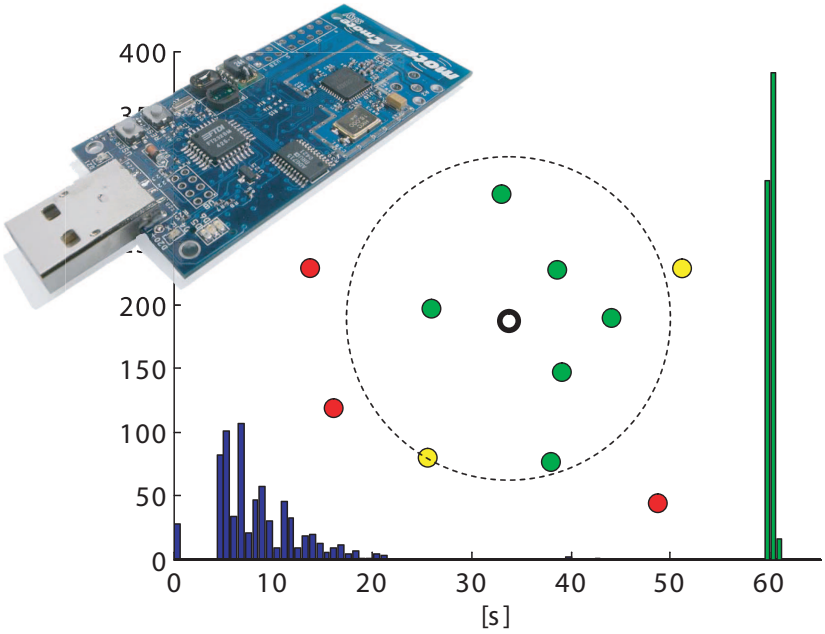
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo

Institut für Technische Informatik
und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Energy Efficient Initialization of Wireless Sensor Networks

Mischa Weise



MASTER THESIS

Summer Term 2007

Supervisor: Andreas Meier
Co-Supervisor: Jan Beutel
Professor: Dr. Lothar Thiele

October 2007

Contents

Tables

Figures

Abstract

This thesis presents NoSE, Neighbor Search & Estimation - an energy-efficient and fast initialization protocol for wireless sensor nodes. In addition to a reliable neighbor discovery NoSE conducts a link-quality estimation of discovered neighbors at the same time. During the deployment and arrangement of the wireless sensor network energy is saved by using a very low duty cycle with passive low power listening. When the deployment is done, the network is centrally induced with a wake-up call announcing a following discovery to all sensor nodes. The sensor nodes can thus be parameterized for an energy-efficient and fast discovery.

The protocol was implemented under TinyOS on the Tmote Sky - a state of the art sensor node. It has been tested in an indoor, office-like environment and an analysis of the test results regarding energy-efficiency and link-quality-estimation success has been conducted. It is shown that in three minutes the test bed can discover all links and gain a good estimation of their quality.

1

Introduction

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices, so called sensor nodes. These sensor nodes monitor physical or environmental conditions such as temperature, sound, or pressure. The sensor nodes have to manage a great deal of limitations. Mostly, the power supply is limited by batteries causing to use low power components. Thus, the low-power radio transceiver has a reduced communication range, the processor a limited computing power, and the memory a downsized capacity. Furthermore, most WSNs have a greater extent than the communication range of a sensor node and therefore require multi-hop routing approaches.

1.1 Motivation

The combination of low-power operation and wireless communications results in unpredictable links between nodes and not deterministic behavior [17, 16, 20]. The complex behavior of low-power wireless networks is a major challenge for routing protocols. Mainly, the unreliability of wireless links has an adverse effect on their performance. Link failures and packet losses occur randomly. Therefore, a careful selection of the used links is likely to increasing the performance of a routing protocol as a bad-quality link leads to many retransmissions and therefore results in higher power consumption.

In common use, sensor nodes are not deployed and turned on simultaneously (or nearly simultaneously), as assumed in many algorithms and protocols. Looking at the deployment of sensor nodes by humans hands (e.g. in a building, on a mountain slope of a volcano [18]), the deployment may take several days to a week. During this time the wireless sensor network is not fully functional and not ready to operate. For instance a single node may not have any neighbors in communication range yet. It is obvious that in this *deployment phase* energy saving is key. As long as the WSN is not completely installed and therefore not yet ready for operation, every amount of energy we save, can be used to operate the WSN a bit longer. With longevity obviously being a target parameter of WSNs.

When all the nodes are deployed there are many things to do before the WSN can operate - we call this the *initialization phase* of the network. The distributed sensor nodes first have to find their neighbors, establish a network and choose an optimal path to the data sink.

Not all available neighbors can be assumed to be well connected. Furthermore, there are usually more then enough links available then required by the routing protocol. Therefore it is possible to choose the neighbors carefully, in particular according their link quality. To minimize packet loss and thus optimize the function of the WSN one wants to categorize the neighbors into different qualities and only communicate with the best available. This is usually done by estimation of the link quality to one's neighbor, but can be augmented with information of the connectivity of a specific neighbor (a very good link to a neighbor at the border of the WSN might be less important then a good link to a very good connected neighbor toward the center of the WSN).

It is the objective of this Master's Thesis to implement a draft of an energy efficient initialization protocol for wireless sensor networks that copes with the specific demands of the deployment phase of wireless sensor networks and establish a profound base for the consecutive operational phase of the application, most importantly neighbor discovery and link estimation.

1.2 Chapters Overview

This thesis is divided into six parts. This introduction forms the first. In Chapter 2, we give an overview of related work, by analyzing and comparing them. Chapter 3 describes our approach to a neighbor discovery and link-quality estimation protocol design. Furthermore, an implementation of this protocol is described in detail in Chapter 4 and analyzed from different aspects in Chapter 5. Finally, Chapter 6 concludes this thesis and gives an outlook on future work.

2

Related Work

2.1 Sensor MAC protocols

Almost all of the published protocols deal with data transmission problems. They are not concerned with the specific problems of the initialization, but focus on the operational optimization. While there are lots of interesting concepts that work for the operational phase, these concepts have to be considered and even may be adapted to form a optimized protocol for the initialization phase.

They can be roughly divided into random access (CSMA) or predefined schedule (TDMA) based protocols. Predefined schedule based protocols like the LMAC protocol proposed by van Hoesel et al. [9] need to organize slot assignment and synchronization. On one hand, this overhead is not needed for an initialization on the other hand, one of the main tasks of the initialization is to first find all neighbors - though, without knowledge of neighbors, the organizing of slot assignment and synchronization is superfluous anyway.

Random access MAC protocols like B-MAC by Polastre et al. [14] are more easily adapted to form an initialization protocol. B-MAC is a carrier-sense media access protocol that uses a preamble sampling scheme for asynchronous low power listening. It is used in the implementation of the radio stack on TinyOS [15], the embedded systems operation system we used for our implementation. X-MAC by Buttner et al. [3] improves B-MAC's excess energy consumption because of a long preamble by using a shortened preamble approach and additionally reduces excess energy consumption at non-target listeners. This improves the advantages of low power listening. Wise-MAC is a synchronous low power listening protocol proposed by El-Hoiydi et al. [7]. It is based on a non-persistent CSMA and uses preamble sampling to minimize power consumption when idle. Transmit and receive power consumption are reduce by exchanging the sampling schedule with one's direct neighbor.

2.2 The Birthday Protocol

The Birthday Protocols are a group of simple low energy deployment and neighbor discovery protocol proposed by McGlynn et al. [12]. They address two problems associated with static ad hoc wireless networks: methods of saving energy during a deployment of the nodes, and efficient methods of performing adjacent neighbor discovery.

Random independent transmissions are used to discover adjacent nodes and a mathematical model and analysis of the protocols is provided. By analysis and simulation it is shown that the birthday protocols are a promising tool for saving energy during the deployment of an ad hoc network as well as an efficient and flexible means of having the nodes discover their neighbors. In Chapter A the implementation of one of these protocols is described. We use it to have a reference to compare our newly designed initialization protocol with (in Section 5.3.1).

2.3 Initialization Protocols

Kuhn et al. [10] propose setting up a clustered network structure without any central control and analyze this in theory. The drawback of a clustered network structure is that some nodes require much more energy. This would lead to either the need to deploy sensor nodes only for the initialization phase, or to early failure of some sensor nodes due to lack of power.

Energy efficiency during the non-operational phase of the network is investigated by Moscibroda et al. [13] for three different algorithms (of which one is the Birthday Protocol by McGlynn as mentioned above) in a theoretical work.

2.4 Neighbor Discovery Protocols

Zheng et al. [21] presents a systematic approach to designing and implementing asynchronous node wakeup mechanisms in ad hoc networks. A neighborhood discovery protocol based on frames (referred to as block) divided into slots is evaluated.

A theoretical study of the impact of interferences on the neighbor discovery process has been done by Hamida et al. [8]. They use a random hello protocol (inspired by aloha) for neighborhood discovery.

Link Assessment (neighborhood discovery and link estimation in one) is introduced by Keshavarzian et al. [1] and mathematically analyzed with two different approaches. It is assumed that time is slotted and nodes are synchronized (in particular synchronous start-up), which we think is difficult to achieve in a deployment of a wireless sensor network.

2.5 Link-Quality Estimation

An investigation of estimators for determining link-quality has been presented by Woo et al. [19]. The preceding master thesis by T. Rein [17] focuses on link-measurements and link-estimation in great detail and shows that effective estimation algorithms use adaptive number of packets in combination with RSSI values.

3

NoSE - A Neighbor Discovery & Estimation Protocol

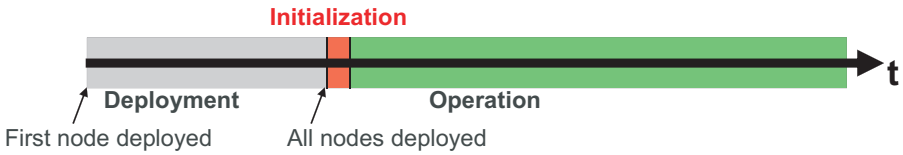
We present the design for our Neighbor Discovery & Estimation Protocol in this chapter. First, we describe the overall concept. Later, on we go deeper into the detailed protocol design.

An initialization protocol needs to consider following three basic issues: First, energy consumption has to be minimized to save energy for the operational phase of the WSN. Second, a single node has first to discover his neighbors before he can act as part of the WSN. Third, we want to find all neighbors and estimate their link-quality in order to choose a set of reliable neighbors for routing purpose.

The focus of the NoSE protocol lies on a very low energy consumption during the *deployment phase* with passive low power listening and an energy efficient and fast Neighbor Discovery and Estimation during an centrally initiated *initialization phase* before the operation of the WSN is started.

3.1 Protocol Concept

We divide the lifetime of a WSN into the three following phases:



Deployment Starts with the first node installed and powered. All the sensor nodes are being deployed at their target location. This phase ends when all nodes are on place - the WSN is fully deployed.

Initialization The WSN has been deployed and a start sequence is initialized at a central point (e.g. the data sink). This phase ends when the WSN is ready for its designed operation.

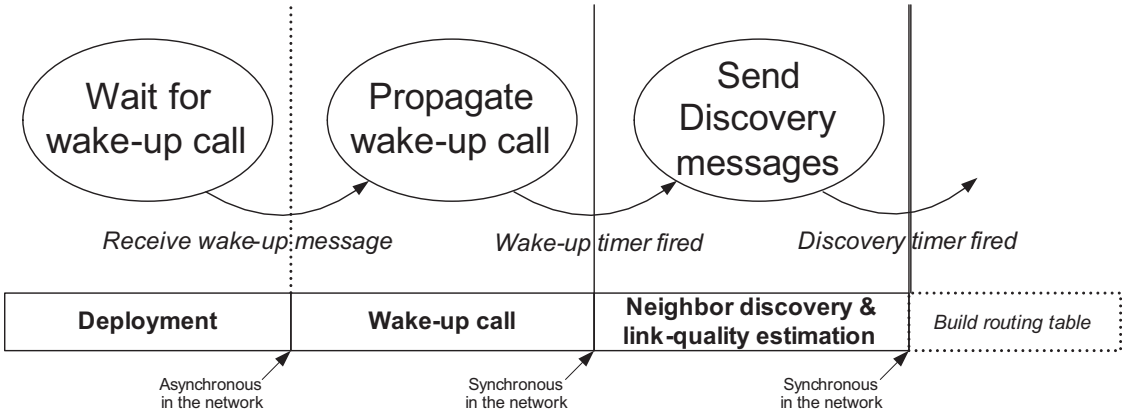
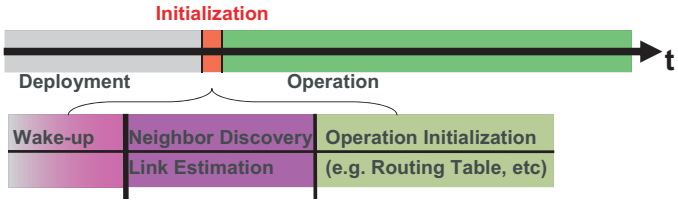


Figure 3-1
NoSE state diagram

Operation In this phase the WSN does the work it was designed for. It is this phase that we want to be best prepared for by ideally parameterizing the preceding phases.

The **initialization phase** is further divided into several NoSE specific steps:

- Wake-up call
- Discovery (Neighbor discovery & link-quality estimation)
- Preparation for operation



The overall concept is to save energy during the deployment phase by using a low duty cycle for the radio in the order of 0.1%. Then wake-up the network during the initialization phase and synchronously initialize all nodes. Because this happens synchronously, taking only a short amount of time, it is profitable to invest more energy in a good neighbor discovery and estimation. This concept is depicted in the state diagram in Figure 3-1.

At this point it is advantageous to adapt the MAC parameters for a more resource efficient point of operation. Because we know that all nodes will almost synchronously begin transmitting discovery messages an optimal trade-off between bandwidth and energy is much different from the one during passive low duty cycle discovery phase or during the operational phase.

It is not the intent of the NoSE protocol to choose links and build a routing table. This is left to routing protocols. However, NoSE does provide well-assessed neighbor information.

