than one second suffices.

This random prolongation of the TDMA rounds introduces the problem of how to predict the exact time of the next parent beacon. Thus, the seed value of the random number generator used for calculating the next random offset is included in each beacon. With this value each child is able to execute the same computation as the parent and to predict when the next beacon message is due. At the parent, the current random number is used as seed value to generate the next random number. Consequently, even if a child misses one or more consecutive beacon messages of its parent it is still able to determine the next beacon arrival time. It therefore recursively draws random numbers until it has compensated for the number of missed beacons.

4.2.3 Data Administration

At the end of the day, Dozer's main task is to transport sensor readings from all nodes to the data sink. While a node's data upload times are strictly defined by the Scheduler module data injection by the application is always possible. Hence, the *Data Manager* module features a message queue buffering injected data pending for transmission. Since data upload to the parent and data reception from the children is unsynchronized, incoming messages from the children are also appended to this queue.

As soon as the Scheduler module signals the beginning of the parent upload slot the Data Manager tries to transmit all queued messages. Each message is explicitly acknowledged and only removed from the queue of the sender if the receiver confirms its correct reception (see Figure 4.3). With the acknowledgment the parent not only takes over responsibility for the packet but also notifies the child about how many more messages it is willing to accept. As a consequence, at most one unnecessary message transmission is possible if the parent is unable or unwilling to handle more messages. The link acknowledgments guarantee that no messages are lost on their path towards the sink despite possible collisions on the wireless links. If a message transfer fails to be acknowledged the child immediately stops its data upload for this round of the schedule since a temporal interruption on the medium may be encountered. In case of consecutive transmission failures over multiple upload slots the Data Manager instructs the Tree Maintenance module to switch to another parent.

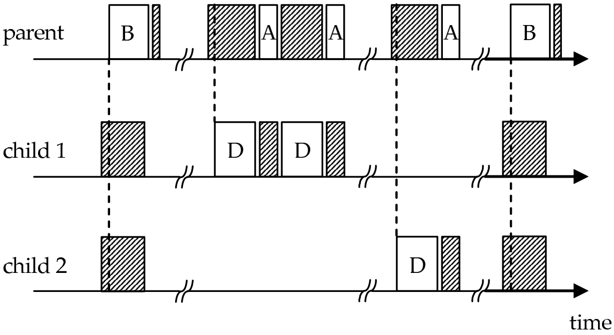

Figure 4.3: Message reception of a parent with two children. Upload slots are determined by parent beacon (B). All data messages (D) are explicitly acknowledged (A).

Due to the limited amount of memory available on current sensor node platforms the queue size is limited. Different buffering strategies may be employed depending on the application requirements. Dozer's default strategy only allows to keep the newest message from each distinct node in the queue; outdated entries are overwritten.

4.2.4 Command Management

While data flow in Dozer is strictly unidirectional towards the sink it is often desirable to be able to send information to one or several nodes in the network. Dozer establishes such a lightweight backward channel by making use of the beacon messages. Commands injected at the data sink are included in the sink's next beacon message. Every node that receives a beacon containing a command temporarily stores the command and includes it in its next beacon. By repeating this procedure at each level of the tree the command is disseminated through the whole network. Besides addressing a command to all nodes in the network the injection of commands for individual nodes is also supported. Nodes that are not directly addressed by a command also relay it to enable propagation to nodes deeper down the tree.

Upon reception of a beacon message from the parent the Tree Management component hands the command to the *Command Manager* module for further processing. The module checks if this node belongs to the set of intended recipients of the command. If this is the case the command is dispatched to the application running on top of the Dozer system. Thus, applications are able to define their own custom commands and corresponding command handlers.

	Current Draw	Power Consum.
uC sleep, radio off	6.0 uA	0.015 mW
uC active, radio idle	12.17 mA	30.43 mW
uC active, radio RX	12.63 mA	31.58 mW
uC active, radio TX	16.10 mA	40.25 mW

Table 4.1: Measured current consumptions of the TinyNode platform in different states at 2.5 volt.

4.3 Experimental Evaluation

In this section we evaluate Dozer's performance under different conditions in real-world testbeds. First, a set of preliminary measurements on a small-scale network are conducted to estimate the scalability of the system. In a second step we present results of a deployed indoor network consisting of 40 sensor nodes. Clock drift compensation is implemented by means of worst-case guard times guaranteeing a prior wake up of the receiver before the sender starts its transmission.

4.3.1 Hardware and Operation System

For all experiments we used the TinyNode 584 sensor platform [47] produced by Shockfish SA was used. It features a MSP430 mirocontroller with 10 kB of RAM and 48 kB of program memory. Furthermore 512 kB of external flash are available. However, due to the high energy costs for access to the flash Dozer does not make use of it. The platform includes a Semtech XE1205 radio transceiver. This radio is known for its good transmission ranges and high data rates of up to 153 kbit/s. For our measurements the nodes were operated at 868 MHz using 0 dBm transmission power and a bandwidth of 75 kbit/s[2]. As a power source two customary 1.2 volt rechargeable batteries were installed with a capacity of 1900 mAh each. The measured current draws for sleep mode, idle listening, receiving, and sending under these conditions are shown in Table 4.1. As can be extracted from the table, on the TinyNode platform idle listening is nearly as expensive as the actual reception of a message. Thus it benefits greatly from Dozer's scarce use of unscheduled random channel overhearing. Furthermore, the cost for transmission and reception of a message are in the same order of magnitude.

Dozer is implemented on top of the TinyOS-1.x operating system. No changes were made to the operating system excepts the replacement of a timer module whose genuine version contains a bug. Under certain conditions

---

[2] As described in [47], at the same transmission power, the XE1205 radio attains higher communication ranges than other state-of-the-art platforms. Hence, we are able to transmit at lower power while still achieving good ranges.

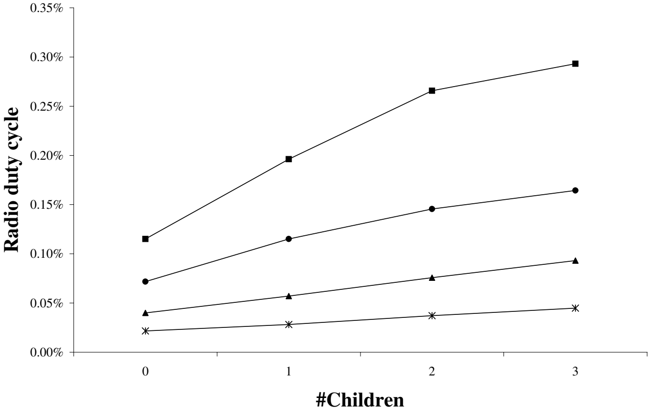

Figure 4.4: Radio duty cycle of a node depending on its number of children. Measurements were performed with beacon intervals of 15 s (square), 30 s (circle), 1 min (triangle), and 2 min (star), respectively.

timer events are triggered too late. In normal operation common TinyOS-applications do not encounter this undesired behavior frequently. However, due to the heavy load on the timer module, this malfunction regularly occurs in the Dozer system with disastrous consequences. A once deferred schedule becomes useless since all relative timings are out of sync. Consequently, a node affected by this problem inevitably looses connectivity and falls back to bootstrap mode. Thus the replacement of the timer module was mission critical.

The memory footprint of Dozer is 20 kB in program memory and 1.7 kB RAM. The message queue of size 20 in the Data Manager module thereby contributes 39% of the RAM usage.

4.3.2 Small Scale Experiments

Measuring the energy drain of a node is a non-trivial task. On the one hand, the measuring interval is too long for high-resolution measurements with an oscilloscope. On the other hand, a voltmeter is too inaccurate to capture short changes in current draw. Hence, we decided to measure energy consumption indirectly. For this purpose, all nodes log their radio duty cycles. This is achieved by summing up the differences between ratio startup and shutdown times. Since spotting the exact switching times from send to receive mode and vice versa is difficult, only the total uptime is recorded ignoring the specific state of the radio. This information is propagated to the base station using Dozer's own data gathering mechanism. The collected information can be converted to power consumption values using Table 4.1. As nodes only provide the overall radio uptime, a worst-case approximation is made. That is, it is assumed that they are always in transmit mode if their radio is active. As a consequence, all results related to energy consumption throughout the remainder of this chapter can be considered as an upper bound for the actual power consumption.

To investigate the relation between a node's power drain, its number of children, and the beacon interval time we conducted a series of experiments on a small network with predefined topology. In each run the node of interest was directly connected to the sink. Over time, up to three children were included in the network and forced to connect to the monitored node. This sequence was repeated with beacon intervals in the range of 15 seconds to two minutes. The data sample interval was set to four times the length of a beacon interval. The results of these experiments are depicted in Figure 4.4.

Originally, the goal of this experiment was to come up with lower bounds for the achievable duty cycles at different positions in the data gathering tree. However, initial test results exhibited unexpected fluctuations when run with different sensor nodes. After closer examination, it became clear that the inherent clock drift is a significant factor influencing the total duty cycle. Thus the following results do not represent precise lower bounds. Nevertheless, they provide an accurate approximation of the radio uptimes in real networks.

Figure 4.4 shows that the duty cycle decreases as the beacon interval grows larger. This elementary observation is based on the fact that the number of messages to be transmitted within one beacon interval is constant independent of its length. Hence, longer intervals lead to prolonged sleeping periods without significantly increasing the radio uptime. Using a similar line of argument, the variable additional costs for a newly connected child at different beacon intervals can be understood.

The reduction of the incurred overhead for the fourth child in the 15 second beacon interval experiment illustrates another phenomenon worth mentioning. Costs for additional children do not necessarily have to grow linearly. Simplified, a parent's costs for a child are twofold. On the one hand, it has to receive the child's data messages. These costs cannot be prevented. On the other hand, the parent has to forward the received messages. Thus, in its next upload slot it has to power up the radio and to send the pending messages. Since the radio start-up consumes a similar amount of time, and thus energy, as the transmission of one data message, its overhead is not negligible. Consequently, if the parent is able to upload data from two or more children in one upload slot it saves the additional overhead of turning on the radio for each of these children individually—costs per child decrease.

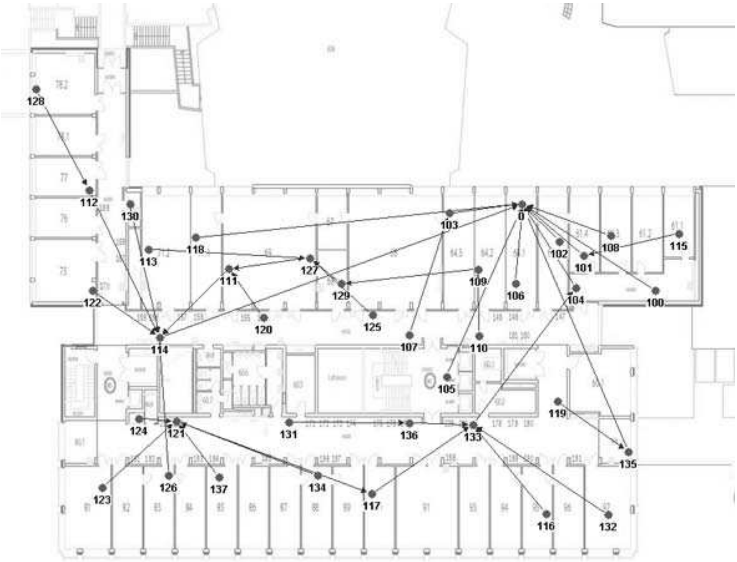

Figure 4.5: Indoor deployment of 40 sensor nodes including a snapshot of Dozer's data gathering tree. Node 0 (upper-right corner) acts as data sink.

4.3.3 Office Floor Experiment

To put Dozer's fitness for real-world deployments to the test, a generic indoor network was run for several weeks. The topology was no longer predefined for this setting but automatically constructed by the Dozer system.

Setting and Protocol Parameters

The considered testbed consisted of 40 TinyNode sensor nodes. The nodes were deployed on one floor of our office building (see Figure 4.5). The dimensions of the building are approximately 70 x 37 meters resulting in an testbed area of more than 2500 square meters. During the whole operation of the network, the floor was populated with more than 80 people during office hours. Thus, the nodes were exposed to frequently changing environmental conditions. Furthermore, during the deployment phase special attention was payed to construct a network with heterogeneous density. While nodes were concentrated in the upper-right part of the building to achieve a dense region, the southern part was only sparsely populated. This allowed the performance evaluation of Dozer in networks featuring different characteristics. In addition to 38 sensing nodes, a base station (Node 0) was placed in the

Beacon interval	30 s
Max. jitter	650 ms
Data sampling interval	120 s
Potential parents update interval	15 min
Overhearing	1 s/4 h
Compensated clock drift	100 ppm
Max. stored potential parents	5
Message queue size	20

Table 4.2: Configuration of the Dozer system for the office floor testbed.

upper-right corner of the map. This location was chosen to get a deep data gathering tree and to enforce multi-hop communication. One further extra node was positioned in the vicinity of the sink for debugging purposes. This node acted as a network sniffer which overheard and logged all network traffic at the base station.

In total Dozer was tested for more than one month on this network. Detailed logging information forming the basis of the evaluation in this section were gathered during one week of operation. Each node thereby sent approximately 5000 data messages to the sink.

As described in Section 4.2 the Dozer system can be tweaked to suite the requirements of a specific application. Table 4.2 shows the important parameters and their assigned values for the office floor testbed. Though the anticipated clock drift in our scenario is less than 50 ppm, Dozer was configured to allow for 100 ppm. Consequently, more energy than strictly necessary was consumed. In return, with this setting, the system is expected to operate properly in outdoor environments facing moderate temperature changes. All other values were chosen to represent a possible demand of a real application in the domain of environmental monitoring.

Tree Topology

Figure 4.5 shows a snapshot of the data gathering tree as it was witnessed during the experiment. Each node features one outgoing arrow pointing to its parent. It can be seen that the base station (Node 0) has numerous children. This has two different reasons. On the one hand, the parent rating function described in Section 4.2.1 promotes connections to the sink since the latter has zero tree depth. As a consequence, each node receiving a sink beacon fist tries to connect to the base station before inquiring any other nodes. On the other hand, the base station was flashed with a slightly modified version of Dozer. Since the sink usually runs on external power—as it is the case in our setting—it is less compelled to economize on its energy resources. Thus,

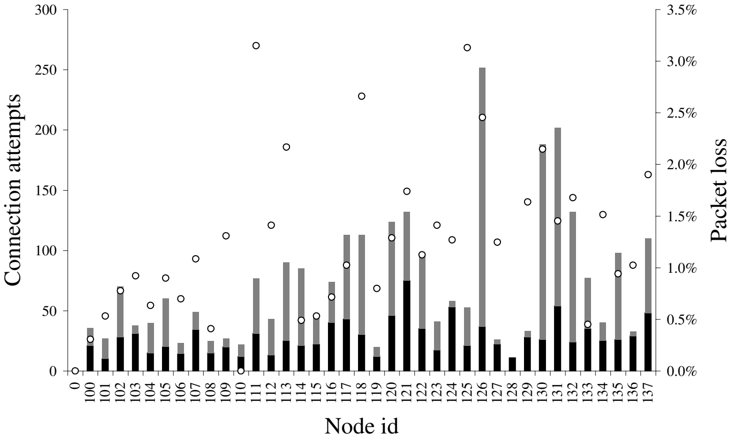

Figure 4.6: Number of successful (black) and failed (grey) connection attempts per node. Per node packet loss on the second y-axis.

the contention window was extended and the sink was configured to accept more than one child per connection phase.

Another observed phenomenon is the fact that hardly any connections passed the central core of the building. We assume that multiple sources of interference led to this barrier. For one, the corridors are lined with solid metal lockers perturbing most radio communication. On the other hand, this zone also comprises the ventilation system, sanitary facilities, and multiple elevators producing additional interference.

We examine the stability of the data gathering tree by investigating topology changes and message loss. Topology changes are indicated by a node exchanging its parent. Both of these values are depicted in Figure 4.6. As hoped for, message loss was low, in average 1.2% and at maximum 3.15%. However, Node 128 is excluded from this analysis. Due to its peripheral position in the network it was only able to connect to one single other node (Node 112). In case of a temporary interruption in the connection to its parent the node went to suspend mode. In addition, the low network density in its vicinity resulted in a low probability for a quick recovery. Thus, the node suffered from message loss of approximately 30%.

The measured high message yield at the base station is evidence of the correct operation of Dozer's Tree Maintenance module. As emerges from Figure 4.6, a significant number of topology changes were necessary to cope with momentary, local channel irregularities.

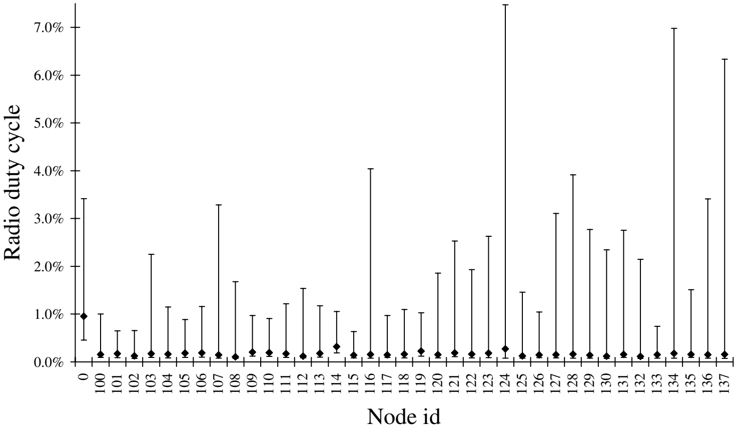

Figure 4.7: Average radio duty cycle of each node including RMS errors.

Energy Consumption

As in the small testbed described in Section 4.3.2, the energy consumptions of the deployed nodes were measured indirectly via their duty cycles. Figure 4.7 depicts the average radio activity of each node in the network. The upward error bar shows the root mean square (RMS) error of all measurements exceeding the average duty cycle; the downward error bar is defined accordingly. The overall average duty cycle of all sensing nodes is 1.67‰ with a standard deviation of 0.0004. Applying the values from Table 4.1 results in an mean energy consumption of 0.082 mW.

Looking at individual nodes, the sink had by far the highest radio uptime of almost 1%. This is not surprising since it had to process the data of the whole network. Additionally, the extended contention window directly affects its duty cycle and explains the considerable difference in comparison to the sensing nodes.

For further investigation of the energy consumption, we take an in-depth look at the two sensing nodes with highest duty cycles—Node 114 and Node 124. Node 124 exhibits a radio uptime of 2.8‰. Figure 4.8 shows a snapshot of 2000 consecutive data messages of this node. As can be seen for most of the time the node ran at a duty cycle of 0.7‰. Comparing this value to the results from Section 4.3.2 leads to the conclusion that the node is a leaf in the data gathering tree. However, three different energy intensive effects can be observed. First, the most dominant peaks exceeding 20% are due to scans for a full beacon interval. This means that the node was forced to establish a new connection but did not find an appropriate potential par-

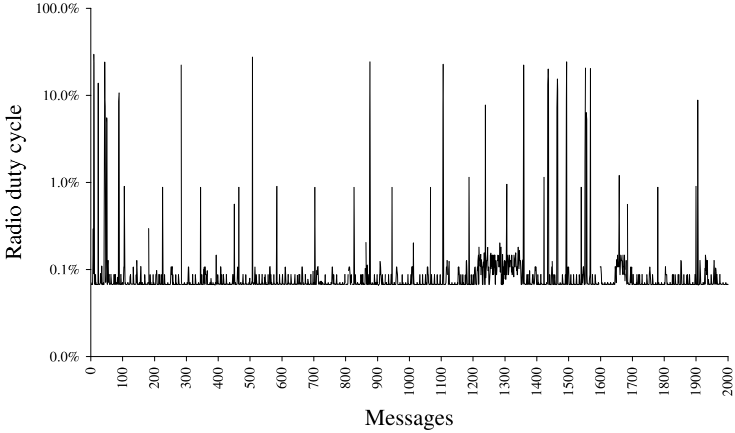

Figure 4.8: Radio duty cycle of Node 124 over a period of three days.

