



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Diploma Thesis

Sensor Networks: Interference Reduction and Possible Applications

Martin Fussen

Prof. Roger Wattenhofer
Aaron Zollinger

Distributed Computing Group
Institute for Pervasive Computing
ETH Zurich

24th March 2004 - 25th July 2004

Contents

1	Introduction	1
2	A Theoretical Approach towards Interference Reduction	3
2.1	Related Work	3
2.2	Interference Model	5
2.3	A Lower Bound	7
2.4	NCC Algorithm	9
2.5	NCC is not an Approximation Algorithm	13
2.6	Conclusion	15
3	Simulation of Average-Case Networks	16
3.1	Simulation Environment	16
3.2	Interference in Average-Case Networks	19
3.3	Conclusion	20
4	Possible Applications in Sensor Networks	21
4.1	A set of Sensors	21
4.1.1	Hardware	21
4.1.2	Software and Development Tools	22
4.2	Sample Applications	25
4.3	Conclusion	29
5	Conclusion	30

Abstract

There are still a lot of open questions in the field of sensor networks. In the first part we address the problem of interference. Energy consumption in general and interference in particular being among the most critical issues in sensor networks, we introduce an explicit definition of interference, based on the number of other nodes by which a given network node can be disturbed. With this definition we show that there exist instances of sensor networks on which no Topology Control algorithm can construct a valid data gathering network with interference less than logarithmic in the number of network nodes n .

Our Nearest Component Connector (NCC) algorithm, asymptotically matches this lower bound, and guarantees to build a valid topology with interference in $O(\log n)$. A comparison of NCC with other previously proposed data gathering structures brings us to the end of part one.

In a second part we have a look at tasks in sensor networks which go beyond simple data gathering. Practical implementations with *ScatterWeb* nodes illustrate possible applications.

Acknowledgements

A great deal of appreciation goes to my advisors Prof. Roger Wattenhofer and Aaron Zollinger, for their guidance, help, encouragement and specially for being my source of inspiration. Last but not least, I would like to thank all members of the *Distributed Computing Group* for always supporting me and for the nice and friendly ambiance during the last four months.

Chapter 1

Introduction

The concept of sensor networks becomes more and more popular and first real-world applications appear. Sensor networks in the near future could be made up of hundreds or thousands of small autonomous nodes and used for a lot of different tasks. Possible applications could be weather forecasts, industrial control systems or video monitoring.

A sensor network consists of sensors deployed in a given region with the task of sensing one or more physical values (such as temperature, humidity, brightness, or motion). The sensors are equipped with radio devices and—in the popular monitoring scenario model—periodically transfer the sensed data to a designated data sink node. Each sensor has a battery and therefore needs to minimize energy consumption in order to extend its lifetime.

There are still a lot of open questions concerning sensor networks. In the first part we address three of them. Which algorithm should be used to build the communication infrastructure in a sensor network? How can we minimize the energy consumption of a sensor? What techniques can be used to minimize interference in a network? We show that an important step towards longer lifetimes of nodes is the reduction of interference in sensor networks. Less interference in a network results in fewer retransmissions of lost messages on the media access layer and therefore leads to lower power consumption.

To study aspects like interference reduction or Topology Control on a theoretical base it is necessary to reduce a physical sensor network to a well defined structure like a graph in the plane. Our model of a sensor network is a directed graph $G(V, E)$ where nodes $\{v_1, \dots, v_n\}$ placed in the plane represent the set of sensors including the sink. Communication links between sensors are modeled as arcs.

To allow all data to be gathered at the sink, a Topology Control algorithm constructs a *sink tree*, a directed tree with all arcs—modeling unidirectional communication links—pointing towards the sink node. In the context of interference reduction, the task of the Topology Control algorithm is to

find such a sink tree with least possible interference. Thereby we account for the fact that in the monitoring scenario communication from the sink to the sensors occurs rarely and can therefore be neglected with respect to interference.

We assume that the transmission power of each node in our sensor network can be adjusted. A higher transmission power allows a node to send messages over a longer distance. With maximal transmission power each node can reach any other node in the network. We further assume that the covered area of a sending node v_i is a disk with v_i in its center.

Assuming a worst-case perspective we show in the first part of our report that there are network instances in which any Topology Control algorithm will construct a resulting network with interference at least $\log n - 1$. We then propose the *Nearest Component Connector (NCC)* algorithm, which provably produces at most $O(\log n)$ interference in any network in polynomial time. In this sense the NCC algorithm is asymptotically optimal. Having proved the worst case optimality of NCC we want to see how our algorithm can be used in an average-case network. We therefor implemented a small simulation environment. The simulation framework can be used to compare different Topology Control algorithms on a sensor network constructed by placing nodes randomly in the plane. We use this simulation to compare our NCC algorithm to previously proposed structures. We thereby show that—besides being asymptotically worst-case optimal—NCC also in the average case produces interference results comparable with other construction methods.

In the last part of our report we have a look at possible applications using sensor nodes which go beyond the simple gathering of sensed data. We show how different capabilities of autonomous sensor nodes—sensors for a range of physical values, actors, wireless communication—can be used to implement solutions for practical problems. We not only present ideas for possible application scenarios but prototypes of all suggested applications. The networks are implemented using *ScatterWeb nodes*. These nodes, developed at Freie Universitaet Berlin and equipped with five sensors, LEDs, a beeper, and a wireless communication device allow fast prototyping and proofs of the feasibility of proposed applications. The five appliances we implemented demonstrate what can be done with comparably simple nodes.

Chapter 2

A Theoretical Approach towards Interference Reduction

2.1 Related Work

The issue of energy efficiency in sensor networks [1, 3, 7]—particularly extending network lifetime—has been mainly studied in the context of optimal sensor placement and energy-efficient routing. Recently also the fact that certain types of sensed data allow for aggregation at sensor nodes [9] and the existence of redundancy in acquired information [4, 6]—for instance correlation between sensed data depending on the distance between sensors—has been considered.

The concept of Topology Control has been studied in the broader context of ad-hoc networks—wireless networks whose application is not confined or targeted to data acquisition and gathering, as is the case for sensor networks—for two decades [10, 11, 12, 13, 16, 17, 18, 20, 21]. Although Topology Control has sometimes been considered the task of generally constructing topologies with certain desired properties (such as network connectivity, planarity, sparseness or locality of construction), interference reduction has often been regarded as one of the main goals of Topology Control. However, most of the proposed Topology Control algorithms are stated to produce low interference implicitly by constructing sparse networks or networks with bounded node degree. As shown in [2], such implicit interference reduction can fail to effectively achieve its goal.

A notable exception to this is [14], which defines an explicit concept of interference between edges and shows—based on a time-step routing model—that there exist inevitable trade-offs between congestion, energy consumption and dilation. While this interference definition is based on the current network traffic, [2] proposes an explicit definition of interference that is in-

dependent of network traffic. This interference definition—adopted and further studied in [15]—is based on the question how many nodes are affected by communication over a given link. In contrast the interference model introduced in this report considers interference at the intended receiver of a message since this is where message collisions actually have their negative effect.