3.2 Protocol Design

The protocol bases on a low power listening based radio stack (such as B-MAC [14]). A node does periodically switch on the radio and senses for a message on the channel staying on just long enough to receive that message and stays off the rest of the time to save energy. Therefore a sender has to send a packet stream for a full period to ensure that every listening node is reached. As the nodes are not synchronized, different nodes switch on at different times and thus receive the same message not at exactly the same moment. This is depicted in Figure 3-2

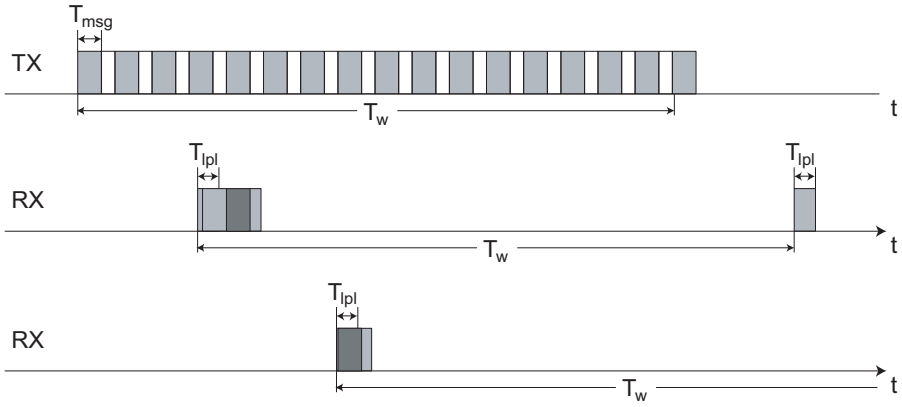


Figure 3-2
Low Power Listening: Transmission in an asynchrony LPL environment

3.2.1 Deployment

During the *deployment phase* all nodes are in a deep low power listening state. Every T_w (e.g. $T_w = 1s$) a node switches on its radio for T_{lpl} (e.g. $T_{lpl} = 2ms$), polls the energy of the channel and goes back to sleep in case there is no traffic on the channel. Since no node is sending any messages, a duty cycle of T_w/T_{lpl} (e.g. $DC = 0.2\%$) is achieved.

3.2.2 Wake-up call

After all nodes are deployed, a wake-up call is flooded, started centrally at the data sink. Usually this is done manually (e.g. by pressing a button or infiltrating a data packet). Optionally a beacon can be induced from an additional node, not belonging to the network. The purpose of this wake-up call is to inform all nodes of the network that the initialization phase has begun and that they have to prepare for the next initialization step (discovery) at a specified time.

The wake-up messages are propagated outwards from the sink node like a wave: From the sink to its immediate neighbor nodes, then to the next ring of nodes, etcetera. The wake-up call contains the network wide *wake-up time* (discovery start time) and parameters for the following initialization steps.

Information distributed by the NoSE wake-up message contains the following:

- Start time; time until begin of the discovery
- Discovery parameters; duration, low power listening period, ...

- Link estimation parameters; number of packets for link estimation

The goal of the wake-up call is now to reach every single sensor nodes with this information. If this is achieved, all sensor nodes of the network will thus synchronously start the next step.

The wake-up call has to be *fast* and *reliable*. Every single node has to be reached, even in a large multi-hop network where hop delay could slow down a wake-up call. For a fast propagation of the wake-up call, we choose a dense and aggressive first wave of wake-up messages, allowing to rapidly reach the border of the network. To be reliable we choose multiple waves of wake-up messages. So if one node is missed by the first, aggressive wave, then a less aggressive second or third wave will reach the node with high probability.

3.2.3 Neighbor Discovery & Estimation

During this step we want to find all the neighbors of the nodes and estimate the quality of these neighbors.

With the wake-up call we assured that this step is conducted simultaneously by all nodes. All nodes now need to discovery their neighbors and estimate the quality of those neighbors, with the goal that a good set of neighbors can be chosen for routing. The discovery takes a determined amount of time (T_d) during which the following algorithm is used:

- Every node broadcast exactly n discovery messages during T_d.
- When a broadcast is received from a neighbor:
  - Store node in the internal neighbor table.
  - Keep counter of the total received messages from that node
  - Add RSSI value of the message to the node information.

When the n discovery messages are sent is part of the chosen strategy. The only constraint to the strategy is that exactly n messages are sent when the discovery is over (after T_d time). We considered the following strategies:

- Random:
  I. Choose randomly n different sending times $t_{s1} \dots t_{sn}$
- Slotted Random:
  I. Divide T_d into n slots $S_1 \dots S_n$ of length T_d/n.
  II. Choose for each slot S_x a time t_{sx} that lies randomly in this slot.

After the discovery phase each node has a list of its neighbor. For the estimation the following information was gained during the discovery:

Packet Reception Rate (PRR) We know that exactly n messages will be sent during the discovery. From the number of received messages we can calculate the PRR and us it as a measurement for the link quality.

Received Signal Strength Indication (RSSI) To each received message the RSSI is automatically saved in the message's meta data and we use it as a further measurement for the link quality.

After the discovery, these two measures can be used to qualify the link quality to all our neighbors.

With the wake-up call we can parameterize the discovery. We want to economize energy and time by clever choosing the parameter of the MAC-protocol and weight this optimization against the quality of the link estimation. In Chap. 5 we explore some of the parameter space.

4

Implementation

This chapter presents our implementation of the NoSE protocol. First an overview is given over the sensor node we use for the implementation, the Tmote Sky. Then we describe TinyOS, an operating system specially designed for wireless embedded sensor networks. Later on, we describe our implementation of the NoSE protocol.

4.1 Platform

4.1.1 Tmote Sky

The Tmote Sky was developed and designed at the University of California, Berkeley and is a next-generation mote platform (see Figure 4-1). It is an enhancement from the Telos Revisions A/B [15] and is commercially distributed by the Moteiv corporation [24] since 2005. Key features of the Tmote Sky platform are the 8 MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash), the fast wakeup from sleep, the ultra low current consumption, and the TinyOS support.

The radio module on Tmote Sky is the CC2420 [4] by Chipcon [23] a single chip 2.4GHz IEEE 802.15.4 compliant packet-based RF transceiver.

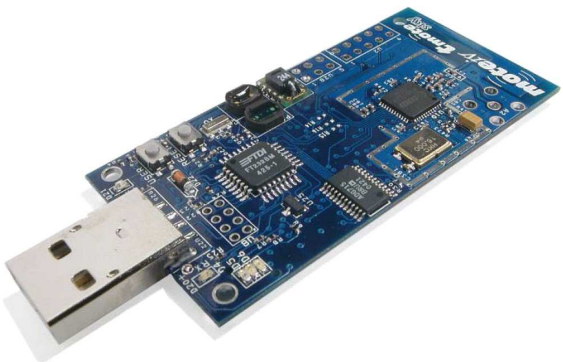


Figure 4-1
Tmote Sky. A next-generation mote platform distributed by Moteiv Corporation.

A special feature of the packet based CC2420 supports time-stamping of sent packets. The transceiver allows modifying a packet in the radio FIFO even when the process of transmitting already has started. Thus a packet can be altered (e.g. for time-stamping).

4.1.2 TinyOS

TinyOS is an event based operating environment designed for use with embedded networked sensors. More specifically, it is designed to support the concurrency intensive operations required by networked sensors with minimal hardware requirements [5]. The programming language of TinyOS is stylized C that uses a custom compiler 'NesC'. The sources for NesC as well as TinyOS are available on SourceForge. TinyOS was initially developed by the U.C. Berkeley EECS Department.

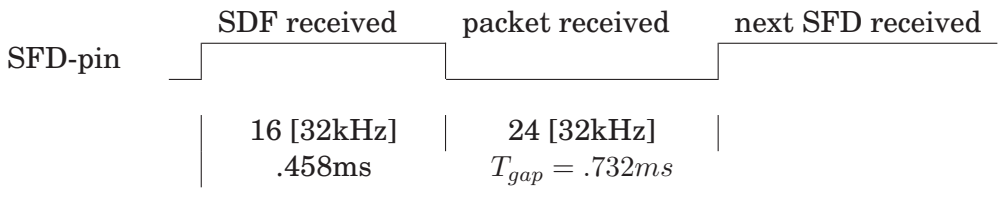
4.1.3 Synchronous Low Power Listening Radio Stack

The TinyOS-2.x radio stack for the CC2420 chip contains an asynchronous low power listening implementation of B-MAC [14] (has to be compiled in through the preprocessor define `LOWPOWERLISTENING`). Roman Amstutz adjusted the TinyOS-2.x LPL-stack in his term project [2] to the synchronous low power listening protocol WiseMac [7].

In Figure 3-2 the simple variant of synchronous low power listening was already shortly introduced: A node is periodically (every T_w) doing a low power listening (a short sampling of the channel for T_{lpl}) and sleeps in between. This short sampling is done with the clear channel assessment (CCA) function of the CC2420 radio chip. The number of CCA-samples `MAX_LPL_CCA_CHECKS` can be set in the header *cc2420/lpl/DefaultLpl.h*. We conducted some measurements of the total on-time of the radio for different CCA values. T_{lpl} is the total time the radio is switched on, the actual duration of the a sampling is less then the measured time:

CCA	T_{lpl} [32kHz]	T_{lpl} [ms]
0	11	0.56
100	97	3.0
400	328	10.0

Because the CC2420 is a packet based radio chip, packet burst are sent and it has to be assured, that the channel is sampled long enough to detect such a burst, even if the radio samples between two packets of a burst: the time T_{lpl} has to be long enough, longer then the time between packets in a burst T_{gap}. The following tests measured the time between a complete reception of the packet and the reception of the SFD of the following packet during a burst:



We choose a save value of 100 CCA samples (results in $T_{lpl} = 3.0$) just to be on the save side that a packet burst cannot be missed by a node when it switches its radio on for T_{lpl} time.

The same has to be considered for the transmission of a packet when a carrier sense (multiple CCAs) is executed before sending the packet burst. To detect a burst form another sender, the carrier sense has to sample for a total of at least $0.75ms$. For this purpose */cc2420/transmit/TransmitP.nc* has been altered to provide the following way: every $0.15ms$ (5 ticks [32kHz]) a CCA is conducted for a total of 6 times. This covers a time of $.92ms$.

If the channel is clear, then the packet burst is sent during T_w (plus one single additional packet to make sure to have covered every listener). In case where the channel is not clear, the sending of the burst is canceled and scheduled for a later time.

4.2 Implementation

The functionality of NoSE was split into two (actually three) parts: *NoseWakeup* to implement the functionality of a wake-up call and *NoseDiscovery* for the discovery and link-quality estimation part. *TestDiscoveryApp* is used as a wrapper to connect the wake-up and discovery part and is the main application.

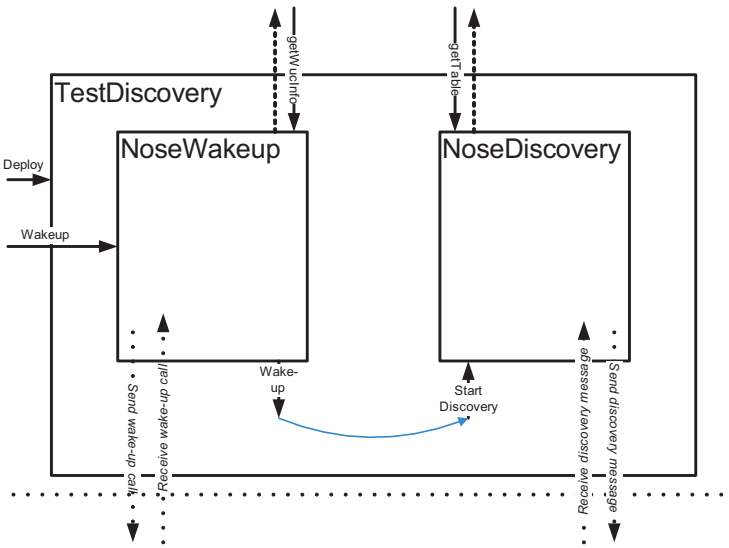


Figure 4-2
Relation between the implementation parts

In figure 4-2 we see those parts. Now a short description of their functionality is given:

- *TestDiscoveryApp*: Application that is run at the beginning. When it receives a `deploy` command the NoSE protocol is initialized with the thus provided parameters and the *NoseWakeup* component activated. Commands and their parameter, as well as log output is described in Table 4-1 and 4-2.

- *NoseWakeup*: Waits for a wake-up call message. An wake-up call is induced at the sink with `wakeup`. When a wake-up call message is received the wake-up timer is started and

  I. The first wake-up call message is sent rather fast (randomly between 0 and $N \cdot T_w$).

  II. The following wake-up call messages are sent (randomly between $N \cdot T_w$ and $2N \cdot T_w$ after the time of last sending)

  III. If the carrier sense reports an congested channel, then a retransmission is scheduled (randomly between $0.5 \cdot T_{lpl}$ and $3T_w$)

  When the wake-up timer is fired the wake-up is signalled to the *TestDiscoveryApp* which start the *NoseDiscovery*.

- *NoseDiscovery*: First the discovery timer is started with the duration of the discovery. Then n sending times $t_{s1} \dots t_{sn}$ are chosen according to the strategy. When we receive a packet, we save the node in the neighbor table and add the RSSI value. If we already have en entry, then the entry is updated and the packet-count is incremented with 1. At a sending time (e.g. t_{s6}) the nodes discovery message is sent. If the channel is not free at the time, a later sending time is rescheduled randomly between 0 and $3T_w$. When the discovery timer is fired, the end of NoSE is signaled to the application.

4.2.0.1 Source Code

The Nose protocol source code is documented and saved under *svn:/ /svn.ee.ethz.ch/siemens-fire/demonstrator/mweise*:

- */devel/NoSe/nose/wakeup/* NoseWakeupC.nc and NoseWakeupP.nc
- */devel/NoSe/nose/discovery/* NoseDiscoveryC.nc and NoseDiscoveryP.nc
- */devel/NoSe/nose/* NoseDiscoveryAppC.nc and NoseDiscoveryC.nc

And the adjusted CC2420 Stack resides in */devel/NoSe/cc2420/**

(a) Commands

Command	***Description***
`deploy`	*Initialize application*
`wakeup`	*Induce wake-up call at sink*
`getState`	*Returns the state of NoSE*
`getWucInfo` Returns:	*Show wake-up information* "WUCINFO_*time*:*@time-fromNode-seqNo-wakeuptime-minDiff-maxDiff-#rcvd-#sent*;..." *@time* Nodetime at first wake-up call message reception *fromNode* ID of the sender node *seqNo* Wave seq-No. of the received message *wakeuptime* Received time till wake-up *minDiff* Min. difference to planned wake-up time if this is not the first message received *maxDiff* Max. difference... *#rcvd* Total number of wake-up call messages received *#sent* Total number of wake-up call messages sent (waves)
`getTable` Returns:	*Show neighbor table* "TABLE_1:*@time-nodeID-lqeRcvd-lastSeqNo-minRssi-maxRssi*;..." "TABLE_2:" second part for long logs... *nodeID* ID of a neighboring node *lqeRcvd* Number of link-quality estiamtion messages received *lastSeqNo* SeqNo. of last received message of this neighbor *minRssi* Min. RSSI detected from this neighbor *maxRssi* Max. RSSI detected from this neighbor When *nodeID* == *thisID*; then *lqeRcvd* is the number of sent packages, *lastSeqNo* is the number of neighbors detected and *maxRssi* is the time in ms that the radio was switched on.

(b) Output

Output	***Description***
`START_[time]`	*logged when a* `deploy` *command has been received*
`GOTWUC_[time]: ...`	*Logged when a wake-up call message has been received, or directly induced with* `wakeup` .
`WAKEUP_[time]`	*Logged when the node starts with the discovery phase*
`DISCOVERYDONE_[time]`	*Logged when the node is done with the discovery phase*

Table 4-1: Commands and log output for the NoSE protocol on the DSN.