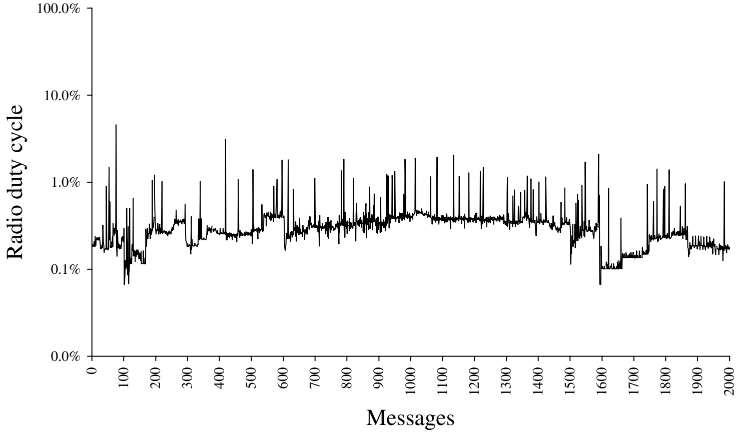

Figure 4.9: Radio duty cycle of Node 114 over a period of three days.

ent in its cache. Second, the overhearing phase once every four hours results in a temporary duty cycle of around 1%. Finally, the potential parents updates lead to the fringes of up to 1‰. These insights and the knowledge that Node 124 was located in a small storage room allows the conclusion that it had but a small neighborhood. Consequently, in times of normal operation it was able to run at nearly optimal duty cycle. However, in case of connection interruptions the interference affected all its possible connections resulting in a fallback to bootstrap mode. Unlike Node 128, it only suffered from brief network disconnections. Thus, it quickly managed to reintegrate in the data gathering tree.

Node 114 features a similar average duty cycle as Node 124, namely 3.2‰. But its power consumption is caused by other reasons as the different RMS error values indicate. Figure 4.9 depicts the radio duty cycle of Node 114 over a period of 2000 successive data messages. Parents updates and overhearing which are always part of a node's normal operation can also be spotted in this chart. However, there is no evidence for a bootstrap phase. In fact, Node 114 acts as a relay for several children and thus cannot reach minimal duty cycles as low as Node 124.

4.4 Clock Drift Compensation

In Section 4.3 Dozer dealt with clock imprecisions using worst-case guard times. For each message reception the radio was turned on in time to guarantee a successful transmission at a relative drift of up to 100 ppm. With a beacon interval and thus synchronization period of 30 seconds these 100 ppm result in a guard time of 4.5 milliseconds. In comparison to the 4.7 milliseconds required to actually transmit a data message this is a significant overhead. Since the majority of all transmissions between two nodes do not require worst-case guard times a more sophisticated drift compensation mechanism has the potential to preserve a large amount of energy.

An appropriate drift compensation component for Dozer has to feature several properties. First, it should not demand additional message exchanges. Considering Dozer's low message rate requirement to maintain the data gathering tree the cost of extra synchronization messages would not only nullify their positive effect but even increase total energy consumption. Second, due to the inherent limitations of current sensor network hardware in terms of memory and computation power complex calculations and data structures are out of scope. Bearing these restrictions in mind, we developed a drift compensation component that solely recycles information produced by the tree maintenance component.

The approach of having the receiver compensate clock skew is reasonable when using fixed guard times but suffers from a scalability problem if dynamic drift compensation is applied. Each node in Dozer's data gathering tree can have a large number of children and would thus be obliged to compute and maintain an individual drift prediction for each of them. Dozer avoids this issue by exploiting the tree structure of the network. It burdens the child with the task of drift compensation for all communication with its parent—independent of whether it is sender or receiver. Hence, each node in the network only has to handle the drift compensation for one bidirectional connection.

With the periodic beacon transmission a message exchange is available which can be used to synchronize a child with its parent. When a child first connects to a new parent it is unable to predict their relative drift. Consequently, it uses a worst-case guard time of 200ppm. On receiving the next beacon the message is immediately time-stamped using the current system time of the child. From the information contained in the parent beacon the child is able to compute how much time has passed between the last two consecutive beacon transmissions according to its parent's clock. Based on this value and its own local timestamps the child derives the current relative drift between the two nodes.[3] The child then incorporates its current drift estimation to compute the next beacon reception time. This process is repeated on each beacon reception in order to maintain an up-to-date drift prediction.

However, a drift compensation system has an additional task besides drift prediction. It does not suffice to know when to expect the next transmission but we also need to know how much the actual drift may vary from the computed prediction. This knowledge is essential as it is required to determine how much in advance and for how long the radio needs to be turned on to maximize packet reception probability while minimizing energy consumption; in other words the system has to decide what dynamic guard times to use. In a completely stable environment this guard time could be set to zero as messages would always be perfectly on time. In real-world scenarios though, drift changes over time. Thus even an ideal drift estimation system fails if the environmental conditions abruptly change in between two synchronization cycles and no additional guard times are used.

Dozer tackles this problem by starting out with a large guard time. As described before a newly connected node uses a worst-case guard time for the first beacon reception. After receiving the next beacon the difference between the predicted and the actual reception time, denoted as estimation error, is used as guard time for further communication. As a consequence, the employed guard times rapidly converge towards zero. As the system still has to cope with sudden changes in the environment we limit the minimum guard time to the maximum expected drift change within one beacon interval. This value is hardware dependent and was empirically determined for the

---

[3] This one-way drift prediction mechanism was shown to provide good relative drift estimation [59].

TinyNode platform. The maximal measured estimation error of 3 jiffies[4] in one beacon interval prompted us to set Dozer's conservative minimum guard time to 20 jiffies.

In practice it still has to be expected that the system sporadically fails to receive transmissions. In this case the guard time is immediately set back to its initial worst-case value. Consequently, a trade-off between the size of the minimum guard time and the number of failed transmissions has to be made. For networks operating at more or less stable temperatures such as indoor deployments, an aggressive minimum guard time results in best performance. On the other hand, networks which have to deal with frequent temperature changes such as most outdoor deployments benefit from a more conservative setting.

4.4.1 Evaluation

The indoor deployment used in Section 4.3.3 serves as a benchmark to assess the influence of the introduced enhancements and modifications on the power consumption of individual nodes. The drift compensation system was set up to allow for up to 200 ppm of relative drift and a maximal drift change of 20 ppm per beacon interval. All other parameters remained the same. We deactivated Node 128 to be able to compare the results with those of Section 4.3.3. Unfortunately, Node 132 failed shortly after the initial deployment and did not recover.[5] As a consequence this node is excluded from all further calculations.

Using worst-case guard times—compensating up to 100 ppm of relative drift—Dozer produces an average radio duty cycle of 1.67‰ on sensing nodes. A repetition of the same experiment with drift compensation results in a mean duty cycle of 1.28‰ which corresponds to an improvement of more than 23%. The lion's share of the gained radio sleep time stems from the drift compensation component. Looking at the minimum duty cycle of a leaf node a reduction from 0.7‰ in Section 4.3.3 to 0.57‰ is found. This observed performance gain can be accredited to the drift compensation component as no other modifications influence this specific value. Consequently, this component accounts for approximately 18% reduced radio uptime. The remaining 5% result from a change in the potential parents mechanism. As described in Section 4.2.1 each node maintains a list of neighboring nodes which may serve as a parent in case of a problem with the current connection. This list is refreshed regularly in order to maintain up-to-date information about known neighbors. In this experiments, however, the update procedure is disabled. The drawback of this step are increased setup costs when a new parent is required as it is harder to time a rendezvous with neighboring nodes based on old information. Yet, these costs are more than compensated by

---

[4] 1 jiffy = 1 clock tick = 1/32768 s

[5] Later investigations unveiled a defective power source to be the root of the problem.

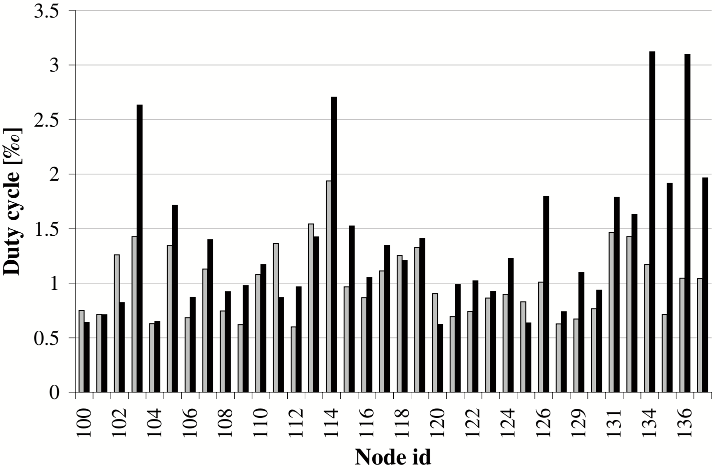

Figure 4.10: Average radio duty cycles in per mill of all nodes with one (black) and two (grey) sinks, respectively.

the amount of energy saved by not communicating regularly with all nodes on the potential parents list. A detailed overview of the average duty cycles for all deployed sensing nodes is given in Figure 4.10.

In the following we investigate the overhead incurred by clock imprecision and guard times at a leaf node of the data gathering tree. We assume an ideal system transmitting the same number of messages as Dozer in the given configuration but with perfectly synchronized clocks. A leaf needs to exchange 4.5 messages per minute. A message including its preamble is 44 bytes long, leading to a transmission time of 4.7 ms at 75 kbps. The radio switching times from sleep to transmit and back add up to 2 ms. These parameters result in an ideal duty cycle of 0.525‰. Thus, using worst-case guard times, thereby exhibiting a duty cycle of 0.7‰, results in an overhead of 33%. In comparison, Dozer with drift compensation exhibiting a duty cycle of 0.57‰ reduces this overhead to 8.57%. This is an improvement by a factor of roughly 4.

4.5 Multiple Sinks

Another challenge every multi-hop data gathering system faces is increased load on nodes close to the sink. Serving as a relay for most data messages these nodes are forced to handle higher traffic rates than other nodes in the network. Consequently, their duty cycles increase and their batteries deplete

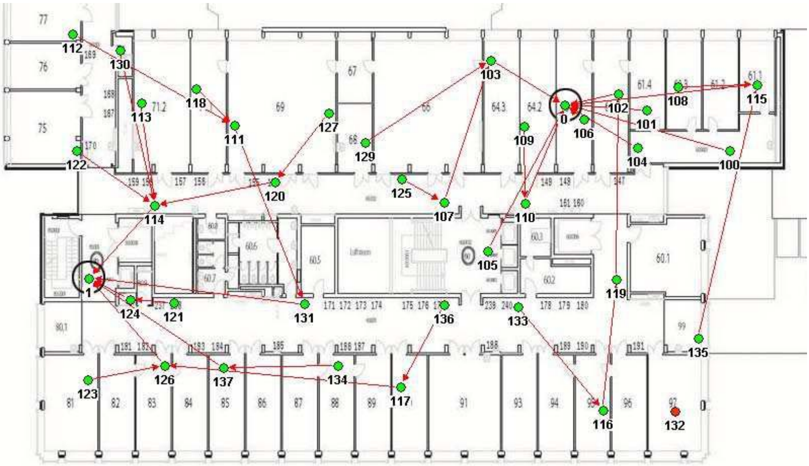

Figure 4.11: Indoor setup of 39 nodes. Node 0 (upper-right corner) and Node 1 (lower-left corner) act as data sinks.

at a faster rate. Furthermore, due to their strategically important position in the network their failure usually also marks the end of the whole network as the remaining nodes cannot reach the sink any longer. Diminishing the increased load at nodes close to the sink is accomplished by reducing the amount of traffic actually reaching the sink. There are two feasible approaches to achieve this goal. First, in-network processing and aggregation as introduced in Chapter 3 may be applied to condense the forwarded information. Depending on the nature of the sampled data this optimization may result in a significant improvement. Unfortunately, in many scenarios neither in-network processing nor aggregation can be applied since the full sampled information of each node is required. Under these conditions the problem can only be countered by the sensible deployment of additional sinks to spread the load on more nodes.

Dozer is designed to handle dynamic addition and removal of sinks. If more than one sink is available a separate data gathering tree for each of them is constructed without any additional overhead. All nodes in the network select a parent with minimal distance to any of these sinks. The different trees are thereby not labeled and thus nodes do not know to which of them they belong. As a consequence, nodes—and with them whole sub-trees—may freely switch from one data gathering tree to another without even noticing. The advantage of this system lies in its flexibility towards load and interference. External interference may cut off parts of the network or enforce long detours to bypass error-prone areas. In such cases the presence of additional sinks may help avoiding data loss and generally reduces maximum tree depths. Hence, not only load at nodes close to the sink is reduced but also the global average duty cycle is improved.

Based on the network introduced in Section 4.3.3 we ran an additional experiment with a second sink. The two data sinks were placed on opposite sides of the building to provide reasonable load balancing. As shown in Figure 4.11 the second sink (Node 1) had the desired effect in that two nearly equal sized trees were constructed. Nodes in the central area of the building regularly switched between parents in both trees since their hop count towards both sinks was balanced. With the decrease in tree depth the maximum observed load measurably dropped resulting in a global average duty cycle of 0.93‰—compared to 1.28‰ with one sink. Figure 4.10 depicts the average duty cycle for each node in the network with one and two sinks, respectively. The majority of all nodes benefit from the presence of the second sink and the reasons for the individual improvements are manifold. For example, Node 134 and 136 are affiliated to the tree rooted at Node 1 avoiding a link through the center of the building. In the single-sink experiment both of them had to rely on this link which apparently suffered from high packet loss. Other nodes such as Node 107 has to service a lighter subtree and therefore spends less energy forwarding messages. Following the same line of argument it can also be explained why some few nodes such as Node 102 suffer from an increased duty cycle in the two sink setup.

4.6 Related Work

Corresponding to the importance of the problem, there have been a plethora of research efforts addressing data gathering in the last few years. Energy efficiency of most existing work [86, 101, 140, 131, 20] stems from the application of generic energy-efficient MAC protocols [119, 133, 144, 155, 157]. These protocols turn off the wireless transceiver whenever possible to save power. Two types of protocols are thereby distinguished: TDMA and contention-based protocols. Protocols falling in the latter category incorporate duty cycling to achieve low power operation. [157] and [100] coordinate the nodes' sleep schedules such that neighboring nodes are awake at the same time. In the active phases CSMA/CA is used to control channel access. To achieve high energy efficiency the active periods must be very small compared to the time nodes are in sleep mode. Since the whole network wakes up at roughly the same time, nodes suffer from high channel contention which reduces network throughput. T-MAC [144] is an improvement of S-MAC [157] handling varying traffic load with adaptive duty cycling. The protocol does however not overcome the inherent limitations of this approach. *Low-power listening* is another strategy to condition contention based MAC protocols to low-power requirements. To avoid idle listening nodes turn off the radio most of time, only periodically probing the channel for the presence of activity. Once network activity is detected the node switches on its radio to listen for the incoming packet. To ensure the receiver is listening a sender has to prefix its packet with a long preamble acting as an in-band busy-tone. A key advantage of asynchronous low-power listening protocols [119, 35] is that the sender and receiver can be completely decoupled in their own duty cycles. However, these protocols suffer from the overhearing problem, since the long preamble also wakes up nodes who are not the intended receiver of a packet. To overcome this drawback, channel polling times of neighboring nodes are synchronized in [158], thereby preventing the protocol from sending long preambles. This move incurs contention during the scheduled channel probing which is resolved by using CSMA. However, a drawback of this protocol is that all nodes require to be tightly synchronized to meet energy efficiency which creates additional costs.

In contrast to the aforementioned protocols, TDMA-based solutions establish a schedule where each node is assigned one or possible multiple timeslots. In each slot nodes are then able to communicate without provoking packet collisions or suffer from overhearing. Pure TDMA protocols are however hardly feasible in reality since they require global time synchronization and are susceptible to topological changes of the network. Hence, most proposed protocols use a combination of pure TDMA and the above mentioned contention-based approach.

In [123] a two phase protocol is proposed. In the first phase a node collects information about its two-hop neighborhood and participates in a distributed slot allocation procedure. In addition, a protocol for network-wide time synchronization is executed during this phase. Once the TDMA schedule is computed in the first phase the protocol switches to the second phase and executes the schedule. FPS [54] and its descendant Twinkle [53] are closely related to the protocol described in this chapter. The coarse grained scheduling of FPS represents a distributed TDMA approach where each node schedules its own children. Although this schedule ensures that parents and their children are contention free, collisions may still occur due to other nodes in the network or poor time synchronization. This contention is handled using CSMA. The protocol does not incorporate a tree construction and is thus dependent on other protocols establishing such a network topology. In contrast to our solution FPS—and thus also Twinkle—requires global time synchronization.

Other related work considers real-time scenarios for mission-critical applications, where the data gathering must be performed within strict latency constraints [97, 160]. DMAC [97] proposes an adaptation of S-MAC optimized for data gathering. The protocol assumes that a routing tree towards the data sink exists. The active periods of the nodes are staggered according to their level in the tree. CSMA is used to arbitrate between children in order to prevent collisions. DMAC achieves low data delivery latency at the sink. However, there is a substantial overhead in case of network instabilities.

Chapter 5

Energy-Efficient Deployment Support

The search for energy-efficient solutions has led to numerous algorithms and protocols that strive to reduce the energy consumption of an operational sensor network. In the previous chapters, we have discussed related work in the context of medium access control protocols, topology control, and data aggregation. This impressive body of work resulted in new insights and several intriguing solutions thereby mainly focusing on the *operational phase* of a network. However, in many applications a crucial loss of energy occurs already *before* the sensor network reaches its operational state, that is, during its deployment.

Consider for instance a water (or power, gas, etc.) metering network for an apartment complex. Each apartment is equipped with a water metering sensor. At midnight, all sensors wake up for a few seconds, the water consumption of each apartment is sent to a base station in multi-hop fashion, and all sensors go back to sleep for another 24 hours. In the operational phase such a sensor network features a gargantuan sleep/awake ratio, allowing even conventional batteries to last several years. To achieve such a long lifetime the node's duty cycle must be significantly below 1%. However, the deployment of the sensor nodes might take days or weeks. With a naive deployment protocol, say, when nodes stay awake until the entire system is deployed, the battery of the node deployed first might be drained before the network even becomes operational. This highlights a problem that is particularly pronounced in settings in which the node's duty cycle during the operational phase is small, but the deployment takes long. Such time-consuming deployment phases are common for data gathering systems considered in this thesis.

Generally, once all sensor nodes are fully deployed, the network should make the transition from the deployment phase to the operational phase as quickly as possible. In particular, we might like to externally trigger a *network discovery* procedure that allows verifying the operability of the newly deployed network (e.g. detect faulty sensor nodes). Clearly, simple solutions to invoke such a system initialization would be to manually switch on all nodes once the deployment phase is completed, or to set a timer at the time of the node's deployment. Unfortunately, in many practical settings, neither of these hands-on solutions is practicable. First, nodes may be deployed in remote or hostile environment in which switching on nodes manually after all nodes are deployed may be impossible. Moreover, in application scenarios featuring a time-consuming deployment phase, predicting the exact duration of the deployment process is usually hard, hence ruling out the possibility of employing a solution based on predefined timers.

So how can the information about the beginning of the operational phase be distributed among the network nodes? Typically, this information is supposed to be *broadcasted by the nodes* in a multi-hop way through the entire network such that, eventually, every sensor node will know that the system is now ready to start its operational phase. Specifically, one or several nodes (in typical sensor network applications, this is usually the base station) are triggered externally. These nodes then try to inform their neighbors, who in turn inform their neighbors, and so forth. We call this externally triggered event that sets off the information broadcast the *launching point.* The trade-off studied in this chapter is about saving energy *during* deployment, yet quickly going into operational mode *after* the launching point.

Ideally, each node should remain in some kind of energy-saving *sleep mode* for the entire duration of the deployment phase preceding the launching point. In sleep mode, nodes do neither send data packets nor listen for incoming messages [134]. The problem is however that individual nodes do not know the exact time of the launching point, or the duration of the deployment phase. As a consequence, a node must periodically leave the sleep mode and listen for incoming messages to learn about the arrival of the launching point from neighbors.[1]

This observation establishes a trade-off between the energy consumption of nodes during the deployment phase and the rapidity of the transition to the operational phase. Neither of the two extremes, *always asleep* and *always awake* during the deployment, is satisfying; any decent protocol is in-between.

We believe that studying the trade-off delay vs. energy efficiency is practically important, even beyond the deployment problem. In particular, there are sensor networks that concentrate on discovering rare events, e.g. sensor

---

[1] Obviously, the problem could be elegantly solved using very low power "trigger" circuits, which operate continuously on small power budgets, and wake up more power-hungry circuits only upon receipt of a suitable signal from a neighboring node. Unfortunately, currently available standard hardware such as the Mica2 [52] wireless sensor nodes do not offer this functionality, and we therefore do not consider this option in this chapter.

networks for seismic surveillance in earthquake and rubble zones, or sensor networks monitoring enemy activity. The pronounced "event" character of such rare events leads to exactly the deployment problem trade-off. Namely, since events occur rarely, sensor nodes should be in sleep mode as often as possible to save energy. These energy savings, however, come at the cost of a prolonged reaction time once a rare event occurs. Hence, this conflict between energy-efficiency and the rapidity of information propagation is fundamental in sensor networks.