2.2 Interference Model

Among the most critical resources in wireless networks with autonomous nodes is energy. One of the foremost approaches to reducing energy consumption consists in minimizing interference between the network nodes and consequently in reducing the number of message collisions and hence required retransmissions. The concept of *Topology Control* confines interference by having the network nodes reduce their transmission power levels and drop long-range connections in a coordinated way. At the same time transmission power reduction has to occur in a controlled manner in order to preserve connectivity of the network.

If we want to minimize interference in sensor networks, we have to look at topologies in which each node sends its data to at most one other node and a valid graph contains a path from every sensor to the sink, which results in a tree with the sink as its root and all arcs pointing towards the root. We call such a tree a *sink tree*.

Definition 1. *Given a set of nodes V and a sink s , a sink tree is a tree spanning V with all arcs pointing towards s .*

Most of the previous work maintains to solve the interference issue in wireless networks implicitly by constructing sparse topologies or topologies with constant-bounded node degrees. Such an implicit notion of interference can however lead to Topology Control algorithms that fail to reduce interference since message transmission can affect nodes even if they are not direct neighbors of the sending node in the resulting topology graph [2]. Besides demonstrating this weakness of implicit interference models, [2] introduces an explicit definition of interference, based on the number of nodes potentially disturbed by communication over a link.

In contrast we assume in this report a receiver-centric perspective and use an explicit model of interference. We explicitly count the number of nodes potentially disturbing the reception of a message. This definition best reflects the fact that interference is a problem occurring at the receiver. Minimizing the interference at each possible receiver (each node in the network) reduces message collisions in the network and therefore lowers the amount of required retransmissions. This saves energy and allows for a longer lifetime of sensors equipped with batteries.

The interference value of a single node is the number of transmission circles by which the node is covered.

Definition 2. *The interference value of a single node v is defined as*

$$I(v) := |\{u | v \in D(u, r_u)\}|$$

where $D(u, r_u)$ stands for the transmission circle with node u in its center and radius r_u .

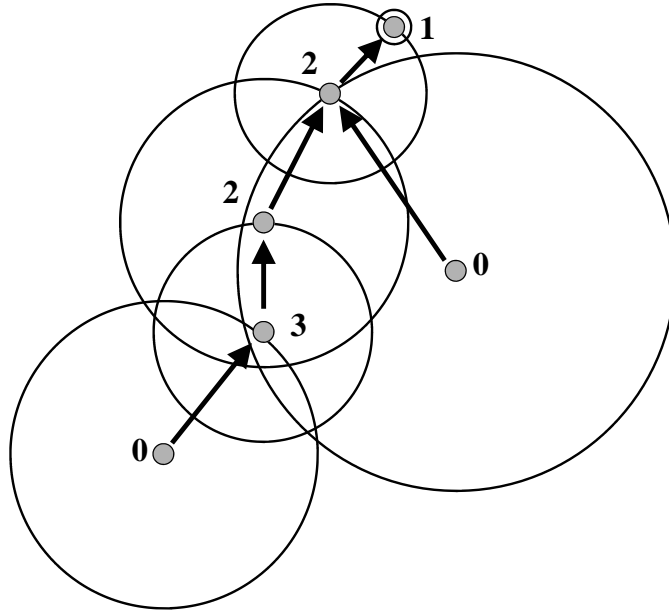


Figure 2.1: A sink tree with 6 nodes. The uppermost node is the sink node. Each node is labeled with its interference value. The interference of the whole network is 3.

As the interference of a whole network we use the maximum of all interference values in the graph (see Figure 2.1).

Definition 3. *The interference of a Graph $G(V,E)$ is defined as*

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

The problem we study in this report consists in finding a sink tree with least possible interference for a given sensor network.

Definition 4. *The Minimum Interference Sink Tree (MIST) problem is defined as the problem of finding a sink tree for a given node set with minimal interference.*

In the remainder of the report we consider Topology Control algorithms with the goal of solving the MIST problem.

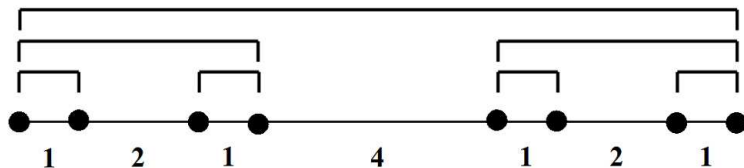


Figure 2.2: A recursive arrangement of 16 nodes on a horizontal line.

2.3 A Lower Bound

In this section we show that n nodes in a sensor network can be arranged in a way that no possible algorithm can construct a sink tree with interference less than $\log(n) - 1$. The existence of such examples constitutes a lower bound with respect to interference.

Theorem 2.3.1. *There exist sensor networks with nodes arranged in a way that no algorithm can construct a sink tree with interference less than $\log(n) - 1$.*

Proof. To prove this Theorem we present an arrangement of $n = 2^s$ nodes which cannot be connected to the given sink in the described way with interference less than $\log(n) - 1$.

The nodes are arranged on a horizontal line. Figure 2.2 shows the arrangement. The first $k = 4$ nodes v_1, v_2, v_3 , and v_4 , are positioned at coordinates 0, 1, 3, and 4. Then a copy of the already positioned nodes is placed in distance $d = \overline{v_1 v_k}$ to the right of node v_k . This construction is recursively repeated until 2^s nodes are placed on the horizontal line.

After the execution of any possible algorithm there must exist a directed path from each node in the set to the global sink. Let G_1 be the node group $\{v_1, \dots, v_{2^{s-1}}\}$ and $G_2 := \{v_{2^{s-1}+1}, \dots, v_{2^s}\}$. If we assume without loss of generality that the node group G_2 contains the global sink, the result of an algorithm has to contain an arc from G_1 to G_2 . Because of the special arrangement of the nodes in our example, the gap between G_1 and G_2 has length equal to the (Euclidean) diameter of the two groups. The arc between G_1 and G_2 cannot be shorter than the gap and therefore interferes with all nodes but one in G_1 . Figure 2.3 illustrates the idea of the proof.

