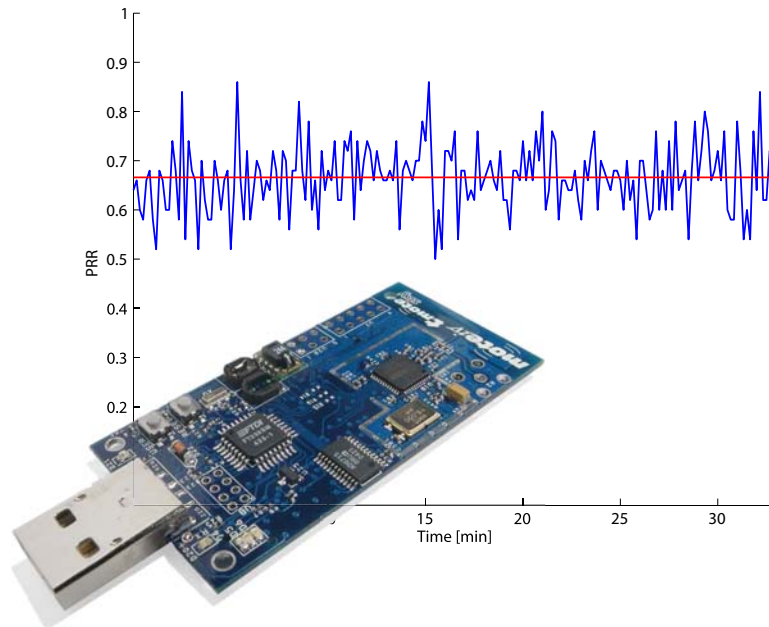


Energy and Time Efficient Link-Quality Estimation for Wireless Sensor Networks

Tobias Rein



MASTER THESIS

Winter Term 2006/07

Supervisor: Andreas Meier

Professor: Dr. Lothar Thiele

April 2007

Contents

<i>1: Introduction</i>	<i>1</i>
1.1 Motivation	1
1.2 Chapters Overview	2
<i>2: Related Work</i>	<i>3</i>
2.1 Metrics	3
2.2 Measurements	4
2.3 Protocols	7
<i>3: Experimental Equipment</i>	<i>9</i>
3.1 Tmote Sky	9
3.2 TinyOS	10
3.3 Deployment Support Network	10
3.4 DSN Analyzer	12
<i>4: LET Platform</i>	<i>13</i>
4.1 Specification & Conception	13
4.2 Implementation	15
4.2.1 Link Test Controller	15
4.2.2 Interface	16
4.2.3 Tmote Sky Software	19
4.3 Validation	21
<i>5: Measurement & Evaluation</i>	<i>27</i>
5.1 Measurement Setup	27
5.2 Measurements	29
5.2.1 Parameters	29
5.2.2 Test Series	30
5.3 Evaluation	31
5.3.1 Temporal Link Behavior	32
5.3.2 Single Link Autocorrelation	34
5.3.3 RSSI, LQI, and PRR Correlation	39

5.3.4	Covariance of Same Source Links	42
5.3.5	Spatial Characteristics of Wireless Links	44
5.3.6	Link Asymmetry	45
<i>6:</i>	<i>Link Estimation</i>	<i>49</i>
6.1	Evaluation Method	49
6.1.1	Link Selection	50
6.1.2	Link Evaluation Tool	52
6.1.3	Record Sampling and Link Estimation Tools	52
6.1.4	Comparison Tool	53
6.1.5	Evaluation of different Estimation Methods	54
6.2	Link-Quality Estimation Algorithms	55
6.2.1	Standard Algorithm	55
6.2.2	Algorithm with different Numbers of Packets	55
6.2.3	Algorithm using a Pattern for the Packet Acquisition	58
6.2.4	Algorithm based on RSSI and LQI values	58
<i>7:</i>	<i>Conclusion and Outlook</i>	<i>65</i>
7.1	Conclusion	65
7.2	Contribution	66
7.3	Outlook	66
<i>A:</i>	<i>Task Description</i>	<i>67</i>
<i>B:</i>	<i>Overview of the Tests</i>	<i>73</i>

Tables

2-1	Overview over past link measurements	6
4-1	Commands for the LET platform	17
4-2	Parameters for the LET commands	18
4-3	Hexadecimal conversion from a byte stream	20
4-4	Channel hopping frequency tests	24
4-5	Results of the validation test	25
4-6	Results of the validation test using the low power mode	25
4-7	Results of the echo validation test	26
5-1	Measurement setup	28
5-2	Parameters for our measurements	29
5-3	Statistics of the data we gathered with our measurements	31
5-4	Relative frequency of the length of packet failure bursts	37
5-5	Relative frequency of the length of packet failure bursts (simulation)	38
5-6	Example for the RNP calculation	39
6-1	RSSI thresholds	60
6-2	Summary of the evaluation of our algorithms.	63

Figures

1-1	Approach for our estimation algorithms	2
3-1	Tmote Sky	10
3-2	Tmote Sky attached to a BTnode	11
3-3	Deployment support network overview	11
4-1	LET test bed overview	16
4-2	LET database	16
4-3	Structure of a LogBlock	19
4-4	LET state graph	22
4-5	Data structure of the external flash	23
5-1	Measurement setup	29
5-2	Histograms to find an interesting transmission power	30
5-3	Temporal behavior with varying numbers of aggregated packets	32
5-4	Temporal behavior of example links	33
5-5	Temporal behavior of unstable Links	34
5-6	Temporal behavior over a week/month	35
5-7	Unstable links over a week/month	36
5-8	Relative frequency (RF) of the length of packet failure bursts	37
5-9	RF of the length of packet failure bursts ($PRR \geq 80\%$)	38
5-10	Required number of packets vs. PRR	39
5-11	Received signal strength vs. PRR	40
5-12	Link quality indicator value vs. PRR	41
5-13	Correlation coefficient of same source links	42
5-14	Conditional probability of packet reception (same source)	43
5-15	Conditional probability of packet reception (same source) for normal distributed data	44
5-16	Spatial characteristics of wireless links	45
5-17	Echo test illustration	46
5-18	Link symmetries	47
5-19	Reception rate of immediate acknowledgments	47
5-20	Sign test for immediate acknowledgments	48

6-1	Evaluation tool chain	49
6-2	Standard deviations of the temporal PRR	51
6-3	Link selection based on the standard deviations	52
6-4	Standard deviation of unstable links (Figure 5-5)	52
6-5	Sliding window for the record sampling	53
6-6	Evaluation of the basic algorithm	56
6-7	Evaluation of the algorithms N-X	56
6-8	Detailed evaluation of algorithms N-30, N-100, N-300	57
6-9	Evaluation of the algorithms P1-X	58
6-10	Detailed evaluation of algorithms P1-1s, P1-10s, P1-1min	59
6-11	Minimal and maximal RSSI values of our measurements	60
6-12	Evaluation of the algorithm R	61
6-13	Detailed evaluation of algorithm R	62
6-14	Minimal and maximal LQI values of our measurements	62

Abstract

The dynamic nature of wireless communication and the strong low-power constraints are major challenges for the design of wireless sensor network applications. Mainly, the instability of wireless links has an adverse effect on the performance, i.e. the reliability of the data transfer.

The main goal of this master thesis is to investigate link-quality estimation for wireless sensor networks. It is shown that efficient estimation algorithms use adaptive numbers of packets in combination with RSSI values.

Moreover, a test bed has been designed and extensive link-quality measurements in an indoor, office-like environment have been performed, using the Tmote Sky sensor node. This implementation of the test bed allows extensions for other sensor nodes and is flexible to move into other environments.

In addition, a profound analysis of wireless link characteristics in a non-deterministic environment is provided. It turned out that some of the links have a highly unstable behavior. Furthermore, comparisons with the results of recent publications have been drawn. These results do not necessarily match with the ones presented in this thesis; most likely due to the change of the environment.

1

Introduction

A wireless sensor network (WSN) consists of spatially distributed autonomous devices, referred to as 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. However, 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 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.

The goal of this thesis is to design an estimation algorithm, which allows estimating the link qualities for WSN in a power and time efficient way. Based on these estimations, an upper-level routing protocol benefits, by using the high-quality links. This increases the stability of the whole network and reduces the power consumption.

Our estimation algorithms should take place in an initial phase of the sensor node (refer to Figure 1-1). This stands in contrast to an adaptive algorithm, which estimates the link qualities continuously during the productive time of the node. With modern low-power MAC protocols, these adaptive algorithms are hardly possible, because these protocols shut down the radio module whenever possible and minimize the reception of packets addressed to other nodes.

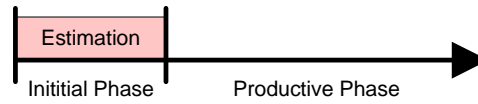


Figure 1-1

This figure illustrates the approach for our estimation algorithms, which take place in the initial phase of the sensor node.

1.2 Chapters Overview

This thesis is divided in six parts. In Chapter 2, we give a detailed overview of related work, by analyzing and comparing them. In Chapter 3, we illustrate the experimental equipment and the software framework we were using. In Chapter 4, we provide implementation details of the test platform, which we designed for our measurements. In Chapter 5, we describe our link-test procedure and analyze various characteristics of the behavior of wireless links in detail, based on extensive link measurements in an indoor environment. In Chapter 6, we present different link estimation algorithms and evaluate them. Chapter 7 concludes this thesis and gives an outlook on future work.

2

Related Work

Many recent studies have discussed the complexity of wireless communication in sensor networks. The behavior of the unpredictable wireless-links has been tested in many experiments with various radio modules, metrics and environments. The procedure of all these link tests is the same: one sender broadcasts packets at a given frequency and one or more nodes listen and record the received packets. Then, conclusions are drawn, based on the results. This chapter shows an overview of these previous works. First, the different metrics that have been used are going to be discussed. Secondly, a summary over the published measurements and their results is given, and finally resulting algorithms for routing protocols are presented.

2.1 Metrics

For wireless-link estimation, the choice of an appropriate characteristic with its metrics is fundamental. A very basic but also highly important indicator is the packet reception rate (PRR) with its complement, the packet loss rate. The PRR is basically the number of received packets at one node in relation to all the sent packets by the transmitter. Most of the earlier studies are based on these indicators, combined with the received signal strength indicator (RSSI) [22]. In addition to the RSSI value, newer radios based on IEEE 802.15.4 standard implement a parameter called link quality indication (LQI). RSSI and LQI have been discussed in detail in the past. First measurements showed that RSSI is a poor indicator of link quality [21]. LQI was believed to be a better indicator than RSSI. Srinivasan and Levis [17] have done measurements with a second generation radio chips (CC2420) and disproved these earlier results. Their observations are clearly different from the general belief that LQI is a good indicator of link quality and RSSI is not. They found that RSSI is a promising quantity to be measured when its value is above a certain sensitivity threshold. Concerning the LQI, they found that a better correlation with the packet reception rate could be achieved by computing the mean LQI over many packets. Lal et al. [9] measured an additional noise value on the channel, right after a packet has been received. With this extra information and the RSSI value, they

build another metric, the so called signal to noise ratio (SNR). They defined a good link as one that has a SNR greater than 25dB and a moderate one, when the SNR fluctuates between 25 and 15dB. Is the SNR larger than 25dB, the PRR measured is 100% for almost all the time.

The link estimation metrics mentioned above do not take the time-related consideration of packet losses into account. To retain this temporal aspect, the required number of packets (RNP) has been introduced by Cerpa et al. [2]. For every sent packet, the number of required retransmissions until a success is evaluated and averaged over a certain window size. Temporal losses of link will increase these metrics much more than uniform distributed link failures.

Son and Heidemann [15] extended the traditional experiments by taking a sender and one or more interferers sending simultaneously. Considering the fact of having interference, they established the signal-to-interference-and-noise-ratio (SINR) for concurrent packet transmission. They could confirm the existence of the SINR threshold. Unfortunately, this threshold is not a constant value but depends on signal strength and the transmitter hardware. These kinds of experiments are beyond the scope of this thesis and are not going to be discussed later on.

2.2 Measurements

This section gives a detailed overview of previous measurements. Table 2-1 at the end of this paragraph summarizes this discussion.

One of the first attempt of systematic measurements of packet delivery in wireless sensor networks has been performed by Zhao et al. [22] in 2003. They placed Mica nodes in a simple linear topology in three different environments: an indoor office building, a habitat with moderate foliage and an open parking lot. Based on their measurements, they divided the communication range of the node in three regions: a region close to the sender in which all nodes receive most of the packets, a region out of range, and the 'gray area' in between, the region at the edge of the communication range in which the reception rate varies dramatically; some nodes see nearly 90% successful reception, while neighboring nodes sometimes have less than 50% reception rate. Against their expectations, this area has a significant extent. While the 'gray area', measured on the parking lot, covers 10% of the total communication range, it covers 30% of the measurements in a habitat and 50% of the measurements in the office building. They refer their findings to multi-path signal delivery. Moreover, they found significant asymmetry in realistic environments but were not able to establish causes for their findings.

Woo et al. [21] measured signal strengths in a uniform grid over a large, essentially unobstructed indoor space with 50 nodes (Mica). The results showed that both, the mean link quality, and the variance in quality are a function of distance. They determined three different regions: the effective region where the reception rate is above 90%, the transitional region where some of the links are good and others are not, and the clear region where no more packets can be received. The borders of these regions lie at about 10 and 40 feets, respectively. These findings are comparable to the

'gray area' defined by Zhao et al. [22]. Furthermore, they confirmed the existence of asymmetric connectivity.

The study of Willig and Mitschke [20] analyzed bit errors of the received packets in a straight line arrangement. Their measurements pointed out that bit errors occur in bursts, but packet failures do not; in 98.4% only a single packet failure happens. Thus, they stated that due to the occurrence of bit error bursts, forward error correction is not power-efficient and therefore not meaningful. However, why do they propose to postpone the retransmission of automatic repeat request (ARQ) schemes for a short time although in the majority of the cases only a single packet gets lost?

In a more recent survey by Srinivasan et al. [16] about long term behaviors, they were not able to reproduce this effects by Willig and Mitschke [20]. They showed that packet losses are highly correlated over short-time periods but are independent over longer terms. Analogously, they found that long-term link asymmetries are rare, even though short-term asymmetries are not uncommon. The dissimilar results of these publications may be founded in the differences of the radio chips [17].