In this chapter, we take a step towards understanding and analyzing the trade-off between energy-efficiency and propagation delay, particularly during the deployment phase. We model the problem in a way that allows to compare different protocols and algorithms and evaluate their respective strengths and weaknesses, independent of application-specific parameters such as node distribution or deployment pattern. Specifically, we analyze the behavior of three different algorithms. The first algorithm [104] has originally been proposed for the purpose of neighbor discovery, but can be applied for the deployment problem as well. In addition, we present two novel algorithms that significantly outperform the algorithm by [104], for both worst-case *and* average-case scenarios. It is interesting to note that one of our algorithms is "semi-structured," in the sense that already deployed nodes structure themselves in a feeble way that allows to incorporate newly deployed nodes with a small energy overhead only. This semi-structured approach is in contrast to, say, tree-based dissemination algorithms in which during the deployment process, a lot of effort—and hence, energy—is required to recognize and integrate new nodes. We believe that constructing *semi-structures* is an interesting concept by itself, with potentially many applications beyond the scope of this thesis.

5.1 Unstructured Radio Networks

Our model of computation is based on the *unstructured radio network model* as introduced in [81]. This model aims to capture the harsh characteristics of newly deployed ad hoc and sensor networks. It encompasses various critical aspects such as asynchronous wake-up, absence of a MAC layer, and scarce knowledge about the network graph. More specifically, the model makes the following assumptions.

- During the deployment phase, sensor nodes *wake up asynchronously* at any time. Moreover, they do not have access to a global clock and hence, upon waking up, they do not know whether or how many other nodes in their neighborhood have already been deployed. Once the launching point is reached, we assume that all nodes have been deployed and therefore, no new nodes join the network. In other words, after the launching point we consider a static network.

- We also assume that nodes have *no built-in knowledge* about other node's distribution or wake-up pattern. Specifically, nodes are completely clueless about the number of nodes in their neighborhood. The only knowledge a-priori given to the nodes is an upper bound n for the total number of nodes deployed in the network. It has been shown in [69] that without any such estimate of n, every algorithm requires at least $\Omega(n/\log n)$ time until one single message can be transmitted without collision. In practice, the number of nodes in a network may not be known exactly, but it can be roughly estimated in advance.

- If a node receives multiple messages at the same time, these messages become garbled and cannot be received properly. Moreover, nodes do not feature a reliable *collision detection mechanism*. That is, nodes are not capable of distinguishing the situation in which two or more neighbors are sending and the situation in which no neighbor is sending. Furthermore, a sending node does not know how many (if any at all) of its neighbors have correctly received its transmission. The unreliable collision detection model is the strongest possible model when analyzing wireless networks. Clearly, algorithms designed for a model as restricted as this can also be employed by systems that are equipped with more sophisticated hardware.

- We model the multi-hop network as a unit disk graph (UDG). In a UDG $G = (V, E)$, with $n = |V|$, two nodes are connected by an edge if their Euclidean distance is at most 1. The network being multi-hop leads to well-known aspects such as the hidden-terminal problem.

- Finally, we assume that both the node's location and wake-up pattern is completely arbitrary, potentially even *worst-case*. Particularly, we do not assume any kind of uniform node distribution or Markovian wake-up pattern.

The various aspects of this model suggest that we deal with a particularly harsh model of computation; a model that captures many of the realistic characteristics of newly deployed sensor networks. We assume time to be divided into time-slots, the length of which are roughly the same at each node. In each time-slot a node can be in exactly one of the three following modes: *transmit* T, *listen* L, or *sleep* S. In sleep mode S, a nodes deactivates its radio subsystem to save energy. That is, a node does not overhear the shared medium in sleep mode and thus misses all messages sent by neighboring nodes. At the communication distances typical in sensor networks, listening for information on the radio channel is of a cost similar to transmission of data [122]. Therefore, the energy consumption $e(v)$ of a node v corresponds to the number of time-slots it spends in either transmit or listen mode. Consequently, reducing merely the node's *sending time* is not sufficient when designing energy efficient algorithms for sensor networks. Instead, the *listening time* must also be minimized.

5.2 The Deployment Problem

Before the sensor network can start performing its intended task, nodes must be deployed, a process that may take several days or even weeks. We divide the *non-operational phase* of a sensor network into two parts, the *deployment phase* and *notification phase* as shown in Figure 5.1. In the deployment phase, sensor nodes are physically positioned at their intended locations. Once this is done for all sensor nodes, the notification phase is triggered, in which the aim is to inform all nodes about the system being up and running. The transition to this second phase is induced by an externally triggered event that is received by at least one node in the network. We call this moment when the first node becomes notified the *launching point* $\mathcal{LP}$. During the notification phase, we call a node *notified* if it has already received the notification message, and *unaware* otherwise. At the launching point, at least 1 node is notified whereas at most $n-1$ nodes are unaware.

During the deployment phase, an algorithm may build an initial structure which can help speed up the notification process later on. On the other hand, the building and maintenance (incorporating newly awakening nodes into a tree, for example) of such a structure requires the nodes to stay awake longer and thus spend more energy. To enable a fair comparison between different algorithmic approaches, our problem definition has to be general enough to account for these various possibilities.

The total energy consumption of a node v in a deployment algorithm $\mathcal{A}$ can be divided into two parts, the *initialization energy* and the *maintenance energy*. The initialization energy $e_{init}(v)$ is the total amount of energy used by v to initially join a desired structure (e.g., decide whether it is a clusterhead or become a part of a tree). A node's initialization energy accrues only once, regardless of the length of the deployment phase. In contrast, the maintenance energy $e_m(v)$ denotes the total amount of energy used by v once it has been properly initialized. Specifically, the maintenance energy $e_m(v)$ includes the node's periodic listen phases which are necessary to learn about the launching point. If $e_m(v)$ is small, the node will require a long time before learning about the $\mathcal{LP}$, thus slowing down the notification phase. Depending on the nature of the algorithm, $e_m(v)$ may comprise additional aspects. Consider for instance an algorithm that is based on maintaining a tree-structure which allows for rapid event dissemination during the notification phase. In this case, already initialized nodes periodically send messages to inform neighbors that may have woken up in the meantime, thus enabling their integration into the tree.

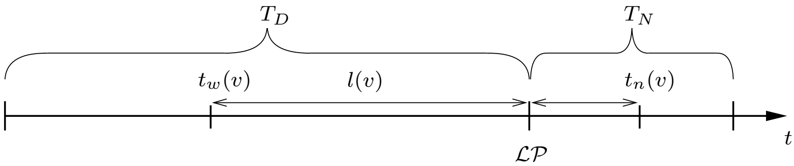

Figure 5.1: The deployment phase is of length T_D, the notification phase is of length T_N.

More formally, let T_D and T_N be the length of the deployment phase and notification phase, respectively. Further, $t_w(v)$ denotes the wake-up point of node v. The time v is active before the launching point is $\ell(v) = t(\mathcal{LP}) - t_w(v)$. Since we consider asynchronous wake-up with an imaginary adversary determining each node's wake-up point (and hence $\ell(v)$), we consider the *average maintenance energy* $a_m(v) = e_m(v)/\ell(v)$. This value describes the maintenance energy used by a node v for a single time-slot between its wake-up and the LP. Note that $a_m(v)$ is independent of a node's $t_w(v)$, $\ell(v)$, or the time of the launching point, because $a_m(v)$ considers only periodical maintenance costs, i.e., no initialization costs.

We still have to come up with a measure for the algorithm's *energy efficiency* that takes into account both the maintenance and the initialization costs, but remains independent of the specific wake-up pattern. For that, we define the energy efficiency of an algorithm $\mathcal{A}$ with regard to a deployment phase of length T_D, denoted by $E(\mathcal{A}, T_D)$, as the *average energy consumption of algorithm $\mathcal{A}$ per node and per time-slot.* That is, an algorithm in which all nodes listen in every time-slot has energy efficiency equal to 1, whereas the algorithm that lets all nodes sleep all the time has energy efficiency 0. With this definition, the measure of an algorithm's energy efficiency does not depend on the particular wake-up pattern of a given problem instance. Instead, it captures the characteristic of the algorithm itself, thus enabling a stringent and concise comparison between different approaches.

Formally, the two main quality measures of a deployment algorithm $\mathcal{A}$ are defined as follows.

Definition 5.1. *Let $\mathcal{A}$ be a deployment algorithm and let T_D be the length of the deployment phase before the launching point. Also, let $f(n)$ be a minimal function such that with probability at least $1 - 1/n$, it holds that $T_N \leq f(n)$. The algorithm's energy and time efficiency, $E(\mathcal{A}, T_D)$ and $T(\mathcal{A}, T_D)$, are defined as*

$$\begin{aligned} E(\mathcal{A}, T_D) &:= \frac{1}{n \cdot T_D} \sum_{v \in V} (e_{init}(v) + T_D \cdot a_m(v)), \\ T(\mathcal{A}, T_D) &:= f(n). \end{aligned}$$

Note that the definition of $E(\mathcal{A}, T_D)$ corresponds to the intuitive notion of *energy efficiency* given above. Particularly, the terms $T_D \cdot a_m(v)$ and $e_{init}(v)$ describe a node v's maintenance and initialization energy during a deployment phase of length T_D, respectively. Adding up these values over all nodes and dividing by $\frac{1}{n \cdot T_D}$, the number of nodes and time-slots leads to the energy efficiency $E(\mathcal{A}, T_D)$. As for the second measure, an algorithm has time efficiency $f(n)$ (for instance n^2) if with high probability, all nodes are notified $f(n)$ time-slots after the launching point.

Definition 5.1 allows us to compare deployment algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ in two ways. First, we can fix the notification time $f(n)$ and compare both algorithm's energy requirements. That is, we demand two algorithms to finish the notification period within the same amount of time. We then compare which algorithm requires more energy during the deployment phase to ensure that all nodes are notified within $f(n)$, i.e., $T_N \leq f(n)$. Alternatively, we can fix the energy consumption $E(\mathcal{A}_1, T_D)$ and $E(\mathcal{A}_2, T_D)$, respectively, of both algorithms and then compare the resulting length of the notification phase. Clearly, both comparison methodologies are two sides of the same coin; they both describe the inherent trade-off between energy efficiency and the rapidity of information dissemination.

5.3 Deployment Algorithms

In this section, we analyze three different algorithms under our model and derive their respective strengths and weaknesses. We begin our exposition by analyzing the so-called *birthday algorithm* proposed in [104] which can be employed as a algorithm for the deployment of sensor networks. In subsequent Sections 5.3.2 and 5.3.3, we propose two novel algorithms that significantly outperform [104].

For the analysis of the algorithms we assume time to be divided into synchronized time-slots. However, notice that none of the algorithms relies on this assumption. This simplification of the analysis is justified due to the standard trick introduced in [128] for the study of slotted versus unslotted ALOHA. In [128], it is shown that the realistic unslotted case and the idealized slotted case differ only by a factor of two. The basic intuition is that a single packet can only cause interference in two consecutive time-slots. By the same token, analyzing the algorithms in an "ideal" setting with synchronized time-slots, we obtain results which are only a factor two better in comparison to results in the more realistic unslotted setting.

Throughout this chapter, we will denote by N_v the set of neighbors of node v, i.e., $N_v = \{u \in V \mid \{u, v\} \in E\}$. Finally, we will make use of the following two facts. The first can be found in standard mathematical textbooks and the second was proven in [69].

Fact 5.1. *For all n,t, with $n \geq 1$ and $|t| \leq n$,*

$$e^t \left(1 - \frac{t^2}{n}\right) \leq \left(1 + \frac{t}{n}\right)^n \leq e^t.$$

Fact 5.2. *Given a set of probabilities $p_1 \cdots p_n$ with $\forall i : p_i \in [0, \frac{1}{2}]$, the following inequalities hold:*

$$\left(\frac{1}{4}\right)^{\sum_{k=1}^{n} p_k} \leq \prod_{k=1}^{n} (1 - p_k) \leq \left(\frac{1}{e}\right)^{\sum_{k=1}^{n} p_k}.$$

5.3.1 Birthday Algorithm

The birthday algorithm $\mathcal{A}_{birth}$ proposed in [104] is conceptually simple. Before being notified, a node v listens in each time-slot with probability p_L and sleeps with probability $1 - p_L$. Once v has learned about the launching point in the notification phase, it sends with probability p_T, which is set to $1/n$, and listens with probability p_L. The choice of the sending probability is motivated by the goal to avoid interference in the case when several notified nodes try to send a message to a common neighbor. Clearly, the broad idea of the algorithm is to let nodes sleep as long as possible. That is, we want to choose p_L as small as possible while still guaranteeing a speedy notification phase.

$\mathcal{A}_{birth}$ has been designed and analyzed for neighborhood discovery, i.e., not for the deployment problem as considered in this chapter. In this section, we will analyze the birthday algorithm's performance in the context of the problem of sensor network deployment. Specifically, we analyze the trade-off exhibited by $\mathcal{A}_{birth}$ in accordance to the definitions given in Section 5.2.

Let $f(n)$ be the time in which we require the notification procedure to finish with high probability, that is, let $f(n)$ be a function such that $T_N(\mathcal{A}_{birth}) \leq f(n)$ with high probability. Given this constraint, we want to optimize the algorithm's energy efficiency. The achievable trade-off is expressed in the following theorem.

Theorem 5.1. *Let $f(n)$ be a function such that the* birthday algorithm *$\mathcal{A}_{birth}$ has time efficiency $T(\mathcal{A}_{birth}, T_D) \leq f(n)$. For arbitrary T_D, $\mathcal{A}_{birth}$'s energy efficiency is*

$$E(\mathcal{A}_{birth}, T_D) \in \Theta\left(\frac{n^2}{f(n)}\right).$$

Proof. The birthday algorithm does not require any initialization and therefore, $e_{init}(v) = 0$, for all $v \in V$. The average maintenance energy for each node corresponds directly to the listening probability, i.e., $a_m(v) = p_L$. Hence, the algorithm's energy efficiency is

$$E(\mathcal{A}_{birth}, T_D) \;=\; \frac{1}{n \cdot T_D} \sum_{v \in V} (T_D \cdot p_L) \;=\; p_L.$$

Consider the network graph $G_b = (V_b, E_b)$ consisting of nodes $v_1, \ldots, v_n$ positioned in a line, i.e., v_i is a neighbor of v_j iff $j = i + 1$ and $1 < j \leq n$. Recall that the nodes themselves have no knowledge about the topology of the network. Finally, let v_0 be the node that is externally triggered at the launching point.

By the construction of G_b, the information about the arrival of the launching point has to traverse the entire network in a hop-by-hop fashion. We call a time-slot t *successful*, if there is a notified node v_i that sends in t and its unaware neighboring node v_{i+1} listens at the same time. Informally, the notification information is passed on by one hop in each successful time-slot.

The probability P_{suc} that a time-slot t is successful is $P_{suc} = p_L \cdot p_T$. To pass the notification through the entire chain, a minimum of $n-1$ successful time-slots are required. In total, the algorithm is allowed to use $f(n)$ time-slots and the broadcast has to succeed with probability at least $1-1/n$. Given these constraints, we want to minimize p_L thus optimizing $E(\mathcal{A}_{birth}, T_D)$. In expectation, the number of successful rounds is $p_L p_T f(n)$. Since we want at least $n - 1$ successes, it follows that

$$\frac{p_L f(n)}{n} \;=\; n - 1 \quad \Rightarrow \quad p_L \;\in\; \Omega\left(\frac{n^2}{f(n)}\right).$$

Finally, we show that for a large enough constant c, $p_L = cn^2/f(n)$ is enough to obtain the high probability argument. Let X be the number of successful rounds. The expected value of X is $\mu = p_L f(n)/n$. We bound the probability of having less than $n-1$ successful rounds using Chernoff Bounds as

$$\begin{aligned} P[X < n-1] \;&=\; P\left[X < \left(1 - \left(1 - \frac{n(n-1)}{p_L f(n)}\right)\right) \frac{p_L f(n)}{n}\right] \\ &<\; e^{-\frac{p_L f(n)}{2n}\left(1-\frac{n(n-1)}{p_L f(n)}\right)^2} = e^{-\frac{cn}{2}\left(1-\frac{1}{c}\right)^2}, \end{aligned}$$

which is smaller than $1/n$ for a suitably large constant c. Notice that setting p_L to a value strictly smaller, i.e., $p_L \in o(n^2/f(n))$ renders the exponent positive thus not yielding the desired result. □

Keep in mind that for the birthday algorithm, the notification phase T_N must be at least of length $\Omega(n^2)$ to guarantee a feasible solution. In the following two sections, we will propose algorithms featuring strictly better trade-offs.

```
Algorithm $\mathcal{A}_{uni}$

upon wake-up do:
listen with probability $p_L$, otherwise sleep
upon notification do:
for $i := \lceil \log n \rceil + 1$ to 1 by $-1$ do
    $p_T := 1/2^i$
    for $\frac{c(\lceil \log n \rceil + 1)}{p_L}$ time-slots do
        send message with probability $p_T$
    end for
end for
```

5.3.2 Uniform Algorithm

In a way, the second algorithm $\mathcal{A}_{uni}$ shares the philosophy of the birthday algorithm, having in common that there are no initialization costs and all nodes perform the same procedure uniformly. Specifically, algorithm $\mathcal{A}_{uni}$ has one input parameter, the listening probability p_L; c is a constant to be defined later.

The main improvement is a simple idea originally stemming from the literature on broadcast in radio networks [8]. When trying to inform an unaware node, notified nodes will exponentially increase their sending probability, thus reducing the average waiting time. Notice that the number of time-slots per sending probability is inversely proportional to the unaware node's listening probability p_L. In comparison with the birthday algorithm $\mathcal{A}_{birth}$ analyzed in Section 5.3.1, $\mathcal{A}_{uni}$ exhibits a strictly better performance trade-off as stated in Theorem 5.2.

Theorem 5.2. *Let $f(n)$ be a function such that the* uniform algorithm $\mathcal{A}_{uni}$ *has time efficiency $T(\mathcal{A}_{uni}, T_D) \leq f(n)$. For arbitrary T_D, $\mathcal{A}_{uni}$'s energy efficiency is at most*

$$E(\mathcal{A}_{uni}, T_D) \in O\left(\frac{n \log^2 n}{f(n)}\right).$$

Proof. Like $\mathcal{A}_{birth}$, $\mathcal{A}_{uni}$ does not require any initialization and all nodes are treated uniformly. Therefore, by the same argument as in Section 5.3.1, $E(\mathcal{A}_{uni}, T_D) = p_L$.

We define the listening probability p_L to be $p_L := cn(\lceil \log n \rceil + 1)^2 / f(n)$. We seek to show that for a constant $c \geq 12$, the probability of the notification message advancing at least *one hop* in time $O(f(n)/n)$ is at least $1 - n^{-2}$. Since the diameter of the network is at most n, the theorem follows from $(1 - n^{-2})^n \geq e^{-1/n} \geq 1 - n^{-1}$.

Let $Z_{v,t}$ denote the event of node v hearing a notification message in time-slot t. Consider an unaware node $v \in V$ and let t_0 be the first time-slot in which at least one node in v's neighborhood N_v is notified. Starting from this round, the sum of sending probabilities $\sum_{w \in N_v} p_T(w)$ increases. Let t^* be the last time-slot in which the sum of sending probabilities is smaller than 1/2. Notice that it takes at most $t^* - t_0 \leq (\lceil \log n \rceil + 1) \cdot \frac{c(\lceil \log n \rceil + 1)}{p_L}$ time-slots until t^* is reached.