If, in a next step, we look into G_1 , there are 2^{s-1} nodes partitioned into two subgroups $G_{1.1}$ and $G_{1.2}$. Assuming, again without loss of generality, that the above arc from G_1 to G_2 originates in $G_{1.2}$, we can observe that there has to exist an arc leading out from $G_{1.1}$, which—bridging $G_{1.1}$'s adjacent gap—interferes with all nodes in $G_{1.1}$ (except for the node at which this arc originates). The existence of such an arc is a consequence to the required

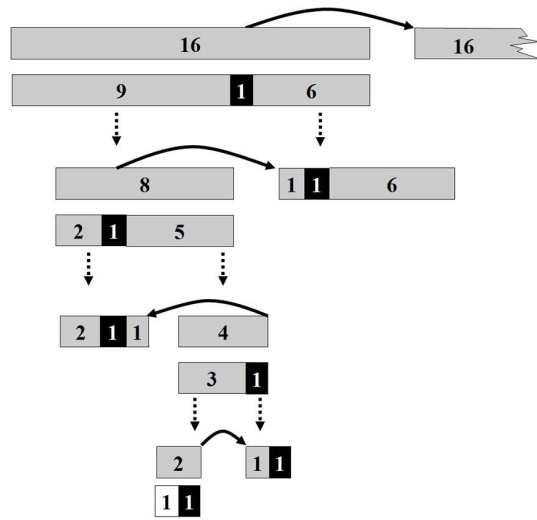


Figure 2.3: Illustration of the proof of Theorem 2.3.1. The numbers stand for the number of nodes in each group. The dark fields represent the unaffected nodes in the last step and the hollow node (bottom) is the one whose interference value was incremented in each step.

directed path from each node to the sink. The same argument recursively holds for all node levels in the arrangement.

This all together proves that the interference value of at least one node is incremented in all steps and because the node set is of size 2^s we get a maximum interference not smaller than $s - 1$ or $\log(n) - 1$. \square

2.4 NCC Algorithm

In this section we present the Nearest Component Connector algorithm (*NCC*).

The general idea of this algorithm is to connect components to their nearest neighbors. This is done in several rounds and leads to a sink tree. A component can be a single node or a group of previously connected nodes. When the algorithm starts, each node in the given sensor network forms a component of its own. First the predefined global sink is treated exactly as a normal node. Whenever two or more components are connected in one round, they form a single component in the following round of NCC. If we have a look at an arbitrary component during the execution of the algorithm we observe that this component has exactly one node all other component members have a directed path to. This means that there is one node which gathers all sensed data of the component. We call this special node the *local sink* of its component.

Whenever a new arc is established during the execution of NCC it goes from a local sink of a component C to the nearest node not in C . If a round produces a cycle, it is broken by removing one of its arcs at the end of the round. This guarantees the construction of a valid sink tree topology. After the last round of NCC however, the root of the resulting tree is not necessarily the global sink. So we need an additional step in which the arc originating from the global sink is deleted and a new arc from the only remaining local sink to the global sink is added.

The detailed steps of NCC can be seen in Algorithm 1 and Figure 2.4 shows a sample execution of the algorithm.

We will now prove that the presented NCC algorithm constructs a valid sink tree topology for a given sensor network consisting of n nodes with an interference value in $O(\log n)$. We will also see that the execution of NCC takes polynomial time only.

Theorem 2.4.1. *The NCC Algorithm constructs a sink tree on a given Graph $G = (V, E)$ with $|V| = n$ with an interference value in $O(\log n)$ in polynomial time.*

Proof. This proof has three parts. In the first one we show that NCC does not need more than $\log n$ rounds (while-loop iterations) to build the sink tree. In the second part we show that in each of these rounds the interference value of a node will not be incremented by more than a constant value. In part three we show that NCC terminates in polynomial time.

To show the first part we use the fact that in each round a local sink s either establishes an arc to the nearest node of another component or that another component establishes an arc to the component s is part of. The two or more connected components together form one component in the next round. Therefore the number of components in round i is at most half

Algorithm 1 The Nearest Component Connector algorithm NCC

Input: V : a set of nodes placed in the plane

$s \in V$: a predefined global sink

$G := (V, E := \emptyset)$;

$lsinks := V$ // set of local sinks

While ($|lsinks| > 1$) do

$lsinks :=$ sinks in G ;

 // sinks are nodes having no outgoing arc

$E' = \{e_1, \dots, e_{|lsinks|}\}$: e_i is an arc from v_i to its nearest neighbor in a different component;

 If $G' := (V, E \cup E')$ contains a cycle, remove one of the arcs in the cycle from E' ;

$G := G'$

od;

Remove the arc originating from the global sink from E and add a new arc from the only remaining local sink to the predefined global sink to E .

Output: G

the number of the components in round $i - 1$. This implies that after at most $\log n$ rounds only one component is left and the algorithm terminates.

Figure 2.5 illustrates the second part of the proof. We use the fact that each sink connects to the *nearest* node in a component different from its own. If a local sink l_i which is part of component C_i connects to a node v_j , its distance to all nodes not in C_i is at least $\bar{l}_i v_j$. So only nodes which are members of component C_i or nodes with the same distance from l_i as v_j are affected by the new arc. Furthermore a component establishes maximally one new arc in a single round and maximally 6 local sinks can establish an arc to the same node.¹ All this shows that the interference value of a node is maximally incremented by a constant in a single round of NCC.

Together with part one of the proof and the fact that the last step adds one single arc to the graph we see that the interference value of any node is incremented at most $\log(n) + 1$ times and each time by at most a constant value. This proves that the interference of the whole network is in $O(\log n)$.

The only remaining part of Theorem 2.4.1 we need to prove states that

¹This is the so-called “kissing number.” It is defined as the number of equivalent spheres that touch an equivalent sphere without intersections. The kissing number in the two-dimensional plane is 6. In three dimensions it is 12.

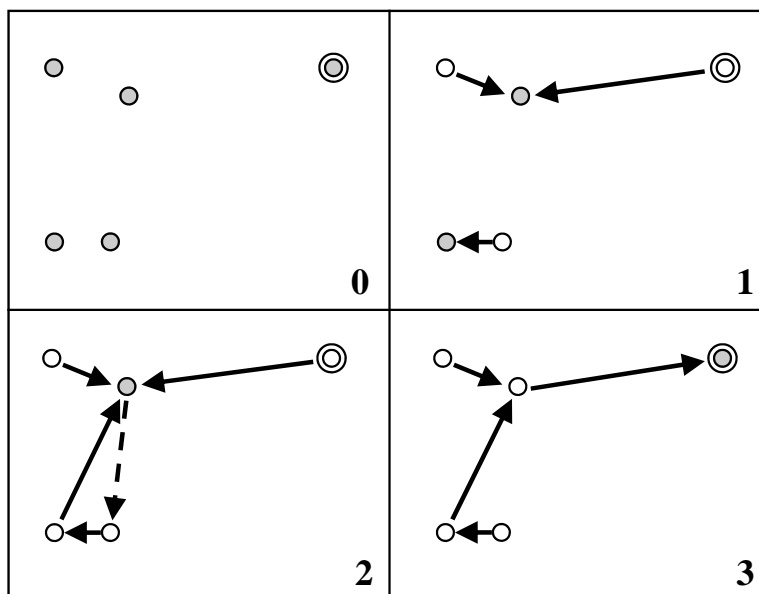


Figure 2.4: A sample execution of NCC on a given set of 5 nodes. Situation 0 shows the given nodes and the predefined sink (top right node). In each of the following two rounds every local sink connects to the nearest node not in its own component. In round 2 a cycle is produced. It is broken at the end of the round by removing one of the involved arcs (dashed arrow). After the last round (situation 3) the arc originating from the global sink is removed and an arc is added from the only remaining local sink to the predefined global sink.