Cerpa et al. [2] performed in-depth analysis of the temporal properties of wireless links. In an indoor office setting, they placed 55 nodes (Mica, RFM TR1000) arranged in a grid structure with approximately 1 m mesh. The tests had durations from 24 up to 96 hours. In general, they could not determine a particular time during the day, where the average quality of all links from a transmitter is significantly better or worse. That means that the global quality bandwidth of the system has very little oscillations over time. They showed that the RNP metric is not directly proportional to $1/PRR$ and suggested using RNP instead of PRR to estimate link quality, because it includes the underlying distribution of losses. In their measurements, they could not quantify the correlation neither between distance and PRR nor between distance and RNP for links, especially in the 'gray area'. They also determined the covariance of same source links, and concluded that using single path routing instead of multiple path strategies is more convenient if only high quality links are involved. If a packet from a given sender arrives at a node connected with a good link, there is a high probability that the same packet arrives at all nodes connected with good quality links to the sender. Furthermore, they analyzed the forward and reverse link correlation and concluded that there is a significant benefit to acknowledge packets immediately, so that the chances of the acknowledgement being received increase.

Based on the findings of Srinivasan et al. [16], Reijers et al. [14] extensively investigated the 'gray area' and the influence of the environment. In a linear setup with 51 nodes, they measured in three different surrounding areas: indoor, outdoor, and open space. They pointed out that the hardest environment is the indoor environment, a corridor in their building. On the tennis court (outdoor) and the hockey field (open space), in fact, the 'gray area' was much less pronounced than in the corridor but still perceptible. They interpreted this outcome as a result of the multi-path reflections, which occur in the indoor environment. Furthermore, they quantified the directionality of the nodes' antennas. The resulting 8-like shape matches reasonably with the theoretical radiation pattern, but again, the indoor environment polled badly. They also did symmetry testing and achieved advantageous results. Just like Srinivasan et al. [16], they distinguished between good, medium and bad

Reference	Environment	Topology	Platform	Focus
Zhao et al. [22]	indoor, habitat, parking lot	linear (up to 60 nodes)	Mica	Prevalence of the 'gray area'
Woo et al. [21]	indoor	grid (50 nodes)	Mica	PRR against distance and link asymmetries
Willig and Mitschke [20]	indoor	linear	EBS (868 MHz)	Bit errors and FEC, ARQ
Srinivasan et al. [16]	indoor and outdoor	random (e.g. Mirage Testbed, 100 nodes)	Telos, MicaZ	Long term behaviour
Cerpa et al. [2]	indoor	grid, 55 nodes	Mica1	Temporal properties
Reijers et al. [14]	indoor, outdoor, open space (hockey field)	linear (51 nodes)	Proprietary node similar to Mica (868 MHz)	Investigate the 'gray area'
Lal et al. [9]	indoor	various	Bosch (916 MHz)	Estimate global link quality out of little data

Table 2-1: This table gives an overview of past link measurements. The table lists the environment, the topology, and the used platform of prior work. Additionally, the main focus of these measurements is given.

links. In approximately 88%, the reverse link of a good one is also good and in 79% of a bad link, the reverse link is bad, too. These findings apply for the full dataset as well as just for the links in the 'gray area'. Concerning the RSSI value, they found a threshold above which reception is consistently good. Unfortunately, this value depends on the environment and if RSSI wants to be used to qualify links as 'good', it either has to be chosen conservatively, or a priori knowledge of the environment is necessary.

Lal et al. [9] focused on the SNR value and on the packet-loss rate, referred to as link inefficiency. They took single measurements out of long-term experiments and tried to estimate the global link quality out of this little data. It seems that around 10 samples are sufficient to get a very good estimation of link inefficiency for bad links. Moreover, their observations show that with a surprisingly few number of samples, a reliable cost metric for grading links can be given. These samples can be taken during the same day or at random time over several days, depending on the specific constraints of an upper level protocol.

2.3 Protocols

The work mentioned above has been done to improve the efficiency of upper layer protocols. Lin et al. [12] designed an algorithm that adapts the transmission power to the link quality based on the relation between RSSI/LQI and transmission power. Stann et al. [18] developed a protocol that resides as a service between the MAC and network layer. With only local information, they could achieve nearly a perfect flooding by requiring moderate broadcast reliability (50-70%) when nodes have many neighbours.

Out of their sound measurements, Cerpa et al. [2] designed a centralized and a distributed routing algorithm. The first is based on the Dijkstra algorithm and considers correlation of successive links in multihop communication. The second uses statistics to establish probabilistic gradients on the forwarding path.

Woo et al. [21] analyzed a set of routing protocols including Shortest Path, Minimum Transmission, and Broadcast algorithms. They established four important metrics to evaluate these protocols: hop distribution, path reliability, end-to-end success rate and stability.

3

Experimental Equipment

In this Chapter, we present the components of our test bed. We first give an overview of the specification and the features of the Tmote Sky, a state of the art sensor node we are using for our link tests. Later on, we briefly describe the TinyOS, an operating system, designed for wireless embedded sensor networks. Moreover, we introduce the Deployment Support Network, and finally, we characterize the DSN-Analyzer.

3.1 Tmote Sky

Developed and designed at the University of California, Berkeley, the Tmote Sky is a next-generation mote platform (see Figure 3-1). It is an enhancement from the Telos Revision A/B 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 integrated AD/DA converter, the DMA Controller, the fast wakeup from sleep, the ultra low current consumption, and the TinyOS support.

Additionally, the Tmote Sky is equipped with the radio chip CC2420 by Chipcon [23] and an integrated on-chip antenna with 50m range indoors and 125m range outdoors. This transceiver is the first IEEE 802.15.4 radio, ZigBee [25] ready and designed for low-power and low-voltage wireless applications. It operates in the 2.4 GHz band, has a data transfer rate of 250 kbps, and uses offset quadratic phase shift keying (OQPSK) with a direct-sequence-spread-spectrum (DSSS) encoding. IEEE 802.15.4 specifies 16 channels numbered from 11 to 26. The centre frequencies are separated by 5 MHz and range from 2.405 GHz, channel 11, up to 2.480 GHz, channel 26. In addition to the Received Signal Strength Indicator (RSSI), IEEE 802.15.4 states to provide a Link Quality Indicator (LQI) value, which is reported with each received packet. LQI is a characterization of the strength and quality of the incoming packets. It can be thought as chip error rate and is calculated over 8 symbols following the start frame delimiter. LQI values generated by the Tmote Sky are usually between 50 and 110 and correspond to minimum and maximum quality frames.

3.2 TinyOS

TinyOS [5] is an open-source framework, designed for wireless sensor networks. It features a component-based architecture, which enables rapid innovation and implementation while minimizing code size, as required by the tight memory constraints of sensor networks. TinyOS's library includes different network protocols, distributed services, sensor drivers, and data acquisition tools. The event-driven execution model of the TinyOS enables fine-grained power management and allows the scheduling flexibility, made necessary by the unpredictable nature of wireless communication and physical world interfaces.

TinyOS has been ported to over a dozen platforms and numerous sensor boards such as TelosB or Mica2. A wide community uses it to develop and test various algorithms and protocols. The latest release TinyOS 2.0 provides many advantages over earlier versions, including greater robustness, integrated power, resource management, and an experimental low-power CC2420 radio stack.

The programming language of TinyOS is a dialect of C, called nesC. It supports components, which provide and require interfaces. An application consists of a configuration that wires different components and other configurations. Important restrictions of nesC are that there exist no function pointers, no dynamic memory allocation, and no dynamic component installation or destruction. On the other hand, these static requirements enable static analysis and optimizations, which are fundamental for the limitations of a sensor node.

3.3 Deployment Support Network

Developing applications for Wireless Sensor Network (WSN) involves various challenges. Monitoring, testing, controlling, and reprogramming the application on many sensor nodes are time-consuming and impractical tasks if every node has to be addressed separately using serial cables. The Deployment Support Network (DSN) [6] is a tool that reduces this exhausting job. With one static connection to the DSN, it is possible to supervise, reprogram, and control all the nodes in a WSN.

The DSN itself is also a Wireless Sensor Network, more precisely a multihop ad-hoc network, based on BTnodes [1] using Bluetooth. One of the BTnodes (called GUI



Figure 3-1
Tmote Sky. A next-generation mote platform distributed by Moteiv Corporation.

node) is connected to a PC, which allows the interaction with the DSN. The other BTnodes are attached to the sensor nodes of the WSN (also referred to as target nodes) by using a target adapter (see Figure 3-2). In doing so, the DSN builds a second, independent, overlying wireless network, as shown in Figure 3-3. On this backbone network, commands and messages can be sent from the target nodes to the GUI node, as well as from the PC to each target node.



Figure 3-2

This image shows the target adapter that connects a Tmote Sky with a BTnode from the DSN.

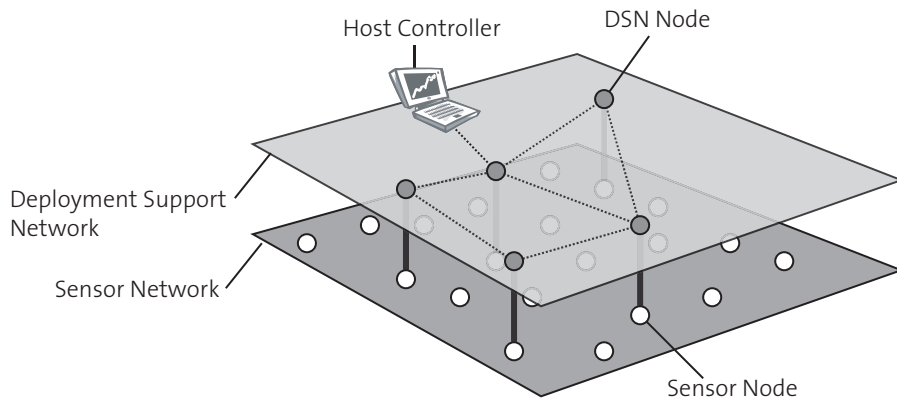


Figure 3-3

This illustration shows the Deployment Support Network (DSN) with the target nodes. The DSN builds an independent, overlying network to monitor and control the sensor network. Several nodes, which need to be observed, must be attached to a BTnode of the DSN.

To profit from the DSN, an interface from the BTnodes to the particular target node needs to be specified. This consists of hardware (target adapter) and of software for the BTnode and for the target node. For the Tmote Sky, these adaptations have been done in a previous master thesis by Roman Lim [11]. The implementation uses the UART1 interface of the Tmote Sky and contains a bootloader to reprogram the target node over the DSN and a TinyOS component, implementing an interface, which abstracts the communication with the DSN.

The DSN has some limitations, which turned out to be very important for the implementation of our link quality measurement tool. The bandwidth of the DSN is

limited to approximately 10 packets per second for the whole network. If the load is too high, the risk of a total crash of the DSN increases drastically. Additionally, the reliability of a successful packet transmission is not given. Even if the load is feasible, some packets might get lost, although there is a packet recovery mechanism implemented. Finally, a DSN node can loose the connection to the DSN temporally. All these restriction impact the conception of the link quality measurement tool, as described in Section 4.1.

3.4 DSN Analyzer

The DSN Analyzer is a java tool to perform link tests. As the name implies, the DSN Analyzer is based on the DSN and additionally, it is developed for the specific target node Siemens A80. Designed in the context of a master thesis by Patrice Oehen [13], the DSN Analyzer automated the execution of a link test. It initializes the nodes with given parameters; it starts the test and gets the resulting data. The graphical user interface of the DSN Analyzer allows the user to generate histograms and other kind of charts out of the measured data. It supports many auxiliary features such as 3-d map of the target nodes placement, graphical analyze view of the links, or the possibility to perform test in series.

4

LET Platform

This chapter outlines the details of our Link Estimation Test (LET) platform, a utility to perform automated link quality tests. In the first section, we state the problems and major challenges of running such measurements, and discuss other specifications of the platform. Additionally, we present our concept and legitimate why the DSNAnalyzer, presented in Chapter 3.4, is not being used. Then, we describe the implementation of the LET Platform. Finally, we give information about the validation of this platform.

4.1 Specification & Conception

Our link quality measurements proceed as follows: A sender broadcasts a number of packets with a given frequency; one or more receivers listen continuously and log the data of received packets. With the measured information, conclusions about link quality can be drawn as qualified in Chapter 5. To avoid interference within our network, there is only one node sending at the same time. Although this seems to be simple, link quality measurements involve many fundamental decisions, which are going to be discussed in this section.

Basically, a link quality test is described by parameters such as transmission power, number of packets to be sent, frequency of sending consecutive packets, packet length, and the channel. Additionally, our tool provides three different scenarios:

One sender This is the basic scenario, described in the preface of this section. There is one sender broadcasting packets and an arbitrary number of receivers log the information about received packets.

Echo Test An echo test takes place between one sender and one receiver. Additionally to the basic test, the receiver immediately responds to every received message by sending a packet back to the sender. Both, the sender and the receiver record the received packets.

Channel Hopping The last scenario is to identify the performance of different channels. Such a test needs one sender and several receivers. Normally, the

receivers are placed side by side and each of them listens to a different channel. The sender broadcasts bursts of messages, this means that every burst consists of one packet on each specified channel.

Furthermore, our platform supports a low power feature, which affords to switch the radio off between two consecutive packets. Since there is no additional synchronization mechanism between the nodes, the receivers simply switch the radio off for a certain uncritical duration after having received a packet. In this context, uncritical duration means that it is guaranteed that the radio is on before the sender broadcasts the next packet, i.e. taking uncertainties like the clock drift into account.

In the following, we discuss other characteristics, which are relevant for our link estimation tests. These specifications affect the implementation directly and are not parameterizable.

Packet reception A very basic issue is the question of which packets are accepted at the receiver. We specified that a packet is correct only if all bits are received correctly. Another approach would be to accept all received packets. Additionally, this manner permits to determine bit errors with their location. However, due to the fact that forward error correction is inefficient in wireless sensor networks [21, 20], we reject incorrectly received packets.

Sequence numbers More important than to detect bit errors is to have packets with sequence numbers. By means of sequence numbers it is possible to trace lost packets in a packet stream, which allows to evaluate if there are bursts or striking patterns of packet losses. Without this feature, it would only be possible to obtain a statistical analysis, i.e. how many packets are received.

Link quality parameters (RSSI, LQI, noise level) Additionally to the sequence number, link quality parameters for each packet are available. As described in Chapter 3.1, IEEE 802.15.4 specifies two of them: Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI). Their quality for link estimation has been intensely disputed in the community (refer to Section 2.1). To reproduce published findings and to form our own opinion, we decided to monitor both, the RSSI and the LQI value.

According to the measurements of Lal et al. [9], there is another value appended to the data record of each packet. This value indicates the noise level of the channel right after a packet has been received.

Based on the specification mentioned above, we established the conception of our link measurement tool. The first idea was to create a TinyOS implementation for the Tmote Sky, which cooperates with the DSNAnalyzer (see Section 3.4). Using an existing test bed would have been the most convenient way to do link quality measurements. Unfortunately, the DSNAnalyzer has several constraints, which do not correspond with the specification given before:

- First of all, the DSNAnalyzer is based on bit errors. This might make sense for nodes with bit stream based radio, but not for the packet based radio CC2420 on the Tmote Sky. Moreover, including sequence numbers in packets while checking for bit errors is unfeasible.