Now, consider the time interval $\mathcal{I} = [t^* + 1, \ldots, t^* + \frac{c(\lceil \log n \rceil + 1)}{p_L}]$. During this interval, notified nodes can at most double their p_T and new nodes will send with the initial sending probability $p_T = \frac{1}{2n}$. At the end of this interval, the sum of sending probabilities is therefore at most

$$\sum_{w \in N_v} p_T(w) \;\leq\; 2 \cdot \frac{1}{2} + \sum_{w \in N_w} \frac{1}{2n} \;\leq\; \frac{3}{2}. \tag{5.1}$$

Therefore, in each time-slot $t \in \mathcal{I}$, the sum of sending probabilities is at least 1/2 and at most 3/2. The probability $P[Z_{v,t}]$ that v receives the notification message from one of its neighbors is

$$\begin{aligned}
P[Z_{v,t}] \quad &= \quad p_L \sum_{w \in N_v} \left(p_T(w) \cdot \prod_{\substack{q \in N_v \\ q \neq w}} (1 - p_T(q)) \right) \\
&\geq \quad p_L \sum_{w \in N_v} p_T(w) \cdot \prod_{q \in N_v} (1 - p_T(q)) \\
&\underset{\text{Fact 5.2}}{\geq} \quad p_L \sum_{w \in N_v} p_T(w) \cdot \left(\frac{1}{4}\right)^{\sum_{q \in N_v} p_T(q)} \\
&\geq \quad \frac{3p_L}{2} \cdot \left(\frac{1}{4}\right)^{3/2} \;>\; \frac{p_L}{6}.
\end{aligned}$$

For large enough functions $f(n)$ and $p_L = cn(\lceil \log n \rceil + 1)^2 / f(n)$, the probability that none of the $\frac{c(\lceil \log n \rceil + 1)}{p_L}$ time-slots $t \in \mathcal{I}$ is successful is at most

$$\begin{aligned}
P[\cap_{t \in \mathcal{I}} \overline{Z_{v,t}}] \quad &\leq \quad \left(1 - \frac{p_L}{6}\right)^{\frac{c(\lceil \log n \rceil + 1)}{p_L}} \\
&= \quad \left(1 - \frac{cn(\lceil \log n \rceil + 1)^2}{6f(n)}\right)^{\frac{f(n)(\lceil \log n \rceil + 1)}{n(\lceil \log n \rceil + 1)^2}} \\
&\underset{\text{Fact 5.1}}{\leq} \quad e^{-\frac{c}{6}(\lceil \log n \rceil + 1)} \;<\; n^{-2}.
\end{aligned}$$

Therefore, with probability exceeding $1 - n^{-2}$, the notification message is passed on at least by one hop in time

$$t^* - t_0 \;\leq\; (\lceil \log n \rceil + 1) \cdot \frac{cf(n)(\lceil \log n \rceil + 1)}{cn(\lceil \log n \rceil + 1)^2} \;=\; \frac{f(n)}{n}.$$

Consequently, by the argument given at the beginning of the proof, the notification message reaches all n nodes within time $f(n)$ with probability at least $1 - \frac{1}{n}$. □

The trade-off obtained by $\mathcal{A}_{uni}$ is strictly better than the one obtained by the birthday algorithm $\mathcal{A}_{birth}$. Moreover, in the case $p_L = 1$, the algorithm allows a feasible solution for functions $f(n) \in \Omega(n \log^2 n)$ as opposed to $f(n) \in \Omega(n^2)$ for the birthday algorithm.

5.3.3 Cluster Algorithm

Finally, our last algorithm is based on a different paradigm than the two previous ones. Instead of treating all nodes identically (uniformly), it forms a *semi-structure* that renders the notification of nodes during the notification phase quicker. On the other hand, installing and maintaining this structure requires additional energy during the deployment phase. Contrary to the first two algorithms, the cluster algorithm $\mathcal{A}_{clu}$ has non-zero initialization costs $e_{init}(v)$ and unequal energy requirements between different nodes. Therefore, $\mathcal{A}_{clu}$ uses the full potential of Definition 5.1.

The design of $\mathcal{A}_{clu}$ aims to mend the main energy dissipation of the two previous algorithms, namely the lack of synchronization. If neighboring nodes had synchronized wake-up points, the notification phase would take significantly less time. Consequently, when demanding the same notification efficiency T_N, the nodes could sleep longer, thus saving energy during the deployment phase. The problem is that synchronization between neighboring nodes incurs additional set-up and maintenance costs and the question is whether these additional costs will equiponderate the gains stemming from the above-mentioned notification speed-up.

Our approach is based on grouping neighboring nodes into synchronized clusters. Within such a cluster, nodes wake-up at the same time. In particular, the algorithm constructs a clustering based on a *maximal independent set* of the underlying network graph $G = (V, E)$. An independent set S of G is a subset of V such that $\forall u, v \in S, (u, v) \notin E$. S is a *maximal independent set* (MIS) if any node v not in S has a neighbor in S. In our particular case, we do not consider a MIS on the original graph G, but we consider a MIS of the graph G' in which two nodes are adjacent if their mutual distance is at most $1/2$. This corresponds to each node setting its transmission range to $1/2$.

```
Algorithm $\mathcal{A}_{clu}$: Code for non-leader $u$

upon wake-up do:
perform MIS algorithm of length $O(W+\log^2 n)$ → decide on leader $s(u)$,
  receive wake-up point $r_3$
loop
    sleep until next wake-up point $r_3$
    for $\eta \log n$ time-slots listen for notification message $\mathcal{M}_n$
    $r_3 := r_3 + I$
end loop
upon notification do:
loop
    upon receiving $\mathcal{M}_a(r_2)$, wait until $r_2$          } $S_1$
    for $i := \lceil \log n \rceil + 1$ to 1 by $-1$ do          ⎫
        for $(\gamma+\eta)(\lceil \log n \rceil + 1)$ time-slots do   ⎪
            send message with probability                          ⎪
            $p_T = 1/2^i$                                          ⎬ $S_2$
            upon receiving $\mathcal{M}_r$, quit for-loops         ⎪
        end for                                                    ⎪
    end for                                                        ⎭
end loop
```

Constructing a MIS efficiently in an unstructured radio network is a non-trivial task. Our clustering algorithm for the deployment problem uses a MIS algorithm proposed in [108]. It is important to note, however, that any other MIS algorithm in the unstructured radio network model can be used instead without affecting the asymptotic energy efficiency of algorithm $\mathcal{A}_{clu}$. We now introduce an adaptation of this algorithm to a level of detail necessary to understand our results.

Each node starts executing the algorithm upon waking up. Nodes that are located in a region which is already covered by an existing MIS node (leader) will learn about their being covered during an initial waiting period of length W. If this is not the case, v will decide whether it joins the MIS or not during the second phase of length $O(\log^2 n)$ time-slots. Hence, in total, every node needs to be awake for $W + O(\log^2 n)$ time-slots before deciding whether it becomes a leader or not. Subsequently, for the entire duration of the deployment phase, leaders have to transmit with a sending probability of $\Theta(\log n/W)$ to inform newly awakening nodes of their being covered. This prevents nodes that wake up later from invalidating the MIS condition. Non-leader nodes do not have any duties and can sleep arbitrarily long. Let $s(v)$ denote the leader of node v and for $u \in S$ let $S(u)$ refer to the set of nodes having u as their leader, i.e., $S(u) = \{v | u = s(v)\}$ for all $u \in S$.

Algorithm $\mathcal{A}_{clu}$: Code for leader v

```
upon wake-up do:
perform MIS algorithm of length $O(W + \log^2 n)$ $\rightarrow$ become leader with cluster S(v)
choose rendezvous point $r_1$
$r_2 := r_1 + \eta\lceil\log n\rceil$, $r_3 := r_2 + (\gamma + \eta)\lceil\log^2 n\rceil$
loop
    sleep or send with probability $\log n / W$ until next wake-up point $r_1$
    for $\eta \log n$ time-slots send $\mathcal{M}_a(r_2)$ with probability $p_{MIS} \in \Theta(1)$    } $S_1$
    for $(\gamma + \eta)(\lceil\log n\rceil + 1)^2 - \eta(\lceil\log n\rceil)$ time-slots listen for $\mathcal{M}_n$    ⎫
    if $\mathcal{M}_n$ received then                                                                              ⎬ $S_2$
        send $\mathcal{M}_r$ for $\eta \log n$ time-slots with probability $p_{MIS}$
    end if                                                                                                        ⎭
    sleep until $r_3$
    if notified then                                                                                              ⎫
        for $\eta \log n$ time-slots send $\mathcal{M}_n$ with probability $p_{MIS}$                            ⎬ $S_3$
        become non-leader
    end if                                                                                                        ⎭
    $r_1 := r_1 + I$
    $r_2 := r_1 + \eta\lceil\log n\rceil$
    $r_3 := r_2 + (\gamma + \eta)\lceil\log^2 n\rceil$
end loop
```

We incorporate a slightly stronger version of the result in [108] into our algorithm $\mathcal{A}_{clu}$. Particularly, we require that the MIS S be connected if we consider all two-hop paths in G. Note that this condition is automatically fulfilled if the network density is reasonably high (for instance, if there is at least one node in every disk of radius $1/4$ in the convex hull of the nodes). In $\mathcal{A}_{clu}$, each leader $v \in S$ coordinates the nodes in $S(v)$ and is responsible for their synchronized waking up. Specifically, a leader v decides on the timing of the *rendezvous windows* for its cluster; a time window during which the nodes $w \in S(v)$ are simultaneously awake. Every node $w \in S(v)$ learns the timing of these rendezvous windows from its leader v. The idea is that once a leader is notified, it can notify all nodes in its cluster at almost the same time.

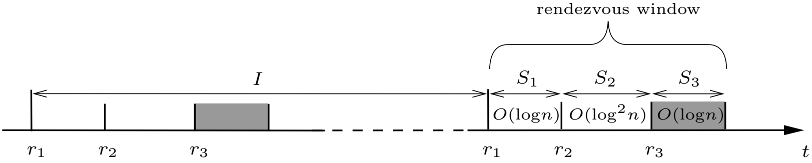

Figure 5.2: Rendezvous interval I and rendezvous window.

Each rendezvous takes place in three steps as shown in Figure 5.2. In the *proclamation step* S_1, leader v announces the rendezvous interval to neighboring nodes which do *not* belong to $S(v)$. The reason is that once a node is notified, it remains listening on the channel. Such a node must be able to notify neighboring leaders, even if it is in a different cluster itself (otherwise, the notification message would not broadcast through the network). In other words, the proclamation step is intended for announcing the notification across cluster boundaries. The conveyance of these messages in the opposite direction is the aim of the second step, the *leader-notification step* S_2. In this step, notified nodes try to inform a neighboring unaware leader.

Finally, the rendezvous is concluded by the *notification step* S_3. A notified leader v attempts to notify all unaware nodes in $S(v)$ during this step. Note that this is the only time-interval during which an unaware non-leader node must be awake. Summarizing, the actions during the rendezvous window are designed as to guarantee that a notification message in the neighborhood of a leader v is, first, passed to v, and second, passed from v to all nodes in $S(v)$. After the rendezvous window, a notified leader becomes a non-leader node to help informing other leaders located in its neighborhood. Finally, notice that all transmissions during a rendezvous are performed using the full transmission range. In the following, we give a more precise description of algorithm $\mathcal{A}_{clu}$ as performed by leaders and non-leaders, in which γ and η are suitably large constants.

Consider a rendezvous window of leader v. In the *proclamation step* S_1, v sends an announcement message $\mathcal{M}_a(r_2)$ containing the starting time of the second step of the rendezvous with a constant probability $p_{MIS} \in \Theta(1)$. Let u be a notified node with $(u, v) \in E$ and $u \notin S(v)$. Notified nodes remain listening in order to eavesdrop an announcement message of neighboring leaders. If node u receives such a message $\mathcal{M}_a(r_2)$ from v, it tries to notify v during the subsequent *leader-notification step.* In the analysis, we will show that with high probability *every* notified node in v's neighborhood will receive $\mathcal{M}_a(r_2)$ from v.

In the *leader-notification step* S_2 all notified neighbors of v try to send a notification message $\mathcal{M}_n$ to v. Notice that if there are no notified neighbors of v, nothing happens during the *leader-notification step.* The procedure of informing a leader follows along the lines of the uniform algorithm presented in Section 5.3.2. Starting with probability $\frac{1}{2n}$, notified nodes exponentially increase their sending probability to speed up the notification. To prevent too much “noise” (i.e., too many nodes sending with high probability at the same time), v starts sending a reception message $\mathcal{M}_r$ with probability p_{MIS} as soon as it has received $\mathcal{M}_n$. In the analysis, we show that the $O(\log^2 n)$ time-slots are sufficient to perform these tasks with high enough probability.

Finally, unaware nodes in $S(v)$ are only awake in the *notification step* S_3 starting from r_3. They are listening during these time-slots, waiting for a possible notification message $\mathcal{M}_n$ from a potentially notified v.

Analysis In the following, we will sometimes omit calculating the exact values of the various constants involved for the sake of clarity and due to lack of space. Instead, we focus our attention on portraying the main ideas and concepts of our algorithm and proofs. Exact constants can be derived by a more rigorous analysis in a straightforward way.

We begin with a simple geometric lemma, saying that the number of leaders (and corresponding clusters) in any disk of radius 1 is bounded by a constant.

Lemma 5.3. *Let v be an arbitrary node. Let $Q := \{s(u) \mid u \in N_v\}$ be the set of all leaders that lead at least one node in v's neighborhood. It holds that $|Q| \leq \varphi$ for a constant φ.*

Proof. The proof follows from a simple area argument. There cannot be more than a constant number of disks of radius 1/4 packed into a disk of radius 1 such that no two disks overlap. □

In the following, let $p_v(t)$ be the sending probability of node v in time-slot t. Further, $\Phi_v(t)$ denotes the sum of the sending probabilities of neighbors of v that are not leaders, formally

$$\Phi_v(t) := \sum_{u \in N_v \setminus S} p_u(t).$$

In the next lemma, we show that given an upper bound on $\Phi_v(t)$, $\eta \log n$ time-slots are sufficient to let a leader inform all its neighbors. Because of cyclic dependencies, it is convenient to formulate this upper bound on $\Phi_v(t)$ as an invariant.

Invariant 5.1. *Let t be an arbitrary time-slot. For all leaders $v \in S$, it holds that $\Phi_v(t) \leq \chi$, for a constant $\chi \leq \frac{3\varphi}{2}$.*

Lemma 5.4. *Let v be a leader and consider a time interval $\mathcal{J}$ of length $\eta \log n$ during which v sends with probability p_{MIS}. Under the condition that Invariant 5.1 holds, all nodes $w \in S(v)$ receive the message during $\mathcal{J}$ with probability $1 - n^{-3}$.*

Proof. Let N_v^2 denote the set of nodes which are in distance at most 2 of v. We call a time-slot successful if v sends, but no other node in N_v^2 sends. In a successful time-slot, all nodes in N_v receive the message from v without collision. The probability $P_{suc}(t)$ that a single time-slot t is successful is at least

$$\begin{aligned} P_{suc}(t) &\geq p_{MIS} \cdot \prod_{\substack{w \in N_v^2 \\ w \neq v}} (1 - p_w(t)) \\ &\underset{\text{Lm 5.3}}{\geq} p_{MIS} \cdot (1 - p_{MIS})^{\varphi - 1} \prod_{w \in N_v^2 \setminus S} (1 - p_w(t)) \\ &\underset{\text{Fact 5.2}}{\geq} p_{MIS} \cdot (1 - p_{MIS})^{\varphi - 1} \left(\frac{1}{4}\right)^{\sum_{w \in N_v^2 \setminus S} p_w(t)} \\ &\geq p_{MIS} \cdot (1 - p_{MIS})^{\varphi - 1} \left(\frac{1}{4}\right)^{\chi\varphi} \in \Theta(1). \end{aligned}$$

where the last inequality follows from Lemma 5.3 and Invariant 5.1 which holds by assumption. Finally, the probability P_{no} that none of the $\eta \log n$ time-slots is successful is bounded by

$$\begin{aligned} P_{no} &\leq \left(1 - p_{MIS}(1 - p_{MIS})^{\varphi - 1} \left(\frac{1}{4}\right)^{\chi\varphi}\right)^{\eta \log n} \\ &\leq \frac{1}{2n^3} \end{aligned}$$

for a suitably large constant η. □

Unfortunately, Lemma 5.4 holds only conditionally; based on the assumption that Invariant 5.1 holds. In the following, we prove this invariant by placing an upper bound on $\Phi_v(t)$ that holds throughout the execution of the algorithm with high probability.

Lemma 5.5. *With probability $1 - n^{-2}$, it holds for all t and for all leaders $v \in S$ that $\Phi_v(t) \leq \chi$, where $\chi \leq \frac{3\varphi}{2}$ is a constant, i.e., Invariant 5.1 holds.*

Proof. At the beginning of the notification phase, Invariant 5.1 clearly holds. For the sake of contradiction, assume that leader v is the first to violate the invariant. Further, notice that Φ_v can only increase if some of its neighboring non-leader nodes are in the *leader-notification step* S_2. The idea is that as soon as v receives the notification message, it starts sending a reception message $\mathcal{M}_r$. We will show that the nodes in N_v receive this message and stop sending. This prevents Φ_v from increasing too much.

We define time-slots t_v^* for a leader v, such that, $\Phi_v(t_v^*) < 1/2$ and $\Phi_v(t_v^* + 1) \geq 1/2$. By the same argument as in the proof of Theorem 5.2 (cf. Inequality (5.1)), we can bound $\Phi_v(t_v^* + (\gamma+\eta)(\lceil \log n \rceil + 1)) \leq 3/2$. That is, for all time-slots t in the interval $\mathcal{J} = [t_v^* + 1, \ldots, t_v^* + (\gamma+\eta)(\lceil \log n \rceil + 1)]$, it holds that $1/2 < \Phi_v(t) \leq 3/2$. The probability $P_{suc}(t)$ that v receives a message without collision in an arbitrary time slot $t \in \mathcal{J}$ is at least

$$\begin{aligned} P_{suc}(t) \;&\geq\; \prod_{w \in S \cap N_v} (1 - p_w(t)) \\ &\quad \cdot \sum_{w \in N_v \setminus S} \left(p_w(t) \cdot \prod_{\substack{q \in N_v \setminus S \\ q \neq w}} (1 - p_q(t)) \right) \\ &\geq\; (1 - p_{MIS})^{\varphi} \cdot \Phi_w(t) \left(\frac{1}{4}\right)^{\Phi_w(t)} \\ &\geq\; (1 - p_{MIS})^{\varphi} \cdot \frac{3}{2} \left(\frac{1}{4}\right)^{3/2} > \frac{(1 - p_{MIS})^{\varphi}}{6}. \end{aligned}$$

We continue the proof by showing that with high probability, the first $\gamma(\lceil \log n \rceil + 1)$ time-slots of $\mathcal{J}$ suffice such that v receives $\mathcal{M}_n$. Specifically, the probability P_{no} that none of these time-slots is successful is

$$P_{no} \;\leq\; \left(1 - \frac{(1 - p_{MIS})^{\varphi}}{6}\right)^{\gamma(\lceil \log n \rceil + 1)}, \tag{5.2}$$

which again can be made $P_{no} \leq n^{-3}/2$ for large enough constants γ. Once, node v receives $\mathcal{M}_n$, it will try to acknowledge by sending $\mathcal{M}_r$. Notice that there are at least $\eta(\lceil \log n \rceil + 1)$ time-slots in $\mathcal{J}$ left during which $1/2 < \Phi_v(t) \leq 3/2$. By the assumption that v is the *first* leader to violate Invariant 5.1, we know that until the end of $\mathcal{J}$, Invariant 5.1 and consequently Lemma 5.4 hold. That is, with probability at least $1 - n^{-3}$, the message $\mathcal{M}_r$ will be received by all nodes in N_v within the $\eta(\lceil \log n \rceil + 1)$ time-slots. Hence, the probability that v is the first node to violate Invariant 5.1 is bounded by $2 \cdot n^{-3}/2$ for suitably large constants γ and η. Because there are at most n leaders in the network and every leader needs to be notified only once, the Lemma holds with probability $1 - n^{-2}$. □

The following Corollary is implicit in the proof of Lemma 5.5 (cf, Inequality (5.2)).

Corollary 5.6. *Consider a leader v and the leader-notification step S_2 of a notification window. If there exists a notified node in $N_v \backslash S$, v will be notified at the end of S_2.*

Thanks to Lemma 5.5, we can now apply Lemma 5.4 throughout the algorithm with high probability.