NCC terminates in polynomial time. Every node v_i is a sink in one iteration of the while loop. (Actually it can be a sink in more than one loop iteration if a cycle is broken by removing the arc originating at that node. The fact however that for every such removed arc at least one other arc is added to the tree in the same round entails only an additional factor 2.) A sink has to find its nearest neighbor in a foreign component. This can be implemented using a list of neighbors, sorted according to their distances for each node and a union-find structure to check if a node is in a foreign component. The n lists of sorted neighbors can be constructed in time $n \cdot n \log(n)$. Maintaining the union-find structure and component membership lookups during the execution of NCC can also be done in time $n \cdot n \log(n)$. These observations prove that NCC terminates in polynomial time. \square

We present NCC in a centralized manner. This reflects the fact that in a sensor network we have an instance (the sink) commonly assumed to have much more computing power and energy than all other nodes (sensors).

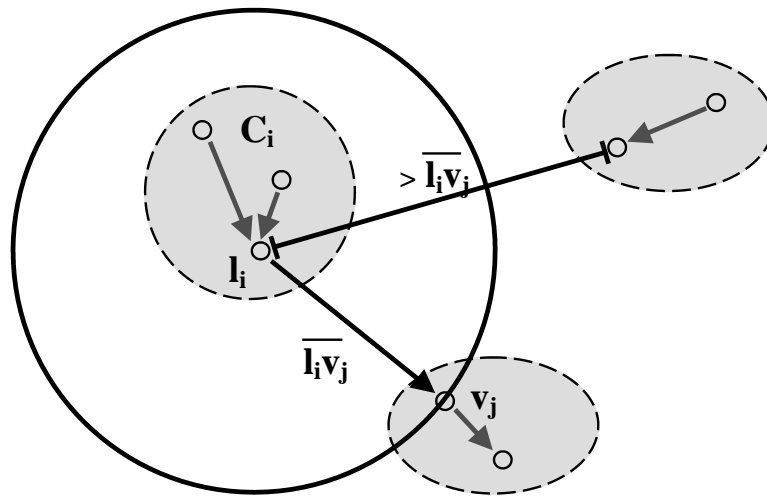


Figure 2.5: An illustration of the second part of the proof of Theorem 2.4.1. An arc from the local sink l_i to v_j only interferes with nodes in l_i 's component and v_j (and nodes in other components only if they are exactly at distance $\overline{l_i v_j}$).

Therefore the sink can run NCC and distribute the topology information of the constructed sink tree in an initialization phase.

Nevertheless a distributed variant of NCC is feasible. This variant would require counters in each node which keep track of the number of component unions the node was involved in since the start of the algorithm. These counters then guarantee that only components in the same “round” can establish new arcs between each other.

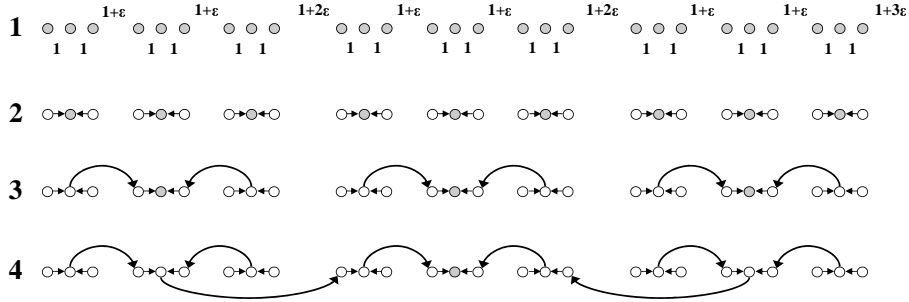


Figure 2.6: A recursive arrangement of nodes on a horizontal line. The four steps illustrate how the NCC algorithm constructs a sink tree for the given nodes (schematic illustration).

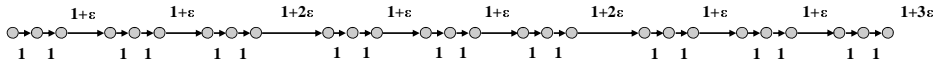


Figure 2.7: The same set of nodes connected linearly (schematic illustration).

2.5 NCC is not an Approximation Algorithm

In this section we show that NCC is not an approximation algorithm. That means there are node arrangements with an optimal solution which is “considerably” better than the one produced by NCC. Formally we say a minimization algorithm A is an approximation algorithm for a given minimization problem if there exists a factor α so that for all possible inputs the solution produced by A is at most the optimal solution times α .

Definition 5. *A minimization algorithm A is an approximation algorithm for a given minimization problem if there exists a constant α such that:*

$$Sol(A) < \alpha \cdot Sol(Opt)$$

where $Sol(A)$ and $Sol(Opt)$ stand for the solutions produced by the minimization algorithm A and an optimal algorithm Opt respectively.

Theorem 2.5.1. *The minimization algorithm NCC is not an approximation algorithm for the problem of finding a sink tree with minimal interference in a given set of nodes placed in the plane.*

Proof. To prove Theorem 2.5.1 we show that n nodes in a sensor network can be arranged in a way that NCC constructs a sink tree with interference in $O(\log n)$ whereas the optimal sink tree has constant interference.

Figure 2.6 shows the construction of a sink tree in a sensor network consisting of recursively placed nodes on a horizontal line. A group of three nodes is placed at coordinates 0, 1, and 2 on a horizontal line. Then two copies of this group are placed to the right of the first group. The distance between the groups is slightly greater than the distance between neighbors inside a group. ($1 + k\epsilon$ in the k^{th} recursion step) This recursive construction goes on until n nodes are placed on the line.

If we look at the steps of NCC during the construction of the sink tree, we can see, that the interference value of the leftmost node is incremented each time NCC proceeds to the next recursion step—an arc is added which connects the leftmost component to the next component on the right—which leads to interference in $O(\log n)$.

On the other hand Figure 2.7 shows that the same node arrangement can be connected linearly which leads to constant interference. In this example the sink tree constructed by NCC has an interference value which cannot be bounded by a factor f times the interference of the optimal solution. So we have shown that NCC is not an approximation according to Definition 2.5.1. \square

2.6 Conclusion

The approach we assume in this report in order to study interference in wireless and particularly sensor networks differs from most of the previous work in two ways: First, we introduce an explicit definition of interference. Second, our definition of interference is receiver-centric and reflects the fact that message collisions prevent proper message reception only if they occur at the receiving node.

With this formalized notion of interference we show on the one hand that there exist instances of sensor networks with n nodes in which it is impossible to construct a sink tree—a valid data gathering structure—with interference less than $\log n - 1$. On the other hand we describe the NCC algorithm asymptotically matching this lower bound in that it provably builds a sink tree with interference at most $O(\log n)$ on any given sensor network.

In this report we focus on the interference aspect in sensor networks and neglect the fact that communication over long links is more energy consuming than over short links. We therefore consider our work to be a first step towards understanding the complex interplay between interference and energy efficiency in sensor networks.