- The DSNAnalyzer has very limited functionalities to evaluate RSSI and noise values. It is possible to measure these data for every received packet, but this implies a high load for the Deployment Support Network. If a node receives a packet, it immediately sends a message with the measured RSSI and noise value back to the DSN. Practical experiences have shown that if all nodes together generate more than 10 packets per second, the DSN crashes and packets get lost.
- Another reason not to use the DSNAnalyzer is the interface between the software running on the nodes and the DSNAnalyzer implemented on the host PC. The DSNAnalyzer uses JSON-RPC, a modification of XML-RPC [4]. This is quite a simple interface, but in our opinion, still too much overhead for a sensor node, and more important the DSN.

Due to these restrictions, we decided to implement our own link test tool. To be as independent from the reliability of the DSN as possible, we determined to divide our tests in two separate steps:

- I. In the first step, the measurements take place. After the nodes have been initialized, the sender broadcasts the specified amount of packets while the receivers store all the received sequence numbers with the associated link quality parameters in the internal memory. Except for the initialization, this step raises no traffic for the DSN and therefore, a DSN crash has no consequences for our test.
- II. In the second step, the data stored by the nodes are collected and written to a database. This leads to a high load for the DSN. Since all relevant data are consistently saved on the nodes themselves, a packet loss of the DSN does not result in a leakage of our measured data; we can re-demand them. Moreover, we are not susceptible on a temporary crash of the DSN.

4.2 Implementation

The implementation of the LET Platform includes two main parts; the software for the Tmote Sky and a server tool, named Link Test Controller, to control the tests and gather the results (see Figure 4-1). For the communication in between, we use the DSN and a well-defined interface.

4.2.1 Link Test Controller

The Link Test Controller is a java tool, running on a windows computer, which controls the process flow of a test and stores the measured data into a database. It uses the DSN and the interface given in Section 4.2.2 to communicate with the nodes. First, the Link Test Controller gets the information for a specific test out of the LET database (see Figure 4-2). In this database, all the parameters, information about participating nodes, and settings for the DSN are stored. With this data, the controller is able to initialize the nodes and to start the test. During the execution of the test, the controller is waiting. As soon as the sender signals that the test is finished, the Link Test Controller starts the collection of the data. Node by node it requests

the data stored in the internal memory of each node. If data packets get lost on the DSN, the controller re-demands the appropriate information until all data of all participant nodes have been collected.

The Link Test Controller includes several mechanisms, which take the unreliability of the DSN into account. In addition, the tool checks if a node is reachable before a message is sent. In so doing, we avoid unnecessary traffic on the DSN.

4.2.2 Interface

A simple interface between the Tmote Sky and the Link Test Controller provides commands to initialize and start tests, and to collect the data. Due to the unreliability of the DSN, all commands coming from the Link Test Controller need to be acknowledged by the Tmote Sky. Table 4-1 shows these commands with the according return message from the Tmote Sky. Additionally, the commands `init`

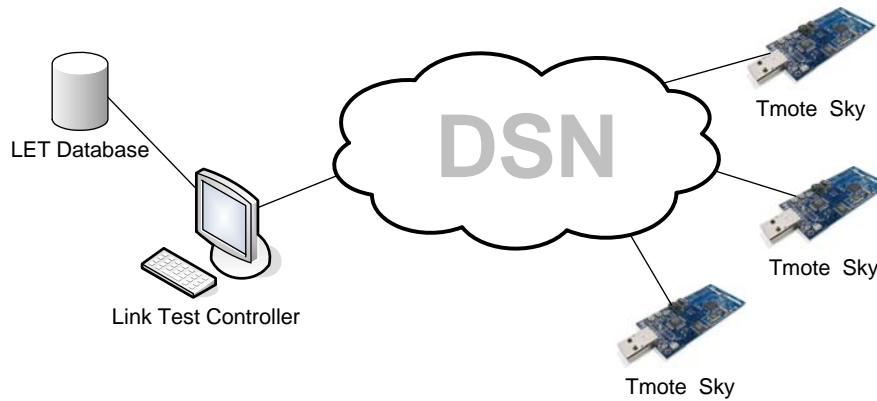


Figure 4-1

This figure gives an overview of the LET test bed. The Link Test Controller controls the test, which runs on the Tmote Skys, using the DSN. It gets the parameters for each test out of the LET database and stores the gathered measurements in it.

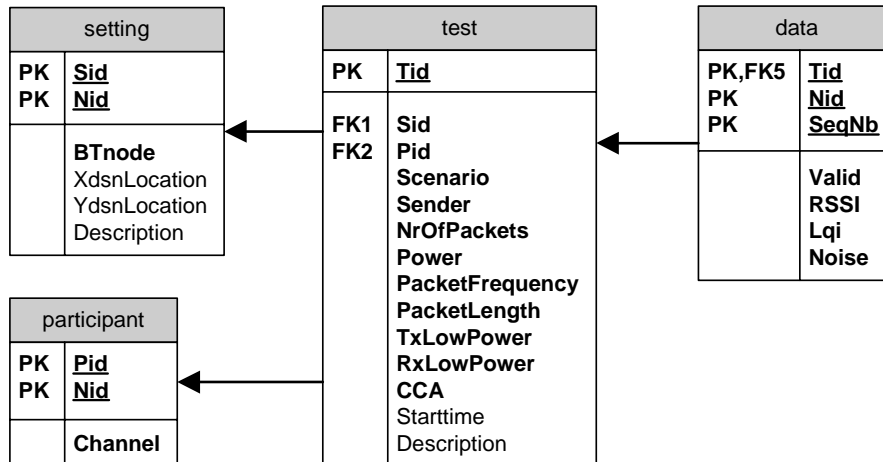


Figure 4-2

Design of the LET database. All settings for a test and the resulting measurements are stored in this database.

Command	Description	Return Message
init	Initializes the node with the given parameters	ready TX or ready RX
start	Starts a test	test started
gData	Collects the data of the node	data collection started: <n>: blocks expected (where <n> stands for the number of blocks stored in the memory)
stop	Stops either a test or the data collection	test stopped or data collection stopped
gStatus	Gives information about the nodes' state	all the above mentioned messages and additionally initializing, test finished and LinkEstimator started

Table 4-1: Commands for the LET platform. With this commands the Tmote Sky can be controlled over the DSN. Some of them can have arguments as shown in Table 4-2.

and `gData` can be passed with parameters as shown in Table 4-2. The syntax of a command with parameters is similar to a function call, known from standard programming languages. The parameters follow the command name, are embraced by brackets, and comma-separated (`command(parameter1, parameter2, ...)`). The commands and the parameters are transmitted as strings; therefore, some of the parameters need to be converted to the recommended data type by the nodes. An advantage of this method is the possibility to convert a parameter into different data types. This has been done for the parameter `channel` in the `init` command, which is either interpreted as single value or as array. Additionally, the parameters can be omitted as often used for the `gData` command. In this case, the default values are taken.

For the transfer of the measured data during the data collection, an efficient interface has been chosen. The measurements of several packet during a test are combined to blocks, called `LogBlock`, as shown in Figure 4-3. Every `LogBlock` is exactly 84 bytes and includes test and node id, block number, sequence number of the first packet logged in this block, a bit mask indicating which packets have been successfully received, and the recorded link quality parameters. To save memory space, we do not store zeros for RSSI, LQI and noise level of not received packet. Therefore, it is not evident in advance which link quality parameters belong to which packet. The sequence number and the bit mask need to be evaluated first to conclude this combination; whereas each bit of the bit mask stands for a consecutive packet applying 1 and 0 to a received or not received packet, respectively. Since each `LogBlock` contains a different number of logged packets, an additional field in the `LogBlock` indicates the valid length of the bit mask. There are two limitations for the quantity of packets in a `LogBlock`. First, only 22 triples of RSSI, LQI, and noise fit into the 66-byte tail of a `LogBlock`. Secondly, the 8-byte bit mask allows a range of 64

Parameter	Description	Range	Default
Parameters for the <code>init</code> command			
test id	A unique id for the test		
scenario	The scenario which is used	1 (One Sender), 2 (Echo), 3 (Channel Hop- ping)	1
sender id	The id of the sender of this test		
power	The transmission power	[0 ... 31]	31
channel or [chan- nel, channel, ...]	The channel used for the test. If the scenario is Channel Hopping, the sender needs to know all the channels	[11 ... 26]	26
number of packets	The number of packets which should be sent during the test		1000
packet frequency	The time between two sent packets in miliseconds	>150	200
packet length	The additional payload for a packet (Four bytes are used for the sequence number)	[0 ... 24]	12
tx low power	Specifies whether the sender uses low power mode or not	1 (true), 0 (false)	0
rx low power	Specifies whether the receivers use low power mode or not	1 (true), 0 (false)	0
clear channel assessment (CCA)	Allows to disable the CCA, which is normally done by the TinyOS	1 (true), 0 (false)	1
Parameters for the <code>gData</code> command			
block id	The specified blocks which need to be retransmitted		0
Number of blocks	The amount of consecutive blocks which needed to be sent	(0 means all packets till the end is reached)	0

Table 4-2: This table shows the parameters for the LET commands. To use these parameters, they must follow the command, be embraced by brackets, and comma-separated. The order corresponds to the order in this table.

sequence numbers from the first to the last received packet.

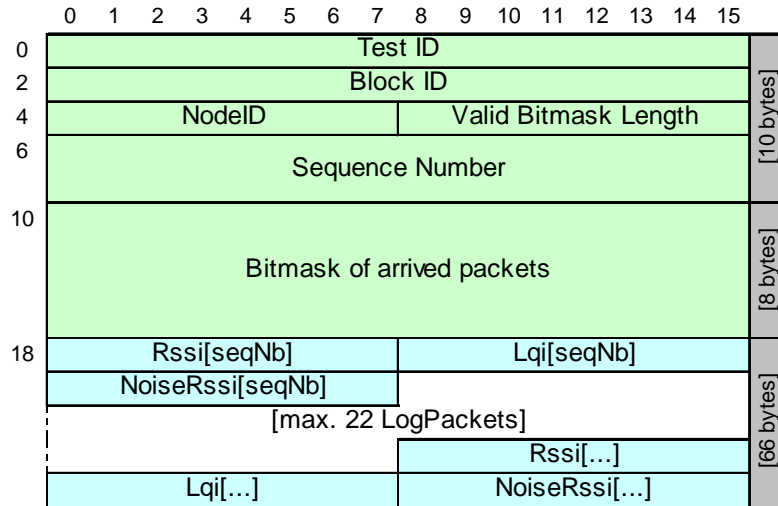


Figure 4-3

This schema shows the structure of the so called LogBlock. The data stored in the memory of each node are structured in this way to transfer them over the DSN to the Link Test Controller.

To avoid complications with the byte encodings of the DSN, we convert each block into a hexadecimal stream as described in Chapter 4.2.3.1. This results in a DSN packet with a length of 168 bytes for each LogBlock. Due to restrictions of the DSN, we limited the maximum frequency of sending such packets successively to 5 packets per second (one packet every 200ms).

4.2.3 Tmote Sky Software

The software running on the Tmote Sky Platforms is a TinyOS application written in nesC (refer to Chapter 3.2). It consists of four main parts; a component for the communication with the DSN, one for parsing incoming commands, the main component executing the link tests, and one handling the data storage. In addition, we use several components for radio control and memory access, provided by the TinyOS 2.0 kernel.

4.2.3.1 DSN Component

The DSN component has been implemented and updated to a TinyOS 2.0 component by Roman Lim [11] within the scope of his master thesis. It provides the communication between the DSN nodes and the Tmote Sky target, using the UART1 interface. It enables to receive commands from the DSN, as well as to send log messages. The communication is based on ASCII encoding with the restriction of the newline character (`\n`), which is used as message delimiter.

Additionally, we added the functionality of writing binary streams to the DSN. This feature has been implemented to transfer the test data efficiently to the Link Test Controller. As mentioned above, the DSN is not able to transfer the newline character (decimal 10, binary '00001010'), because it represents the message delimiter.

Byte	Hex Symbols	DSN Byte Stream
0000 0000	0 0	00110000 00110000
0000 0001	0 1	00110000 00110001
0000 0010	0 2	00110000 00110010
0000 0011	0 3	00110000 00110011
⋮	⋮	⋮
0000 1010	0 A	00000000 01000001
0000 1011	0 B	00000000 01000010
0001 1100	0 C	00000000 01000011
0001 1101	0 D	00000000 01000100
0000 1110	0 E	00110000 01000101
0000 1111	0 F	00110000 01000110
0001 0000	1 0	00110001 00110000
0001 0001	1 1	00110001 00110001
⋮	⋮	⋮
1111 1100	F C	01000110 01000011
1111 1101	F D	01000110 01000011
1111 1110	F E	01000110 01000101
1111 1111	F F	01000110 01000110

Table 4-3: This table shows the hexadecimal conversion of a byte stream. Each byte is divided in two parts, converted to hexadecimal symbols and interpreted as ASCII characters.

Moreover, it turned out that the DSN backend has several other forbidden characters, which cannot be treated correctly. Therefore, we decided to convert the binary message into a hexadecimal data stream. This means that every byte of the bits stream is divided in two 4-bit parts, each representing a hexadecimal symbol (0-9 and A-F). This results in a string consisting of only hexadecimal symbols. By sending this stream to the DSN, the symbols are interpreted as ASCII characters and converted to the appropriate byte stream (refer to Table 4-3). With this procedure, we are able to send an arbitrary byte sequence over the DSN, without having any troubles. Unfortunately, the conversion to hexadecimal characters doubles the load for the DSN.

4.2.3.2 Parser Component

The Parser component parses the messages received from the DSN component. There are well defined commands, which are understood (refer to Section 4.2.2). If a received message matches one of the commands, the parser reads the arguments and converts the ASCII characters into the required data type if necessary. In addition, it sets undefined parameters to their default value. Finally, an event to the main component, called Link Estimation Test, is signaled. Should a message not match one of the defined commands, an error is sent back to the DSN.

4.2.3.3 Link Estimation Test Component

The Link Estimation Test component is the main component of the Tmote Sky software and implements the state machine, represented in Figure 4-4. Driven by commands coming from the Parser component, it is responsible for the workflow of the link test. There exist three main states: `SEND`, `RECEIVE`, and `COLLECT`.