(a) deploy

Parameter	Description	Comments
W	Number of wake-up message to send per node	
P	Number of discovery messages to send for link-quality estimation	
N	Number of estimated neighboring nodes	used for backoff calculation
R_{tx}	Radio transmission power	between with 0 =off 31 =full

(b) wakeup

Parameter	Description	Comments
T_{wuc}	Time until wake-up (start of discovery phase)	in $10ms$
T_d	Duration of discovery phase	in $10s$
T_w	Low power listening period	in ms

Table 4-2: Parameter description for the deploy *and* wakeup *command.*

5

Analysis

In this chapter we analyze the performance of the NoSE protocol implementation. First an overview of the setup of the testbed and the link quality between the nodes is given. Then we show the effectiveness of a wake-up call. Further, we compare the discovered neighbors of the NoSE protocol with the ones discovered by the Birthday Protocol. To gain a first hint of the energy efficiency we do a theoretical analysis of the NoSE protocol, then compared it with our measurements and extrapolate the relations of the parameters. Finally we sum up the analysis and draw our conclusion.

5.1 The Test Bed

We tested and analyzed all applications in an office-like indoor environment: The Deployment Support Network (with Tmote Sky target nodes) installed in the ETZ building of the Electrical Engineering Department, ETH Zurich. In Figure 5-1 we see the distribution of the sensor nodes.

Key numbers of the setup are:

- 25 Sensor Nodes (Tmote Sky)
- 224 Links
- Longest Path: 6 Hops

5.1.1 Deployment Support Network - DSN

Development of applications on WSNs faces many difficulties. Testing and debugging of embedded systems per se is hard, and an application that is running on many wireless sensor nodes is even harder to do so. Reprogramming and monitoring is time consuming when each node needs to be programmed individually.

The Deployment Support Network (DSN) [6] is a toolkit to help develop WSNs. With a connection to the DSN we can connect and control all the sensor nodes of the WSN. The DSN itself is a wireless sensor network based on BTnodes [22] using Bluetooth

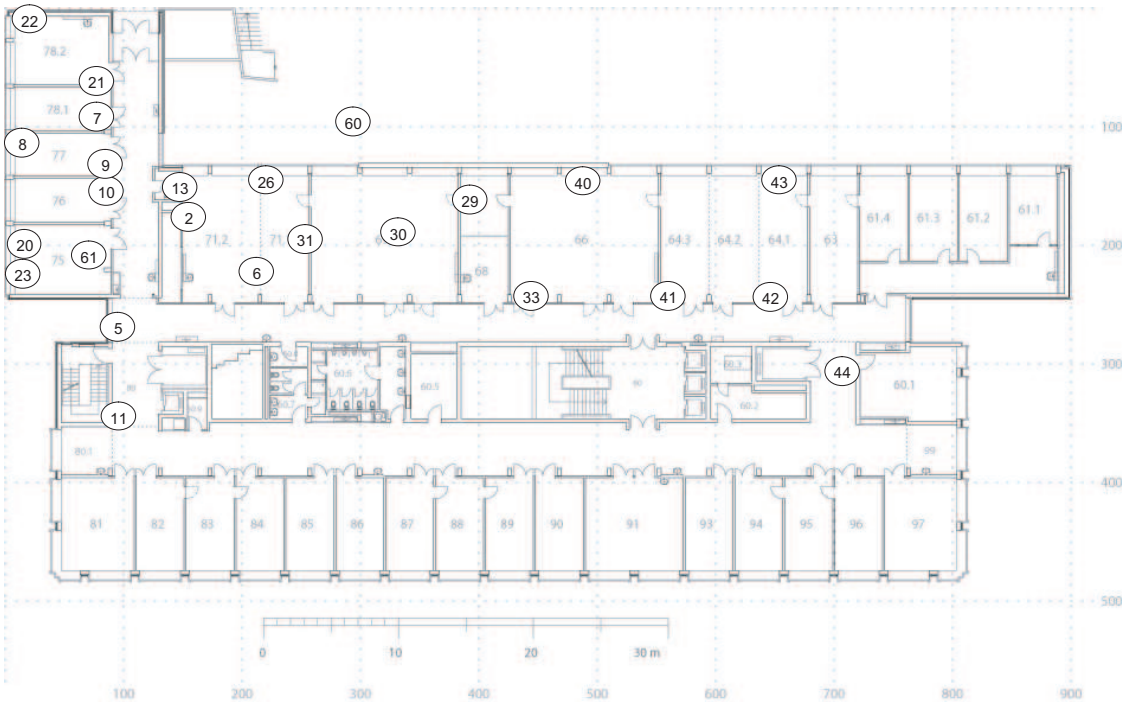


Figure 5-1
The node placement of our test bed on the ETZ G-Floor, ETH Zurich

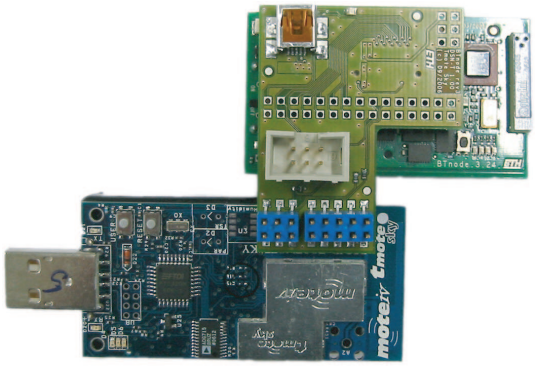


Figure 5-2
A Tmote Sky is connected to a BTnode (above) from the DSN

to build a multi-hop network. All BTnodes (except one root BTnode) are attached to a sensor node of the WSN (referred to as target node) by using an adapter. This is shown in Figure 5-2. The DSN builds a second, independent and stable overlying wireless network (see Figure 5-3). On this reliable overlay network commands and messages can be sent to and received from each individual target sensor node. In a previous master's thesis by Roman Lim [11] the adaptation of the DSN to the Tmode Sky platform has been done and he has implemented a TinyOS interface for easy communication with the DSN.

The Tmote setup at the ETH ETZ-Floor is connected to such a DSN. With the help of the DSN it makes the task of programming, testing and analyzing the implemen-

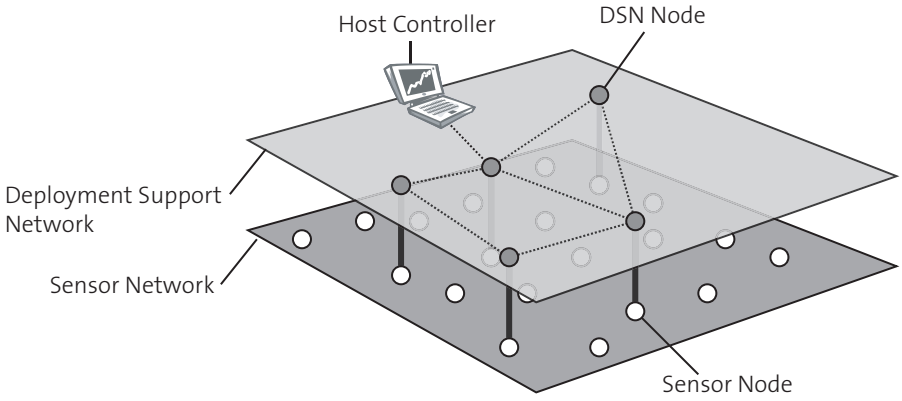


Figure 5-3
Deployment Support Network (DSN): A wireless overlay network to monitor and control target sensor network.

tation of the NoSE protocol much easier: (1) The application can be deployed on all target nodes (the Tmote Sky nodes) or on a specified subset. (2) Commands can be sent to the application on each sensor and thus one can interact with each individual sensor node during runtime. (3) Log messages generated by the target nodes are stored centrally in a database where they can be retrieved for detailed analysis later.

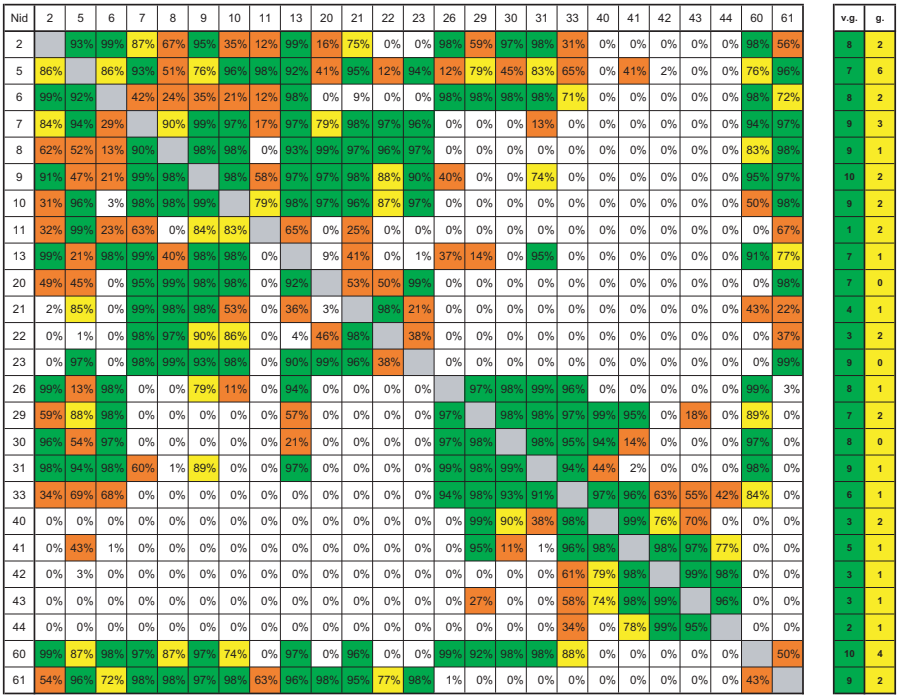
5.1.2 Link-Quality of the Setup

For the analysis of the link-quality estimation of our protocol, we need to have an accurate understanding of the link-quality in our setup. We used an adjusted version of our implementation of McGlynn et al.'s Birthday Protocol to test the links between the nodes of our setup (see Appendix A, and for details regarding the link testing see Section A.3).

Several measurements of the link-quality were executed and this at different time (every other day between other test runs). In figure 5-4 the average link-quality of all this link tests is shown. When we look at subsets, measured at different time, then we see that some good links deteriorated and new links show up. We see that in the deepest of night there are less links found then during other hours but most of them are really good links. On the other hand, in our measurements during afternoon and evening hours show the mentioned change in quality (e.g. see sensor node 2 in Figure 5-5 (a) and (b)).

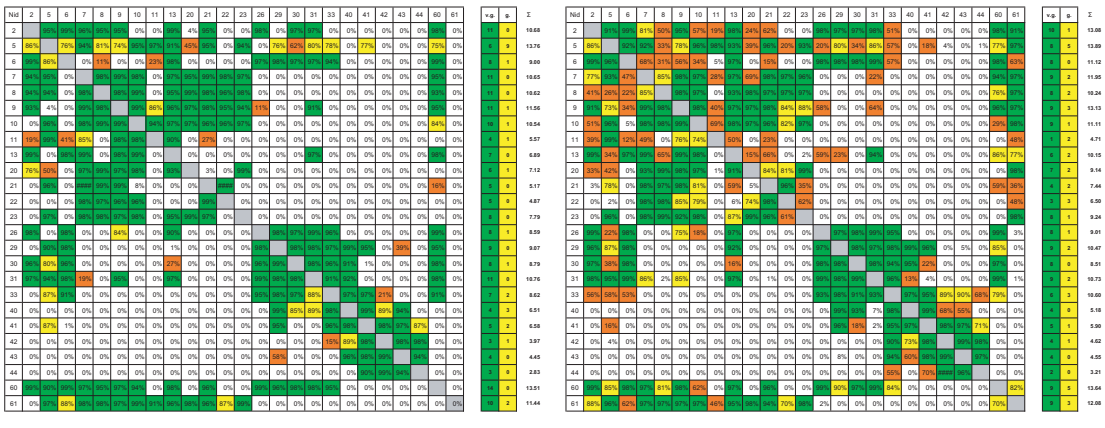
We assume that this is due to open office doors and presence of people during these hours. This changes the environment and the wave reflection and propagation slightly. Additionally, closing and opening doors and people moving around may change the quality of a link during measurements.

All in all a link-quality measurement or estimation does want to find those links that are not greatly affected by these chance events over time. So one can argue that only the very good links over all the link measurements are to be considered as high-quality links.