Chapter 3

Simulation of Average-Case Networks

In order to see if NCC can be used in average-case networks, we implemented a small simulation environment. With the help of this environment we compared NCC to some previously proposed construction methods for sink trees in sensor Networks. The results prove that NCC can be used in real sensor networks, but is slightly outperformed by a simple construction method based on a Minimum Spanning Tree. Nevertheless the comparable results together with its worst-case optimality makes NCC a usable algorithm for practical applications.

3.1 Simulation Environment

Our simulation environment allows a comparison of five different construction methods for sink trees in sensor networks. With the help of this tool it is possible to see how different algorithms behave in different randomly generated planar sensor networks.

The nodes in our simulations are distributed uniformly in a square field. Also the sink is chosen randomly. The user can adjust the node density (number of nodes in the unit square) and test all five algorithms with the same node set. The five implemented construction methods are:

1. The NCC construction method presented in Chapter 2.
2. The Minimum Spanning Tree (MST) with weights equal to the Euclidean edge lengths and all edges pointing towards the global sink. The algorithm to construct the MST starts with a set—called *connected set*—containing the sink only. In each step of the algorithm one new node is added to the connected set. MST chooses the shortest edge from an unconnected node to a node in the connected set

which does not generate a circle and adds it to the growing sink tree. The previously unconnected node is added to the connected set.

3. The Shortest Path Tree (SPT) with respect to the energy metric. (The SPT contains the shortest paths from all nodes to the sink.) SPT is constructed in a similar way to MST applying Dijkstra's algorithm.
4. A Greedy algorithm which chooses the next arc to insert in the growing sink tree, according to its influence on the maximal interference in the set of given nodes. (The arc is chosen with least negative effect on the temporal maximum in the Graph. If two or more arcs increment the maximum by the same number the shortest of them is chosen.)
5. The LEX construction which inserts in each step the arc producing the graph with smallest possible lexicographical order. The lexicographical representation of interference in a graph is defined as the sorted list — highest to lowest— of all node interference values in the graph. Example: The graph with interference values $\{5,4,4,3,2,2,2,1\}$ has a higher lexicographical order than the graph with values $\{5,4,3,3,2,2,2,1\}$.

Automated Simulation An important feature of our simulation environment is its capability of processing predefined simulation scripts. These scripts make it possible to carry out long simulation series. All results of the simulated algorithms are collected and can be used for further analysis and generation of diagrams. The commands used in the scripts allow the generation of sensor networks with different node densities and application of all algorithms.

Extendable Framework The simulation environment is implemented in Java. The object oriented design makes the framework extendable. The program design mainly consists of a node and a graph class. An instance of the node class represents a single sensor node in the set of given sensors. It contains methods for adding links, incrementing its interference value and contains a list with all distances to the other nodes in the graph. The graph class represents the whole sensor network and contains methods for the generation of node sets, adding and removing links, and calculating distances between nodes. In addition to these two central classes the framework contains five algorithm classes for the five simulated construction methods. Adding more algorithms to the simulation tool therefore only requires new algorithm classes.

Time intensive calculations Applying Greedy and Lex algorithms involves lots of time consuming calculations. Therefore the network size for these construction methods should not exceed 300 nodes. With the other

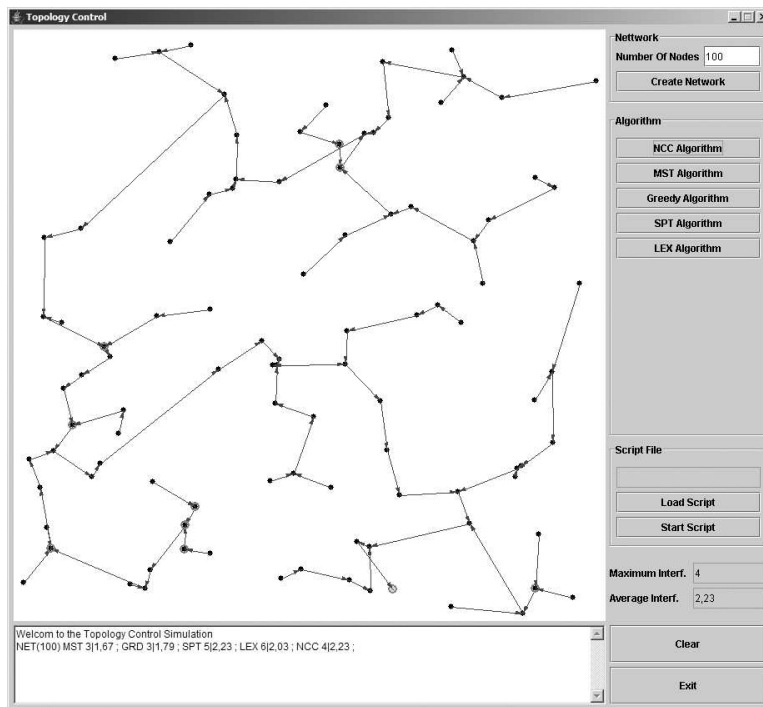


Figure 3.1: A screenshot of our simulation environment. On the left side a graphical representation of the node set and the constructed sink tree is shown. At the bottom the results are presented as a semicolon separated list which can be copied to a spreadsheet application.

algorithms simulations with network sizes of up to 1000 or even 2000 nodes are feasible.

Simple to use GUI A simple frontend allows simulating different scenarios, gives a graphical and textual feedback of the achieved results and constructed graphs and provides the possibility of loading scripts for automated simulations. Figure 3.1 shows a sample execution of the NCC algorithm on a network of 200 nodes.

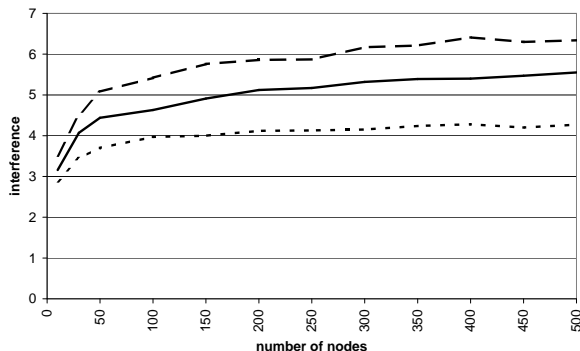


Figure 3.2: Simulation results for the Shortest Path Tree algorithm (dashed), the Nearest Component Connector algorithm (solid) and the Minimum Spanning Tree algorithm (dotted) in a range from 10 to 500 nodes. Due to their time intensive calculations the Greedy and LEX algorithm have not been simulated.

3.2 Interference in Average-Case Networks

We have seen that the NCC algorithm has asymptotically optimal worst-case behavior in the sense that it produces interference not greater than $O(\log n)$ for all possible node arrangements. In this section we will have a closer look at the average case behavior of our algorithm. We compare NCC to the Minimum Spanning Tree algorithm (MST) and the Shortest Path Tree algorithm (SPT).