During the initialization procedure, the parameters are set and the flash is deleted. Afterwards, the node changes to the `SEND` (`SEND_READY`) or `RECEIVE` state, depending on the parameters of the `init` command. If the sender receives the `start` command, it advances to the `SEND_SENDING` sub-state and starts broadcasting the packet stream. All specifications have been transmitted with the `init` command: sending power, channel, frequency, packet length, number of packets, low power, and CCA mode. In this state, the sender accepts only the `stop` and the `gStatus` commands; all the others are ignored. On the other hand, all nodes in receive state log the received packets. Depending on the assigned scenario, the receiver sends an echo back to the sender for every received packet; this one writes the received answers in its flash, too. As soon as the sender finishes the test, a message is generated, saying the test has been finished and is sent to the DSN. The sender itself switches to the `SEND_DONE` sub-state.

After performing the link test, the data collection takes place. If a node receives the command `gData`, it first reads the whole flash to get information about the number of DSN packets it has to send and moves to the `COLLECT_COLLECTING` state. Accordingly, it starts sending these packets to the DSN. Once all the data has been transmitted, it changes to the `COLLECT_DONE` sub-state. If packets got lost on the DSN and need to be retransmitted, the `gData` command can be recalled with parameters, indicating specific packets to be resent.

A special feature of this component allows collecting the data out of the `UNINITIALIZED` state. This has been implemented for the case a node crashes or has been removed from the power supply.

4.2.3.4 Data Storage Component

The Data Storage component provides the interface to write data to the external flash (ST M25P80). It stores the information coming from the received packets as shown in Figure 4-5. During the initialization, an `InitBlock` is written at the very beginning of the flash, containing all the information specified in the `init` command. While receiving packets, the measured data are internally stored until a block, called `LogBlock`, has been filled up. These `LogBlocks` have the same structure as the blocks discussed in Section 4.2.2. During the data collection, the Data Storage Component supports reading these blocks either consecutively or individual and returns them to the Link Estimation Test Component.

4.3 Validation

The LET Platform has been validated by testing several critical components individually. In addition to these separate tests, a full link test with 5 nodes has affirmed the whole functionality of our tool.

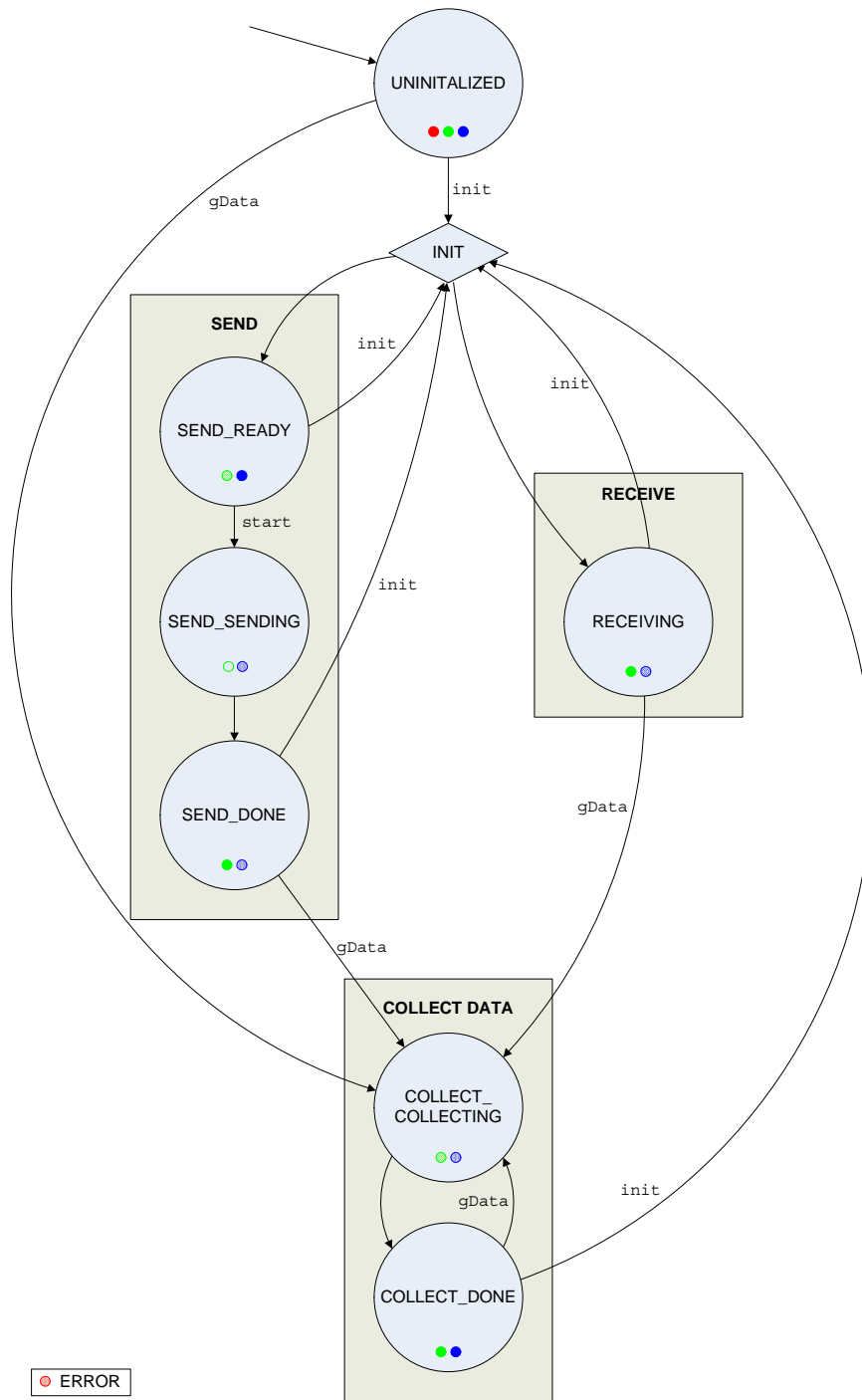


Figure 4-4

This illustration shows the state graph implemented by the Tmote Sky software. There are basically three stats, but some of them are divided into sub-states. The labels on the transitions are commands coming from the Link Test Controller (see Table 4-1). The dots in each state indicate the configuration of the LEDs on the node. If the red LED is on, an error has occurred in the application.

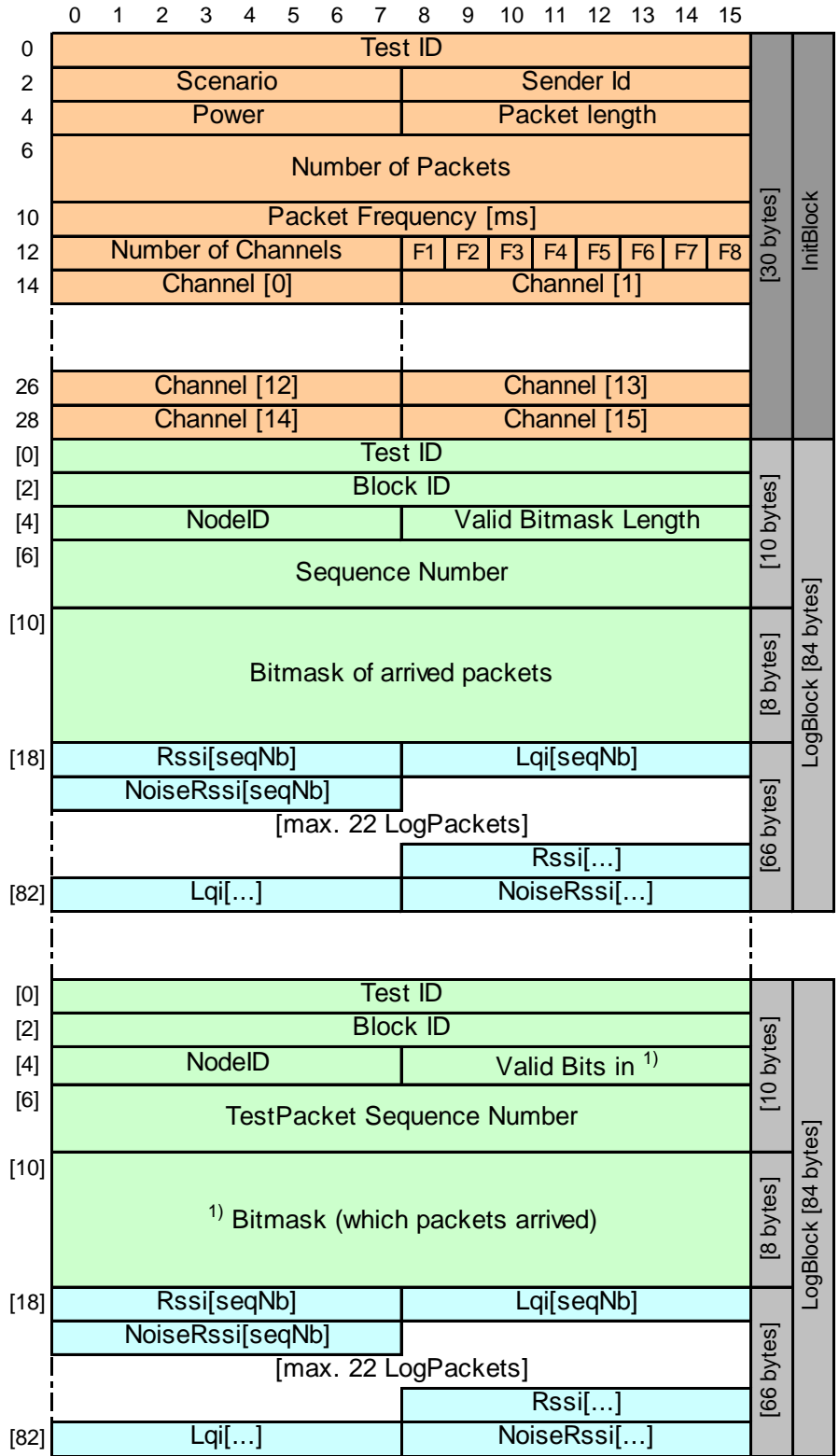


Figure 4-5
 This is the data structure of the external flash of the Tmote Sky. First, an InitBlock is written. Then, several Blocks with the same structure are stored. These blocks, called LogBlocks, contain the measured data.

Node id	Time [ms]		Maximum time per channel [ms]
	Minimum	Maximum	
3 channels (consecutive): 11, 12, 13			
3	27	58	19.3
6	27	54	
12	26	51	
3 channels: 26, 11, 19			
3	26	53	17.7
6	27	53	
12	25	53	
8 channels: 11, 14, 16, 18, 20, 22, 24, 26			
3	90	127	15.9
6	91	123	
12	92	124	
16 channels: 11 ... 26			
3	184	245	15.5
6	178	248	
12	175	245	
3 channels: random order			
3	180	245	15.6
6	176	250	
12	173	249	

Table 4-4: This table shows the result of the test, concerning the channel hopping frequency. The measurements are out of 200 bursts

A crucial value for the implementation on the Tmote Sky is the frequency of incoming packets, which could be proceeded from the receiver nodes. An important factor concerning this value is the time constraint of writing data to the external flash. The datasheet of the flash device specifies a page (up to 256 bytes) program time of 0.64ms, typically. In addition, the time for the arbitration of the SPI bus needs to be considered. Practical measurements on different nodes showed that the maximum packet frequency, which could be handled is at about 1000 packets per second. This frequency is high enough to avoid restrictions due to memory access. Furthermore, we investigate the speed of reading data in the same way. It turned out that reading one block takes between 5 and 10ms. Again, this is fast enough if we consider the maximum load of the DSN of 10 packets per second.

For the Channel Hopping scenario, the question of how fast the radio can switch the channels came up. Elaborate tests indicate that this value is highly fluctuating as shown in Table 4-4. We fixed the minimum time required for one channel to 25ms, which means that if we have 4 different channels, the packet frequency has to be greater or equal to 100ms.

Another timing factor is the stability of the sending frequency. On the one hand,

	40	41	42	43	44
40		99.7	99.8	99.7	99.8
41	99.8		99.6	99.1	99.6
42	99.8	99.6		99.5	99.5
43	99.8	98.6	99.5		99.3
44	99.9	99.6	99.7	99.5	

Table 4-5: This table shows the packet reception rate for each link of the first validation test. Each row stands for a test, in which the node with the id specified in the row header has been the sender. The column header specifies the id of the receiving nodes.

	40	41	42	43	44
40		98.9	99.7	99.7	99.4
41	99.6		99.9	98.8	99.9
42	99.8	99.9		99.1	99.7
43	99.8	99.1	99.9		99.0
44	99.9	99.9	99.6	99.3	

Table 4-6: This table shows the packet reception rate for each link of the validation test using the low power mode. Each row stands for a test, in which the node with the id specified in the row header has been the sender. The column header specifies the id of the receiving nodes.

a receiver-node must switch on the radio before the next packet arrives. On the other hand, the duration of the off-state of the radio should be as long as possible. Measurements result in a deviation of ± 10 ms of the mean sending frequency if the clear channel assessment (CCA) of the sending Tmote Sky is on. Without using CCA, the maximum deviation is less than 1ms. The radio itself takes another 10ms to switch off and on [11]. To guarantee the reception of a potential packet, the duration of the effective low power state is set to 40ms less than the packet frequency.

After having tested all the individual components, we performed several entire link tests with 5 nodes, placed in a circle with approximately 30cm diameter on a table. For all these tests, we chose a transmission power of 31 (maximum), a frequency for sending packets of 5 packets per second (200ms per packet), and a packet length of 12 bytes. The channel is set to 27. In these test series, the Link Test Controller and the DSN came into operation for the first time. We performed three different series; First, a series over 10000 packets, secondly, a series over 1000 packets, using the low power mode, and thirdly, an echo test over 1000 packets with nodes 40 and 41. The Tables 4-5, 4-5, and 4-5 show the resulting packet reception rate. Most of the nodes received more than 99.5% of the packets. There are only two links with a PRR of less than 99.0%.

	40	41
40	99.0	99.3
41	99.6	99.1

Table 4-7: This table shows the packet reception rate for each link of the echo validation test. Each row stands for a test, in which the node with the id specified in the row header has been the sender. The column header specifies the id of the receiving nodes.

5

Measurement & Evaluation

In this Chapter, we provide detailed descriptions and results of our link quality tests. First, we picture the setup of our indoor measurements. Then, we briefly describe the test series we executed, and finally, we give an extensive evaluation of the measured data.

5.1 Measurement Setup

Our measurements took place in an office building of the Electrical Engineering Department, ETH Zurich. We placed 18 Tmote Skys heterogeneously distributed on the G-Floor (see Figure 5-1 and Table 5-1). Except of nodes 21/22, nodes 11/20, and nodes 42/43, all nodes were placed in separate rooms. They were positioned on solid sub surfaces and were adjusted parallel to each other. A mark on the base allows to exactly reassign the position of each node. Node 2, 41, 42 and 44 are put on a chest (1.9m heigh); the others are stationed on desks (0.80m heigh) or on cabinets (1.15m heigh).