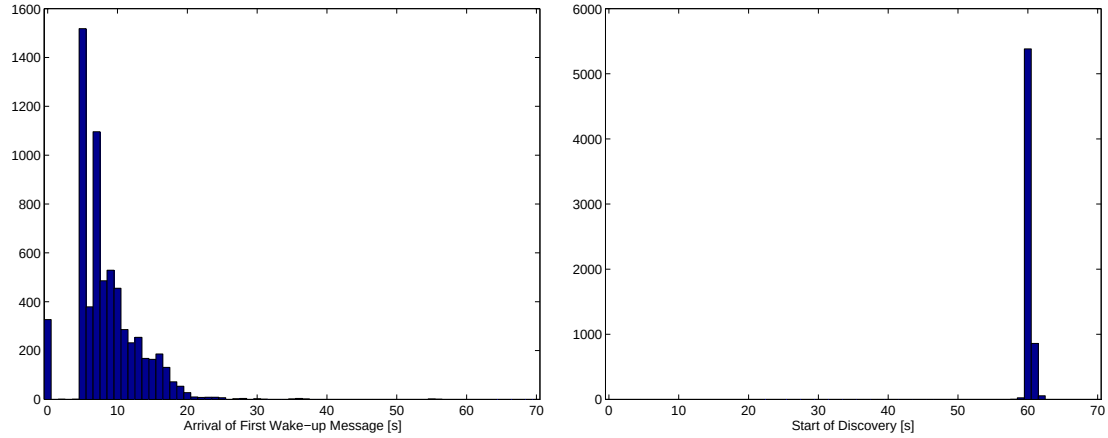
Nid	2	5	6	7	8	9	10	11	13	20	21	22	23	26	29	30	31	33	40	41	42	43	44	60	61	v.g.	g.
2		93%	99%	87%	67%	95%	35%	12%	99%	16%	75%	0%	0%	98%	59%	97%	98%	31%	0%	0%	0%	0%	0%	98%	56%	8	2
5	86%		86%	93%	51%	76%	96%	98%	92%	41%	95%	12%	94%	12%	79%	45%	83%	65%	0%	41%	2%	0%	0%	76%	96%	7	6
6	99%	92%		42%	24%	35%	21%	12%	98%	0%	9%	0%	0%	98%	98%	98%	98%	71%	0%	0%	0%	0%	0%	98%	72%	8	2
7	84%	94%	29%		90%	99%	97%	17%	97%	79%	98%	97%	96%	0%	0%	0%	13%	0%	0%	0%	0%	0%	0%	94%	97%	9	3
8	62%	52%	13%	90%		98%	98%	0%	93%	99%	97%	96%	97%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	83%	98%	9	1
9	91%	47%	21%	99%	98%		98%	58%	97%	97%	98%	88%	90%	40%	0%	0%	74%	0%	0%	0%	0%	0%	0%	95%	97%	10	2
10	31%	96%	3%	98%	98%	99%		79%	98%	97%	96%	87%	97%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	50%	98%	9	2
11	32%	99%	23%	63%	0%	84%	83%		65%	0%	25%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%	1	2
13	99%	21%	98%	99%	40%	98%	98%	0%		9%	41%	0%	1%	37%	14%	0%	95%	0%	0%	0%	0%	0%	0%	91%	77%	7	1
20	49%	45%	0%	95%	99%	98%	98%	0%	92%		53%	50%	99%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	98%	7	0
21	2%	85%	0%	99%	98%	98%	53%	0%	36%	3%		98%	21%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	43%	22%	4	1
22	0%	1%	0%	98%	97%	90%	86%	0%	4%	46%	98%		38%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	37%	3	2
23	0%	97%	0%	98%	99%	93%	98%	0%	90%	99%	96%	38%		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	99%	9	0
26	99%	13%	98%	0%	0%	79%	11%	0%	94%	0%	0%	0%	0%		97%	98%	99%	96%	0%	0%	0%	0%	0%	99%	3%	8	1
29	59%	88%	98%	0%	0%	0%	0%	0%	57%	0%	0%	0%	0%	97%		98%	98%	97%	99%	95%	0%	18%	0%	89%	0%	7	2
30	96%	54%	97%	0%	0%	0%	0%	0%	21%	0%	0%	0%	0%	97%	98%		98%	95%	94%	14%	0%	0%	0%	97%	0%	8	0
31	98%	94%	98%	60%	1%	89%	0%	0%	97%	0%	0%	0%	0%	99%	98%	99%		94%	44%	2%	0%	0%	0%	98%	0%	9	1
33	34%	69%	68%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	94%	98%	93%	91%		97%	96%	63%	55%	42%	84%	0%	6	1
40	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	99%	90%	38%	98%		99%	76%	70%	0%	0%	0%	3	2
41	0%	43%	1%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	95%	11%	1%	96%	98%		98%	97%	77%	0%	0%	5	1
42	0%	3%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	61%	79%	98%		99%	98%	0%	0%	3	1
43	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	27%	0%	0%	58%	74%	98%	99%		96%	0%	0%	3	1
44	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	34%	0%	78%	99%	95%		0%	0%	2	1
60	99%	87%	98%	97%	87%	97%	74%	0%	97%	0%	96%	0%	0%	99%	92%	98%	98%	88%	0%	0%	0%	0%	0%		50%	10	4
61	54%	96%	72%	98%	98%	97%	98%	63%	96%	98%	95%	77%	98%	1%	0%	0%	0%	0%	0%	0%	0%	0%	0%	43%		9	2

Figure 5-4
The average link-quality of the test bed measured with the Birthday Protocol link test. This is the average over all link tests (at 4 different times, 12 instances total). On the right we see the number of very good links (v.g., link-quality of 90% *or higher) in green and the number of good to mediocre links (g., link-quality of* $70 - 90\%$ *in yellow).*



(a) Night (b) Afternoon and evening

Figure 5-5
This shows the difference between link tests in the depth of night (a) and during the afternoon and evening hours (b).



(a) Arrival time of first wake-up

(b) Wake-up Time

Figure 5-6
Histogram of the Wake-up call with wake-up time of $60s$*, listening period* $Tw = 1s$*. First wake-up message is sent randomly between* 0 *to* $5s$ *after receipt of a wake-up message. The second message is sent between* 5 *to* $10s$ *after sending of the first. At* $t = 0s$ *at the data sink a wake-up message is induced to wake-up the whole WSN.*

5.2 Wake-up Call

In order to verify the functionality and accuracy of the wake-up call we gathered data of the wake-up calls of all the tests. In the Figure 5-6(a) we show the arrival time of the first wake-up message at each sensor node. The arrivals at $t = 0s$ represent our wake-up message manually induced at the data sink. The constant gap at the beginning is due to the pseudo-randomness of the sending time, which is always the same random number for one node (as the seed is constant for a node). Because we have chosen only one node to act as sink node, the first message is therefore sent at a constant time in our tests, but this time is “randomly” chosen.

We see that in a first aggressive wave many nodes are reached. Actually in our network there is never a case where a node is not reached by the first aggressive message. Node B may miss a message of node A because of interference or a bad link, but there is a high probability that nodes C and D (both common neighbors of A and B) have heard the message from A and later node B receives one of the first messages sent by either C or D.

Due to the inaccurateness of time measurement in the DSN overlay network we cannot exactly define the accuracy of the actual wake-up (the start of the discovery phase). Directly monitoring global time on the target nodes of the DSN is impossible. The best we can do is reset all nodes simultaneously and then assume a common reset time of the internal timers of the sensor nodes and calculate the time delays from this common point. We can make sure that a command reaches all sensor nodes, but we cannot do so simultaneously (due to multi-hop delays in the DSN overlay network). For all this an accurate measurement is currently not possible with the DSN.

The actual wake-up (the start of the discovery) of the sensor network happens at

the planned time with the accuracy of about a second, as depicted in Figure 5-6(b). Measurements during development suggest that the wake-up call is much more accurate then measured on the test bed. For the purpose of the wake-up call, to synchronously start a (intense) discovery on all sensor nodes, an accuracy in the magnitude of a second is sufficient.

5.3 Discovery

This section covers the analysis of NoSE's discovery phase.

5.3.1 Comparison with the Birthday Protocol

Here we compare the NoSE implementation with the Birthday Protocol (described in Section 2.2 and implemented as described in Appendix A). The Birthday Protocols can be parameterized to save energy during deployment and have a very high probability of discovering neighbors after an asynchronous discovery is started. But there are two main disadvantages. (1) The radio is operating at 100% duty cycle during discovery and (2) there is no defined end point after that an operational phase may begin. A third disadvantage can be added when compared to the NoSE protocol: (3) The Birthday Protocol does not make any link-quality estimation.

In Table 5-1 both protocols are compared. The columns correspond to a certain link-quality as measured with our link test and the values in the rows show of discovered links of that category. We see that both the Birthday Protocol and the NoSE Neighbor Discovery find all neighbors with high probability.

PRR [%]	$> 95\%$	$85 - 95\%$	$50 - 85\%$	$< 50\%$
NoSE	97.8%	88.8%	80.8%	59.3%
Birthday	97.0%	91.9%	82.5%	70.4%

Table 5-1: Comparison of the neighbor discovery performance. Both protocols find almost all high-quality links, but also find a substantial amount of the available low-quality links.

What we see here is that a good neighbor discovery not only finds the high-quality links, but all medium-quality and poor-quality links too. For a reliable operation of a wireless sensor network only high-quality links are wanted and therefore an additional link-quality estimation would be desirable during the initialization.

5.3.2 Energy Optimization

In this section we analyze the energy efficiency of the NoSE neighbor discovery. We have three parameters that we can vary for the discovery phase and we explore their impact on the duty cycle:

- T_w the low-power-listening period
- T_d the duration of the neighbor discovery
- P the number of packets to send

Theoretically one can assume that the energy consumption of the radio is modeled as follows. The radio is sleeping and periodically, every T_w, it wakes up for a low-power-listening of T_{lpl}. If there is a message on the channel, the radio stays powered to receive this message, this takes T_r on average. To send a message to its neighbors the radio must send a packet burst for a duration of T_w. If we assemble those times to make a rough approximation for the duty cycle of the radio, we get

$$DC(T_w, P, T_d) = \frac{P \cdot T_w + N \cdot P \cdot T_r + \frac{T_d}{T_w} \cdot T_{lpl}}{T_d} \tag{5.1}$$

With P the number of packets to send during the discovery duration T_d, and N the average numbers of neighbor of a node. This equation does not include any energy lost by the clear channel assessment if the channel is found occupied.

In Figures 5-7 the duty cycle of the radio (from (5.1)) is plotted for different discovery durations T_d. From measurements of our implementation we set $T_{lpl} = 3.5ms$, $T_r = 4.4 - T_{lpl} = 0.9ms$ and $N = 10$ to have an approximation of our test bed.

From those theoretical figures we can derive some interesting properties:

Energetic optima exists There is an optimal, energy-efficient low-power-listening period T_w for a defined combination of duration T_d and number of packets P

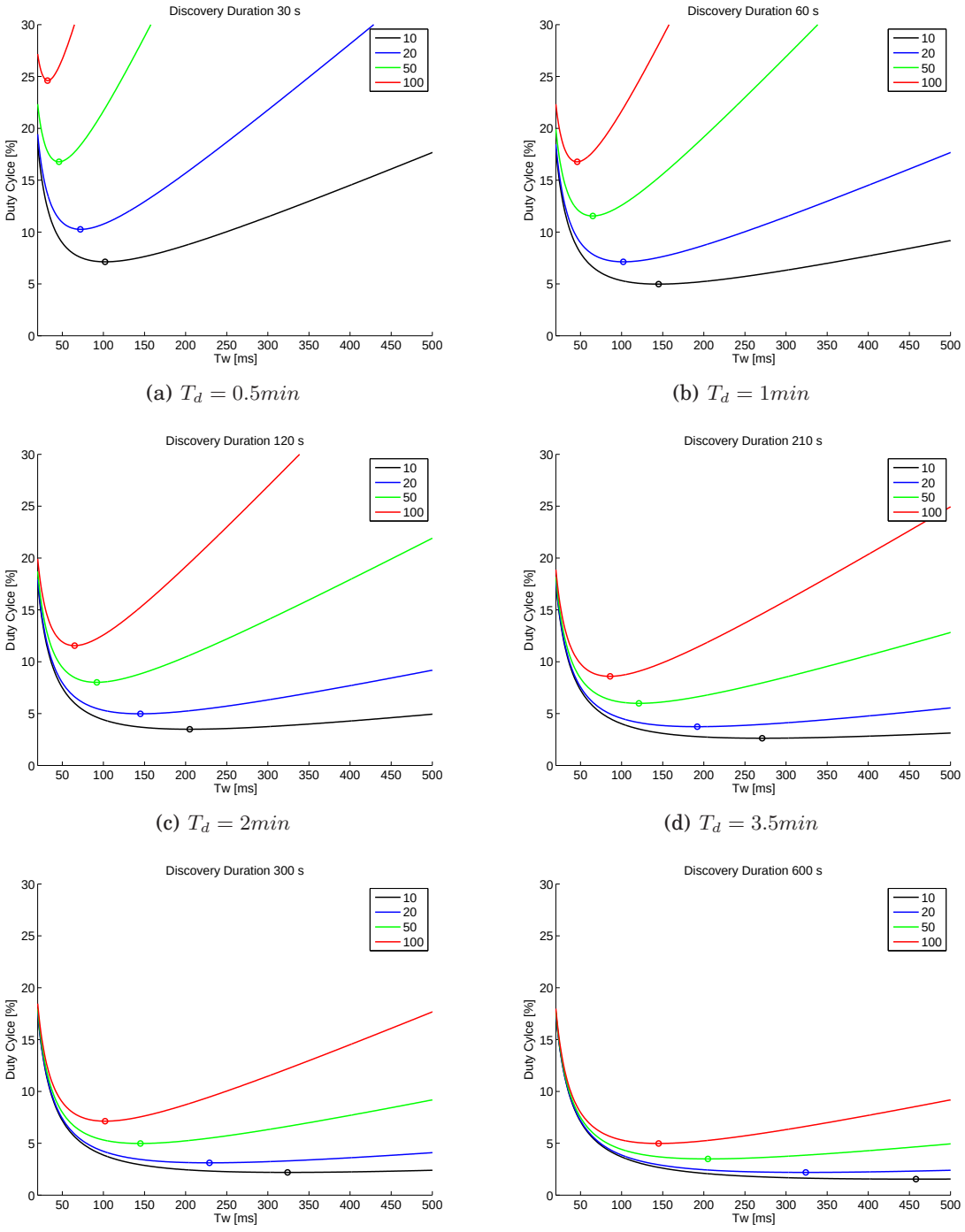
Energy not linear over time Duration and duty cycle are *not* linearly dependent for fixed number of packets. E.g. when we compare (a) and (e): T_d is multiplied by 10 from $T_d^{(a)} = 30s$ to $T_d^{(b)} = 300s$, the duty cycle is only reduced by roughly 3.5 from $DC^{(a)}(T_w, 10, 30) = 7$ to $DC^{(b)}(T_w, 10, 300) = 2$.
For a fixed number of packets to send, a longer discovery duration is more energy extensive.

Figure 5-8 shows the average radio duty cycle depending on discovery duration T_d and the low-power-listening period T_w for the case of 20 packets being sent. When we compare this with the theoretical lines for 20 packets, we find vaguely the same curves:

- For $T_d = 5min$, one can surmise a minimum at $T_w = 230ms$ and duty cycle that is about 1.5 to 2.0 times larger then expected.
- The curve for $T_d = 2min$ shows a minimum near $T_w = 150ms$ and duty cycle that is about the same amount larger too.
- The curve with $T_d = 1min$ behaves the same way, but for a value near or greater $T_w = 100ms$ the duty cycle rises disproportionately
- $T_d = .5min$ cannot be compared with the theoretical values. It shows a disproportionate rise almost from the beginning.

We don't know where the factor of around 1.5 regarding the duty cycle comes from.

The curve for $T_d = 30s$ ($0.5min$) deviates from the theoretical calculations due to a very high channel occupancy. If we assume an average of $N = 10$ neighbors and each sends $P = 20$ packets, a total of $N * P = 200$ slots will be occupied. With a slot period



(a) $T_d = 0.5min$ (b) $T_d = 1min$

(c) $T_d = 2min$ (d) $T_d = 3.5min$

(e) $T_d = 5min$ (f) $T_d = 10min$

Figure 5-7
Theoretical duty cycle for NoSE Neighbor Discovery & Link Estimation for different number of packets sent randomly over a certain discovery duration.