In order to allow for evaluation in different conditions all three algorithms constructed sink trees for networks with different node densities. We simulated networks from 10 to 500 nodes distributed in a unit square. Plotted in the diagram are the averaged values over 100 runs for each simulated node density.

Figure 3.2 shows that all three algorithms produce rising interference with increasing node densities. Closer observation yields that our NCC algorithm performs better than the Shortest Path Tree algorithm but worse than the Minimum Spanning Tree algorithm. This is quite intriguing as the very simple MST algorithm, which was not explicitly designed to reduce interference, seems to outperform our NCC algorithm in average-case networks. Note however that MST is not asymptotically worst-case optimal and can produce interference of $n - 2$ for a sensor network consisting of n nodes. A sample of such an arrangement is shown in Figure 3.3.

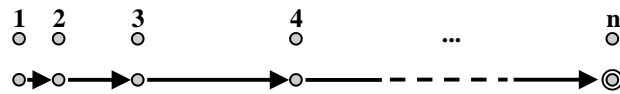


Figure 3.3: Nodes arranged on a horizontal line with exponentially increasing distances and the rightmost node chosen as global sink. Applied on this setting the MST algorithm produces interference $n - 2$.

3.3 Conclusion

The presented simulation environment gave us the possibility to see how NCC behaves in average-case networks and to compare its results to the results of other sink tree construction methods. Intriguingly the latter results show that—although the interference values produced by NCC fall roughly in the same range as those of other constructions—a simple minimum-spanning-tree-based structure keeps interference at a lower level than NCC in average-case networks. This is quite surprising, as MST was not explicitly designed to reduce interference.

Chapter 4

Possible Applications in Sensor Networks

The price of producing small autonomous sensors nodes drops continuously and larger sensor networks become more and more affordable. While the first companies start producing sensor network equipment three universities already sell ready to use prototyping systems [5, 8, 19] But what kind of real-world problems can be tackled with a set of sensor nodes? In this chapter we present five possible applications using sensors nodes equipped with five sensors, some actors and a wireless communication device. The prototypes of the presented ideas show how a concept can be transferred into a real world appliance.

4.1 A set of Sensors

In this section we present the hard- and software we used to implement our ideas of possible applications with autonomous sensor nodes.

4.1.1 Hardware

The nodes we used have been developed at *Freie Universitaet Berlin* and are called Embedded Sensor Boards (ESBs). They are part of the Scatter-Web project. Figure 4.1 shows one of the sensor nodes used. Each ESB is equipped with the following six sensors:

- Motion sensor (passive infrared sensor)
- Brightness sensor (infrared sensor)
- Temperature sensor (integrated with Real-time clock)
- Vibration/Tilt sensor

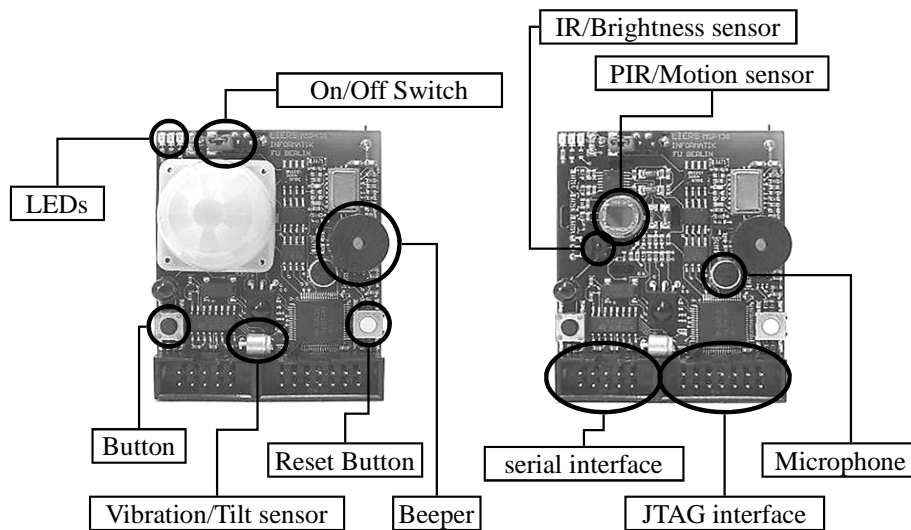


Figure 4.1: A picture of an Embedded Sensor Board (ESB) with and without protecting cover.

- Microphone
- Button

Additionally each sensor has two kinds of actors:

- 3 LEDs (Red, Yellow, Green)
- Beeper

In order to communicate with other sensor boards each node is equipped with a radio transceiver. Applications can control the transmission power directly. With minimal sending power, the transmission radius is smaller than 0.3 meters. With maximal power a transmission radius of up to 300 meters is achievable in open space. Using the serial interface an ESB can be connected to a PC. This allows applications running on the ESB to communicate via a standard terminal. The *JTAG* interface (left/front) can be used—as one of three possibilities, as described below—to transfer programs onto the sensor node.

4.1.2 Software and Development Tools

The sensor nodes have two layers of software on board. The first one, which can be seen as a kind of an operating system is called *firmware*. This layer

contains the controlling main-loop, some device drivers with their interrupts, and other input/output handlers. The second layer is simply called *program* and contains all application-dependent functions. The separation into two parts is mainly a consequence of the requirement for applications to be exchangeable via radio. Therefore the code which actually does the reprogramming of application code must stay in place while the new application instructions are received. The big advantage of this structure is that one can simply compile the same firmware with different programs and then send only the binary of a program to an ESB.

ScatterWeb nodes come with some sample user applications. They show how programming of new applications works and demonstrate most sensor functions in a simple way. Studying and altering the code of these small programs helps understanding the concept of ScatterWeb.

There are three possibilities to exchange an application on an ESB.

- The first one, mentioned earlier in this chapter, is via the JTAG interface. One single node is connected to the PC using a cable and the binary code of the compiled C program is transferred to the ESB. This method can be used to transfer both firmware and programs to the ESB.
- The second possibility is via a USB device called *ScatterFlasher*. The ScatterFlasher is connected to the USB port and with a special driver which comes with the device it is possible to send a new application over the air to a selected node. The nodes can be selected from a list of all active sensors in the communication range of the flasher. This method can be used to transfer programs only to the ESB.
- The third possibility is mainly useful when a lot of sensors have to be flashed. The desired application is transferred to one single node (using method one or two) and then broadcasted from this sensor over the air to all nodes in its range. The distribution of a program has two phases. In a first phase all application packets are broadcasted. In a second phase each node can request retransmissions of missing packets. This method can be used to transfer programs only to the ESB.

Callbacks An important part of all applications running on ScatterWeb nodes are callback functions. Callbacks are invoked whenever a special event occurs. The callback functions have to be registered for a certain kind of events—new value of a sensor, a radio packet was received, data arrived over the serial interface. This is usually done at initialization. A special callback function can be registered for the main-loop. This callback is invoked in each main-loop iteration and can be used for periodical tasks.