Theorem 5.7. *With probability at least $1 - 1/n$, the algorithm works as demanded, that is, each leader v successfully announces to all its neighbors about the proclamation step S_1 for the entire duration of the notification phase. Furthermore, as soon as there exists a notified non-leader in N_v, v will be notified in the following leader-notification step S_2. And finally, a notified leader v will inform all its neighbors $u \in N_v$ in the notification-step S_3 following v's notification.*

Proof. The steps S_2 and S_3 follow directly from Lemma 5.4, Corollary 5.6, and the fact that there are at most n leaders, each of which is notified at most once. By Lemma 5.4, every attempt of sending a $\mathcal{M}_a$ message is successful with probability $1 - n^{-3}$. Each of the n nodes needs to send at most n messages $\mathcal{M}_a$ during the notification phase. The proof is concluded because the set of leaders is connected if we consider all two-hop paths in G. □

Of particular interest is the energy efficiency and its comparison to the two previous algorithms. Let $m \leq n$ be the number of leader nodes in the network and let ξ denote the energy efficiency $E(\mathcal{A}_{clu}, T_D)$. Clearly, the ratio m/n depends on the *density* of the network. The following theorem quantifies the achieved trade-off.

Theorem 5.8. *Let $f(n)$ be a function such that algorithm $\mathcal{A}_{clu}$ has time efficiency $T(\mathcal{A}_{clu}, T_D) \leq f(n)$. Let m be the number of* leaders *chosen by $\mathcal{A}_{clu}$. For a given T_D, $\mathcal{A}_{clu}$'s energy efficiency $\xi = E(\mathcal{A}_{clu}, T_D)$ is bounded by*

$$\xi \in O\left(\frac{\frac{f(n)}{n \log n} + \log^2 n}{T_D} + \frac{n \log n}{f(n)} + \frac{m \log^2 n}{f(n)}\right).$$

Proof. The choice of I's length determines the trade-off between energy-efficiency and the speed of notification. We have to choose I such that with high probability, the notification broadcast is finished within time $f(n)$. We do so by setting I to a value guaranteeing that the notification proceeds at least one hop in time $f(n)/n$ with high probability. That is, we set I to $\lceil f(n)/n \rceil$.

Upon waking up, each node v has initialization costs $e_{init}(v) \in O(W + \log^2 n)$. For the maintenance costs during the deployment phase, we distinguish between leaders and non-leader nodes. Non-leaders are awake for the duration of $\eta \log n$ during each rendezvous interval of length I. Thus, for non-leaders, $a(v) = \eta\lceil \log n\rceil/I$. Leaders must be awake longer in each rendezvous interval, namely $2\eta \log n + (\gamma+\eta)(\lceil \log n\rceil + 1)^2 \in O(\log^2 n)$ time-slots. Additionally, leaders need to send with probability $\log n/W$ in each time-slot.

Therefore, for appropriate constants α, $\delta > \gamma + \eta$, and η the energy efficiency $E(\mathcal{A}_{clu}, T_D)$ of $\mathcal{A}_{clu}$ is at most

$$\begin{aligned}
\xi &= \frac{1}{nT_D}\Bigg[\sum_{v\in S} \alpha\left(W + \log^2 n\right) \\
&\quad + T_D \sum_{v\in S}\left(\frac{\log n}{W} + \frac{\delta \log^2 n}{I}\right) \\
&\quad + \sum_{v\in V\setminus S}\left(\alpha(W + \log^2 n) + \frac{T_D \eta \log n}{I}\right)\Bigg].
\end{aligned}$$

Setting $I = \lceil f(n)/n\rceil$ and $W = I/\log n$, we obtain

$$\begin{aligned}
\xi &= \frac{\alpha(W + \log^2 n)}{T_D} + \frac{m}{n}\left(\frac{\log n}{W} + \frac{\delta \log^2 n}{I}\right) \\
&\quad + \frac{n-m}{n}\cdot\frac{\eta \log n}{I} \\
&\leq \frac{\alpha\left(\frac{f(n)}{n \log n} + \log^2 n\right)}{T_D} + \frac{m}{n}\cdot\frac{(\delta+1) n \log^2 n}{f(n)} \\
&\quad + \frac{n-m}{n}\cdot\frac{\eta n \log n}{f(n)} \\
&\in O\left(\frac{\frac{f(n)}{n\log n} + \log^2 n}{T_D} + \frac{m \log^2 n}{f(n)} + \frac{n \log n}{f(n)}\right).
\end{aligned}$$

□

Observe that the first asymptotic term of Theorem 5.8 contains T_D in the denominator. This highlights the notion that the amount of energy spent on initializing a structure weighs more or less heavily, depending on the respective length of the deployment phase. Specifically, this term can be neglected if the deployment phase is long. As for the two remaining terms, they express the energy efficiency of leaders and non-leaders, respectively.

Algorithm $\mathcal{A}$	Energy Efficiency $E(\mathcal{A}, T_D)$
$\mathcal{A}_{birth}$ [104]	$\Theta(1)$
$\mathcal{A}_{uni}$	$\Theta(\frac{\log^2 n}{n})$
$\mathcal{A}_{clu}$	$\Theta(\frac{\log n}{n} + \frac{m \log^2 n}{n^2})$

Table 5.1: A comparison of the energy efficiency of the three algorithms for a fixed $T(\mathcal{A}, T_D) = f(n) \in \Theta(n^2)$ and large enough T_D.

5.3.4 Discussion

In this section, we discuss the results obtained in Theorems 5.1, 5.2, and 5.8. These theorems yield a concise comparison between the three algorithms analyzed in this chapter.

For the comparison, we demand all three algorithms to finish their notification phase within a fixed amount of time $f(n) \in \Theta(n^2)$, $f(n)$ being the same for all algorithms. This allows us to compare the energy efficiency $E(\mathcal{A}, T_D)$ each algorithm is required to invest to ensure that the notification is finished within time $f(n)$. As mentioned in Section 5.2, we obtain the same results when asking the question the other way around, i.e., when fixing the algorithm's energy efficiency and comparing the resulting time efficiency $T(\mathcal{A}, T_D) = f(n)$. Table 5.1 shows the results derived from Theorems 5.1, 5.2, and 5.8 under the assumption that the length of the deployment phase T_D is long enough compared to $f(n)^2$.

First, we emphasize that both $\mathcal{A}_{clu}$ and $\mathcal{A}_{uni}$ significantly outperform $\mathcal{A}_{birth}$, regardless of the network density or, generally, the ratio between leaders vs. non-leaders. It is interesting to study the relative strengths of $\mathcal{A}_{clu}$ and $\mathcal{A}_{uni}$. Asymptotically, the trade-off achieved by $\mathcal{A}_{clu}$ is strictly better than $\mathcal{A}_{uni}$ if $m \in o(n)$, that is, if less than a constant fraction of the nodes are leaders. If, for instance, $m \in O(n/\log n)$, the resulting asymptotic energy-efficiency is $E(\mathcal{A}_{clu}, T_D) \in O(n \log n / f(n))$, which is better than $\mathcal{A}_{uni}$ by a $O(\log n)$ factor. In case the number of leaders is a constant fraction of n, the asymptotic energy efficiency is $O(n \log^2 n / f(n))$, which equals the trade-off achieved by $\mathcal{A}_{uni}$. Hence, depending on the *network density* and the resulting number of leaders, the asymptotic energy efficiency of $\mathcal{A}_{clu}$ is either better or equal than that of $\mathcal{A}_{uni}$. Intuitively, high network densities render the number of leaders m small relative to n and hence, $\mathcal{A}_{clu}$ is more efficient than $\mathcal{A}_{uni}$. That is, the higher the network density, the more worthwhile it becomes to invest initial energy on obtaining a cluster-based semi-structure.

---

[2] Note that $f(n) \in \Theta(n^2)$ is the smallest value for $f(n)$ so that $\mathcal{A}_{birth}$ is capable of finishing its notification phase within $f(n)$ in arbitrary networks. Similar results as shown in Table 5.1 can be obtained for higher values of $f(n)$ in a straightforward way.

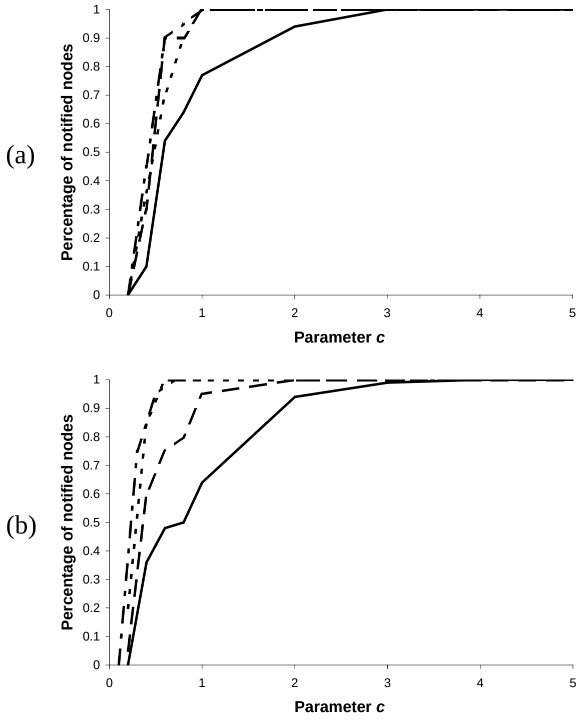

Figure 5.3: Percentage of notified nodes for a given c of algorithm $\mathcal{A}_{uni}$. $\mathcal{A}_{uni}$ is thereby simulated with $p_L = 1$ (solid), 0.5 (dashed), 0.1 (dotted), and 0.01 (dash-dotted) at network densities 5 (a) and 20 (b).

5.4 Simulations

In this section we evaluate the performance of the three algorithms proposed in Section 5.3 on average-case Euclidean graphs, that is on graphs with randomly placed nodes. In particular networks were constructed by placing nodes randomly and uniformly on a square field of size 10 by 10 units and subsequently computing for each node set the Unit Disk Graph—defined such that an edge exists if and only if its Euclidean length is at most one unit. The resulting Unit Disk Graphs were then employed as input networks for the algorithms under consideration.

The two newly introduced algorithms $\mathcal{A}_{uni}$ and $\mathcal{A}_{clu}$ make use of several parameters. In Section 5.3, an exact value for the parameter c of $\mathcal{A}_{uni}$ is given whereas minute bounds for the parameters of $\mathcal{A}_{clu}$ are omitted for the sake of clarity. However, it is important to notice that Section 5.3 considers a worst-case scenario while we assume average-case networks in this section. Hence, we can presumably set the parameters for the two algorithms to lower values than determined in the previous section.

Figure 5.3 shows the mean percentage of notified nodes if algorithm $\mathcal{A}_{uni}$ is executed on networks with density 5, and density 20 respectively, against the parameter c ranging from 0 to 5 and various input parameters p_L. The network density is thereby defined as the number of nodes per unit square throughout the rest of this section. Figure 5.3 leads to the important observation that we can lower the parameter c with decreasing listening probability p_L on condition that all nodes are notified after the termination of $\mathcal{A}_{uni}$. Furthermore, as apparent in Figure 5.3(a) and 5.3(b), the parameter c is quite independent of the network density and can thus be chosen exclusively dependent on the input parameter p_L of $\mathcal{A}_{uni}$. As a consequence, we define the parameter c as follows. If the listening probability p_L of algorithm $\mathcal{A}_{uni}$ is above 0.75 we set $c = 3$. If p_L is between 0.5 and 0.75 c is set to 2 and $c = 1$ if p_L is less than 0.5.

Contrary to $\mathcal{A}_{uni}$, algorithm $\mathcal{A}_{clu}$ has more than one parameter, namely γ, η, and p_{MIS}, which leads to multi-dimensional optimization. By starting with relatively high values for these parameters and reducing them individually until full notification could not be guaranteed with high probability we determined the following values for γ, η, and p_{MIS}: $\gamma = 5$, $\eta = 5$, and $p_{MIS} = 0.2$.

Using the above determined parameters, the respective performance of the three algorithms of Section 5.3 was evaluated by simulating their corresponding notification phase for different node densities and given a particular energy efficiency. The node nearest to the top-left corner was notified by an externally triggered event at the outset of the simulation, i.e., at the launching point. Notice that algorithm $\mathcal{A}_{birth}$, as described in [104], does not terminate after a fixed number of time-slots once a node is notified. We therefore executed $\mathcal{A}_{birth}$ without termination criterion and stopped the simulation series once all node were notified. On the other hand we assumed the deployment phase to be much longer than the notification phase. As a consequence we did not consider the initialization energy of a node since the maintenance energy becomes the dominant factor of energy consumption.

We found that the two newly proposed algorithms $\mathcal{A}_{uni}$ and $\mathcal{A}_{clu}$ outperform algorithm $\mathcal{A}_{birth}$ not only in the worst-case consideration as described in Section 5.3 but also in average case networks (cf. Figure 5.4). Figure 5.4(a) depicts the performance of the algorithms on networks with density 5. Algorithm $\mathcal{A}_{uni}$ is able to notify all nodes more than twice as fast than $\mathcal{A}_{birth}$ with high probability. $\mathcal{A}_{clu}$ lies roughly in the middle of the other two. If we consider a network density of 15, $\mathcal{A}_{clu}$ and $\mathcal{A}_{uni}$ need approximately the same number of time-slots to notify all nodes (cf. Figure 5.4(b)). That is,

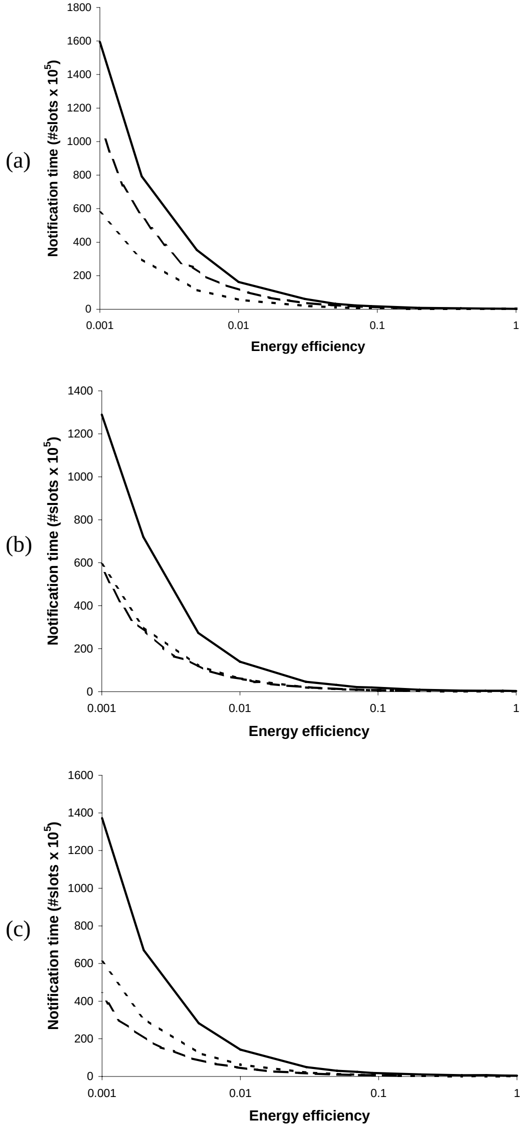

Figure 5.4: Mean values of the number of time-slots required to notify a network given a particular energy efficiency. The algorithms $\mathcal{A}_{birth}$ (solid), $\mathcal{A}_{uni}$ (dotted), and $\mathcal{A}_{clu}$ (dashed) are simulated at network densities 5 (a), 15 (b), and 30 (c).

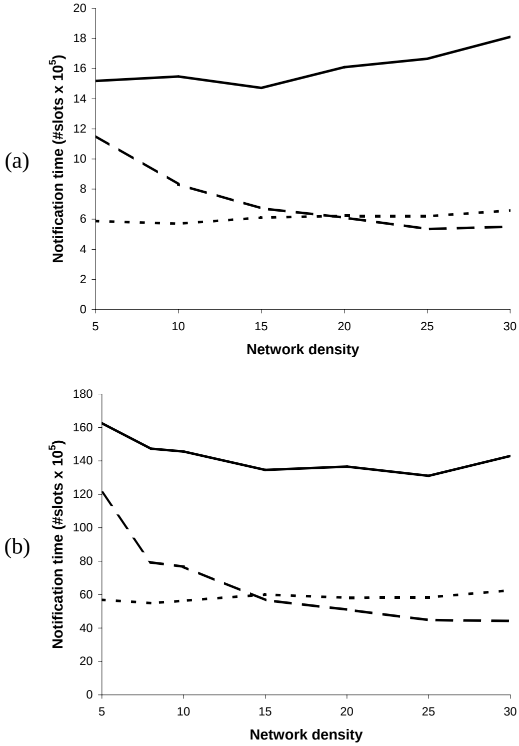

Figure 5.5: The number of time-slots required to notify all nodes for algorithm $\mathcal{A}_{birth}$ (solid), $\mathcal{A}_{uni}$ (dotted), and $\mathcal{A}_{clu}$ (dashed) for different network densities. The energy efficiency of the algorithms is thereby 0.1 (a) and 0.01 (b).

in comparison to the simulations at network density 5, $\mathcal{A}_{clu}$ is now able to notify the nodes faster with the same energy efficiency. This is due to the fact that with increasing density the ratio between leaders and non-leaders in $\mathcal{A}_{clu}$ decreases and non-leaders spend less energy during the deployment phase than nodes in algorithm $\mathcal{A}_{uni}$. This is exactly the same behavior that we analytically derived in Theorem 5.8. Considering network density 30, Figure 5.4(c) shows that the tide has turned in favor of $\mathcal{A}_{clu}$, which now outperforms the other two algorithms.

This results confirm the assumption that algorithm $\mathcal{A}_{uni}$ is well suited for low network densities while $\mathcal{A}_{clu}$ is dedicated for dense networks. To investigate the critical network density where $\mathcal{A}_{clu}$ starts to outperform $\mathcal{A}_{uni}$ more closely we simulated the algorithms for a fixed energy efficiency against increasing network densities ranging from 5 to 30 (cf. Figure 5.5). What strikes from Figure 5.5 is that the performance of algorithms $\mathcal{A}_{birth}$ and $\mathcal{A}_{uni}$ is more or less independent from the given network density. This is also shown in Figure 5.4 where the curves for both algorithms, $\mathcal{A}_{birth}$ and $\mathcal{A}_{uni}$, look approximately the same in all three plots. In contrast, the behavior of $\mathcal{A}_{clu}$ is quite different. Since the network density has a direct impact on the ratio of leader to non-leader nodes, its performance is highly dependent on the network density. Figure 5.5(a) depicts the performance of the three algorithms under the constraint of attaining an energy efficiency of 0.1. If the network density exceeds 19, algorithm $\mathcal{A}_{clu}$ needs less time-slots than $\mathcal{A}_{uni}$ to notify the entire network and should thus be preferred. If we require the algorithms to attain energy efficiency 0.01, $\mathcal{A}_{clu}$ outperforms $\mathcal{A}_{uni}$ already at density 15 (see Figure 5.5(b)).

These simulations complement the worst-case results derived in the previous section, showing that our algorithms $\mathcal{A}_{uni}$ and $\mathcal{A}_{clu}$ are also efficient in average-case scenarios. Moreover the simulations, in particular Figure 5.5, give a clear indication as to when the concept of clustering or the usage of semi-structures is worthwhile in the deployment process.

5.5 Related Work

To the best of our knowledge, the only previous work to explicitly address the problem of saving energy *during* the deployment of wireless ad hoc and sensor networks has been [104]. McGlynn and Borbash [104] propose an energy-saving method for performing adjacent neighbor discovery after the deployment of a network. Their algorithm is inspired by the well known *birthday paradox*. Using a similar idea to access the shared medium, a node randomly schedules its periodic wake-up to listen for incoming messages. The rest of the time the node powers down its radio subsystem to reduce energy consumption. In Section 5.3.1, we have given a succinct analysis of the birthday algorithm's performance in the context of the deployment problem. In [106], the authors present a probabilistic broadcast algorithm for wireless sensor networks. However, their approach relies on a previously established, functioning MAC layer and sleep scheduling mechanism. The paper therefore cannot be directly compared to our work.

In the literature on wireless sensor networks, various other problems in the context of *deployment* have been studied. Most notably, several papers have investigated problems related to the *placement* of nodes such that certain coverage requirements be fulfilled. The deployment of mobile nodes for coverage of a sensing field has been considered in [56, 147, 165]. In [9], the problem of covering and exploring an unknown dynamic environment using a mobile robot is addressed. An algorithm for this problem is presented that makes use of a deployed network of radio beacons which assists the robot in coverage. Other work in this area associated with the term *deployment* includes the placement of a given number of sensor nodes to reduce communication cost [72] or an optimal sensor placement for a given target distribution [117].