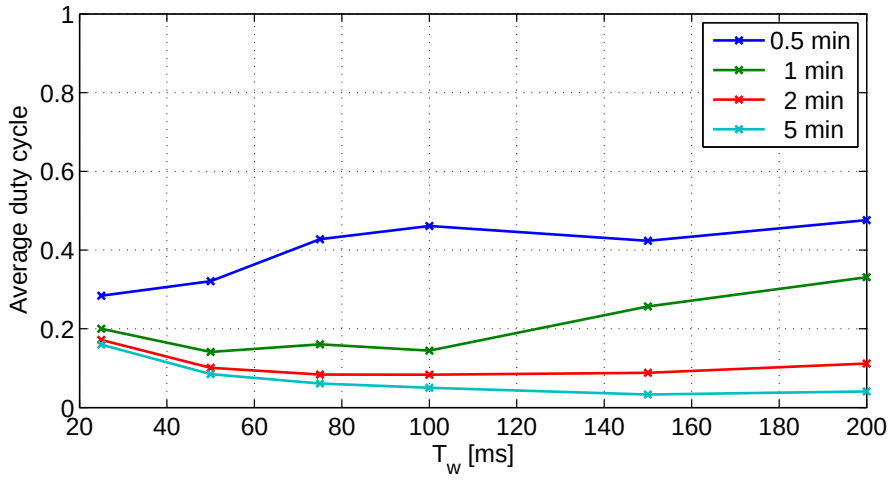


Figure 5-8
Experimental duty cycle for NoSE Neighbor Discovery & Link Estimation for a total 20 *packets send randomly over the duration of the discovery. Depending on the discovery duration and the low-power-listening period.*

of $T_w = 50ms$ we have a total of $T_d/T_w = 600$ slots during the discovery phase. Thus the slot occupancy is $1/3$. For $T_w > 50ms$ it is therefore likely to have a significant amount of sending retries due to channel congestion. This causes difference in duty cycle from the theoretical calculations. The same applies proportionally to $T_d = 60s$ ($1min$) for the area $T_w > 100ms$. The values for T_w are more or less chosen arbitrarily, but one can clearly see the disproportionately raise of the curve compared to the theoretical curve around and after these values.

To gain more insight about these issues, they would have to be examined more in depth.

Table 5-2 shows the radio duty cycle depending on the number of packets P being sent and the discovery duration T_d for a fixed low-power-listening period ($T_w = 100ms$). We see that if we multiply P by the same factor as T_d the duty cycle stays constant (around 10%). So we can say that regarding to the duty cycle it does not matter whether we send 10 packets in 1 minute or 50 packets in 5 minutes. But obviously the absolute energy-consumption is greater for a longer discovery duration.

	1 min	**2 min**	**3.5 min**	**5 min**
10 Packets	0.096	0.085	0.081	0.079
20 Packets	0.141	0.101	0.087	0.085
35 Packets	0.259	0.129	0.099	0.091
50 Packets	0.435	0.174	0.115	0.100

Table 5-2: Energy consumption (radio duty cycle) for the NoSE Neighbor Discovery & Link Estimation depending on the number of packets being sent and the discovery duration (for $T_w = 100ms$*).*

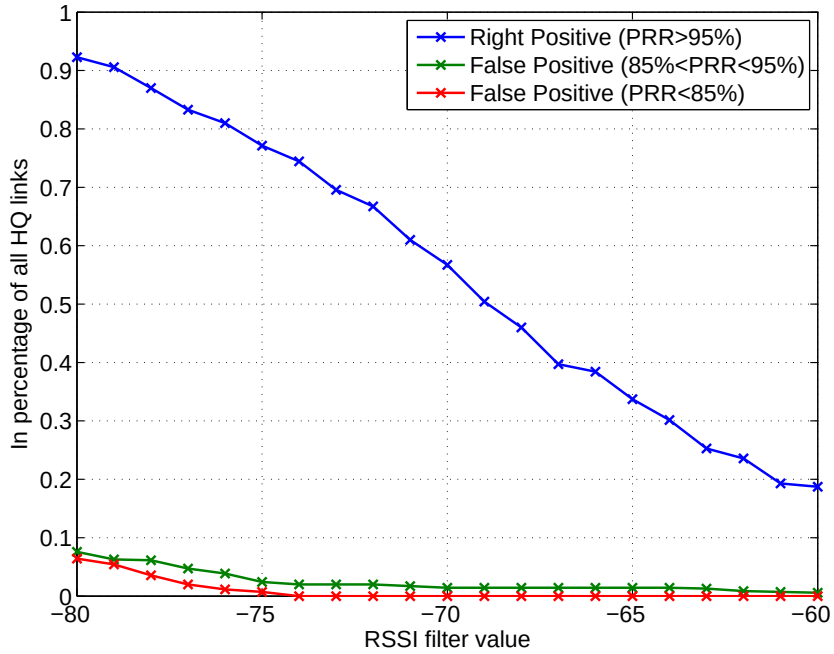


Figure 5-9
Usage of RSSI to quantify high-quality links

5.3.3 Link-quality Estimation

In the next step we want to find out, whether we can gain in quality of the link estimation if we invest more time and send more packets or if it is even more profitable to take more time and send fewer packets.

With the NoSE discovery two values are gained to quantify link-quality:

Packet Reception Rate (PRR) The PRR is directly calculated from the received packets during the discovery and the knowledge of the number of sent packets.

Received Signal Strength Indicator (RSSI) The RSSI value is provided with each received packet.

In a first step we choose only the links that have a PRR of greater 95% and then use the RSSI value as filter. In Figure 5-9 we use three PRR ranges and analyze the impact of the RSSI filter value on the success rate in finding high-quality links ($> 95\%$ in link tests) and avoiding false positives.

For a RSSI value worse then -75 the number of false positive (especially the range of bad links with $PRR < 85\%$) increases suddenly. Therefore a RSSI of -75 or -77 should result in a high number of successfully estimated high-quality links.

Figure 5-10 shows the relative number of discovered high quality links. From this figure we can derive the following two main conclusions:

- For a short discovery duration and a high number of packets sent the success rate drops significantly. This is because not all nodes are able to send their messages during the discovery duration, when the slot occupancy grows

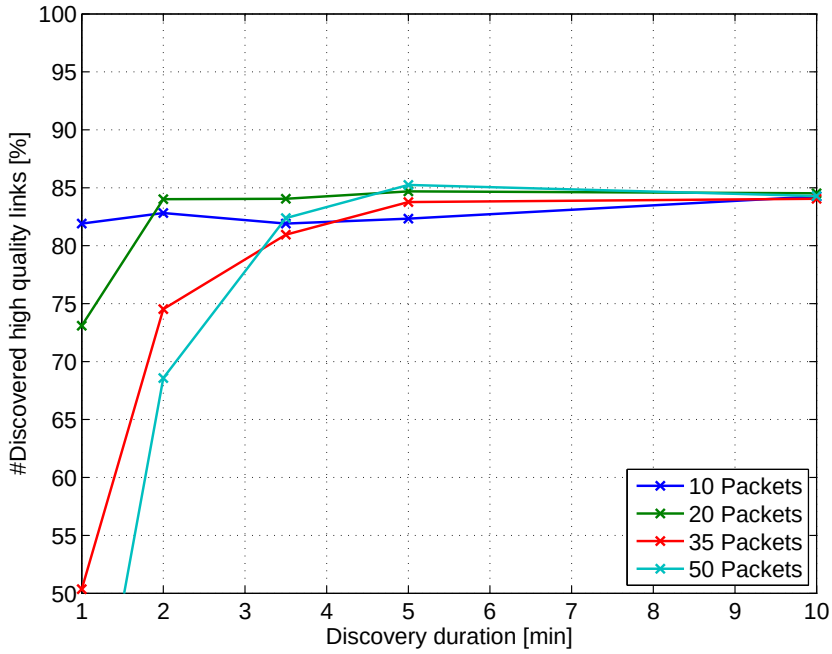


Figure 5-10
A Link-Quality Estimation. Relative number of discovered high quality links ($>= 95\%$) dependent on discovery duration and number of packets sent ($Tw = 100ms$). Quantification criteria: $PRR > 90\%$ and $RSSI > -77$.

too high. (With $T_w = 100ms$ and neighbors $N = 15$ the maximal theoretical throughput would be 40 messages per minute)

- For long discovery duration the number of packets sent has almost no impact on the discovery rate for high quality link.
- A short discovery ($1min$) with a total of 10 packets sent, can yield a relatively good result.
- As sending is energy-expensive it is better to use not more then 20 packets for the discovery. For short durations this yields a better result then a discovery with more packets. And for longer discovery duration the difference in the success rate to discovery high quality links is negligible.

We further propose, based on this Figure, that for the setup of our test bed the following two alternatives would yield best result for a link estimation:

I. A discovery duration of $T_d = 1min$ with 10 packets sent.

II. A discovery duration of $T_d = 2min$ with 20 packets sent.

From the previous energy-related analysis we know that the duty cycle for those two alternatives is the same and the minimum would lie in the area of $8 - 10\%$ duty cycle with a period $T_w = 150ms$ (this is taken from the experimental data in Figure 5-8). So now one has to decide if it is worth to invest double the amount of energy to gain $3 - 4\%$ better link-quality estimation. We think in the short time span of the discovery we can invest some energy to gain a better link-quality estimation.

5.4 Tests used for analysis

Table 5-3 shows a list of all executed tests used for analysis and their quantity. For detailed test parameterization see Appendix B.

Description	**Quantity**	**Tid**	*Comments*
Linktests (full power)		677	
Linktests (half power)		676	
Birthday Protocol	240	150 - 185	
NoSE Wake-up call only	259	1000 - 1231	
NoSE Discovery	228	2000 - 2223	No energy measurements (Iid 1-3)
NoSE Discovery	60	12170 - 12189	No energy, 2nd discovery strategy
NoSE Discovery	120	2200 - 2328	Energy (Iid 20-23)
NoSE Discovery	75	13000 - 13024	2nd discovery strategy (Iid 1-3)
NoSE Discovery	75	3000 - 3024	1nd discovery strategy (Iid 1-3)
NoSE Wake-up call	259	2000 - 13024	always with same parameters

Table 5-3: Overview over all executed tests and their quantity.

6

Conclusion

The area of wireless sensor networks is a very popular area of research. There exist many algorithms and protocols that deal with the operational phase of WSNs. But these protocols are not optimized for the phase when the sensor nodes are deployed and the network needs to be initialized. During the deployment phase of the network it is important to establish a profound base for the consecutive operation of the application. One need to consider the energy consumption: The installation of the nodes might take a long time, maybe some days or even weeks. Active beaconing during this phase is very likely to be useless, since the neighbors might not even be deployed yet.

This master thesis proposes an energy-efficient neighbor discovery and link estimation protocol: NoSE - Neighbor Search & Estimation. During deployment all sensor nodes are in a very low power mode. Asynchronously a centrally triggered wake-up call informs all the nodes of the time for the synchronous neighbor discovery phase. During this energy and time optimized discovery phase link-quality estimation data is gathered, that can be used for a link-quality estimation at the end of this phase.

Based on the implementation of the NoSE protocol and extensive tests performed in an office-like environment with 25 Tmote Sky sensor nodes, the following conclusions have been drawn:

- Comparing data of our link test gathered at different hours, confirms that some links tend to be unstable over time [17] and others are related to time (i.e. office hours).
- It is relatively easy to find all links in a network with high probability, but not always energy-efficient.
- Finding neighbor links alone is not sufficient, as even very bad links are discovered. Therefore NoSE provides link estimation.
- Based on the NoSE protocol analysis:
  - There exists a duty cycle optimized low-power-listening period T_w for a combination of number of packets being sent and the discovery duration.

  - For a fix period T_w the duty cycle is linear for the number of packets being sent and the discovery duration.
  - Link-quality estimations show that for a long discovery duration, the number of packets sent does not matter for the quality of the estimation. But for a short discovery 10 or 20 sent packets result in good link-quality estimation, with 20 using more energy than 10 but on the other hand delivers better link-quality estimation.
- NoSE achieves a fast and energy-efficient neighbor discovery combined with good link-quality estimation in our test bed in *tree minutes*.

6.1 Contribution

The contribution of this master thesis includes the following:

- NoSE - a neighbor discovery and link-quality estimation protocol has been designed to meet the specific tasks of the deployment and initialization phase of wireless sensor networks, as well as cope with their general limitations.
- The designed protocol has been implemented under TinyOS on the Tmote Sky platform and been deployed and tested on a testbed in an office-like environment at the Department of Electrical Engineering at ETH Zurich.
- One variant of the Birthday Protocol proposed by McGlynn et al. [12] has been implemented on the same platform and was compared with the implementation of NoSE.
- An analysis of the NoSE implementation has been made by running a series of tests on the testbed. This allows further analysis and finding a parameterization with an ideal trade-off between energy-saving, duration and link-quality estimation for a specific deployment.

6.2 Future Work

As the main focus of this thesis was on design and implementation, there is generally more analysis to do. One could change and optimize the strategies for the wake-up call and the discovery. The effect of the discovery time on the link-estimation is another possible focus: How much time and energy should be invested to gain the desired level of estimation-quality.

A next step would be to integrate the NoSE initialization protocol into a full application and analyze how this does effect the application regarding energy-efficiency and link-reliability.

A

The Birthday Protocol

This is the specification of our implementation of the Birthday Protocols proposed by McGlynn et al. [12]. First the general model of a group of Birthday Protocols is portrayed, then our specific implementation is described and in the end we show how we used the Birthday Protocols to conduct a link test.

A.1 Model of the Protocol

The Birthday Protocols it is assumed that:

- time is slotted. We have n time slots. Each time slot has length T_s.
- n is large.
- the total number of nodes is known.
- the nodes do not coordinate their actions in any way.
- the nodes are placed randomly in some area.
- nodes are distinguishable by an ID such as a MAC address.
- each node has some internal memory to record local topology.

A node can be in one of three states transmit (T), listen (L), or energy-saving (S). during the *transmission state*, a node is broadcasting a discovery message advertising itself. Such a message would consist of, at a minimum, the address of the broadcaster. During *listen state*, a node is listening for discovery messages. If such a message is heard, the nodes records the source address in its local topology table. During *energy-saving state*, a node spends zero energy on its radio subsystem.

Each node will choose randomly to enter one of the three states at the beginning of each time slot. Its choice will depend on the mode the node is in, and two globally fixed values π_t and π_l. The first quantity π_t represents the probability to transmit in BLT mode. The quantity π_l represents the probability of listening in both BLT and BL mode.

A node in BLT mode can be in any of the three states, L, T and S. In each time slot it chooses state T with probability p_t, L with probability p_l and S with probability p_s. A node which transits into BLT mode sets

$$\begin{aligned} p_t &\leftarrow \pi_t \\ p_l &\leftarrow \pi_l \\ p_s &\leftarrow 1 - \pi_t - \pi_l \end{aligned} \tag{A.1}$$

A node in BL mode does not transmit. It alternates between states L and S. On each time slot it picks randomly between the two states, choosing L with probability p_l and S the rest of the time. A node which transits into BL mode sets

$$\begin{aligned} p_t &\leftarrow 0 \\ p_l &\leftarrow \pi_l \\ p_s &\leftarrow 1 - \pi_l \end{aligned} \tag{A.2}$$

Because one distinguishes between the events "X hears Y" and "Y hears X", Neighbor discovery is defined as the discovery of "unidirectional links". A clique C is defined as a set of nodes which are all within radio range of each other and N be the number of nodes in a clique.