Sending and Receiving Data The firmware provides methods for sending and receiving radio packets to and from other sensor nodes and a callback can be registered for the event of an arriving packet. Each node has an identifier. This ID is used for the addressing of radio packets. Firmware layer functions guarantee that only the addressed nodes receive a packet.

Development Toolkit The ScatterWeb toolkit contains a C programming environment, a C compiler, a tool for flashing ESBs via the serial interface and a tool for over the air flashing. The applications are written in C. It is also possible to alter the firmware but this requires recompiling of all applications with the changed firmware. Another useful tool for developing larger applications is the debugger which allows stepping through an application on a connected sensor node. Unfortunately not included in the toolkit is an emulation environment for testing new applications directly on the development computer.

4.2 Sample Applications

In this section we present five applications implemented with ScatterWeb nodes. The first three appliances were created to become familiar with the programming environment, the possibilities and limitations of embedded sensor boards, and the concept of distributed autonomous nodes. The fourth application is an alarm which detects vibrations and the fifth application is a prototype of a system to control lamps in a room with a radio remote control.

RadioTester This application can be used to test the radio range of an ESB. It consists of two nodes. One of them, called *base station* continuously sends radio packets, the other node called *receiver* counts how many packets it receives in a certain amount of time. If the number of received packets drops under a defined level, this is indicated by changing the status from green to yellow. If no packets at all arrive in a time slot, the status changes to red. With this application it is possible to see what influence walls or other obstacles in a building have on the range of a node. And we used it to find out at how far the signal of a node can be received with different transmission powers.

InTheMiddle We wrote this small application to see if adjusting the sending radius of a node can be used to determine the node's position relative to other nodes. The set of nodes contains three ESBs. They are placed on a straight line. Then the node in the middle (M) tries to find out if it is closer to the node on its left hand side (L) or the node on its right hand side (R). To determine its relative position, M sends out test packets with a continuously reduced transmission power. L and R reply with maximal transmission power if they receive a test packet. This process goes on until R or L does not reply any more. If that happens, M knows that it is closer to the sensor still replying. The application works quite well and can determine the position of node M relative to L and R for node arrangements with $\min(\overline{LM}, \overline{MR}) \lesssim \frac{\overline{LR}}{3}$.

LightsOn This application shows how sensor events can be interpreted using a callback function and how newly defined packets can be sent to other nodes. The appliance consists of two nodes. One is called *lightSensor* the other *lamp*. Whenever the sensed intensity of light at the lightSensor falls under a certain predefined value the lightSensor sends a packet via radio to the lamp. The lamp receives the packet and in a callback function the three LEDs are turned on. If the light intensity at the sensor rises again, the LEDs are turned off.

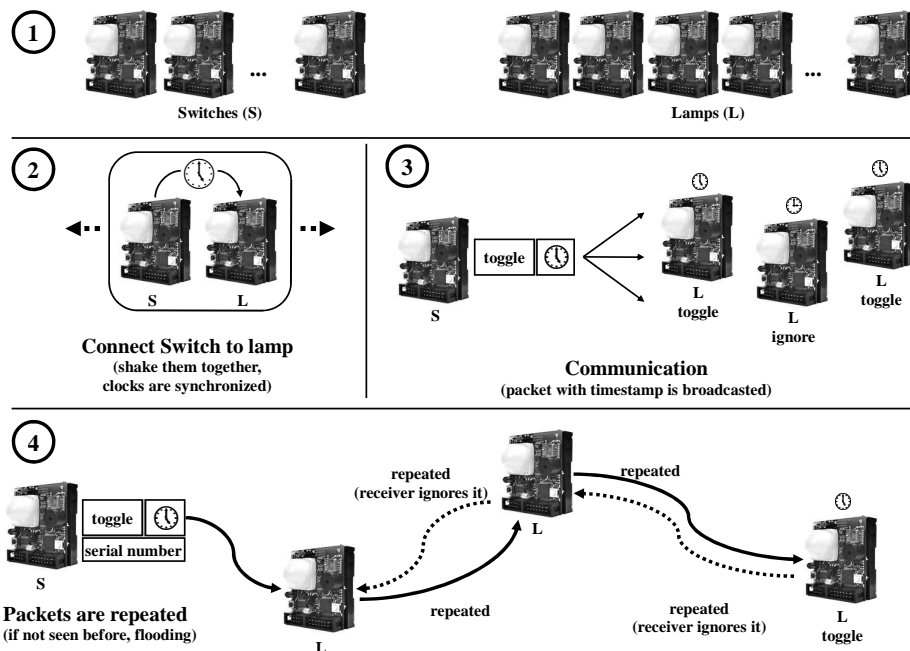


Figure 4.2: A schematic illustration of the RLS concept.

Alarm This system of sensors is a prototype of a burglar alarm. A set of sensors—distributed in a house or a museum—detects vibrations. Whenever a single sensor detects a vibration with an intensity higher than a defined threshold, it goes into alarm state, flashes its red LED and tries to inform other sensors via radio. For the information of other sensors a flooding algorithm is used. A so informed sensor goes into alarm state as well.

The interesting part of this prototype is its concept to save energy. All sensors periodically go to a low-power state (sleep) in which they are able to detect vibrations but do not listen to radio signals. With wake to sleep ratios $\frac{wake}{sleep} \simeq 0.01$ this method helps to save a lot of energy which in turn extends the lifetime of the nodes. To guarantee that an alarm signal reaches all nodes, a node remains in alarm state and tries to inform other nodes for a time slightly longer than the sleeping time of the nodes.

Over the serial interface the sleeping time of nodes can be adjusted and the transmission power can be set.

RemoteLightSwitch The (RLS) Application is a prototype of a concept for a light switch and lamp system communicating via radio. As can be seen in Figure 4.2, Illustration 1 the system consists of light switches and lamps. In our prototype both components are implemented with ScatterWeb sensor nodes. The button of the lamp nodes is used to toggle the connected lamps,

and the LEDs of a lamp indicate its status.

The idea of the system is to give its user the possibility of moving and rearranging light switches in a building. Each switch can control any lamp, but only the owner of a lamp has the right to turn it on and off. In reality the switches work with battery power whereas the lamps would be permanently fixed and connected to the power supply system. This implies that the switches need to keep their energy consumption as low as possible. The lamps on the other hand can afford a higher energy consumption. The power needed for sending messages and storing data can be neglected compared to the energy the bulb itself consumes.

The requirement that only the owner of a lamp is allowed to control it is addressed with the following strategy.

- Before a switch is allowed to control a lamp it has to be shaken together with the lamp. This guarantees that only the owner of the lamp *and* the switch can control the lamp. As can be seen in Figure 4.2, Illustration 2, whenever a lamp and a switch are shaken, the lamp sets its local copy of the switch-clock to the received time. From now on this copy is continuously updated and kept accurate.
- Each time a switch sends a command to a lamp it adds its local time to the radio packet. A lamp only accepts an instruction if the received time is equal—up to a certain accuracy—to its locally running clock for the sending switch. This strategy prevents reply attacks where an attacker records a command from the owner of a lamp and retransmits it later. Of course this security measure requires that the time information is encrypted, which we neglected for our prototype system. Figure 4.2, Illustration 3 shows the process of sending a toggle packet with an added time stamp. To of the three depicted lamps accept the toggle command and one ignores it.