The model of computation used throughout this chapter was introduced in the domain of the *initialization* of wireless radio networks, and in particular ad hoc and sensor networks. Early works on radio networks can for example be found in [8, 69]. Most recently, fast algorithms for computing initial structures from scratch, based on which more sophisticated algorithm can subsequently be applied have been given in [81] and [108].

5.6 Concluding Remarks

The trade-off between energy-efficiency and the rapidity of event dissemination lies at the heart of wireless sensor network design. In this chapter, we have analyzed this key trade-off in the important non-operational phase by formalizing the problem of sensor network deployment, thus allowing a stringent analysis and comparison of different protocols.

Specifically, we have presented two algorithms, the first being entirely unstructured, the second using the idea of clustering. These algorithms can be regarded as archetypal representatives of an *unstructured* and *semi-structured* approach to the deployment problem, respectively. Interestingly, currently used standard MAC protocols such as B-MAC [119] or S-MAC [157] can be classified into these two approaches. Specifically, while the B-MAC approach is unstructured, S-MAC sets up some weak notion of clustering during the deployment, i.e., it uses a semi-structure.

Having a formal model that allows comparing these two schemes yields results that bear relevance to theoreticians and practitioners alike, because they give concise and sound answers to the question which deployment algorithm should be employed in a certain application scenario. Furthermore, notice that our results also shed new light on the intriguing question whether and in which cases *clustering* (as opposed to unstructured solutions that do not require any maintenance costs) is really worthwhile. This is of particular interest in view of the multiplicity of clustering algorithms proposed in the recent literature, e.g. [21, 29, 81, 82, 150].

Chapter 6

Differential Application Updates

In the previous chapter, we have studied different algorithms allowing sensor nodes to save energy during the deployment of a network. In this chapter, we address another issue arising during the lifetime of a sensor network which is orthogonal to the actual task of the network: The ability to reprogram the sensor network [50, 148]. Software updates are necessary for a variety of reasons. Iterative code updates on a real-world testbed during application development is critical to fix software bugs or for parameter tuning. Once a network is deployed the application may need to be reconfigured or even replaced in order to adapt to changing demands.

Once deployed, sensor nodes are expected to operate for an extended period of time. Direct intervention at individual nodes to install new software is at best cumbersome but may even be impossible if they are deployed in remote or hostile environments. Thus, network reprogramming must be realized by exploiting the network's own ability to disseminate information via wireless communication. Program code injected at a base station is required to be delivered to all nodes in its entirety. Intermediate nodes thereby act as relays to spread the software within the network. Given the comparatively small bandwidth of the wireless channel and the considerable amount of data to be distributed, classical flooding is prone to result in serious redundancy, contention, and collisions [143]. These problems prolong the update completion time, i.e. the time until all nodes in the network fully received the new software. Even worse, sensor nodes waste parts of their tight energy budgets on superfluous communication. Current code distribution protocols for sensor networks try to mitigate the broadcast storm problem by incorporating transmission suppression mechanisms or clever sender selection [137, 60, 85].

The radio subsystem is one of the major cost drivers in terms of energy consumption on current hardware platforms. Therefore, communication should be limited to a minimum during reprogramming in order not to reduce the lifetime of the network too much. Orthogonal to the above mentioned efforts the amount of data that is actually disseminated throughout the network should be minimized. Data compression seems to be an adequate answer to this problem. As knowledge about the application currently executed in the sensor network is present[1] differential compression, also known as delta compression, can be applied. Delta algorithms compress data by encoding one file in terms of another; in our case encoding the new application in terms of the one currently running on the nodes. Consequently, only the resulting delta file has to be transferred to the nodes which are then able to reconstruct the new application by means of their current version and the received delta. There exists a rich literature proposing a plethora of different algorithms for delta compression, e.g. [62, 142, 99, 141, 118, 1]. These algorithms excel at very large files. However, neither time nor space complexity is crucial considering the small code size of today's sensor network applications. It is much more important to account for the asymmetry of disposable computational power at the encoder and the decoder. While almost unlimited resources are available to generate the delta file on the host machine, special care has to be taken to meet the stringent hardware requirements when decoding on the nodes.

In this chapter we present an efficient code update mechanism for sensor networks based on differential compression. The delta algorithm is pursuing a greedy strategy resulting in minimal delta file sizes. The algorithm operates on binary data without any prior knowledge of the program code structure. This guarantees a generic solution independent of the applied hardware platform. We refrain from compressing the delta any further as this would exceed the resources available at the decoder. Furthermore, in contrast to other existing work we directly read from program memory to rebuild new code images instead of accessing flash memory which is slow and costly. The delta file is also structured to allow sequential access to persistent storage. All this leads to a lean decoder that allows fast and efficient program reconstruction at the sensor nodes.

Our work is tightly integrated into Deluge [60], the standard code dissemination protocol for the TinyOS platform. Deluge has proven to reliably propagate large objects in multi-hop sensor networks. Furthermore, it offers the possibility to store multiple program images and switch between them without continuous download. We support code updates for all program images even if they are not currently executed. Performance evaluations show that update size reductions in the range of 30% for major upgrades to 99% for small changes are achieved. This translates to a reprogramming speedup by a factor of about 1.4 and 100, respectively.

---

[1] This assumption is based on the fact that sensor networks are normally operated by a central authority.

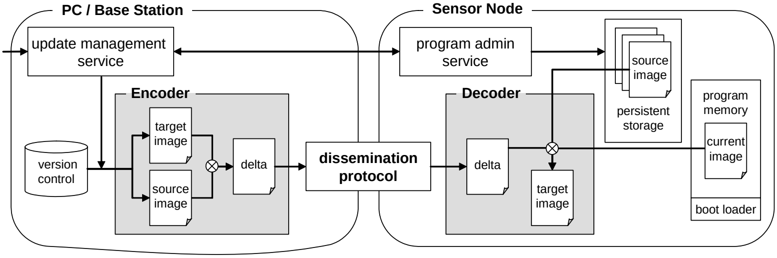

Figure 6.1: Components involved in the process of wireless reprogramming.

6.1 Overview

Updating code in wireless sensor network is a non-trivial task and requires the interaction of multiple system components. In general, application reprogramming can be broken down into three steps: image encoding, image distribution, and image decoding. Figure 6.1 shows a schematic view of all involved components and how they are interrelated in our code update mechanism. On the left-hand side, all services are consolidated that run on the host machine or the base station, respectively. On the right, the required components on a sensor node are depicted. The dissemination protocol is responsible for reliably distributing the encoded update in the entire sensor network. We make use of Deluge as it is widely accepted as the standard dissemination protocol and it has shown its robustness in various real-world deployments. We give a brief overview of Deluge's data management as it has direct implications on all other system components.[2]

Deluge enables a sensor node to store multiple application images. It divides the external flash memory (EEPROM) into slots, each of them large enough to hold one image. In conjunction with a bootloader Deluge is then able switch between these images. To manage the program image upload, Deluge divides images into pages of fixed size.[3] Deluge then transmits images at page granularity. That is, all packets of the last page in use were distributed no matter how many of them actually contain data of the new application image. The residual space of the last page is thereby filled with zero bytes. This overhead of up to one kilobyte might be of minor concern if the application image is transmitted in its entirety. However, it becomes unacceptable in the context of small changes leading to delta files of only a few bytes. Deluge was therefore adapted to just transmit packets containing vital information about the new image. The remaining bytes of the last page are then padded with zeros on the sensor node itself to enable a 16-bit

---

[2] The interested reader is referred to [60] for a detailed description of Deluge.

[3] In the current version of Deluge one page sums up to 1104 bytes. In turn, this results in 48 data packets per page.

cyclic redundancy check on the pages. By requiring a node to dedicate itself to receiving a single page at a time, it is able to keep track of missing packets using a fixed-size bit vector. Packets also include CRC checksums. Redundant data integrity checks at both packet and page level is critical as erroneous data is otherwise propagated throughout the whole network due to the epidemic nature of Deluge.

The protocol also incorporates an administration service that allows the base station to retrieve information about all stored images including which one is currently running. On the host machine, Deluge offers an update management service to inject new images into the network. To allow differential updates a version control system is required at the host machine in order to know all application images currently residing on the sensor nodes. In the current version a file-system based image repository is used to archive the latest program versions stored in each slot on the sensor nodes. If a new target image is supposed to be injected to a given slot, the update manager first queries the nodes to retrieve metadata about all loaded images. Based on this information a crosscheck in the version control system is performed to ensure that the latest image version for the requested slot is present in the repository. Once the validity of the source image in the repository is verified it is used as input for the delta encoder along with the target image. The encoder processes both images and generates the corresponding delta file. The delta is then disseminated using Deluge as if it was a normal application image. However, it is not stored in the designated slot of the target image but in an additional EEPROM slot reserved for delta files. Upon complete delta reception, a node starts the decoding process using additional information from external flash memory and program memory. The target image is thereby directly reconstructed in its intended EEPROM slot. In the meantime, the delta is further disseminated within the network. We now give a detailed description of the encoding algorithm employed on the host machine as well as of the decoder that resides on the sensor nodes.

6.2 Update Mechanism

All delta algorithms introduced in Section 6.4 use some kind of heuristic to speed up the generation of copy commands and consequently to reduce the overall execution time of the encoder. In [126] a greedy algorithm is presented that optimally solves the string-to-string correction problem which lies at the heart of differential updating. While its time complexity is undesirable for very large input files it poses no problem in the context of sensor networks where program size is limited to a few hundred kilobytes.[4] Hence, the design

---

[4] The maximal application memory footprint of state-of-the-art sensor network hardware is limited to 48kB for nodes equipped with MSP430 microcontrollers or 128kB for ATmega128 platforms, respectively.

Instruction	Code	Arguments	Cost [bytes]
`shift`	xxxxx100	none	1
`run`	xxxxx101	byte to be repeated	2
`copy`	xxxxx110	start address	3
`add`	xxxxx111	data to be added	1+#bytes

Table 6.1: Instruction codes, arguments, and overall costs in bytes if the instructions reconstruct less than 32 bytes. The length of the instruction is encoded in the first five bits of the instruction code.

of our delta encoder is based on the findings in [126]. Before we give a detailed description of the encoder itself we specify the employed instruction set and how instructions are arranged in the delta file.

6.2.1 Delta Instructions and Delta File Organization

We adopt the set of delta instructions specified in VCDIFF [75] which is a portable data format for encoding differential data. It is proposed to decouple encoder and decoder implementations to enable interoperability between different protocol implementations. It distinguishes three types of instructions: `add`, `copy` and `run`. The first two instructions are straightforward; `add` appends a number of given bytes to the target image and `copy` points to a section in the source image to be copied to the target image. The `run` instruction is used to encode consecutive occurrences of the same byte efficiently. It has two arguments, the value of the byte and the number of times it is repeated. Making use of the fact that the target image is decoded into the same slot in external memory where the source image resides, we introduce a fourth instruction. The `shift` instruction is used to encode sections of the image that have not changed at all from one version to the next. It is used to prevent unnecessary EEPROM writes. The only effect of a `shift` instruction is the adjustment of the pointer to the target image at the decoder.

These instructions are designed to minimize the overhead of the delta file. Each instruction code has a size of one to three bytes dependent on the number of bytes in the target image the corresponding instruction encodes. Table 6.1 comprises the arguments and costs of all four instruction types if they reconstruct less than 32 bytes of the target image. The actual length is directly encoded in the first 5 bits of the instruction code in this case. The cost of an instruction increases by one if the encoded fragment spans up to 255 bytes, or by two if it is larger than that, as the instruction length occupies one or two additional bytes, respectively.

We refrained from using the same delta file organization as proposed in VCDIFF. It splits the file in three sections, one for data to be added, one for the actual instructions, and one for the addresses of the `copy` instructions. This enables better secondary compression of the delta file. As we try to keep the decoder complexity to a minimum to meet the nodes' hardware limitations no such compression is applied. We could still use the VCDIFF format without secondary compression. However, the fact that an instruction has to gather its arguments from different places within the delta file results in unfavorable EEPROM access patterns. Random access to external memory—as it would be the case if the VCDIFF format was employed—results in increased overhead during the decoding process. This is caused by the discrepancy between the small average delta instructions and the rather coarse-grained EEPROM organization. On average, the delta instruction length is below four bytes for all experiments described in Section 6.3. In contrast, external memory access is granted at a page granularity with page sizes of 256 bytes for flash chips of modern sensor network hardware. As EEPROM writes are expensive (see Table 6.2) current flash storage incorporates a limited amount of cached memory pages to mitigate the impact of costly write operations.[5] However, it is important to notice that even though a read itself is cheap, it may force a dirty cache page to be written back to EEPROM which renders a read operation as expensive as a write.

To allow for the above mentioned EEPROM characteristics the delta file is organized by appending instructions in the order of their generation. That is, each instruction code is directly followed by its corresponding arguments. Furthermore, all instructions are ordered from left to right according to the sections they are encoding in the target file. This permits a continuous memory access during the execution of the delta instructions.

6.2.2 Delta Encoder

The severe hardware constraints of wireless sensor networks let our delta encoder differ in various points from common delta compression algorithms to optimize the decoding process. Besides the objective to minimize the delta file size one also has to consider the energy spent on reconstructing the new image at the nodes. In particular, special care has to be taken to optimize external flash memory access.

The only instruction that requires additional information from the sources image to reconstruct its section of the target image is `copy`. To avoid alternating read and write requests between source image and delta file potentially causing the above discussed EEPROM cache thrashing problem we derive the data required by `copy` instructions directly from program memory. This decision has several implications. Most important, copies must be generated based on the currently executed image even if it is not identical to the image we would like to update. That is, the decoder is actually using a third

---

[5] The Atmel AT45DB041B flash chip on the TinyNode platform has a cache size of two pages.

Operation	Current Draw	Time	Rel. Power Drain
Receive a packet	14 mA	5 ms	1
Send a packet	33 mA	5 ms	2.36
Read EEPROM page	4 mA	0.3 ms	0.017
Write EEPROM page	15 mA	20 ms	4.29

Table 6.2: Relative energy consumption of different operations in comparison to a packet reception for the TinyNode platform.

input file, namely the currently executed image, to reconstruct the target image. Second, decoding is sped up since reading from program memory is fast in comparison to accessing external flash memory. Third, we are able to directly overwrite the source image in external memory without wasting an additional slot during the reconstruction process. This renders `shift` instructions possible. As a drawback, it is no longer allowed to use an already decoded section of the target image as origin for later copies. This potentially results in larger delta files. However, the aforementioned positive effects compensate this restriction.

In a first phase, the encoder analyzes both source and target image. The algorithm runs simultaneous over both input files and generates `shift` instructions for each byte sequence that remains unchanged. Then, the target image is inspected and `run` instructions are produced for consecutive bytes with identical values. In a third pass, for each byte in the target image a search for the longest common subsequence in the source image is performed. A `copy` is then generated for each byte with a matching sequence of size at least three as `copy` instructions of length three or larger start to pay off compared to an `add`.

In a second phase a sweep line algorithm is employed to determine the optimal instruction set for the target image minimizing the size of the resulting delta. All instructions produced in the first phase reconstruct a certain section of the target image determined by their start and end address. The algorithm processes the image from left to right and greedily picks the leftmost instruction based on its start addresses. Then, the next instruction is recursively chosen according to the following rules. First, the instruction must either overlap with or be adjacent to the current instruction. Second, we choose the instruction among those fulfilling the previous requirement whose endpoint is farthest to the right. The instruction costs are used for tie breaking. To avoid redundancy in the delta file the new instruction is pruned to start right after the end of the current one if they overlap. If no instruction satisfies these demands an `add` is generated. These `add` instructions span the sections not covered by any of the other three instruction types. Once the algorithm reaches the end of the target image the delta file is generated according to the rules stated in the previous section.

6.2.3 Delta Decoder

The delta decoder is mapped as a simple state machine executing delta instruction in rotation. Prior to the actual decoding metadata is read from the head of the delta file. This information is appended by the encoder and contains the delta file length and additional metadata for the target image required by Deluge. The length is used to determine completion of the decoding. The metadata comprises version and slot information of the target image. This information is used to verify the applicability of the delta to the image in the given slot. In case of failure the decoding process is aborted and the image is updated traditionally without the help of differential reprogramming. If the delta is valid, the instructions are consecutively executed to rebuild the target image.

Once the decoder has fully reconstructed the image it signals Deluge to pause the advertisement process for the newly built image. This has the effect that the delta is disseminated faster within the network than the actual image enabling all nodes to reprogram themselves by means of the delta.

6.3 Experimental Evaluation

In this section we analyze the performance of our differential update mechanism on real sensor network hardware. As in section Section 4.3, we run the experiments on TinyNode sensor nodes operating TinyOS. To prove the fitness of the proposed approach in a wide range of application scenarios five different test cases are consulted ranging from small code updates to complete application exchanges. Except one, all applications are part of the standard TinyOS distribution. Before evaluating the performance of our reprogramming approach the different test settings are discussed in the following.

- Case 1: This case mimics micro updates like they occur during parameter tuning. We increase the rate at which the LED of the `Blink` application is toggled. This change of a constant has only local impact and should therefore result in a small delta.

- Case 2: The `Blink` application is modified to facilitate concurrent program executions. The application logic is therefore encapsulated in an independent task (see `BlinkTask`). This leads to additional calls to the TinyOS scheduler and a deferred function invocation.

- Case 3: The `CntToLeds` application, which shows a binary countdown on the LED's, is extended to simultaneously broadcast the displayed value over the radio (`CntToLedsAndRfm`). This is a typical example of a software upgrade that integrates additional functionality.

- Case 4: In this setting `Blink` is replaced with the `Oscilloscope` application. The latter incorporates multi-hop routing to convey sensor

Case	Target Size	Delta Size	Size Reduction	Encoding Time
1	28684	322	98.88%	8453 ms
2	27833	5543	80.08%	5812 ms
3	28109	7383	73.73%	6797 ms
4	34733	17618	49.28%	7563 ms
5	21508	14904	30.70%	4781 ms

Table 6.3: Target and delta sizes in bytes for the different settings. Additionally, times required to encode the images are given.

readings towards a base station. Both application share a common set of system components. This scenario highlights the ability of our protocol to cope with major software changes.

- Case 5: Here we switch from `Blink` to `Dozer` [17], an energy-efficient data gathering system. Among other features, Dozer employs a customized network stack such that the commonalities between the two applications are minimal. `Dozer` is the only application in this evaluation that has no built-in Deluge support.

Using the original Deluge, `Blink` produces a memory footprint of 24.8 kB in program memory and 824 bytes RAM. In comparison, the enhanced Deluge version including the delta decoder sums up to 27.6 kB ROM and 958 bytes of RAM. Consequently, our modifications increase the memory footprint of an application by 2.8 kB in program memory and 134 bytes of RAM.

The performance of our delta encoder for all five cases is shown in Table 6.3. For Case 1, the encoder achieves a size reduction by a factor of 100. Actually, the mere delta is only 32 bytes long. The other 290 bytes consist of metadata overhead introduced by Deluge such as 256 bytes of CRC checksums. As mentioned in the case descriptions, the increasing delta sizes indicate that the similarity between source and target image decreases from Case 1 to 5. The delta file produced due to major software changes is still only about half the size of the original image. Furthermore, even if we replace an application with one that has hardly anything in common with the former, such as in Case 5, the encoder achieves a size reduction of about 30%. Thus, our application update mechanism reduces the power consumption of the code dissemination by the same percentage as it compresses the input data since the energy spent for the image distribution is directly proportional to the transmitted amount of data. Table 6.3 also contains the execution times of the encoder for the five different scenarios. Note that the encoding was computed on a customary personal computer. The encoding process for the considered settings takes up to nine seconds. Compared to the code distribution speedup achieved by smaller delta files this execution time is negligible.

Table 6.4 shows the number of occurrences of all four instruction types in all five settings. Furthermore, it contains the average number of bytes covered by one instruction. One can see that the modifications in Case 1 are purely local as only 14 bytes have to be overwritten and the rest of the image stays untouched. For the other four scenarios, `copy` and `add` instructions constitute the dominating part of the delta files. It is interesting to see that the average size of a `copy` is larger if the source and target images have a higher similarity. The opposite is true for the `add` instruction. In the case of minor application updates many code blocks are only shifted to a different position within the code image but not changed at all. This fact is exploited by the `copy` instructions enabling a relocation of these sections with constant overhead. However, if the new application is completely unrelated to the one to be replaced as in Case 5, the image exhibits less opportunities for copies. Consequently, more `add` instructions are necessary to rebuild the target image.