A third mode is proposed. In this "probabilistic round robin" (PRR) mode the fraction of discovered links is maximized by choosing the "right" transmission probability. For a particular value of N this is when $p_t = \frac{1}{N}$. A node which transits into PRR mode sets

$$\begin{aligned} p_t &\leftarrow \tfrac{1}{N} \\ p_l &\leftarrow 1 - \tfrac{1}{N} \\ p_s &\leftarrow 0 \end{aligned} \tag{A.3}$$

There is no energy conservation in PRR mode, and the number of links discovered is maximized. The fraction F of links discovered becomes

$$F = 1 - \left(1 - \frac{p_l}{N}\right)^n$$

with n equals the number of slots the PRR mode lasts.

A scenario is described where nodes are deployed randomly in BL mode. The nodes save energy yet are listening with some p_l to hear some neighbor. After the deployment phase is over, one node is put in PRR mode to launch the initialization. When in BL mode, and a message is heard, the node must transit to PRR mode for a certain fixed period, then fall back to BL mode. It is shown that with this BL-PRR protocol the two main goals are comfortably satisfied. During long deployments energy conservation is achieved in BL mode, While PRR mode quickly discovers neighbors.

A.2 Implementation

Our goal is to implement this BL-PRR protocol on the Tmote Sky platform [15], [5] described in chapter 4. For reasons concerning the implementation on real nodes, we change or extend the protocol in the following manner:

Name	Bytes	*Description*
uniqueId	2	`0xB886` (*Birthday protocol identifier*)
nodeId	2	*Node identifier*

Table A-1: BirthdayProtocolMsg*: Birthday protocol message format*

Name	Bytes	*Description*
nodeId	2	*Node identifier of a neighbor*
nodeCount	4	*Message quantity that has been received from this neighbor*

Table A-2: neighborTableEntry_t*: Neighbor table format*

- A node has no information about the state and the timer of the other nodes. Therefore, the slots are *not* synchronized on different nodes. Problems of overlapping slots (e.g. still receiving when slot time is expired) resulting from this are ignored and handled by the operation system implementation of the node.
- During the transmission state, only one discovery message is sent. The rest of the slot time the node just ignores every other action (mostly a received package).

The operation system used on the Tmote (TinyOS) has a timer component. Such a periodic timer is used to represent one slot. The program flow of the implementation is described in Fig. A-1. First variables are initialized and the timer is started. When the timer expires the slot is over and the random change of state (depending on mode) has to be set as well as some action has to be taken regarding the new state.

When in *transmit state* a message has to be broadcast via the radio. This message consists of a birthday protocol identifier and of the node identifier, see Table A-1. The (fixed) birthday protocol identifier is needed to distinguish between other messages and birthday protocol messages when receiving a packages that matches the correct payload length. When in *listen state* the nodes listens on the radio for a broadcasted message. And in *sleep state* (energy-saving state) the radio is turned off to minimize energy consumption of the node.

Figure A-2 shows the modes and their transitions for this implementation.

The node starts in BL mode during deployment. In this mode the node mostly sleeps to conserve energy and with probability π_l wakes up to listen for a discovery message. If it receives such a message it transmits into PRR mode. In PRR mode it either transmits a message with probabilty $\pi_t = 1/N$ or listens for a message of its neighbors. All the discovered neighbors are saved to the *neighbor table*. The format of the neighbor table is described in Table A-2. After a certain time (n number of slots) the protocol falls back into BL mode, where it stays until an artificial end of the whole protocol. The initialization parameters are shown in Table A-3. They represent the same situation as simulated in the birthday protocol publication: fraction of links discovered to be $F = 95\%$, energy gain to be 100.

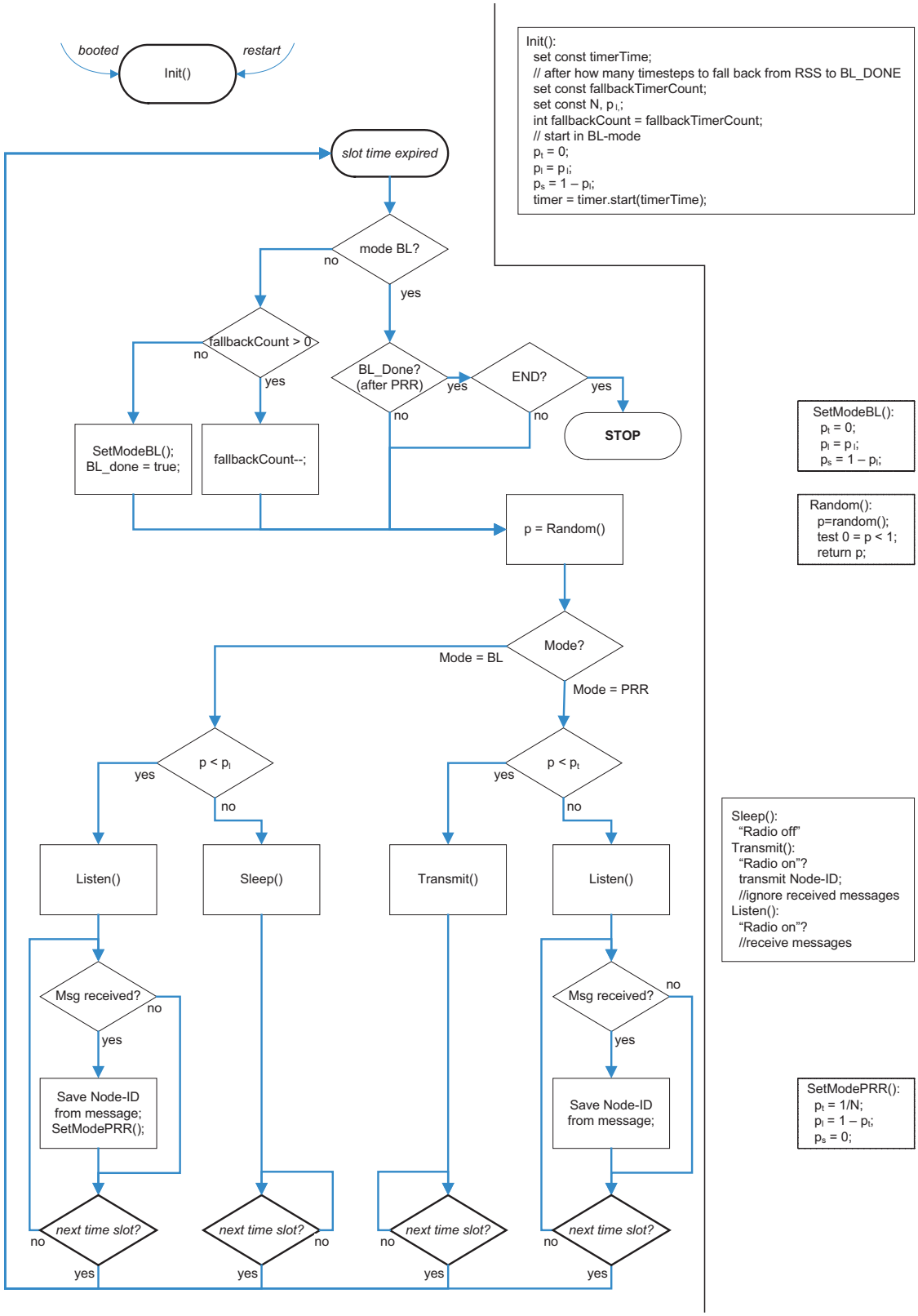


Figure A-1
The flowchart of the BL-PRR birthday protocol implementation.

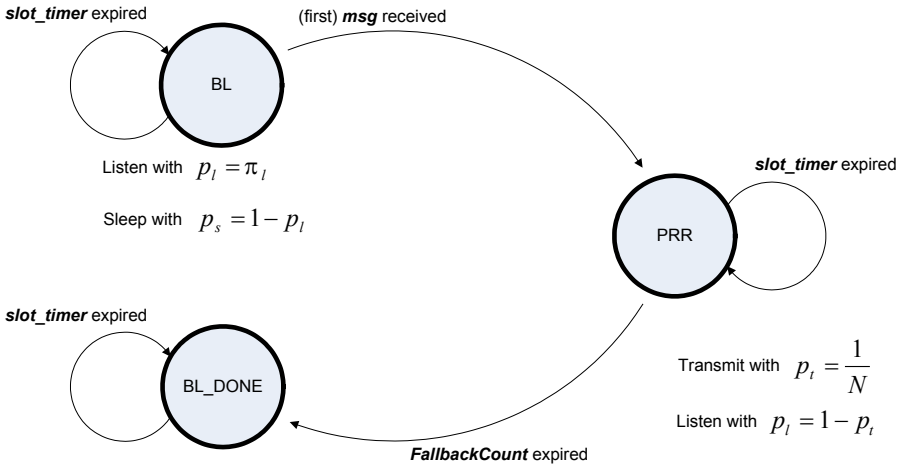


Figure A-2
The overlaying state machine of the BL-PRR birthday protocol implementation.

Property	Value	*Description*
T_s	20	*Slot length in* ms
N	10	*Number of nodes in a clique*
π_l	.01	*Probability to Listen in BL mode*
n	3000	*Number of slots to stay in PRR mode* ($= 1min$)
$\emptyset$	50000	*Artificial end of protocol* ($= 8min20sec$)
	100	*Maximal number of entries in the neighbor table*
	`0xB886`	*Birthday protocol identifier (hex)*
	216	*Birthday protocol AM Type (TinyOS)*

Table A-3: Parameters of the implementation as defined in `BirthdayProtocol.h`

Command	*Description*
`s`	*Stop Protocol on this node*
`r`	*Restart Protocol on this node*
`p`	*Pause / continue protocol on this node*
`startPRR`	*Start PRR mode (used to start protocol in WSN)*
`getTable`	*Show neighbor table* "NT_*thisID*:*nodeID*(*count*);..." *thisID* ID of this node *nodeID* ID of a neighboring node *count* Quantity of messages received from the corresponding node
`getMode`	*Show protocol mode* *thisID* ID of this node
`getParam`	*Show protocol parameter* *thisID* ID of this node

Table A-4: Commands and log output for the Birthday Protocol on the DSN.

A.2.1 Radio Stack

This implementation uses CC2420 Radio Stack of TinyOS with the following settings:

- Messages are broadcasts of TinyOS Active Messages (AM).
- Acknoledgemnts have been turned off.
- The Low Power Listening functionality of the CC2420 Stack is not used.

A.2.2 Interaction over the DSN

In Table A-4 the commands to interact with the Birthday Protocol implementation are described. Additionally, the log output to these commands is given.

To start a run, at sink node the PRR mode must be started with the command `startPRR`. When the PRR mode is over (that can be testes with `getMode`) the neighbor table can be gathered by use of `getTable`. The first entry of the neighbor table is always an entry with *thisID*=*nodeID* and *count* represents the number of actually sent messages.

A.3 Link measurements with the Birthday Protocol

To gain an exact link measurement we choose a listen probability in BL mode of $\pi_l = 1$ and a very small transmission probability π_t during a very long PRR mode of n slots. The nodes are always listening and only very rarely transmitting a packet.

With these settings we can assume that every packet that has been transmitted by a node can be received by all its neighbors with no packet loss because of collisions. Thus we define the link quality between two nodes as

$$\text{Link-quality} = \frac{\text{packets received}}{\text{packets transmitted}}$$

Since the Birthday Protocols are of probabilistic nature, we cannot assume a exact number of packets transmitted. But by calculating the probability that at least B number of packets are transmitted and setting an upper bound (of B) for the sending of packets, we can assume with great certainty how many packets have been transmitted. This number is Poisson distributed (with λ being the average numbers of packets send during n slots): $\lambda = n \cdot \pi_t$.

For sufficiently large values of λ the Poisson distribution can be approximated by a normal distribution with mean λ and variance λ. And thus we calculate the probability that at least B packets are transmitted by the sender node:

$$Pr(X \geq B) = 1 - \Omega(B; \mu = \lambda, \sigma^2 = \lambda)$$

With Ω being the cumulative distribution function of the normal distribution with $\mu = \lambda$ and $\sigma^2 = \lambda$.

From this we derive the probability that at least B packets are sent for the chosen values for our link test, $B = 1000$ and $\pi_t = \frac{1}{200}$ (shown in Tab. A-5).

Length of PRR modus $n = \lambda \cdot \frac{1}{\pi_t}$	Avg. packets sent λ	$Pr(X \geq B)$
200000 slots	1000	50.00%
205000 slots	1025	78.52%
210000 slots	1050	94.30%
220000 slots	1100	99.87%
225000 slots	1125	99.99%

Table A-5: Probability of packet transmission for the Birthday Protocol link test

The following adjustments were made to our original implementation of the Birthday Protocol:

- A node accepts messages during a transmission slot (if it is not transmitting the whole slot)
- The number of packets transmitted is limited by a upper bound B

The Link Test was carried out during two different day times with multiple consecutive runs to achieve a somewhat relevant data base. The parameters in Table A-6 were choosen for the implementation.

Now because we can set the upper bound of packets sent to 1000, any node sends exactly 1000 packets with a probability of greater then 99.99%. Thus, we have an accurate base to compare the link-estimations of both our implementation of the NoSE discovery and the Birthday Protocol.

Note that because the PRR mode has no energy saving (100% duty cycle), this link test is only practicable for an environment where there are no energy constraints on the nodes.

Slot time	$T_s = 10ms$
Listening prob. in BL mode	$\pi_l = 1$
Transmission prob. in PRR mode	$\pi_t = 1/200$
PRR mode length	$n = 225000$ Slots

This adds up to

Avg. number of transmissions	$\lambda = 1125$
Probability of than 1000 packets sent	$Pr(X \geq 1000) > 99.99\%$

Table A-6: Parameters for the Birthday Protocol link test

B

Test Descriptions

This appendix contains all the information about our test and their parameterization. First, the data for the Birthday Protocol tests, including the link test that we conducted with a specific parameterization of the Birthday Protocol. Second, the data for the NoSE wake-up call and discovery.