Another requirement for a usable system were long switch lifetimes with one battery set. We solved this problem by implementing a forwarding mechanism. This allows us to lower the transmission power of a switch and therefore to save energy. The command does not need to reach all lamps it is intended for; reaching only one arbitrary lamp is sufficient. In a future version of the system the switch should be able to find out on its own what transmission power is needed to reach the nearest lamp. This lamp then forwards the command to all lamps with maximal transmission power. For the forwarding we use a classical flooding algorithm where packets are only forwarded if they are new to a lamp. Each lamp therefore remembers the highest packet serial number it received for each switch in the set of nodes. A packet is only repeated if its number is greater than the stored one. Figure 4.2, Illustration 4 shows how the implemented flooding algorithm works.

Because in a radio based system packets can be lost, we added a function to the system which allows to switch off—instead of toggling—all lamps a light switch is controlling. This function brings all lamps to the same state which would be impossible with toggle commands only.

Future work could replace this function by a strategy in which all lamps keep track of the other member in their groups and synchronize all states according to a majority decision. Other future addons to the system could be possibilities to dim lamps, handing over a whole group of lamps from one switch to another without the need of shaking dozens of lamps, the already mentioned encryption of radio packets, keeping the energy consumption even lower by letting switches sleep while they are not used and a scheme of acknowledgements which prevents loss of packets.

We tested our system, consisting of 2 switches and 6 lamps in three rooms of a building. The toggling of lamps as well as the forwarding of packets worked very well. Only the coupling of a switch to a lamp was not really user-friendly and failed from time to time. This implies that for a real world application the shaking of switch and lamp should be replaced. Other possibilities would be the simultaneous pressing of a button, physical contact or an infrared signal from the switch to the lamp.

4.3 Conclusion

We presented three demo applications and two prototypes implemented on ScatterWeb nodes. While the first three programs had the purpose of getting familiar to the whole ScatterWeb system, the fourth one showed clearly what different and complex tasks can be tackled with autonomous wireless nodes.

Sensor nodes compared with some actors can be used for much more than simple data gathering. We know that our prototype is only a first step towards a usable system of radio controlled light switches but nevertheless we proved the feasibility of such a system.

The implementations presented in this section show how manifold applications can be implemented with ScatterWeb nodes. Our experiences with the system brings us to the conviction that ScatterWeb is a very powerful system for prototyping and testing of concepts.

Chapter 5

Conclusion

In the first theoretical part of this report we introduced an explicit, receiver-centric definition of interference in sensor networks. It best reflects the fact that message collisions prevent proper message reception only if they occur at the receiver. Based on this definition, we then presented and proved that in certain instances of sensor networks—with specially arranged sensor nodes—no algorithm can construct a sink tree with interference less than logarithmic in the number of nodes.

The NCC algorithm, as described in Chapter 2 asymptotically matches this lower bound. In this sense we found a new worst-case optimal solution for the problem of finding a minimal interference sink tree (MIST) in a given sensor network.

The Nearest Component Connector algorithm keeps interference in a network on a low level, which results in less packet retransmissions and helps to reduce energy consumption. On the other hand NCC does not try to replace long arcs by short ones which would additionally help to save battery power. We therefore consider our work to be a first step towards extending lifetimes of nodes in sensor networks.

In addition to the worst-case observations we evaluated the NCC algorithm in average networks. We therefore implemented a small simulation environment in Java, which gave us the possibility of comparing different construction methods for sink trees. Intriguingly the results showed that—although the interference values produced by NCC fall roughly in the same range as those of other constructions—a simple minimum-spanning-tree-based structure keeps interference at a lower level than NCC in average-case networks.

In Chapter 4 we presented—in addition to some demo applications which we implemented to become familiar with ScatterWeb—a prototype of a remote light switch. This application clearly shows how powerful ScatterWeb can be for prototyping concepts of future real world applications. Our appliance demonstrates sensor-data processing, radio communication, and

concepts for packet repeating—all important parts of sensor and ad-hoc networks—in a running system.

Altogether this report demonstrates how the lifetime of sensor nodes can be extended, what methods should be used for the generation of communication infrastructures in sensor networks, and what future applications based on autonomous sensor nodes could look like.

Bibliography

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38:393–422, 2002.
- [2] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Does Topology Control Reduce Interference? In *Proc. of the 5th ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MOBIHOC)*, 2004.
- [3] C.-Y. Chong and S. P. Kumar. Sensor Networks: Evolution, Opportunities, and Challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.
- [4] 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 Communications Societies (INFOCOM)*, 2003.
- [5] Computer Engineering and Networks Laboratory (TIK) and the Research Group for Distributed Systems at ETH Zurich. BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks. <http://www.btnode.ethz.ch/>.
- [6] R. Cristescu, B. Beferull-Lonzano, and M. Vetterli. On Network Correlated Data Gathering. In *Proc. of the 23rd Conference of the IEEE Communications Society (INFOCOM)*, 2004.
- [7] D. Estrin et al. *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*. The National Academies Press, Washington DC, 2001.
- [8] Freie Universitaet Berlin, Fachbereich Mathematik und Informatik. Scatterweb Project. <http://www.scatterweb.net>.
- [9] A. Goel and D. Estrin. Simultaneous Optimization of Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

- [10] T. Hou and V. Li. Transmission Range Control in Multihop Packet Radio Networks. *IEEE Trans. on Communications*, 34(1):38–44, 1986.
- [11] L. Hu. Topology Control for Multihop Packet Radio Networks. *IEEE Trans. on Communications*, 41(10), 1993.
- [12] 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.
- [13] 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.
- [14] F. Meyer auf der Heide, C. Schindelhauer, K. Volbert, and M. Grünewald. Energy, Congestion and Dilation in Radio Networks. In *Proceedings of the 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2002.
- [15] K. Moaveni-Nejad, W.-Z. Song, W.-Z. Wang, Y. Wang, and X.-Y. Li. Low-Interference Topology Control for Wireless Ad Hoc Networks. In *Proc. of the 1st Int. Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks (TAWN)*, 2004.
- [16] 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)*, 2000.
- [17] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE J. Selected Areas in Communications*, 17(8), 1999.
- [18] H. Takagi and L. Kleinrock. Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. *IEEE Trans. on Communications*, 32(3):246–257, 1984.
- [19] University of California Berkeley. Berkeley Motes and TinyOS. <http://www.tinyos.net/>.
- [20] Y. Wang and X.-Y. Li. Localized Construction of Bounded Degree Planar Spanner. In *Proc. of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2003.
- [21] 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)*, 2001.