This setup has been chosen to obtain measurements from a realistic environment for wireless sensor networks. Most of the previous measurements [22, 21, 20, 2, 14] proceeded in artificial line-ups such as straight lines, grids, or circles, which have to be considered when we compare the results of our tests with others.

We performed link measurements during the night, as well as during the day. At night, all doors are closed and no human activity takes place. During the day, people are moving around, doors open and close, and elevators, located near to node 5 and node 42, may operate.

The basic setup we were using for most of our tests includes the 10 nodes with an id, which is less than 25. In a second step, we expanded our test bed with 8 additional nodes to measure different links with other distances.

Node Id	Location [m]		Description
22	2.1	1.3	room 78.2, desk
21	5.5	5.2	room 78.2, cabinet
7	7.4	7.2	room 78.1, cabinet
9	7.6	11.9	room 77, cabinet
20	1.1	18.2	room 75, desk
10	7.1	13.5	room 76, cabinet
11	6.1	17.2	room 75, cabinet
5	7.9	23.4	corridor, windowsill
13	12.9	13.6	broom closet, top of an unused fridge
2	13.5	15.4	room 71.2, chest
Extension of the basic test bed			
31	22.0	17.3	room 71.1, desk
30	28.3	16.2	room 69, desk
32	37	16.8	room 67, desk
2	18.5	19.3	room 71.2, has been moved to a desk
40	42.9	12.7	room 66, windowsill
41	49.3	21.1	room 64.3, chest
42	57.2	20.9	room 64.1, windowsill
43	57.9	12.8	room 64.1, chest
44	62.5	27.2	corridor, chest

Table 5-1: Setup of the Tmote Skys for our indoor measurements. The locations are based on their grid of the floor plan.

Parameter	Value
Transmission Power	19
Channel	26
Packet Frequency	200
Packet Length	12

Table 5-2: This table shows the values for the parameters we are using in our measurements.

5.2 Measurements

This section provides an overview of the measurements we performed. First, we define the parameters we are using for our tests, and then, we briefly describe the test series we run. Detailed specifications about all tests can be found in Appendix B, where each test with all its parameters is listed.

5.2.1 Parameters

Before we started our measurements, we assigned some of the parameters of the LET Platform (refer to Table 4-2) to constant values, as shown in Table 5-2. We set the length of the packet to 12 bytes for all our tests, we settled the sending frequency to 5 packets per second (200 ms for each packet), and we decided to use the channel 26. According to Srinivasan et al. [16], this channel has the least interference with the Wireless LAN (802.11b).



Figure 5-1

This is the floor plan of the G-Floor of the ETZ building, ETH Zurich, which shows the node placement of our indoor test bed.

It is of interest to have an adequate transmission power for our environment. This means that we would like to measure links of all qualities. If we are using a low transmission power, many links will hardly receive any packets, and if we are using a high transmission power, most of the links will receive all packets. Therefore, we performed short tests with 10 nodes, 1000 packets, and four different transmission powers: 15, 20, 25 and 31, respectively. Figure 5-2 shows the results of these tests, by means of histograms with the number of links for a given packet reception rate. We determined that, for high transmission power, the bars slightly accumulate to the right side; to high PRR. On the one hand, for our evaluation, links with PRR values between 0.7 and 1.0 are the most interesting ones to estimate. On the other hand, we wanted to measure as many links as possible, which means that the number of links with PRR equal 0 should be as small as possible. Therefore, we defined a transmission power of 19 for our tests. For the CC2420 radio chip, a transmission power of 19 corresponds to -5dBm output power.

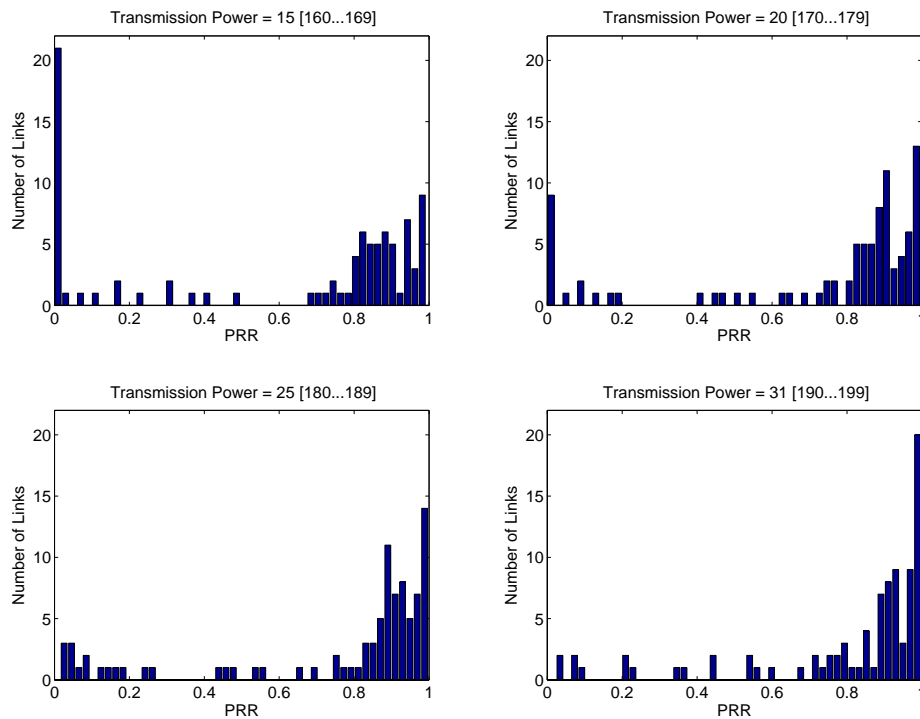


Figure 5-2

These four histograms show the number of links with a given packet reception rate (PRR) of a test series. Each histogram is generated out of 90 links (10 tests with 9 receiving nodes). The numbers in squared brackets indicate the test ids the diagrams are based on.

5.2.2 Test Series

For our measurements, we defined different test series. A test series consists of several tests with the same parameters and the same participating nodes. In each test of the series, one node is the sender and the others are the receivers. Typically,

Number of Tests		Number of Packets
Total	318	2'262'000
Validation & Testing	37	82'000
Tests to evaluate TX power	120	120'000
Measurements (total)	161	2'060'000
One Sender	81	1'260'000
Echo	80	800'000

Table 5-3: This table shows statistics of the number of tests we performed and the amount of data we gathered in all our measurements.

all involved nodes are dedicated to be the sender once in each series in a round robin like manner.

We run the first test series at daytime with 10 nodes over 100'000 packets. With the given packet frequency of 200 ms, each test had a duration of nearly 6 hours. Therefore, we run only 5 tests with pre-elected nodes: node 2, 7, 10, 20, and 22, respectively. The selection of these nodes is based on the evaluations of the test series we run to determine the transmission power.

After these long-term tests, we run shorter test series with 10'000 packets during the night. We repeated this night-time measurements one week and one month later and performed similar ones during the day. Subsequently, we did a test series using low power mode for the sender and the receivers with a total of 10'000 packets.

In between these tests, we ran various measurements with the Echo Scenario. These series consist of only two tests, in which both participating nodes are once the sender and once the receiver.

In addition, we extended our setting and carried out further tests, including additional echo tests.

Table 5-3 gives a summary about the number of tests we performed and the amount of data we gathered in all our measurements.

5.3 Evaluation

This section shows the evaluations we performed with the measured data from the link tests. Most of the analysis is based on link evaluations performed and described in other publications. The main difference is that we were using data out of measurements in a realistic environment. Therefore, we determined remarkable differences in some of the evaluations.

The analysis presented in the Section 5.3.1, 5.3.2, and 5.3.3 concerns a single link, while the analysis in Section 5.3.4 takes all links of the same test into account. The evaluations described in the Section 5.3.5 is a global evaluation, which considers all the links of all the tests. At the end of this section, we analyze the symmetry of the links.

5.3.1 Temporal Link Behavior

To obtain an overview of the measured data, we analyzed the temporal behavior of the packet reception rate (PRR) of a link during a test. Since our traces consist only of '1' (received packet) and '0' (not received packet), one packet cannot indicate a packet reception rate. Therefore, we computed a temporal PRR by aggregating a certain number of packets. With a sliding window like technique, we resulted in a graph, which shows the temporal behavior of the PRR. The number of packets we were using for the aggregation has an impact of the smoothness of the graph as shown in Figure 5-3.

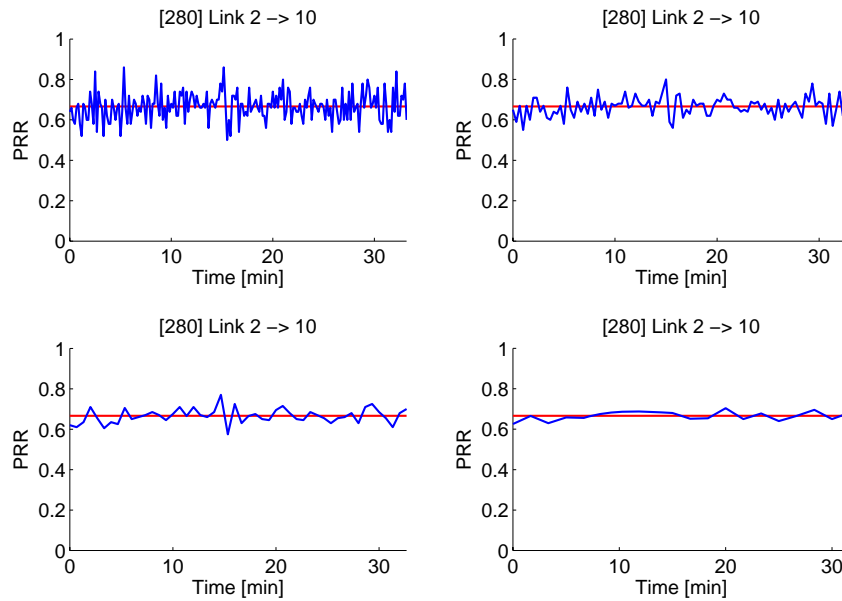


Figure 5-3

These figures show the temporal behavior of the same link four times, but with varying numbers of aggregated packets. The more packets are aggregated, the smoother the graph looks. In this example, we used 50, 100, 200, and 500 packets to calculate an aggregated packet reception rate.

Figure 5-4 shows some resulting plots of the temporal behavior analysis, using an aggregations of 50 packets. We determined that most of our links are stable but established also some very unstable links as shown in Figure 5-5. There are links fluctuating extremely over the full period of the measurement, links having local discontinuities, links following a clear trend, and links changing stepwise. In our measurements, about 10% of the links are very unstable (refer to Section 6.1.1). Such links are not deterministic and quite hard to estimate. Consider the link with the step function of Figure 5-5. If we examine the first part of the link, it seems to be impossible to predict the second part with a high PRR. The existence of such links is a first result of this thesis.

Moreover, we analyzed the temporal behavior of the links over weeks and months. Therefore, we performed similar test series at intervals of a week and a month,

respectively. About 68% of all links have a variance of the PRR of these three measurements of less than 0.01; 76% of all links less than 0.04 (refer to Figure 5-6). However, almost one fourth of all links have a variance of 0.04 or greater. In other words, these links vary more than 20% referring to the PRR. In Figure 5-7, some examples of links with great variances are given. Some links have a trend, but others do not, and some links have an outlier in just one of the three tests. The causes for this behavior have not been determined yet. The instability discussed in the previous paragraph might have an effect on this result, but there exist also links having stable behavior in all the three tests, but with extremely different PRR.

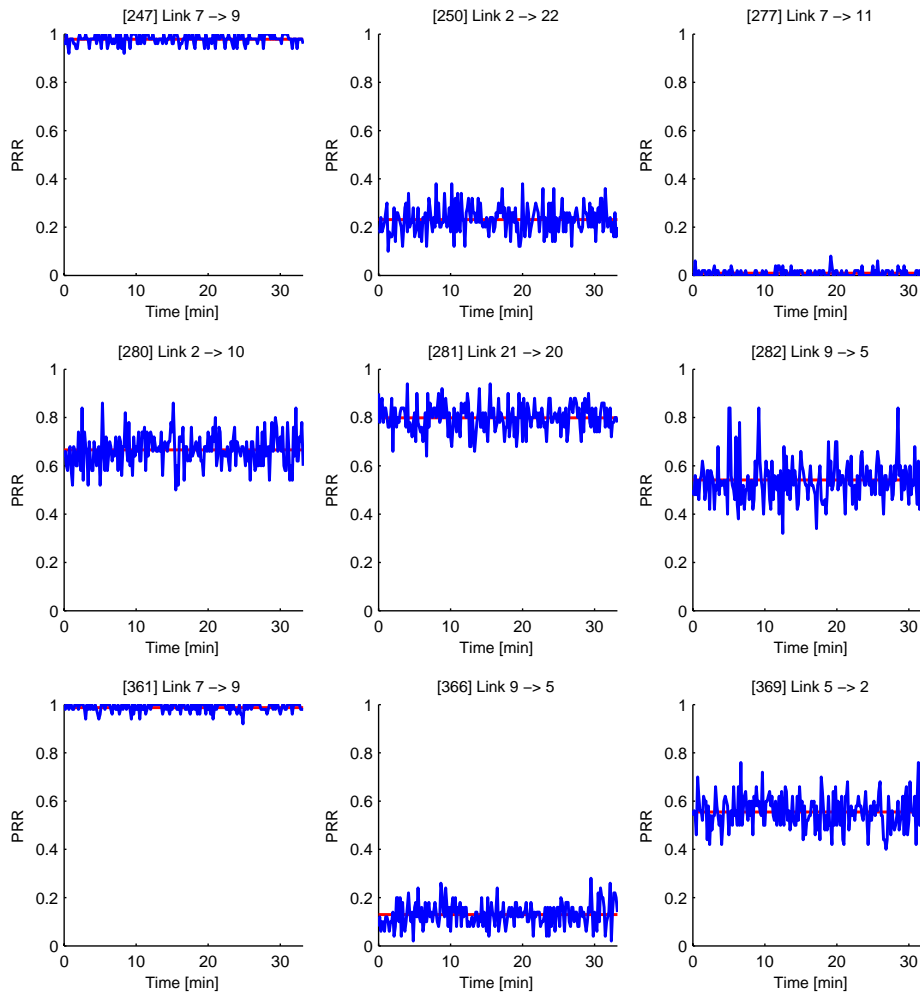


Figure 5-4

These figures show the temporal behavior of example links. The figures plot the time versus the aggregates PRR. The red line indicates the PRR over the whole link. The number of aggregated packets is set to 50. On the top of each figure, the number in squared brackets indicates the test id, followed by numbers, defining the sender and the receiver, respectively.