Table B-1: Tests: Linktest

Tid	**SinkId**	**TxPower**	T_s[ms]	N	$L = 1/\pi_l$	n{#slots in PRR}
666	29	31	10	200	1	250000
667	29	24	10	200	1	250000
668	29	16	10	200	1	250000
669	29	8	10	200	1	250000
670	29	7	10	200	1	250000
671	29	3	10	200	1	250000
676	29	15	10	200	1	250000
677	29	31	10	200	1	250000

Table B-2: Tests: Birthday Protocol

Tid	**SinkId**	**TxPower**	T_s[ms]	N	$L = 1/\pi_l$	n{#slots in PRR}
150	29	31	10	5	100	1500
151	29	31	10	10	100	3000
152	29	31	10	15	100	4500
153	29	31	10	5	100	900
154	29	31	10	10	100	1700

Continued on next page

Table B-2 – continued from previous page

Tid	**SinkId**	**TxPower**	T_s[ms]	N	$L = 1/\pi_1$	n{#slots in PRR}
155	29	31	10	15	100	2500
160	29	31	20	5	100	1500
161	29	31	20	10	100	3000
162	29	31	20	15	100	4500
163	29	31	20	5	100	900
164	29	31	20	10	100	1700
165	29	31	20	15	100	2500
170	44	31	10	5	100	1500
171	44	31	10	10	100	3000
172	44	31	10	15	100	4500
173	44	31	10	5	100	900
174	44	31	10	10	100	1700
175	44	31	10	15	100	2500
180	44	31	20	5	100	1500
181	44	31	20	10	100	3000
182	44	31	20	15	100	4500
183	44	31	20	5	100	900
184	44	31	20	10	100	1700
185	44	31	20	15	100	2500

Table B-3: Tests: NoSE, Wake-up call only

Tid	**SinkId**	**TxPower**	**W**	**N**	T_{wuc} [10ms]
1000	20	31	3	10	18432
1001	20	31	2	5	12288
1002	20	31	3	5	18432
1003	20	31	4	5	24576
1004	20	7	2	5	12288
1005	20	7	3	5	18432
1006	20	7	4	5	24576
1011	20	31	2	10	12288
1012	20	31	3	10	18432
1013	20	31	4	10	24576
1014	20	7	2	10	12288
1015	20	7	3	10	18432
1016	20	7	4	10	24576
1021	20	31	2	15	12288
1022	20	31	3	15	18432
1023	20	31	4	15	24576
1024	20	7	2	15	12288

Continued on next page

Table B-3 – continued from previous page

Tid	SinkId	TxPower	W	N	T_{wuc} [10ms]
1025	20	7	3	15	18432
1026	20	7	4	15	24576
1099	20	31	2	15	12288
1100	20	31	2	5	3072
1101	20	31	2	5	6144
1102	20	31	2	10	6144
1103	20	31	3	5	6144
1104	20	31	3	10	6144
1105	20	31	4	5	6144
1106	20	31	2	5	12288
1107	20	31	2	10	12288
1108	20	31	2	15	12288
1109	20	31	2	20	12288
1110	20	31	3	5	12288
1111	20	31	3	10	12288
1112	20	31	3	15	12288
1113	20	31	4	5	12288
1114	20	31	4	10	12288
1115	20	31	2	5	18432
1116	20	31	2	10	18432
1117	20	31	2	15	18432
1118	20	31	2	20	18432
1119	20	31	2	30	18432
1120	20	31	3	5	18432
1121	20	31	3	10	18432
1122	20	31	3	15	18432
1123	20	31	3	20	18432
1124	20	31	4	5	18432
1125	20	31	4	10	18432
1126	20	31	4	15	18432
1200	20	31	2	5	3072
1201	20	31	3	5	3072
1202	20	31	2	5	6144
1203	20	31	3	5	6144
1204	20	31	2	5	12288
1205	20	31	3	5	12288
1206	20	31	2	5	30720
1207	20	31	3	5	30720
1208	20	31	2	15	3072
1209	20	31	3	15	3072
1210	20	31	2	15	6144
1211	20	31	3	15	6144

Continued on next page

Table B-3 – continued from previous page

Tid	SinkId	TxPower	W	N	T_{wuc} [10ms]
1212	20	31	2	15	12288
1213	20	31	3	15	12288
1214	20	31	2	15	30720
1215	20	31	3	15	30720
1216	20	15	2	5	3072
1217	20	15	3	5	3072
1218	20	15	2	5	6144
1219	20	15	3	5	6144
1220	20	15	2	5	12288
1221	20	15	3	5	12288
1222	20	15	2	5	30720
1223	20	15	3	5	30720
1224	20	15	2	15	3072
1225	20	15	3	15	3072
1226	20	15	2	15	6144
1227	20	15	3	15	6144
1228	20	15	2	15	12288
1229	20	15	3	15	12288
1230	20	15	2	15	30720
1231	20	15	3	15	30720

Table B-4: Tests: NoSE Neighbor Discovery & Link-quality Estimation

Tid	SinkId	TxPower	W	N	T_{wuc} [10ms]	T_d [10s]	T_w [ms]	P
2000	20	31	2	10	12288	30	30	10
2001	20	31	2	10	12288	30	50	10
2002	20	31	2	10	12288	30	100	10
2003	20	31	2	10	12288	30	30	50
2004	20	31	2	10	12288	30	50	50
2005	20	31	2	10	12288	30	100	50
2010	20	31	2	10	12288	30	30	10
2011	20	31	2	10	12288	30	50	10
2012	20	31	2	10	12288	30	100	10
2013	20	31	2	10	12288	30	30	50
2014	20	31	2	10	12288	30	50	50
2015	20	31	2	10	12288	30	100	50
2100	20	31	2	5	6144	0.75	30	10
2101	20	31	2	5	6144	1.25	50	10
2102	20	31	2	5	6144	2.5	100	10

Continued on next page

Table B-4 – continued from previous page

Tid	SinkId	TxPower	W	N	T_{wuc} [10ms]	T_d [10s]	T_w [ms]	P
2103	20	31	2	5	6144	5	200	10
2104	20	31	2	5	6144	1.5	30	10
2105	20	31	2	5	6144	2.5	50	10
2106	20	31	2	5	6144	5	100	10
2107	20	31	2	5	6144	10	200	10
2108	20	31	2	5	6144	3	30	10
2109	20	31	2	5	6144	5	50	10
2110	20	31	2	5	6144	10	100	10
2111	20	31	2	5	6144	20	200	10
2112	20	31	2	5	6144	6	30	10
2113	20	31	2	5	6144	10	50	10
2114	20	31	2	5	6144	20	100	10
2115	20	31	2	5	6144	40	200	10
2150	20	31	2	5	6144	30	30	10
2151	20	31	2	5	6144	30	50	10
2152	20	31	2	5	6144	30	100	10
2153	20	31	2	5	6144	30	30	20
2154	20	31	2	5	6144	30	50	20
2155	20	31	2	5	6144	30	100	20
2156	20	31	2	5	6144	30	30	50
2157	20	31	2	5	6144	30	50	50
2158	20	31	2	5	6144	30	100	50
2159	20	31	2	5	6144	30	30	100
2160	20	31	2	5	6144	30	50	100
2161	20	31	2	5	6144	30	100	100
2170	20	31	2	5	6144	6	100	10
2171	20	31	2	5	6144	12	100	10
2172	20	31	2	5	6144	30	100	10
2173	20	31	2	5	6144	60	100	10
2174	20	31	2	5	6144	6	100	20
2175	20	31	2	5	6144	12	100	20
2176	20	31	2	5	6144	30	100	20
2177	20	31	2	5	6144	60	100	20
2178	20	31	2	5	6144	6	100	35
2179	20	31	2	5	6144	12	100	35
2180	20	31	2	5	6144	30	100	35
2181	20	31	2	5	6144	60	100	35
2182	20	31	2	5	6144	6	100	50
2183	20	31	2	5	6144	12	100	50
2184	20	31	2	5	6144	30	100	50
2185	20	31	2	5	6144	60	100	50
2186	20	31	2	5	6144	6	100	100

Continued on next page

Table B-4 – continued from previous page

Tid	SinkId	TxPower	W	N	T_{wuc} [10ms]	T_d [10s]	T_w [ms]	P
2187	20	31	2	5	6144	12	100	100
2188	20	31	2	5	6144	30	100	100
2189	20	31	2	5	6144	60	100	100
2200	20	31	2	5	6144	3	25	20
2201	20	31	2	5	6144	3	50	20
2202	20	31	2	5	6144	3	75	20
2203	20	31	2	5	6144	3	100	20
2204	20	31	2	5	6144	3	150	20
2205	20	31	2	5	6144	3	200	20
2206	20	31	2	5	6144	6	25	20
2207	20	31	2	5	6144	6	50	20
2208	20	31	2	5	6144	6	75	20
2209	20	31	2	5	6144	6	100	20
2210	20	31	2	5	6144	6	150	20
2211	20	31	2	5	6144	6	200	20
2212	20	31	2	5	6144	12	25	20
2213	20	31	2	5	6144	12	50	20
2214	20	31	2	5	6144	12	75	20
2215	20	31	2	5	6144	12	100	20
2216	20	31	2	5	6144	12	150	20
2217	20	31	2	5	6144	12	200	20
2218	20	31	2	5	6144	30	25	20
2219	20	31	2	5	6144	30	50	20
2220	20	31	2	5	6144	30	75	20
2221	20	31	2	5	6144	30	100	20
2222	20	31	2	5	6144	30	150	20
2223	20	31	2	5	6144	30	200	20
2230	20	31	2	5	6144	21	25	20
2231	20	31	2	5	6144	21	50	20
2232	20	31	2	5	6144	21	75	20
2233	20	31	2	5	6144	21	100	20
2234	20	31	2	5	6144	21	150	20
2235	20	31	2	5	6144	21	200	20
2300	20	31	2	5	6144	6	50	10
2301	20	31	2	5	6144	12	50	10
2302	20	31	2	5	6144	21	50	10
2303	20	31	2	5	6144	6	50	35
2304	20	31	2	5	6144	12	50	35
2305	20	31	2	5	6144	21	50	35
2306	20	31	2	5	6144	6	50	50
2307	20	31	2	5	6144	12	50	50
2308	20	31	2	5	6144	21	50	50

Continued on next page

Table B-4 – continued from previous page

Tid	SinkId	TxPower	W	N	T_{wuc} [10ms]	T_d [10s]	T_w [ms]	P
2310	20	31	2	5	6144	6	75	10
2311	20	31	2	5	6144	12	75	10
2312	20	31	2	5	6144	21	75	10
2313	20	31	2	5	6144	6	75	35
2314	20	31	2	5	6144	12	75	35
2315	20	31	2	5	6144	21	75	35
2316	20	31	2	5	6144	6	75	50
2317	20	31	2	5	6144	12	75	50
2318	20	31	2	5	6144	21	75	50
2320	20	31	2	5	6144	6	150	10
2321	20	31	2	5	6144	12	150	10
2322	20	31	2	5	6144	21	150	10
2323	20	31	2	5	6144	6	150	35
2324	20	31	2	5	6144	12	150	35
2325	20	31	2	5	6144	21	150	35
2326	20	31	2	5	6144	6	150	50
2327	20	31	2	5	6144	12	150	50
2328	20	31	2	5	6144	21	150	50
2330	20	31	2	5	6144	30	50	10
2331	20	31	2	5	6144	30	75	10
2332	20	31	2	5	6144	30	150	10
2333	20	31	2	5	6144	30	50	35
2334	20	31	2	5	6144	30	75	35
2335	20	31	2	5	6144	30	150	35
2336	20	31	2	5	6144	30	50	50
2337	20	31	2	5	6144	30	75	50
2338	20	31	2	5	6144	30	150	50
3000	20	31	2	5	6144	6	100	10
3001	20	31	2	5	6144	12	100	10
3002	20	31	2	5	6144	21	100	10
3003	20	31	2	5	6144	30	100	10
3004	20	31	2	5	6144	60	100	10
3005	20	31	2	5	6144	6	100	20
3006	20	31	2	5	6144	12	100	20
3007	20	31	2	5	6144	21	100	20
3008	20	31	2	5	6144	30	100	20
3009	20	31	2	5	6144	60	100	20
3010	20	31	2	5	6144	6	100	35
3011	20	31	2	5	6144	12	100	35
3012	20	31	2	5	6144	21	100	35
3013	20	31	2	5	6144	30	100	35
3014	20	31	2	5	6144	60	100	35

Continued on next page

Table B-4 – continued from previous page

Tid	SinkId	TxPower	W	N	T_{wuc} [10ms]	T_d [10s]	T_w [ms]	P
3015	20	31	2	5	6144	6	100	50
3016	20	31	2	5	6144	12	100	50
3017	20	31	2	5	6144	21	100	50
3018	20	31	2	5	6144	30	100	50
3019	20	31	2	5	6144	60	100	50
3020	20	31	2	5	6144	6	100	100
3021	20	31	2	5	6144	12	100	100
3022	20	31	2	5	6144	21	100	100
3023	20	31	2	5	6144	30	100	100
3024	20	31	2	5	6144	60	100	100
12170	20	31	2	5	6144	6	100	10
12171	20	31	2	5	6144	12	100	10
12172	20	31	2	5	6144	30	100	10
12173	20	31	2	5	6144	60	100	10
12174	20	31	2	5	6144	6	100	20
12175	20	31	2	5	6144	12	100	20
12176	20	31	2	5	6144	30	100	20
12177	20	31	2	5	6144	60	100	20
12178	20	31	2	5	6144	6	100	50
12179	20	31	2	5	6144	12	100	50
12180	20	31	2	5	6144	30	100	50
12181	20	31	2	5	6144	60	100	50
12182	20	31	2	5	6144	6	100	100
12183	20	31	2	5	6144	12	100	100
12184	20	31	2	5	6144	30	100	100
12185	20	31	2	5	6144	60	100	100
13000	20	31	2	5	6144	6	100	10
13001	20	31	2	5	6144	12	100	10
13002	20	31	2	5	6144	21	100	10
13003	20	31	2	5	6144	30	100	10
13004	20	31	2	5	6144	60	100	10
13005	20	31	2	5	6144	6	100	20
13006	20	31	2	5	6144	12	100	20
13007	20	31	2	5	6144	21	100	20
13008	20	31	2	5	6144	30	100	20
13009	20	31	2	5	6144	60	100	20
13010	20	31	2	5	6144	6	100	35
13011	20	31	2	5	6144	12	100	35
13012	20	31	2	5	6144	21	100	35
13013	20	31	2	5	6144	30	100	35
13014	20	31	2	5	6144	60	100	35
13015	20	31	2	5	6144	6	100	50

Continued on next page

Table B-4 – continued from previous page

Tid	SinkId	TxPower	W	N	T_{wuc} [10ms]	T_d [10s]	T_w [ms]	P
13016	20	31	2	5	6144	12	100	50
13017	20	31	2	5	6144	21	100	50
13018	20	31	2	5	6144	30	100	50
13019	20	31	2	5	6144	60	100	50
13020	20	31	2	5	6144	6	100	100
13021	20	31	2	5	6144	12	100	100
13022	20	31	2	5	6144	21	100	100
13023	20	31	2	5	6144	30	100	100
13024	20	31	2	5	6144	60	100	100

C

Task Description

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering and Networks Lab (TIK)

Summer Term 2007

MASTER THESIS

for
Mischa Weise

Supervisor: Andreas Meier
Co-Supervisor: Jan Beutel

Start date: 2. April 2007
End date: 1. October 2007

Energy Efficient Initialization of Wireless Sensor Networks

Introduction

Wireless Sensor Network (WSN) nodes are small battery-powered platforms usually equipped with a micro controller, a radio module and a sensor. These nodes can, for instance, be deployed in a house to measure data (i.e. temperature) and forward this data to a base station. However, the base station is usually not in communication range with all sensor nodes. This requires that the sensor nodes build an ad-hoc network to forward the data over multiple hops, as illustrated in Figure 1. According to a vision of Stankovic et al. [1], this enables a "seamless integration of computing with the physical world via sensors and actuators".