To evaluate the decoder we measure the reconstruction time of the target image on a sensor node. Table 6.5 shows the decoding time for all cases dependent on the available buffer size at the decoder. If no input buffer is available, each delta instruction is read separately from external memory before it is processed. If an input buffer is allocated, the decoder consecutively loads data blocks of the delta file from EEPROM into this buffer. Before a new block of data is fetched, all instructions currently located in the buffer are executed. Similar to the input buffer handling, the decoder writes the result of a decode instruction directly to external memory if no output buffer is present. In contrast, if an output buffer is available, it is filled with the outcomes of the processed delta instructions and only wrote back to EEPROM if it is full. We limit the maximum buffer size to 256 bytes thereby matching the EEPROM page size of the TinyNode platform.

In the first scenario, the decoding process takes approximately 1.2 seconds no matter which buffer strategy is applied. This can be explained by the fact that only 10 delta instructions are involved (see Table 6.4) whereas five of them are `shift` instructions which do not lead to EEPROM writes. In contrast, the decoder takes about 15 seconds to update from `Blink` to the `Oscilloscope` application if neither input nor output buffers are used. However, decoding time decreases to 8.68 or 5.63 seconds if an input buffer of 256 bytes or both, input and output buffers of size 256 bytes are employed, respectively. That is, decoding with maximum input and output buffers reduces the execution time by a factor of 2.7 in Case 4.

Due to the promising results with large buffer sizes, we also study the impact of varying buffer sizes on the decoding speed. The reconstruction time for all scenarios was evaluated for buffer sizes of 16, 32, 64, 128, and

Setting	#`shift`	avg. size	#`run`	avg. size	#`copy`	avg. size	#`add`	avg. size
Case 1	5	57334	0	0	0	0	5	2.80
Case 2	16	79.19	3	244	884	27.43	859	1.83
Case 3	71	26.01	5	91	1183	20.13	1153	1.72
Case 4	30	37.13	12	40.32	2757	9.68	2472	2.64
Case 5	25	7.78	20	54.70	2219	6.44	1858	3.19

Table 6.4: The number of occurrences of each instruction type for all scenarios. Furthermore, the average number of bytes encoded by one instruction is given.

Setting	none | none	256 | none	256 | 256	128 | 128	64 | 64	32 | 32	16 | 16
Case 1	1.20	1.24	1.24	1.24	1.24	1.23	1.23
Case 2	7.03	5.05	4.11	4.20	4.38	4.74	5.40
Case 3	8.51	5.52	4.25	4.36	4.54	5.02	5.74
Case 4	15.14	8.68	5.63	5.79	6.06	6.82	7.84
Case 5	11.27	6.47	4.08	4.15	4.35	4.74	5.45

Table 6.5: Time in seconds to reconstruct the target image on a sensor node as a function of the available input and output buffer sizes in bytes at the decoder (input | output).

256 bytes. Table 6.5 shows that the execution times increase with decreasing buffer sizes. However, the increases are moderate: Reducing the buffers from 256 to 16 bytes and thus saving 480 bytes of RAM results in an at most 40% longer decoding time. Furthermore, the execution times with buffers of 16 bytes are roughly the same as if an input buffer of 256 bytes only is employed.

6.4 Related Work

The earliest reprogramming systems in the domain of wireless sensor networks, e.g. XNP [28], did not spread the code within the network but required the nodes to be in transmission range of the base station in order to get the update. This drawback was eliminated by the appearance of MOAP [137] which provides a multi-hop code dissemination protocol. It uses a publish-subscribe based mechanism to prevent saturation of the wireless channel and a sliding window protocol to keep track of missing information.

Deluge [60] and MNP [85] share many ideas as they propagate program code in an epidemic fashion while regulating excess traffic. Both divide a code image into equally sized pages, pipelining the transfer of pages and thus making use of spatial multiplexing. A bit vector is used to detect packet loss within a page. Data is transmitted using an advertise-request-data handshake. Deluge uses techniques such as a sender suppression mechanism borrowed from SRM [38] to be scalable even in high-density networks. In contrast, MNP aims at choosing senders that cover the maximum number of nodes requesting data. There have been various proposals based on the above-mentioned protocols, e.g. [132, 115], that try to speed up program dissemination. However, all these approaches share the fact that the application image is transmitted in its entirety. This potentially induces a large amount of overhead in terms of sent messages but also in terms of incurred latency.

There have been efforts to update applications using software patches outside the sensor network community in the context of differential compression, an encoding technique that arose from the string-to-string correlation problem [146]. Delta compression is concerned with compressing one data set, referred to as the *target image*, in terms of another one, called the *source image*, by computing a *delta*. The main idea is to represent the target image as a combination of copies from the source image and the part of the target image that is already compressed. Sections that cannot be reconstructed by copying are simply added to the delta file. Examples of such delta encoders include *vdelta* [62], *xdelta* [99], and *zdelta* [141]. They incorporate sophisticated heuristics to narrow down the solution space at the prize of decreased memory and time complexity as it is important to perform well on very large input files. However, these heuristics result in suboptimal compression. The *zdelta* algorithm further encodes the delta file using Huffman coding. This raises the decoder complexity to a level which does not match the constraints of current sensor network hardware. In [76], the *xdelta* algorithm is used to demonstrate the efficiency of incremental linking in the domain of sensor networks. However, the authors do not give a fully functional network reprogramming implementation but use the freely available *xdelta* encoder to evaluate the fitness of their solution. There exists other work in the domain of compilers and linkers trying to generate and layout the code such that the new image is as similar as possible to a previous image. Update-conscious compilation is addressed in [91] where careful register and data allocation strategies lead to significantly smaller difference files. In [145] incremental linkers are presented that optimize the object code layout to minimize image differences. All these approaches are orthogonal to our work and can be integrated to further increase the overall system performance. We refrain from including one of them since they are processor specific and thus do not allow a generic solution.

Similar to the above mentioned delta algorithms, *bsdiff* [118] also encodes the target image by means of copy and insert operations. The algorithm does not search for perfect matches to copy from but rather generates approximate matches where more than a given percentage of bytes are identical. The differences inside a match are then corrected using small insert instructions. The idea is that these matches roughly correspond to sections of unmodified source code and the small discrepancies are caused by address shifts and different register allocation. Delta files produced by *bsdiff* can be larger than the target image but are highly compressible. Therefore, a secondary compression algorithm is used (in the current version *bzip2*) which makes the algorithm hardly applicable for sensor networks. The authors of [127] propose an approach similar to *bsdiff*. Their algorithm also produces non-perfect matches which are corrected using repair and patch operations. The patch operations work at the instruction level[6] to recognize opcodes having addresses as arguments which must be moved by an offset given by the patch operation. The algorithm shows promising results but depends on the instruction set of a specific processor.

In [142] *rsync* is presented that efficiently synchronizes binary files in a network with low-bandwidth communication links. It addresses the problem by using two-stage fingerprint comparison of fixed blocks based on hashing. An adaptation to *rsync* in the realm of sensor networks is shown in [65]. As both the source image and the target image reside on the same machine, various improvements can be introduced. The protocol is integrated into XNP. Besides the fact that XNP does only allow single-hop updates, the protocol does not overcome the limitations of *rsync* and performs well only if the differences in the input files are small.

FlexCup [102] exploits the component-based programming abstraction of TinyOS to shift from a monolithic to a modular execution environment.

---

[6] Their work is based on the MSP430 instruction set.

Instead of building a single application image, FlexCup produces separate object files which are then linked to an executable on the sensor node itself. Thus, code changes are handled at the granularity of TinyOS components. This solution does no longer allow global optimizations. Furthermore, since the linking process requires all the memory available at the sensor node, FlexCup is not able to run in parallel to the actual application. In [31], dynamic runtime linking for the Contiki operating system [32] is presented.

Besides TinyOS, there exist other operating systems for sensor networks which are inherently designed to provide a modular environment [32, 49]. They provide support for dynamic loading of applications and system services as a core functionality of the system. However, this flexibility implies additional levels of indirection for function calls which add considerable runtime overhead. The update process for changed components is limited to these components as they are relocatable and address independent.

Virtual machine architectures for sensor networks [88, 89, 7] push the level of indirection one step further. They conceal the underlying hardware to offer high-level operations to applications through an instruction interpreter. Updates are no longer native code but normally considerable smaller application scripts. This renders reprogramming highly efficient. However, the execution overhead of a virtual machine is considerable and outweighs this advantage for long-running applications [88, 31].

The authors of [39] investigate network reprogramming with the assistance of a mobile robot. The robot thereby first profiles a sensor node to determine the current software version. In a second step the robot compiles a new image which is then transmitted to the node.

A temporary alternative to supply in-network programmability inside the sensor network itself is to provide a parallel maintenance network [33]. This is particularly useful during the development process as one does not have to rely on the network being operational to update it. Furthermore, new protocols can be tested and evaluated without the reprogramming service distorting results.

Chapter 7

Conclusion

Wireless sensor networks are about to shift from the laboratory to the environment. It has become apparent in these deployments that data gathering is the prevalent application for sensor networks. Even if a sensor network is not intended to provide continuous information about the monitored environment, an operator still wants to be kept informed about the network state to guarantee the required quality of service. As the lifetime of wireless sensor networks is generally determined by their ability to economize on the available energy resources, it is of prime importance to develop energy-efficient data gathering techniques.

In this dissertation, we studied different strategies to decrease energy consumption of wireless sensor networks in general and data gathering applications in particular. While some problem fields were considered from a theoretical point of view, others were addressed by giving practical implementations on real sensor network hardware. With the often discussed gap between theory and practice in mind it may seem unorthodox to incorporate techniques, methods, and solutions from both sides in one thesis. Looking back however, it was the right decision as it led to a fruitful interaction between these two disciplines. On the one hand, we gained valuable insight into the limitations of sensor nodes and the "real" problems in practical networks by delving into system software and hardware specifications of currently available sensor network platforms. These findings often led to intriguing theoretical questions. The deployment problem studied in Chapter 5 for example arose from difficulties we faced during a testbed installation for the Dozer system described in Chapter 4. On the other hand, theoretical analysis often supported the practical protocol design process.

This deep involvement with the peculiarities of practical sensor networks gave also rise to a better understanding of which network model abstractions are allowed without impairing the significance of theoretical results. In this respect, it becomes apparent that the interference models of Chapter 2 are too simplistic. However, these initial considerations opened up a new field full of fascinating questions forming the basis for ongoing research. Recent work even starts to replace these oversimplified interference models by the realistic signal-to-interference-plus-noise model. The harsh physical conditions of real wireless sensor network is closely captured by the unstructured radio network model in Chapter 5. Yet, it still allows stringent mathematical analysis of the proposed algorithms. The assumption that sensor nodes do not feature a reliable collision detection mechanism makes sense from a worst-case perspective. Nevertheless, in practical networks nodes are often able to detect most occurring collisions. This opens the algorithmically interesting question of how much does collision detection help in general. Are we able to devise a strictly better algorithm for the deployment problem than those proposed in Chapter 5?

From a practical point of view, this dissertation demonstrates that in real-world deployments, multi-hop communication and lifetimes of several years are not mutually exclusive and that the vision of a reliable long-term data gathering networks is within reach. The Dozer network stack facilitates the collection of information acquired by individual sensor nodes consuming only a small amount of energy. So far, data is continuously trickling towards the information sink. However, some applications, e.g. surveillance, control, or emergency response systems, require more reactivity. Critical events must be handled almost instantly resulting in a trade-off between network latency and energy efficiency. Future work will have to show if Dozer can be extended to reconcile the conflicting demands of low latency, high message yield, and low power consumption or if custom-tailored systems optimized for individual application demands are unavoidable.

Bibliography

[1] R. C. Agarwal, K. Gupta, S. Jain, and S. Amalapurapu. An approximation to the greedy algorithm for differential compression. *IBM Journal of Research and Development*, 50(1):149–166, 2006.

[2] K. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):408–421, 2003.

[3] K. Alzoubi, P.-J. Wan, and O. Frieder. Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks. In *Proc. of the 3^{rd} International Symposium on Mobile Ad-Hoc Networking and Computing (MOBIHOC)*, pages 157–164, 2002.

[4] A. Arora, P. Sinha, E. Ertin, V. Naik, H. Zhang, M. Sridharan, and S. Bapat. ExScal Backbone Network Architecture. In *Proc. of the 3^{rd} International Conference on Mobile Systems, Applications, and Services (MOBISYS)*, 2005.

[5] B. Awerbuch, A. Baratz, and D. Peleg. Cost-Sensitive Analysis of Communication Protocols. In *Proc. of the 9^{th} Symposium on Principles of Distributed Computing (PODC)*, pages 177–187, 1990.

[6] M. Bahramgiri, M. Hajiaghayi, and V. S. Mirrokni. Fault-Tolerant and 3-Dimensional Distributed Topology Control Algorithms in Wireless Multi-Hop Networks. *Wireless Networks*, 12(2):179–188, 2006.

[7] R. Balani, C.-C. Han, R. K. Rengaswamy, I. Tsigkogiannis, and M. Srivastava. Multi-Level Software Reconfiguration for Sensor Networks. In *Proc. of the 6^{th} International Conference on Embedded Software (EMSOFT)*, pages 112–121, 2006.

[8] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the Time-Complexity of Broadcast in Radio Networks: an Exponential Gap between Determinism and Randomization. In *Proc. of the 6^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 98–108, 1987.

[9] M. A. Batalin and G. S. Sukhatme. Coverage, Exploration, and Deployment by a Mobile Robot and Communication Network. In *Proc. of the 2^{nd} International Workshop on Information Processing in Sensor Networks (IPSN)*, pages 376–391, 2003.

[10] K. Bharath-Kumar and J. M. Jaffe. Routing to Multiple Destinations in Computer Networks. *IEEE Transactions on Communications*, 31(3):343–351, 1983.

[11] D. Bilò and G. Proietti. On the Complexity of Minimizing Interference in Ad-Hoc and Sensor Networks. In *Proc. of the 4^{th} International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 13–24, 2006.

[12] D. Blough, M. Leoncini, G. Resta, and P. Santi. The K-Neigh Protocol for Symmetric Topology in Ad Hoc Networks. In *Proc. of the 4^{th} International Symposium on Mobile Ad-Hoc Networking and Computing (MOBIHOC)*, pages 141–152, 2003.

[13] F. Bock. An algorithm to construct a minimum spanning tree in a directed network. *Developments in Operations Research*, 1:29–44, 1971.

[14] S. Borbash and E. Jennings. Distributed Topology Control Algorithm for Multihop Wireless Networks. In *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, pages 355–360, 2002.

[15] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. In *Proc. of the 3^{rd} International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 48–55, 1999.

[16] T. Brooke and J. Burrell. From Ethnography to Design in a Vineyard. In *Proc. of the Conference on Designing for user experiences (DUX)*, pages 1–4, 2003.

[17] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In *Proc. of the 6^{th} International Conference on Information Processing in Sensor Networks (IPSN)*, pages 450–459, 2007.

[18] N. Burri, P. von Rickenbach, R. Wattenhofer, and Y. Weber. Topology Control Made Practical: Increasing the Performance of Source Routing. In *Proc. of the 2^{nd} International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pages 1–12, 2006.

[19] P. M. Camerini. A note on finding optimum branchings. *Networks*, 9:309–319, 1979.

[20] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer. A Reactive Soil Moisture Sensor Network: Design and Field Evaluation. *Journal of Distributed Sensor Networks*, 1(2):149–162, 2005.

[21] M. Chatterjee, S. K. Das, and D. Turgut. WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. *Cluster Computing Journal*, 5(2):193–204, 2002.

[22] J. Chou, D. Petrovic, and K. Ramchandran. A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Comunications Societies (INFOCOM)*, pages 1054–1062, 2003.

[23] Y. J. Chu and T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

[24] V. Chvátal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[25] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit Disk Graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

[26] W. S. Conner, J. Chhabra, M. Yarvis, and L. Krishnamurthy. Experimental Evaluation of Topology Control and Synchronization for In-Building Sensor Network Applications. *Mobile Networks and Applications*, 10(4):545–562, 2005.

[27] R. Cristescu, B. Beferull-Lonzano, and M. Vetterli. On Network Correlated Data Gathering. In *Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Comunications Societies (INFOCOM)*, pages 2571–2582, 2004.

[28] Crossbow Technology Inc. Mote In-Network Programming User Reference. `http://www.xbow.com`, 2003.

[29] F. Dai and J. Wu. On Constructing *k*-Connected *k*-Dominating Set in Wireless Networks. In *Proc. of the 19th International Parallel and Distributed Processing Symposium (IPDPS)*, 2005.

[30] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[31] A. Dunkels, N. Finne, J. Eriksson, and T. Voigt. Run-Time Dynamic Linking for Reprogramming Wireless Sensor Networks. In *Proc. of the 3rd International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 15–28, 2006.

[32] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proc. of the 29th International Conference on Local Computer Networks (LCN)*, pages 455–462, 2004.

[33] M. Dyer, J. Beutel, L. Thiele, T. Kalt, P. Oehen, K. Martin, and P. Blum. Deployment Support Network - A Toolkit for the Development of WSNs. In *Proc. of the 4th European Conference on Wireless Sensor Networks (EWSN)*, pages 195–211, 2007.

[34] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240, 1967.

[35] A. El-Hoiydi and J.-D. Decotignie. WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks. In *Proc. of the 2nd International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 18–31, 2004.

[36] G. Even, Z. Lotker, D. Ron, and S. Smorodinsky. Conflict-Free Colorings of Simple Geometric Regions with Applications to Frequency Assignment in Cellular Networks. In *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–136, 2002.

[37] U. Feige. A Threshold of ln n for Approximating Set Cover. *Journal of the ACM*, 45(4):634–652, 1998.

[38] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. *IEEE Transactions on Networking*, 5(6):784–803, 1997.

[39] G. Fuchs, S. Truchat, and F. Dressler. Distributed Software Management in Sensor Networks using Profiling Techniques. In *Proc. of the 1st International Conference on Communication System Software and Middleware (COMSWARE)*, pages 1–6, 2006.

[40] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. Technical Report UCLA/CSD-TR 02-001, University of California, Los Angeles, 2002.

[41] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete Mobile Centers. In *Proc. of the 17th Annual Symposium on Computational Geometry (SCG)*, pages 188–196, 2001.

[42] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric Spanner for Routing in Mobile Networks. In *Proc. of the 2nd International Symposium on Mobile Ad-Hoc Networking and Computing (MOBIHOC)*, pages 45–55, 2001.

[43] A. Ghosh, Y. Wang, and B. Krishnamachari. Efficient Distributed Topology Control in 3-Dimensional Wireless Networks. In *Proc. of the 4nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 91–100, 2007.

[44] O. Gnawali, B. Greenstein, K.-Y. Jang, A. Joki, J. Paek, M. Vieira, D. Estrin, R. Govindan, and E. Kohler. The Tenet Architecture for Tiered Sensor Networks. In *Proc. of the 4th International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 153–166, 2006.

[45] A. Goel and D. Estrin. Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk. In *Proc. of the 14th Symposium on Discrete Algorithms (SODA)*, pages 499–505, 2003.

[46] B. Greenstein, C. Mar, A. Pesterev, S. Farshchi, E. Kohler, J. Judy, and D. Estrin. Capturing High-Frequency Phenomena Using a Bandwidth-Limited Sensor Network. In *Proc. of the 4th International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 279–292, 2006.

[47] H. Dubois-Ferrier and R. Meier and L. Fabre and P. Metrailler. TinyNode: a comprehensive platform for wireless sensor network applications. In *Proc. of the 5th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 358–365, 2006.

[48] M. Halldórsson and T. Tokuyama. Minimizing Interference of a Wireless Ad-Hoc Network in a Plane. In *Proc. of the 4th International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 71–82, 2006.

[49] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A Dynamic Operating System for Sensor Nodes. In *Proc. of the 3rd International Conference on Mobile Systems, Applications, and Services (MOBISYS)*, pages 163–176, 2005.

[50] C.-C. Han, R. Kumar, R. Shea, and M. Srivastava. Sensor network software update management: a survey. *International Journal of Network Management*, 15(4):283–294, 2005.

[51] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proc. of the 33rd Hawaii International Conference on System Sciences (HICSS)*, page 8020, 2000.

[52] J. Hill and D. Culler. Mica: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro*, 22(6):12–24, 2002.

[53] B. Hohlt and E. Brewer. Network Power Scheduling for TinyOS Applications. In *Proc. of the 2nd International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 443–462, 2006.