The impact of this links on the performance of a routing protocol has to be considered. Therefore, we propose to do an over-provisioning in terms of number of links, which means that if a link becomes very badly, it can be discarded from the neighborhood table.

5.3.2 Single Link Autocorrelation

This section discusses the occurrence of bursts of subsequent packet failures and the length of these bursts. It is of interest for routing protocols to know if a link fails just for one packet or for a longer period of time. Especially the retransmission of packets can be designed more efficient if the behavior of packet failures is known. Willig and Mitschke [20] analyzed this characteristic by accumulating the number of bursts at a given burst length. Their measurements are based on a setup with 10 receivers, arranged in an array on a plank. There is a single sender in the middle of the plank, broadcasting 9000 packets in each test. The distance between two sensor nodes is 30cm, which results in an extent of 3m for the whole setup and a maximum distance between the sender and a receiver of 1.5m. In this environment, they found that single-packet bursts make up 72% of all bursts, and that 90% of all bursts have a length of three packets or less.

It is surprising that we measured similar results in our office-like environment with an extent of more than 30m. An evaluation based on all our measurements shows

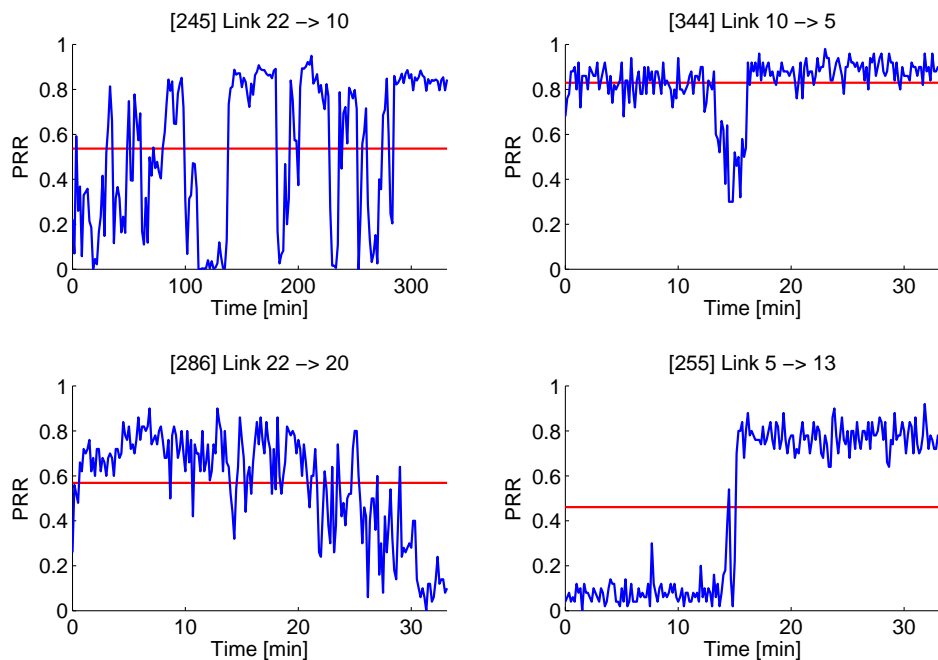


Figure 5-5

These figures show the temporal behavior of very unstable links. They are fluctuating over the full measurement, having local discontinuities, following a clear trend, or changing stepwise.

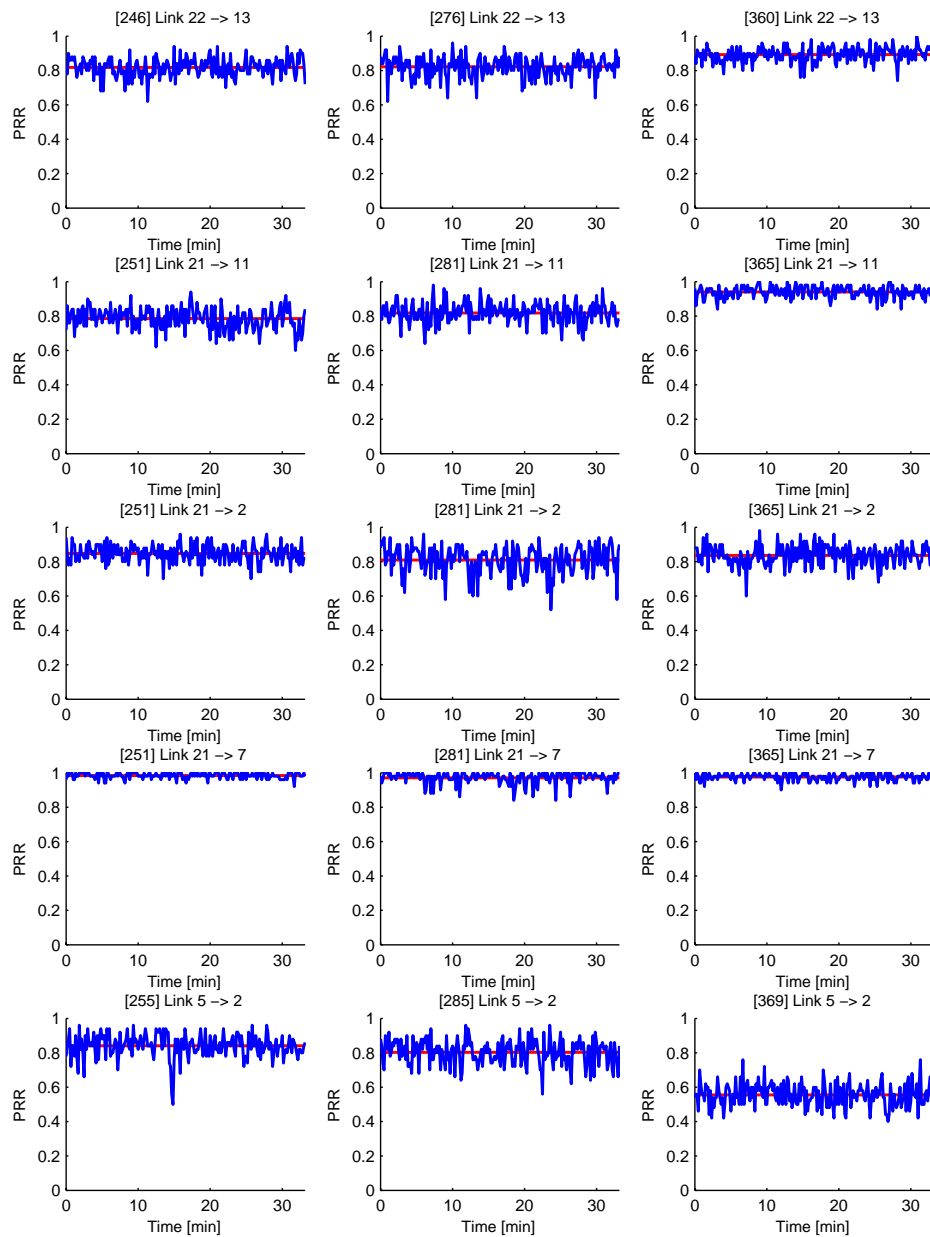


Figure 5-6

These figures show the temporal behavior of example links in terms of a week and a month. In each row, there are three measurements of the same link; the time lag between the first and the second measurement is one week; between the second and the last measurement one month.

that 76% of all bursts are single-packet bursts, and in 94% of all bursts, three or less packets get lost as shown in Table 5-4 and Figure 5-8(a). The average length of a burst is 2.82 packets, and the standard deviation 74 packets. Compared to an average of 1.77 packets and a variation coefficient of 1.65 in the measurements of Willig and Mitschke, our results are clearly different. The high standard deviation may be founded in the fact that we have links with little reception and on which most of the packets get lost. Burst sizes of several thousands packets are not uncommon. Therefore, we performed the same analysis, limited to the links with a $PRR \geq 80\%$. In this evaluation, single-packet burst make up 86% of all bursts and 99% of the burst have a length less or equal three (refer to Table 5-4 and Figure 5-9(a)). The average burst length scaled down to 1.2, and the standard deviation to 0.92 packets.

An interesting question is, if this result is remarkable or just the implication of a normal distribution. For this analysis, we performed a simulation. For each measured link in our tests, we generated a normal distributed binary vector with the same length and the same packet reception rate as the measured data. Afterwards, we evaluated this simulated data with exactly the same methods as for the original data. Again, we distinguished between all links and links having a $PRR \geq 80\%$. The result is listed in Table 5-5 and visualized in Figures 5-8(b) and 5-9(b). The average burst size of 2.3 and 1.1, respectively, matches reasonably well with the measured data. The standard deviations of 27 and 0.4 differ clearly from our measurements and have an obvious impact on the shape of the graphs. The figures based on the simulated data descend faster than the ones based on the measured data.

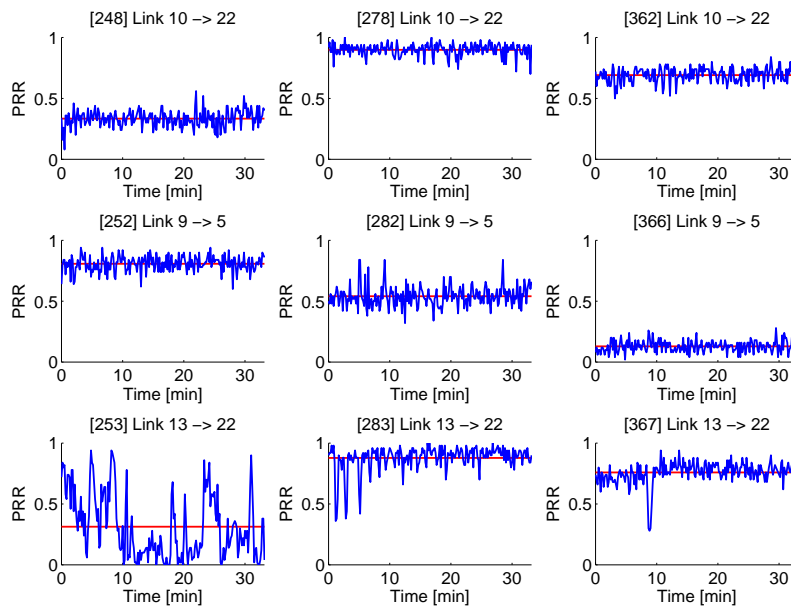
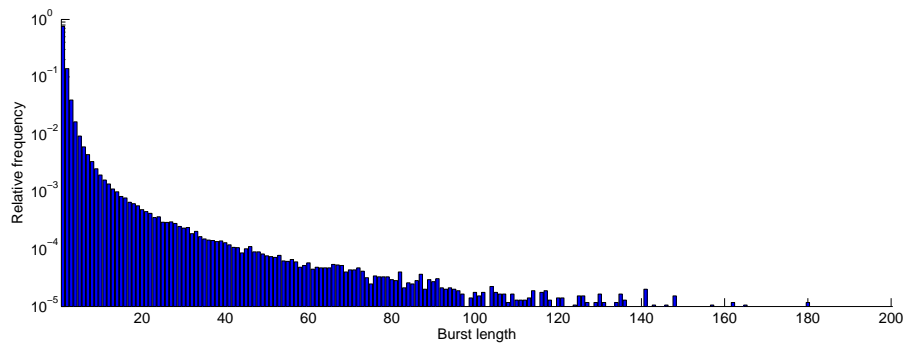


Figure 5-7

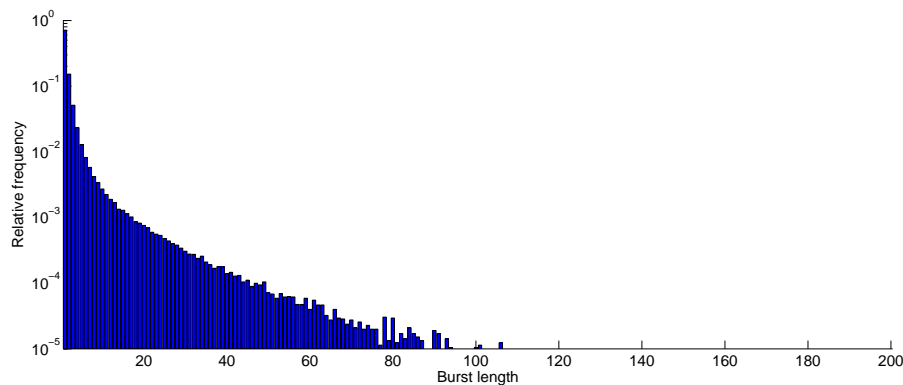
These figures show the temporal behavior of unstable links in terms of a week and a month. The first and the second measurement have a time lag of one week; the second and the third of one month.

Burst Length	All Links		Links with PRR \geq 80%	
	Relative Frequency (RF)	Accumulated RF	Relative Frequency (RF)	Accumulated RF
1	75.89%	75.89%	86.07%	86.07%
2	13.90%	89.79%	11.01%	97.08%
3	3.95%	93.75%	1.90%	98.98%
4	1.65%	95.39%	0.45%	99.43%
5	0.93%	96.32%	0.19%	99.62%
6	0.60%	96.93%	0.10%	99.72%
7	0.44%	97.37%	0.10%	99.82%
8	0.33%	97.70%	0.05%	99.87%
9	0.25%	97.95%	0.03%	99.90%
10	0.20%	98.15%	0.02%	99.91%

Table 5-4: This table shows the relative frequency of the length of packet failure bursts. Columns two and three show the measured results of all our tests, while columns four and five are based on links having PRR \geq 80%.



(a) Measured data



(b) Simulated data

Figure 5-8

These histograms show the relative frequency of the length of packet failure bursts. The axis for the relative frequency is in logarithmic scale.

Burst Length	All PRRs		PRR \geq 80%	
	Relative Frequency (RF)	Accumulated RF	Relative Frequency (RF)	Accumulated RF
1	71.25%	71.25%	88.41%	88.41%
2	15.25%	86.50%	10.02%	98.43%
3	5.12%	91.62%	1.34%	99.77%
4	2.32%	93.93%	0.20%	99.97%
5	1.30%	95.23%	0.028%	99.99%
6	0.83%	96.06%	0.006%	99.999%
7	0.58%	96.64%	0.001%	100.000%
8	0.43%	97.07%	0.000%	100.000%
9	0.34%	97.41%	-	-
10	0.27%	97.68%	-	-

Table 5-5: This table shows the relative frequency of the length of packet failure bursts out of a simulation in which the packet losses are normally distributed over the full dataset length.

Thus, it appears that the packet losses in a real environment are not normally distributed. This might be an indication that packets losses tend to occur in bursts, but out of these evaluations, it is still not clear to what extent.

The Required Number of Packets (RNP) defined by Cerpa et al. [2] is another metrics to evaluate the single link auto correlation. This value indicates the number of necessary retransmissions until a successful transmission of the packet happens. Table 5-6 shows an example of the calculation of this value.