The battery-powered nodes are not only constrained by the very limited power supply (battery) but also in processing power, memory size and communication possibilities to name a few. These limitations result in various difficulties when implementing a WSN application. For instance, an optimized MAC protocol is required to duty cycle the node's radio, whereas the routing protocol has to deal with the very limited memory and processing capabilities. Especially the combination of low-power operation and wireless

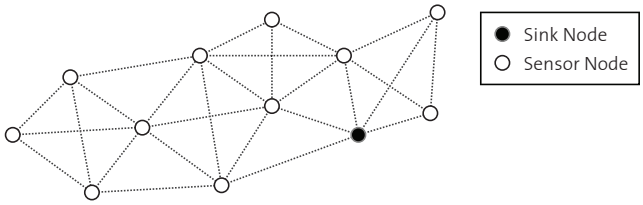


Figure 1: Schematic of a wireless sensor network (WSN): There is usually a dedicated (sink) node acquiring the sensor data gathered by the sensor nodes. Not all sensor nodes are in direct communication range with the node, requiring them to form a multi-hop ad-hoc network.

communication seems to be a crux for the robustness and reliability of WSNs. This combination results in unpredictable communication channels (links) between the nodes due to reasons of interference and fading. Various research groups analyzed this behaviour [2, 3] and showed that the quality of different links varies greatly and that some links show a very unpredictable and not deterministic behavior [4].

Recent deployments [5, 6] of WSN's pointed out the difficulty of reliable and robust operation. In particular, it has been shown that many of the sent data packets did not arrive at the sink node. One of the prime reasons seems to be, as indicated above, the rather undeterministic communication channel in combination with the resulting difficulties routing the data accordingly. It is therefore important that the communication links in use (usually there are many more links available than required by the routing protocol) are chosen carefully, i.e. good-quality links, for a sound operation of the network and application.

Especially during the deployment phase of the network (initialization) it is important to establish a profound basis for the consecutive operational phase of the application. Among other thins, the deployment phase needs to consider following three basic issues:

1. Energy Consumption: The deployment/arrangement of the nodes might take a lot of time, maybe even a couple of weeks. During this phase, it is important that the nodes do not drain a substantial fraction their battery power. For instance, it might not be the best idea to be aggressively looking for neighbors (by broadcasting lots of hello messages), since the neighbors might not even be deployed yet (i.e. switched on) [7, 8, 9].
2. Neighbor Discovery: In order to build up a robust and reliable network "good-quality" communication links should be used. However, this requires being aware of these links/neighbors first and therefore an exhaustive but energy efficient neighbor discovery is essential [7, 10, 11].
3. Link-Quality Estimation: In a last step, the quality of the links to the different neighbors needs to be estimated in order to minimize the chances to be using low-quality links for the routing. This issue has also been raised in [12, 13, 14, 4].

The validation and testing of WSN algorithms and software components is an indispensable part of the design-flow. However, this is not a trivial task because of the limited resources of the nodes. Considering further the distributed nature of the algorithms, the large number of nodes, and the interaction of the nodes with the environment, leads to the fact that the nodes are not only hard to debug, but also hard to access. Access to the state of the nodes, often referred to as visibility, is fundamental for testing and debugging. A tool for debugging, controlling and updating node's software is the so called 'Deployment Support Network' (DSN) [15]. It provides a secondary (wireless) network for a flexible, reliable and direct access to the sensor nodes being in deployment as illustrated in Figure 2.

Problem Definition

The main goal of this thesis is to design and implement an neighbor-discovery and link-quality estimation (NDLQE) protocol for WSNs. This is especially important during the initialization of the network, but might also be required during the operational phase of the network due to changes in the environment or node configuration.

In order to reach this goal, the project should proceed according the following steps:

1. The student should write a project plan and identify its milestones (thematic and time wise). In particular there should be enough room for the final presentation and the report.

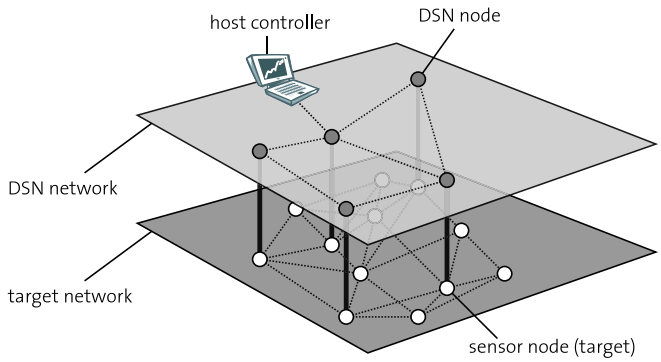


Figure 2: The target nodes are connected to the node of the 'Deployment Support Network' (DSN) with a short wire. This secondary (wireless) network provides a flexible access to the target nodes, in particular it allows updating the application code, logging data from and sending commands to the targets.

2. The student should study related work in the area of WSNs, focusing on the issue of link-quality [2, 3, 4] estimation [12, 13, 14, 4], neighbor discovery [7, 10, 11], initialization protocols [7, 8, 9] and low-power MAC protocols [16, 17, 18]. The student should also look out for further related work in this area. The results of this literature research should be written down as a first chapter of the report.
3. The NDLQE protocol will be implemented on the Tmote Sky [19] platform with the TinyOS 2.x [20] framework. In order to get used to the issue of embedded software engineering the Birthday Protocol [7] should be implemented and will later be used as a reference implementation. Furthermore the student should familiarize himself with the various utilities/tools, such as version control (SVN), mailing lists, online documentations, tutorials and exemplary applications.
4. The JAWS Deployment-Support Network [15, 21] should be set up and be tested with the Birthday Protocol (i.e. flash the Tmote Sky nodes, start the initialization and send log messages).
5. Design an NDLQE protocol that can be implemented on a sensor node.
6. Implement the NDLQE protocol on the Tmote Sky node.
7. Analyze the protocol's performance and compare it with the Birthday Protocol.

Organization

- Duration of the Work:
  This Master Thesis starts 2. April 2007 and has to be finished no later than 1. October 2007.
- Project Plan:
  A project plan with its milestones is held and updated continuously. Unforeseens difficulties that change the project plan have to be documented and should be discussed with the supervisors.
- Weekly Meetings/Reports:
  In regular (weekly) meetings with the supervisors, the current state of the work, potential difficulties as well as future directions are discussed. The day before the weekly meeting a brief status report should be sent to the supervisors commenting on these issues, in order to allow an adequate meeting preparation for the student and the supervisors.

- Research Journal:
  The work's progress is written down in a research journal that is handed in to the supervisor at the end of the project.
- Beginners Presentation:
  Approximately two to three weeks after the start you will shortly present the objectives of the work as well as some background on the topic. The presentation should be no longer than 5 minutes and consist of maximally two slides.
- Final Presentation:
  By the end of the project, you will present the achieved result. The presentation should not exceed 20 minutes.
- Documentation:
  At the end of the project, no later than 1. October 2007, you will have to hand in a written report. Together with the system implementation/software this report is the main outcome of the project. Document your work accurately. Additionally make sure to comment your code extensively, allowing a follow-up project.
- Evaluation of the work:
  The criteria for grading the work are described in [22].
- Finishing up:
  The required resources should be cleaned up and handed back in.

References

[1] J. Stankovic, I. Lee, A. Mok, and R. Rajkumar, "Opportunities and obligations for physical computing systems.," IEEE Computer, vol. 38, no. 11, pp. 23–31, 2005.

[2] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in First Int'l Workshop on Embedded Software (EMSOFT 2001), pp. 1–13, 2003.

[3] N. Reijers, G. Halkes, and K. Langendoen, "Link Layer Measurements in Sensor Networks," in Proc. 1st Int'l Conf. on Mobile Ad-hoc and Sensor Systems (MASS '04), Oct. 2004.

[4] T. Rein, "Energy and Time Efficient Link-Quality Estimation for WSNs," Master's thesis, Computer Engineering and Networks Lab, Swiss Federal Institute of Technology (ETH) Zurich, 2007.

[5] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," Commun. ACM, vol. 47, no. 6, pp. 34–40, 2004.

[6] 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 c-sensys05, (New York, NY, USA), pp. 51–63, ACM Press, 2005.

[7] M. J. McGlynn and S. A. Borbash, "Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks," in Proc. 2nd ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc 2001), (New York, NY, USA), pp. 137–145, ACM Press, 2001.

[8] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Initializing newly deployed ad hoc and sensor networks," in Proc. 10th ACM/IEEE Ann. Int'l Conf. Mobile Computing and Networking (MobiCom 2004), pp. 260–274, ACM Press, New York, 2004.

[9] T. Moscibroda, P. von Rickenbach, and R. Wattenhofer, "Analyzing the energy-latency trade-off during the deployment of sensor networks," in Proc. 25th Ann. Joint IEEE Conf. Computer Communication Soc. (Infocom 2006), Apr. 2006.

[10] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in Proc. 4th ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc 2003), (New York, NY, USA), pp. 35–45, ACM Press, 2003.

[11] E. B. Hamida, G. Chelius, and E. Fleury, "Revisiting neighbor discovery with interferences consideration," in PE-WASUN '06: Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks, (New York, NY, USA), pp. 74–81, ACM Press, 2006.

[12] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in Proc. 1st ACM Conf. Embedded Networked Sensor Systems (SenSys 2003), (New York, NY, USA), pp. 14–27, ACM Press, 2003.

[13] A. Keshavarzian, E. Uysal-Biyikoglu, F. Herrmann, and A. Manjeshwar, "Energy-efficient link assessment in wireless sensor networks," in Proc. 23rd Ann. Joint IEEE Conf. Computer Communication Soc. (Infocom 2004), vol. 3, pp. 1751–1761, Mar. 2004.

[14] I. D. Chakeres and E. M. Belding-Royer, "The utility of hello messages for determining link connectivity," in Proc. 5th Int'l Symp. Personal Wireless Multimedia Communications (WPMC 2002), vol. 2, pp. 504–508, Oct. 2002.

[15] 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. 4th European Workshop on Sensor Networks (EWSN 2007), vol. 4373 of Lecture Notes in Computer Science, pp. 195–211, Springer, Berlin, Jan. 2007.

[16] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004), pp. 95–107, ACM Press, New York, 2004.

[17] A. El-Hoiydi and J. Decotignie, "WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks," in Proc. 1st Int'l Workshop Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004) (S. Nikoletseas and J. Rolim, eds.), vol. 3121 of Lecture Notes in Computer Science, pp. 18–31, Springer, Berlin, June 2004.

[18] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," in Proc. 4th ACM Conf. Embedded Networked Sensor Systems (SenSys 2006), (New York, NY, USA), pp. 307–320, ACM Press, 2006.

[19] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in Proc. 4th Int'l Conf. Information Processing Sensor Networks (IPSN '05), pp. 364–369, IEEE, Piscataway, NJ, Apr. 2005.

[20] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, Ambient Intelligence, ch. TinyOS: An Operating System for Sensor Networks, pp. 115–148. Springer, Berlin, 2005.

[21] "BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks." http://www.btnode.ethz.ch.

[22] TIK, "Notengebung bei Studien- und Diplomarbeiten." Computer Engineering and Networks Lab, ETH Zürich, Switzerland, May 1998.

[23] "NCCR-MICS: Swiss National Competence Center on Mobile Information and Communication Systems." http://www.mics.org.

Bibliography

[1] F. Herrmann A. Keshavarzian, E. Uysal-Biyikoglu and A. Manjeshwar. Energy-efficient link assessment in wireless sensor networks. In *Proc. 23rd Ann. Joint IEEE Conf. Computer Communication Soc. (Infocom 2004)*, volume 3, pages 1751–1761, March 2004.

[2] Roman Amstutz. Wake-up time estimation for a wireless mac protocol, August 2007.

[3] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proc. 4th ACM Conf. Embedded Networked Sensor Systems (SenSys 2006)*, pages 307–320, New York, NY, USA, 2006. ACM Press.

[4] Chipcon. *CC2420, Single Chip 2.4 GHz RF Transceiver for IEEE 802.15.4 and ZigBee*, 2003.

[5] D. Culler et al. Tinyos: An operating system for networked sensors. http://www.tinyos.net.

[6] 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. 4th European Workshop on Sensor Networks (EWSN 2007)*, volume 4373 of *Lecture Notes in Computer Science*, pages 195–211. Springer, Berlin, January 2007.

[7] A. El-Hoiydi and J.D. Decotignie. WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. In S. Nikoletseas and J.D.P. Rolim, editors, *Proc. 1st Int'l Workshop Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004)*, volume 3121 of *Lecture Notes in Computer Science*, pages 18–31. Springer, Berlin, June 2004.

[8] Elyes Ben Hamida, Guillaume Chelius, and Eric Fleury. Revisiting neighbor discovery with interferences consideration. In *PE-WASUN '06: Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, pages 74–81, New York, NY, USA, 2006. ACM Press.

[9] L. van Hoesel and P. Havinga. A lightweight medium access protocol (LMAC) for wireless sensor networks. Tokyo, Japan, June 2004.

[10] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proc. 10th ACM/IEEE Ann. Int'l Conf. Mobile Computing and Networking (MobiCom 2004)*, pages 260–274. ACM Press, New York, 2004.

[11] Roman Lim. Wireless fire sensor network demonstrator. Master's thesis, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2006.

[12] Michael J. McGlynn and Steven A. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 137–145, New York, NY, USA, 2001. ACM Press.

[13] Thomas Moscibroda, Pascal von Rickenbach, and Roger Wattenhofer. Analyzing the Energy-Latency Trade-off during the Deployment of Sensor Networks. In *25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Barcelona, Spain*, April 2006.

[14] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 95–107. ACM Press, New York, 2004.

[15] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. 4th Int'l Conf. Information Processing Sensor Networks (IPSN '05)*, pages 364–369. IEEE, Piscataway, NJ, April 2005.

[16] N. Reijers, G. Halkes, and K. Langendoen. Link Layer Measurements in Sensor Networks. In *Proc. 1st Int'l Conf. on Mobile Ad-hoc and Sensor Systems (MASS '04)*, October 2004.

[17] T. Rein. Energy and Time Efficient Link-Quality Estimation for WSNs. Master's thesis, Computer Engineering and Networks Lab, ETH Zurich, Switzerland, April 2007.

[18] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.

[19] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27, New York, NY, USA, 2003. ACM Press.

[20] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *First Int'l Workshop on Embedded Software (EMSOFT 2001)*, pages 1–13, 2003.

[21] Rong Zheng, Jennifer C. Hou, and Lui Sha. Asynchronous wakeup for ad hoc networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 35–45, New York, NY, USA, 2003. ACM Press.

[22] BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks. http://www.btnode.ethz.ch.

[23] Chipcon. http://www.chipcon.com.

[24] Moteiv. http://www.moteiv.com.