[54] B. Hohlt, L. Doherty, and E. Brewer. Flexible Power Scheduling for Sensor Networks. In *Proc. of the 3rd International Conference on Information Processing in Sensor Networks (IPSN)*, pages 205–214, 2004.

[55] T. Hou and V. Li. Transmission Range Control in Multihop Packet Radio Networks. *IEEE Transactions on Communications*, 34(1):38–44, 1986.

[56] A. Howard, M. J. Matarić, and G. S. Sukhatme. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In *Proc. of the 6th International Conference on Distributed Autonomous Robotic Systems (DARS)*, 2002.

[57] L. Hu. Topology Control for Multihop Packet Radio Networks. *IEEE Transactions on Communications*, 41(10):1474–1481, 1993.

[58] H. Huang, A. W. Richa, and M. Segal. Approximation Algorithms for the Mobile Piercing Set Problem with Applications to Clustering in Adhoc Networks. In *Proc. of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 52–61, 2002.

[59] P.-H. Huang and B. Krishnamachari. Analysis of Existing Approaches and a New Hybrid Strategy for Synchronization in Sensor Networks. In *Proc. of the 3rd IEEE Workshop on Embedded Networked Sensors (EMNETS)*, 2006.

[60] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 777–789, 2004.

[61] P. A. Humblet. A distributed algorithm for minimum weighted directed spanning trees. *IEEE Transactions on Communications*, 31(6):756–762, 1983.

[62] J. J. Hunt, K.-P. Vo, and W. F. Tichy. Delta Algorithms: An Empirical Analysis. *ACM Transactions on Software Engineering and Methodology*, 7(2):192–214, 1998.

[63] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. of the* 6^{th} *Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 56–67, 2000.

[64] J. Janssen. Channel Assignment and Graph Labeling. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, chapter 5, pages 95–117. John Wiley & Sons, Inc., 2002.

[65] J. Jeong and D. Culler. Incremental Network Programming for Wireless Sensors. In *Proc. of the* 1^{st} *International Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 25–33, 2004.

[66] L. Jia, R. Rajaraman, and R. Suel. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. In *Proc. of the* 20^{th} *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 33–42, 2001.

[67] T. Johansson and L. Carr-Motyčková. Reducing Interference in Ad Hoc Networks through Topology Control. In *Proc. of the* 3^{rd} *ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 17–23, 2005.

[68] D. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[69] T. Jurdziński and G. Stachowiak. Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks. In *Proc. of the* 13^{th} *International Symposium on Algorithms and Computation (ISAAC)*, pages 535–549, 2002.

[70] K. Kalpakis, K. Dasgupta, and P. Namjoshi. Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks. In *Proc. of the IEEE International Conference on Networking (ICN)*, pages 685–696, 2002.

[71] N. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming. *Combinatorica*, 4(4):373–396, 1984.

[72] T. Kasetkasem and P. K. Varshney. Communication structure planning for multisensor detection systems. *IEE Proceedings Radar, Sonar and Navigation*, 148(1):2–8, 2001.

[73] V. Kawadia and P. Kumar. Power Control and Clustering in Ad Hoc Networks. In *Proc. of the* 22^{nd} *Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 459–469 vol.1, 2003.

[74] S. Khuller, B. Raghavachari, and N. Young. Balancing Minimum Spanning and Shortest Path Trees. In *Proc. of the* 4^{th} *Symposium on Discrete Algorithms (SODA)*, pages 243–250, 1993.

[75] D. Korn, J. MacDonald, J. Mogul, and K. Vo. The VCDIFF Generic Differencing and Compression Data Format. RFC 3284 (Proposed Standard), June 2002.

[76] J. Koshy and R. Pandey. Remote Incremental Linking for Energy-Efficient Reprogramming of Sensor Networks. In *Proc. of the* 2^{nd} *European Workshop on Wireless Sensor Networks (EWSN)*, pages 354–365, 2005.

[77] E. Kranakis, H. Singh, and J. Urrutia. Compass Routing on Geometric Networks. In *Proc. of the* 11^{th} *Canadian Conference on Computational Geometry (CCCG)*, pages 51–54, 1999.

[78] B. Krishnamachari. *Networking Wireless Sensors.* Cambridge University Press, New York, NY, USA, 2005.

[79] B. Krishnamachari, D. Estrin, and S. Wicker. The Impact of Data Aggregation in Wireless Sensor Networks. In *Proc. of the* 22^{nd} *International Conference on Distributed Computing Systems (ICDCS)*, pages 575–578, 2002.

[80] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Local Approximation Schemes for Ad Hoc and Sensor Networks. In *Proc. of the* 3^{rd} *ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 97–103, 2005.

[81] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proc. of the* 10^{th} *Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 260–274, 2004.

[82] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the Locality of Bounded Growth. In *Proc. of the* 24^{th} *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 60–68, 2005.

[83] F. Kuhn and R. Wattenhofer. Constant-Time Distributed Dominating Set Approximation. In *Proc. of the* 22^{nd} *ACM Symposium on the Principles of Distributed Computing (PODC)*, pages 25–32, 2003.

[84] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Routing: Of Theory and Practice. In *Proc. of the 22nd ACM Symposium on the Principles of Distributed Computing (PODC)*, pages 63–72, 2003.

[85] S. S. Kulkarni and L. Wang. MNP: Multihop Network Reprogramming Service for Sensor Networks. In *Proc. of the 1st International Conference on Distributed Computing Systems (ICDCS)*, pages 7–16, 2005.

[86] K. Langendoen, A. Baggio, and O. Visser. Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture. In *Proc. of the 14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, pages 1–8, 2006.

[87] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Saunders College Publishing, Fort Worth, 1976.

[88] P. Levis and D. Culler. Maté: A Tiny Virtual Machine for Sensor Networks. *ACM SIGOPS Operating System Review*, 36(5):85–95, 2002.

[89] P. Levis, D. Gay, and D. Culler. Active Sensor Networks. In *Proc. of the 2nd International Conference on Networked Systems Design and Implementation (NSDI)*, pages 343–356, 2005.

[90] N. Li, C.-J. Hou, and L. Sha. Design and Analysis of an MST-Based Topology Control Algorithm. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[91] W. Li, Y. Zhang, J. Yang, and J. Zheng. UCC: Update-Conscious Compilation for Energy Efficiency in Wireless Sensor Networks. In *Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 383–393, 2007.

[92] X.-Y. Li, G. Calinescu, and P.-J. Wan. Distributed Construction of Planar Spanner and Routing for Ad Hoc Wireless Networks. In *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[93] X.-Y. Li, W.-Z. Song, and W. Wang. A Unified Energy Efficient Topology for Unicast and Broadcast. In *Proc. of the 11th International Conference on Mobile Computing and Networking (MOBICOM)*, pages 1–15, 2005.

[94] S. Lindsey, C. Raghavendra, and K. Sivalingam. Data Gathering in Sensor Networks using the Energy*Delay Metric. In *Proc. of the 15th International Parallel & Distributed Processing Symposium (IPDPS)*, pages 2001–2008, 2001.

[95] S. Lindsey and C. S. Raghavendra. PEGASIS: Power-Efficient Gathering in Sensor Information Systems. In *Proc. of the IEEE Aerospace Conference*, pages 3–1125–3–1130, 2002.

[96] C.-J. Lu. A Deterministic Approximation Algorithm for a Minimax Integer Programming Problem. In *Proc. of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 663–668, 1999.

[97] G. Lu, B. Krishnamachari, and C. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Sensor Networks. *Wireless Communications and Mobile Computing*, 7(7):863–875, 2007.

[98] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. *Journal of the ACM*, 41(5):960–981, 1994.

[99] J. MacDonald. File System Support for Delta Compression. Masters thesis. Department of Electrical Engineering and Computer Science, University of California at Berkeley, 2000.

[100] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.

[101] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proc. of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 88–97, 2002.

[102] P. J. Marrón, M. Gauger, A. Lachenmann, D. Minder, O. Saukh, and K. Rothermel. FlexCup: A Flexible and Efficient Code Update Mechanism for Sensor Networks. In *Proc. of the 3rd European Conference on Wireless Sensor Networks (EWSN)*, pages 212–227, 2006.

[103] K. Martinez, P. Padhy, A. Elsaify, G. Zou, A. Riddoch, J. Hart, and H. Ong. Deploying a Sensor Network in an Extreme Environment. In *Proc. of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, pages 186–193, 2006.

[104] M. J. McGlynn and S. A. Borbash. Birthday Protocols for Low Energy Deployment and Flexible Neighborhood Discovery in Ad Hoc Wireless Networks. In *Proc. of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 137–145, 2001.

[105] F. Meyer auf der Heide, C. Schindelhauer, K. Volbert, and M. Grünewald. Energy, Congestion and Dilation in Radio Networks. In *Proc. of the 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 343–370, 2002.

[106] M. J. Miller, C. Sengul, and I. Gupta. Exploring the Energy-Latency Trade-off for Broadcasts in Energy-Saving Sensor Networks. In *Proc. of The 25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 17–26, 2005.

[107] K. Moaveni-Nejad and X.-Y. Li. Low-Interference Topology Control for Wireless Ad Hoc Networks. In *Proc. of the 2nd IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2005.

[108] T. Moscibroda and R. Wattenhofer. Maximal Independent Sets in Radio Networks. In *Proc. of the 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 148–157, 2005.

[109] T. Moscibroda and R. Wattenhofer. Minimizing Interference in Ad Hoc and Sensor Networks. In *Proc. of the 3rd ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 24–33, 2005.

[110] T. Moscibroda and R. Wattenhofer. The Complexity of Connectivity in Wireless Networks. In *Proc. of the 25th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1–13, 2006.

[111] T. Moscibroda, R. Wattenhofer, and A. Zollinger. Topology Control Meets SINR: The Scheduling Complexity of Arbitrary Topologies. In *Proc. of the 7th International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 310–321, 2006.

[112] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[113] L. Narayanan. Channel Assignment and Graph Multicoloring. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, chapter 4, pages 71–94. John Wiley & Sons, Inc., 2002.

[114] J. Paek, K. Chintalapudi, R. Govindan, J. Caffrey, and S. Masri. A wireless sensor network for structural health monitoring: performance and experience. In *Proc. of the 2nd IEEE Workshop on Embedded Networked Sensors (EMNETS)*, pages 1–9, 2005.

[115] R. K. Panta and I. K. S. Bagchi. Stream: Low Overhead Wireless Reprogramming for Sensor Networks. In *Proc. of the 26th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 928–936, 2007.

[116] S. Pattem, B. Krishnamachari, and R. Govindan. The Impact of Spatial correlation on Routing with Compression in Wireless Sensor Networks. In *Proc. of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 28–35, 2004.

[117] D. E. Penny. The Automatic Management of Multi-Sensor Systems. In *Proc. of the 1st International Conference on Information Fusion (FUSION)*, pages 748–756, 1998.

[118] C. Percival. Naive Differences of Executable Code. `http://www.daemonology.net/papers/bsdiff.pdf`, 2003.

[119] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 95–107, 2004.

[120] R. Prakash. Unidirectional Links Prove Costly in Wireless Ad-Hoc Networks. In *Proc. of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 15–22, 1999.

[121] P. Raghavan and C. Thompson. Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs. *Combinatorica*, 7(4):365–374, 1987.

[122] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-Aware Wireless Microsensor Networks. *IEEE Signal Processing Magazine*, 19(2):40–50, 2002.

[123] V. Rajendran, J. Garcia-Luna-Aceves, and K. Obraczka. Energy-Efficient, Application-Aware Medium Access for Sensor Networks. In *Proc. of the 2nd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2005.

[124] R. Ramanathan and R. Rosales-Hain. Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment. In *Proc. of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 404–413 vol.2, 2000.

[125] T. Rappaport. *Wireless Communications: Principles and Practice.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[126] C. Reichenberger. Delta Storage for Arbitrary Non-Text Files. In *Proc. of the* 3^{rd} *International Workshop on Software Configuration Management (SCM)*, pages 144–152, 1991.

[127] N. Reijers and K. Langendoen. Efficient Code Distribution in Wireless Sensor Networks. In *Proc. of the* 2^{nd} *International Conference on Wireless Sensor Networks and Applications (WSNA)*, pages 60–67, 2003.

[128] L. G. Roberts. Aloha Packet System with and without Slots and Capture. *ACM SIGCOMM, Computer Communication Review*, 5(2):28–42, 1975.

[129] V. Rodoplu and T. H. Meng. Minimum Energy Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 17(8), 1999.

[130] P. Santi. *Topology Control in Wireless Ad Hoc and Sensor Networks.* John Wiley & Sons Ltd, Chichester, UK, 2005.

[131] T. Schmid, H. Dubois-Ferrière, and M. Vetterli. SensorScope: Experiences with a Wireless Building Monitoring Sensor Network. In *Proc. of the Workshop on Real-World Wireless Sensor Networks (REALWSN)*, 2005.

[132] R. Simon, L. Huang, E. Farrugia, and S. Setia. Using multiple communication channels for efficient data dissemination in wireless sensor networks. In *Proc. of the* 2^{nd} *IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2005.

[133] S. Singh and C. S. Raghavendra. PAMAS - Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. *ACM SIGCOMM Computer Communication Review*, 28(3):5–26, 1998.

[134] A. Sinha and A. Chandrakasan. Dynamic Power Management in Wireless Sensor Networks. *IEEE Design and Test*, 18(2):62–74, 2001.

[135] I. Solis and K. Obraczka. The Impact of Timing in Data Aggregation for Sensor Networks. In *Proc. of the IEEE International Conference on Communications (ICC)*, 2004.

[136] A. Srinivasan. Improved Approximations of Packing and Covering Problems. In *Proc. of the* 27^{th} *ACM Symposium on Theory of Computing (STOC)*, pages 268–276, 1995.

[137] T. Stathopoulos, J. Heidemann, and D. Estrin. A Remote Code Update Mechanism for Wireless Sensor Networks. Technical Report CENS-TR-30, UCLA, 2003.

[138] H. Takagi and L. Kleinrock. Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[139] R. E. Tarjan. Finding Optimum Branchings. *Networks*, 7:25–35, 1977.

[140] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A Macroscope in the Redwoods. In *Proc. of the 3^{rd} International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 51–63, 2005.

[141] D. Trendafilov, N. Memon, and T. Suel. zdelta: An Efficient Delta Compression Tool. Technical Report TR-CIS-2002-02, Polytechnic University, 2002.

[142] A. Tridgell. *Efficient Algorithms for Sorting and Synchronization*. PhD thesis, Astralian National University, 1999.

[143] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. *Wireless Networks*, 8(2/3):153–167, 2002.

[144] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of the 1^{st} International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 171–180, 2003.

[145] C. von Platen and J. Eker. Feedback Linking: Optimizing Object Code Layout for Updates. In *Proc. of the ACM SIGPLAN/SIGBED Conference on Language, Compilers, and Tool Support for Embedded Systems (LCTES)*, pages 2–11, 2006.

[146] R. A. Wagner and M. J. Fischer. The String-to-String Correlation Problem. *Journal of the ACM*, 21(1):168–173, 1974.

[147] G. Wang, G. Cao, and T. L. Porta. Movement-Assisted Sensor Deployment. *IEEE Transactions on Mobile Computing*, 5(6):640–652, 2006.

[148] Q. Wang, Y. Zhu, and L. Cheng. Reprogramming Wireless Sensor Networks: Challenges and Approaches. *IEEE Network*, 20(3):48–55, 2006.

[149] Y. Wang and X.-Y. Li. Localized Construction of Bounded Degree Planar Spanner. In *Proc. of the 1^{st} Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 59–68, 2003.

[150] Y. Wang, W. Wang, and X.-Y. Li. Distributed Low-Cost Backbone Formation for Wireless Ad Hoc Networks. In *Proc. of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 2–13, 2005.

[151] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks. In *Proc. of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1388–1397, 2001.

[152] R. Wattenhofer and A. Zollinger. XTC: A Practical Topology Control Algorithm for Ad-Hoc Networks. In *Proc. of the 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, 2004.

[153] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring Volcanic Eruptions with a Wireless Sensor Network. In *Proc. of the 2nd European Workshop on Sensor Networks (EWSN)*, pages 108–120, 2005.

[154] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *Proc. of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 381–396, 2006.

[155] A. Woo and D.-E. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proc. of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 221–235, 2001.

[156] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A Wireless Sensor Network For Structural Monitoring. In *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 13–24, 2004.

[157] W. Ye, J. S. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1567–1576 vol.3, 2002.

[158] W. Ye, F. Silva, and J. S. Heidemann. Ultra-Low Duty Cycle MAC with Scheduled Channel Polling. In *Proc. of the 4th International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 321–334, 2006.

[159] Y. Yu, B. Krishnamachari, and V. Prasanna. Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks. In *Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Comunications Societies (INFOCOM)*, 2004.

[160] Y. Yu, B. Krishnamachari, and V. Prasanna. Energy Minimization for Real-Time Data Gathering in Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 5(11):3087–3096, 2006.

[161] Y. Yu, V. Prasanna, and B. Krishnamachari. *Information Processing and Routing in Wireless Sensor Networks*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.

[162] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware Design Experiences in ZebraNet. In *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 227–238, 2004.

[163] F. Zhao and L. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[164] C. Zhou and B. Krishnamachari. Localized Topology Generation Mechanisms for Self-Configuring Sensor Networks. In *Proc. of the 46th Annual IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1269–1273 vol.3, 2003.

[165] Y. Zou and K. Chakrabarty. Sensor Deployment and Target Localization Based on Virtual Forces. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1293–1303 vol.2, 2003.

Curriculum Vitae

May 18, 1978	Born in Zug, Switzerland
1984–1998	Primary, secondary, and high schools in Oberägeri and Zug, Switzerland
1998–1999	Studies in architecture, ETH Zurich, Switzerland
1999–2004	Studies in computer science, ETH Zurich, Switzerland
April 2004	M.Sc. in computer science, ETH Zurich, Switzerland
2004–2008	Ph.D. student, research and teaching assistant, Distributed Computing Group, Prof. Roger Wattenhofer, ETH Zurich, Switzerland
May 2008	Ph.D. degree, Distributed Computing Group, ETH Zurich, Switzerland Advisor: Prof. Roger Wattenhofer Co-examiners: Prof. Bhaskar Krishnamachari, University of Southern California, CA, USA Prof. Magnús M. Halldórsson, Reykjavik University, Iceland

Publications

The following lists all publications written during the four years of my being Ph.D. student at ETH Zurich.

1. *Decoding Code on a Sensor Node.* Pascal von Rickenbach and Roger Wattenhofer. 4th International Conference on Distributed Computing in Sensor Systems (DCOSS), Santorini Island, Greece, June 2008.

2. *Sensor Networks Continue to Puzzle: Selected Open Problems.* Thomas Locher, Pascal von Rickenbach, and Roger Wattenhofer. 9th International Conference on Distributed Computing and Networking (ICDCN), Kolkata, India, January 2008.

3. *Dozer: Ultra-Low Power Data Gathering in Sensor Networks.* Nicolas Burri, Pascal von Rickenbach, and Roger Wattenhofer. 9th International Conference on Information Processing in Sensor Networks (IPSN), Cambridge, Massachusetts, USA, April 2007.

4. *Topology Control Made Practical: Increasing the Performance of Source Routing.* Nicolas Burri, Pascal von Rickenbach, Roger Wattenhofer, and Yves Weber. 2nd International Conference on Mobile Ad-hoc and Sensor Networks (MSN), Hong Kong, China, December 2006.

5. *Analyzing the Energy-Latency Trade-off during the Deployment of Sensor Networks.* Thomas Moscibroda, Pascal von Rickenbach, and Roger Wattenhofer. 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Barcelona, Spain, April 2006.

6. *Interference in Cellular Networks: The Minimum Membership Set Cover Problem.* Fabian Kuhn, Pascal von Rickenbach, Roger Wattenhofer, Emo Welzl, and Aaron Zollinger. 11th International Computing and Combinatorics Conference (COCOON), Kunming, Yunnan, China, August 2005.

7. *A Robust Interference Model for Wireless Ad-Hoc Networks.* Pascal von Rickenbach, Stefan Schmid, Roger Wattenhofer, and Aaron Zollinger. 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN), Denver, Colorado, USA, April 2005.

8. *Gathering Correlated Data in Sensor Networks.* Pascal von Rickenbach and Roger Wattenhofer. 2nd ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), Philadelphia, Pennsylvania, USA, October 2004.

9. *Does Topology Control Reduce Interference?* Martin Burkhart, Pascal von Rickenbach, Roger Wattenhofer, and Aaron Zollinger. 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), Roppongi Hills, Tokyo, Japan, May 2004.