We analyzed the correlation between the PRR and the RNP and achieved similar results as in the previous paragraph. The measurements shown in Figure 5-10 clearly

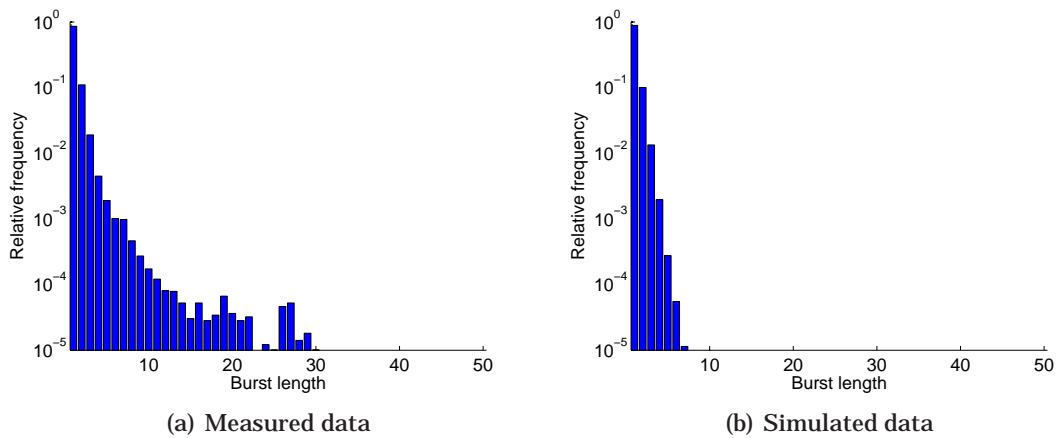


Figure 5-9

These figures show the relative frequency of the length of packet failure bursts, based on the data of the links with PRR \geq 80%. The axis for the relative frequency is in logarithmic scale

Trace	RNP
1	1
0	2
1	1
0	3
0	2
1	1
1	1
0	5
0	4
0	3
0	2
1	1
Average	2.17

Table 5-6: This table shows an example of the calculation of the RNP value. The values in the first column represent if a packet has been received (1) or if it got lost (0). The numbers in the second column are the required retransmissions until a success for each position in the trace.

differ from the result of an experiment with normal distributed packet losses. The conclusion that can be drawn out of this finding is undetermined and further investigation needs to be done. However, many links seems to be uniformly. Maybe special events or interrupts will impact the result.

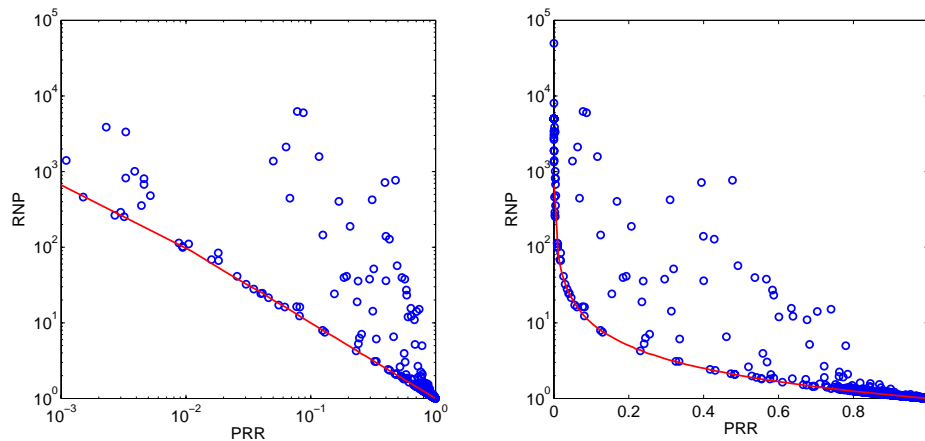


Figure 5-10

This plots show the correlation between the Required Number of Packets (RNP) and the PRR in our measurements. In the left graph, both axes are scaled logarithmic, in the right one, only the y-axis. The red line is the result of a normal distributed random experiment.

5.3.3 RSSI, LQI, and PRR Correlation

In this section, we discuss the correlation between the packet reception rate and the link-quality parameters measured for each received packet.

First, we compared the mean RSSI with the PRR of a link. Srinivasan et al. [16] performed a similar evaluation with the same radio chip (CC2420) as we are using. For short packet bursts, they observed a strong correlation between RSSI and PRR with a sharp cliff around -87dBm: most of the links stronger than -87dBm have a $PRR \geq 99\%$. In a long-term measurement, they could achieve the same relation, but with more outliers. Our measurements show slightly different results as pictured in Figure 5-11. The scatter in our plot is much greater than in the plot by Srinivasan et al. This implicates that we do not have the sharp cliff around -87dBm. At signal strength of -85dBm, we still have many links with $PRR \leq 90\%$. To guarantee a $PRR \geq 95\%$, we would require a mean RSSI value of almost -60dBm.

However, the standard deviations of our RSSI values are comparable to the ones of Srinivasan et al. They stated that most of their links have a standard deviation of less than 1dBm; we measured a mean value of 0.98dBm.

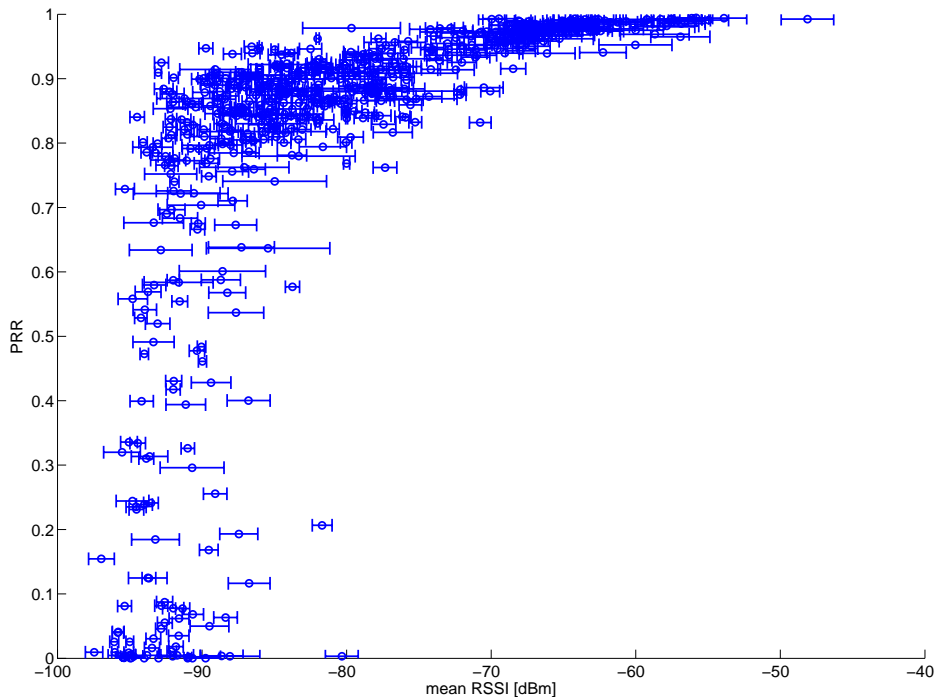


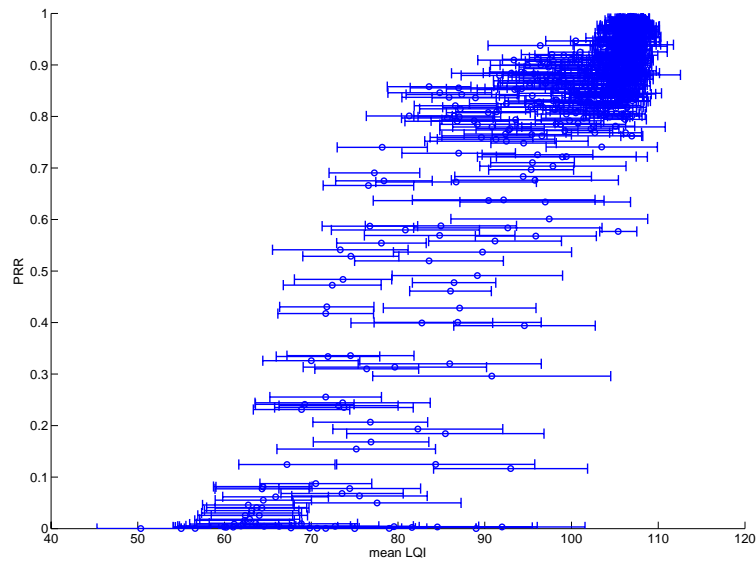
Figure 5-11

This plot shows the correlation between the mean Received Signal Strength (RSSI) of a link in a test and the packet reception rate. The bars indicate the standard deviation for each link.

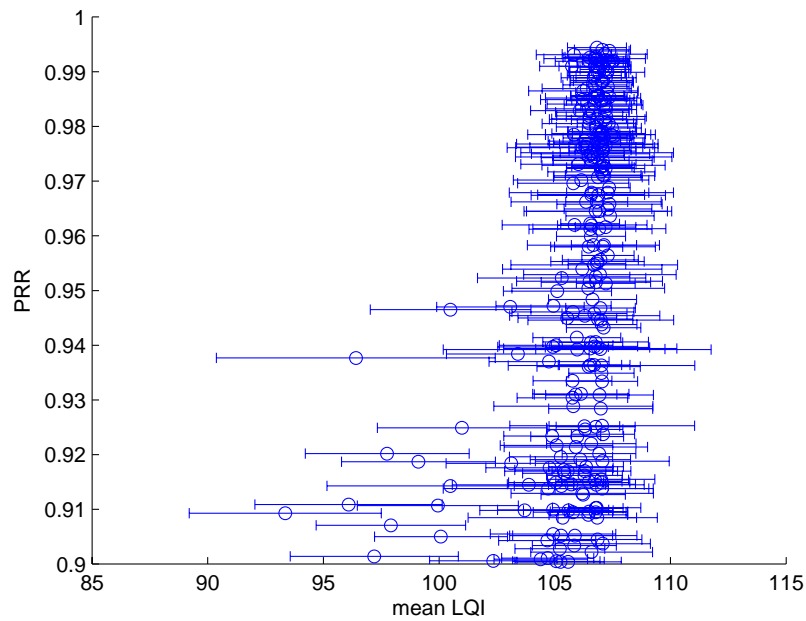
Furthermore, we analyzed the correlation between the link quality indicator (LQI) value and the PRR (see Figure 5-12(a)). Remarkable are the large standard deviations that make the LQI value unattractive for link estimation at first glance. However, there is an accumulation of data around the LQI value 107 with $PRR \geq 80\%$ as presented in Figure 5-12(b). This cluster can be used to detect high quality links.

Again, Srinivasan et al. realized a similar evaluation and their results match ours

reasonably well.



(a) Evaluation of all links



(b) Evaluation based on the links with $PRR \geq 90\%$

Figure 5-12

These figures picture the correlation between the Link Quality Indicator (LQI) value and the packet reception rate. Each dot represents one link in one test. The bars show the standard deviation of the measurements.

5.3.4 Covariance of Same Source Links

We also studied properties of links with the same source. An interesting question is if packet losses on these links are correlated. This means: If a packet has not been received at one receiver, what is the probability that the same packet does not receive at another receiver as well? To analyze this relation, we computed the correlation coefficients of the measurements for paired links

$$cor(X, Y) = \frac{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (5.1)$$

For the calculation we used a binary vector X for the first link and a binary vector Y for the second link. The elements of these vectors represent if a packet has been received ($x_i = 1, y_i = 1$) or it got lost ($x_i = 0, y_i = 0$). \bar{x} and \bar{y} are the expectations of vector X and Y or in other words the PRR of the link.

The results of different tests look quite similar. Therefore, only two examples are shown in Figure 5-13(a). The correlation coefficients seem to be negligible. However, if we compute the correlation coefficients of normal distributed data, the values are more or less zero as shown in Figure 5-13(b). Thus, it is not evident what to conclude out of our measurements. There seems to be a small correlation between packet losses of different receivers with the same sender.

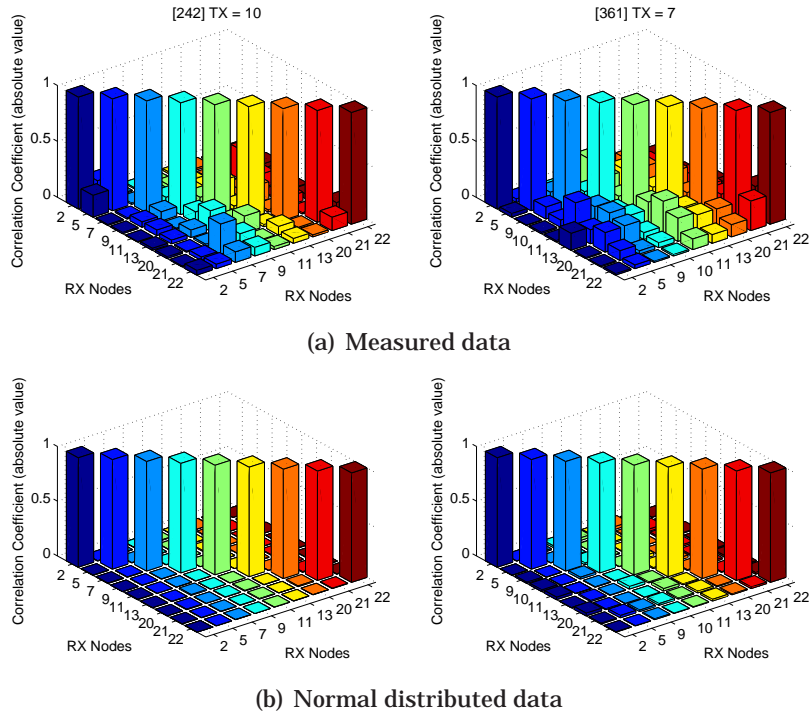


Figure 5-13
 This figures show the correlation coefficients defined in Equation 5.1 of same source links. The graphs are symmetric with respect to the diagonal. The values in the diagonal are 1 since the correlation coefficient of a link with itself equals 1.

Cerpa et al. [2] investigated this relation in a grid lineup. They calculated the conditional probability of successfully receiving a packet if a node in another link also received it ($1 \rightarrow 1$), and the conditional probability of successfully receiving a packet if a node in another link did not receive it ($0 \rightarrow 1$). They distinguished between good, medium, and bad link quality. If a packet receives over a good link, they conclude that most likely all the other links will receive the same packet almost directly proportionally to their PRR. If a packet gets lost on a good link, Cerpa et al. showed that with a high probability, none of the other links will receive that packet either, even not other high quality links.

We performed the same evaluations with our measured data and got pretty different results (refer to Figure 5-14). The relative strong correlation in the measurements of Cerpa et al. for the probability of successfully receiving a packet if a node in another link also received it ($1 \rightarrow 1$) can be confirmed with our measurements. However, the figures for the probability of successfully receiving a packet if a node in another link did not receive it ($0 \rightarrow 1$) look completely different. Our results show greater linear correlations for the transition $0 \rightarrow 1$ than Cerpa et al. measured.

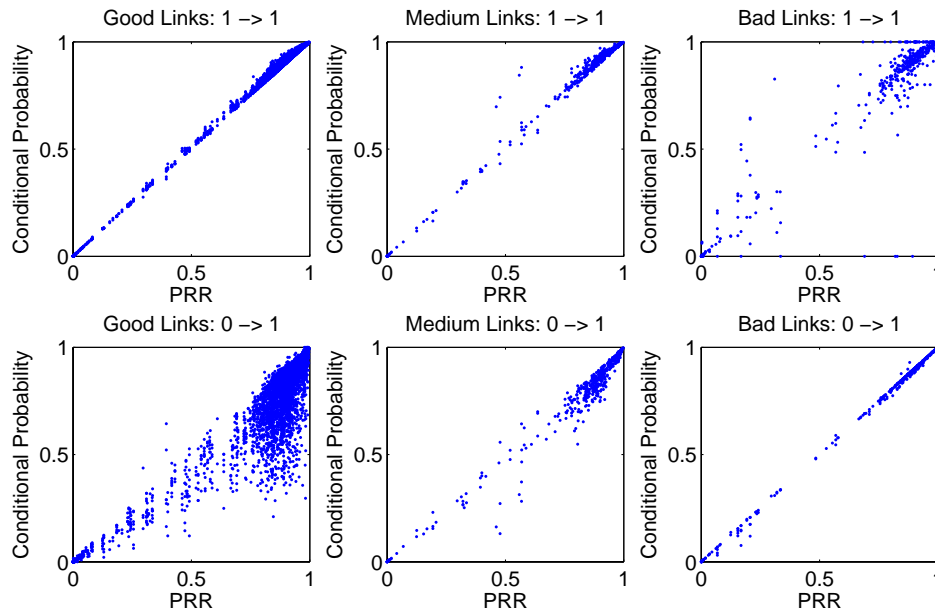


Figure 5-14

The figures in the first row show the conditional probability of receiving a packet at link X if the same packet has been received over another link Y ($1 \rightarrow 1$) against the PRR of link X. The analysis is divided in good, medium, and bad link qualities for link Y. The figures in the second row show the conditional probability of receiving a packet if the same packet get lost on another link ($0 \rightarrow 1$).

Similar to the evaluations in Section 5.3.2, we performed an analysis with normal distributed data. For each link, we generated a normally distributed random vector with the same length and PRR as the original link, and determined the probabilities described in the previous paragraph. The results show more or less linear dependencies as pictured in Figure 5-15.

On closer inspection, differences between the normal distributed data and the measurements can be detected. For the probability $1 \rightarrow 1$, our data are in the average slightly above the diagonal located. This means that if another link receives a packet, the probability of receiving this packet increases. For the probability $0 \rightarrow 1$, the same effect occurs with a negative sign. This means that if a packet get lost over a link, the probability to receive this packet decreases. This tends to the conclusion that there is a correlation between packet failures of different links with the same source. However, our results are not that evident than the measurements of Cerpa et al.

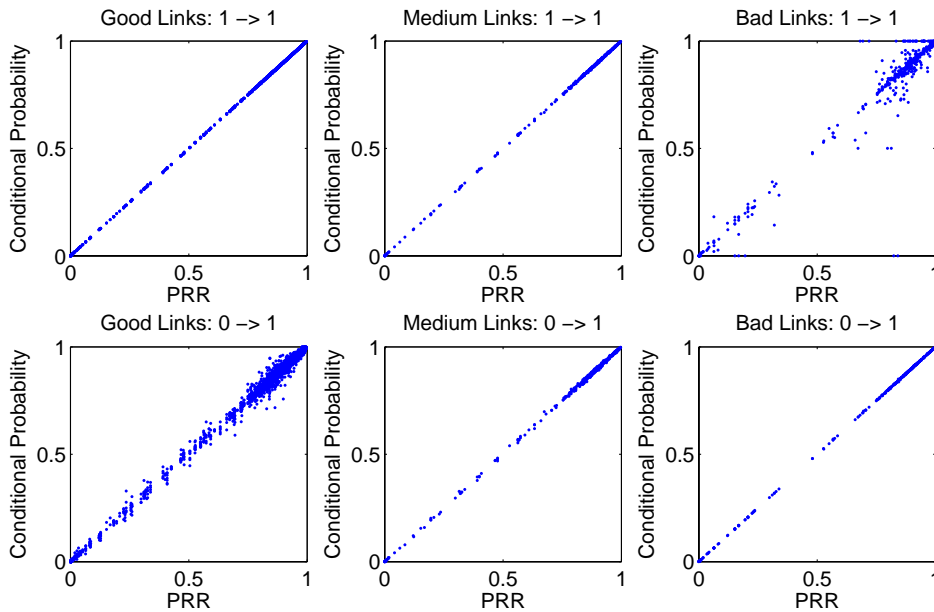


Figure 5-15

These graphs show the same correlation as Figure 5-14 but based on normal distributed data. Remarkable are the near linear dependencies.

5.3.5 Spatial Characteristics of Wireless Links

In this section, we describe the spatial characteristics of the packet reception rate. We analyzed how the distance between the sender and the receiver correlates with the PRR. The result is plotted in Figure 5-16 and based on all measurements we performed, i.e. 171 potential links with a distance from 1.7m to 65m.

In Figure 5-16, three regions can be recognized: in the first one (up to 5m), the packet reception rates are uniformly high; in the second one (from 5m to 28m), the PRRs vary highly; and in the last one (more than 28m), no packets can be received.

This classification has also been seen from Zhao and Govindan [22]. Their evaluations focused on the extent of the second region, they referred to as 'gray area'. In their indoor measurement, the 'gray area' covers 50% of the whole communication range (refer to Chapter 2). In our environment, the 'gray area' has an extent of 23m, which accounts for more than 80% of the full transmission range. If we look at the

number of links, we see that 78.8% of them belong to the 'gray area' and only 8.1% to the area close to the sender. Furthermore, the 'gray area' of our measurements starts at a distance of 5m, compared to 10m in their analysis. These significant differences may be founded in the node placement; most of our nodes are placed in separate rooms, while Zhao and Govindan used a linear arrangement in a corridor. Out of this analysis, we conclude that there is hardly any correlation between distance and PRR in our environment, especially in the 'gray area'.

An interesting question is, if the links in the 'gray area' are more unstable than the links in the area close to the sender. The question might be answered with yes, because all of the links we specified unstable (refer to Section 6.1.1) lie in the 'gray area'. Additionally, if we consider the temporal behavior of all links in the gray area, we measured an average standard deviation of 0.049 for an aggregation of 100 packets. This is 7.5 times more than the average deviation in the area close to the sender. The reason for this result might be the medium quality links, which have mostly a higher temporal variance. Therefore, we performed a further analysis with only good links ($PRR \geq 90\%$) of the 'gray area'. However, the standard deviations measured in the 'gray area' are still 4 times greater than the one in first area.

5.3.6 Link Asymmetry

In the first part of this section, we discuss the asymmetry of wireless links. Furthermore, we analyzed the performance of immediate packet acknowledgements.

The evaluations in this section are based on the test series we run with the Echo Scenario. Each of this series contains of two tests with two participating nodes. The sender in the first test is the receiver in the second one and vice versa. Figure 5-17 illustrate such a test series and visualizes the meaning of the different packet reception rates: PRR_1 , ARR_1 , PRR_2 , and ARR_2 .

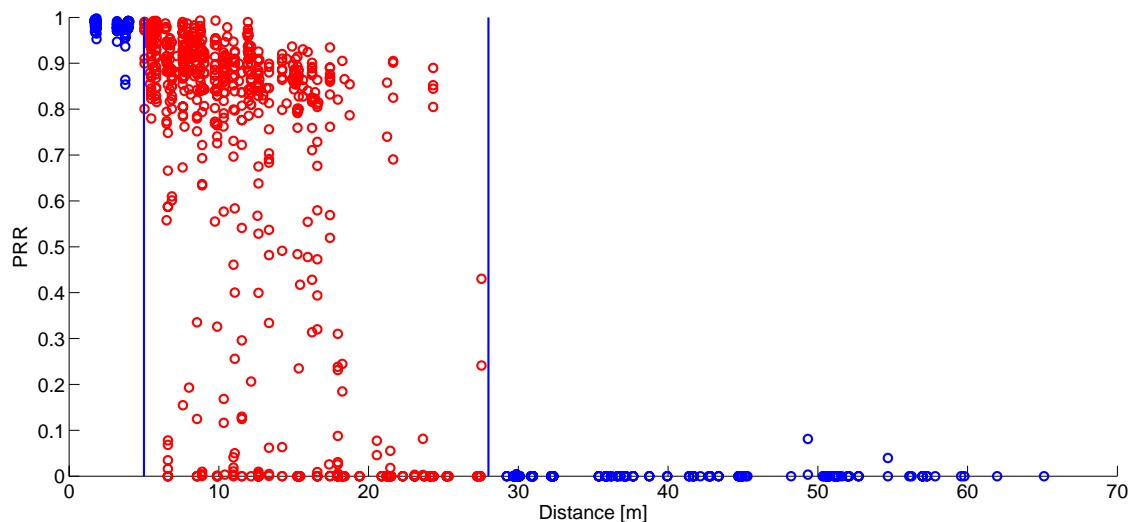


Figure 5-16

This figure shows the PRR in correlation with the distance between the sender and the receiver. The plot is based on all our measurements. The measurements in the 'gray area' are colored red.

To analyze the asymmetries of a wireless links, we plotted PRR_1 versus PRR_2 and vice versa. This resulted in a symmetric graph, shown in Figure 5-18. For the evaluation of all links, we resulted in a standard deviation of 0.19. That means that the PRR of a link differs 19% from the PRR of its reverse link. If we did this evaluation for a selection, in which one of the links has a $PRR \geq 0.8$, we measured a standard deviation of 0.04.

This evaluation confirms the occurrence of link asymmetries. We can conclude that for a link estimation algorithm, each node has to perform its own estimations for each of its neighboring links and can not rely on measurements from its neighbors. Therefore, it might happen that a link is chosen from the node at the one end, but not from the node at the other end of a link. Hence, this link is going to be used unidirectional.

Furthermore, we analyzed the relation between the packet reception rates (PRR) and the reception rates of the acknowledge packets (ARR). The assumption is, that the ARR is generally higher then the PRR. This would mean that if a packet receives at the receiver, the channel seems to have little interference and an immediate transmission of the acknowledge packet success with a high chance.

To quantify this statement, we plotted PRR_1 versus ARR_2 and PRR_2 versus ARR_1 in the same graph (see Figure 5-18). This means that we compared the reception rates of a directed link between normal packets and acknowledge packets. If we consider all links, the average improvement of the ARR compared to the PRR is 2.03%. However, there are some outliers and if we omit these by taking just the links with $PRR \geq 0.8$, we obtain an improvement of 0.19%, which is negligible.

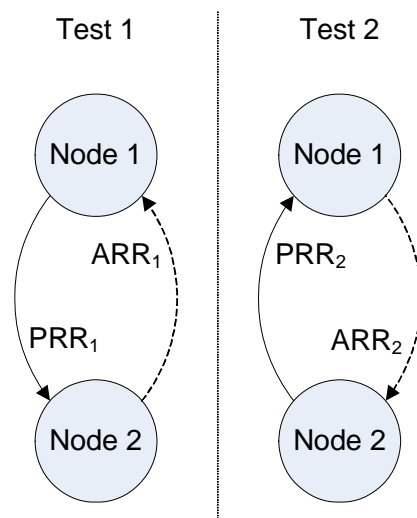


Figure 5-17

This figure illustrates a test series with the Echo Scenario. In the first test, node 1 is the main sender and in the second one, node 2 is the main sender. The receiver acknowledges each received packet, represented by the dashed arrows in the figure. In these tests, four packet reception rates can be measured: the Packet Reception Rate (PRR_1) at the receiver in the first test, the Acknowledgment packet Reception Rate (ARR_1) in the first test and analog for the second test (PRR_2 and ARR_2).

We approve our finding by performing a sign test. The null hypothesis says that the APP does not differ from the corresponding PRR. For our test, we have $N=74$ pairs of (PRR, APP) values. We count the observations in which the APP value is higher than the PRR and obtain a value $X^+=39$. For a two-tailed 5% interval of the binomial distribution with $N=74$ we determine the edges at $c_1=28$ and $c_2=46$. That means the interval $[0 \dots 28] \cup [46 \dots 74]$ covers 5% of the binomial distribution. Our result is not part of this interval as shown in Figure 5-20, therefore we can not discard the null hypothesis. This means that statistically seen, the ARR is not

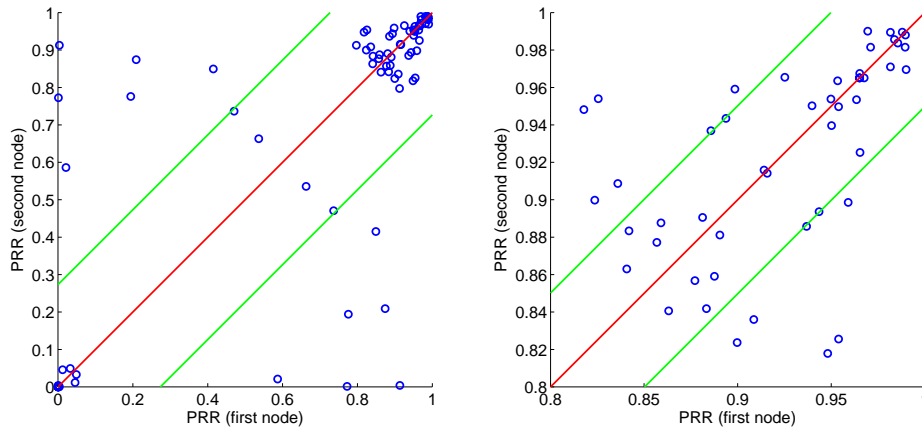


Figure 5-18

These figures show the link symmetries, we measured in our Echo tests for all links and the links with $PRR_1 \geq 0.8$ and $PRR_2 \geq 0.8$, respectively. The graphs are symmetric with respect to the diagonal, due to the fact that we plotted PRR_1 versus PRR_2 and PRR_2 versus PRR_1 . The green lines represent the standard deviation of the measurement.

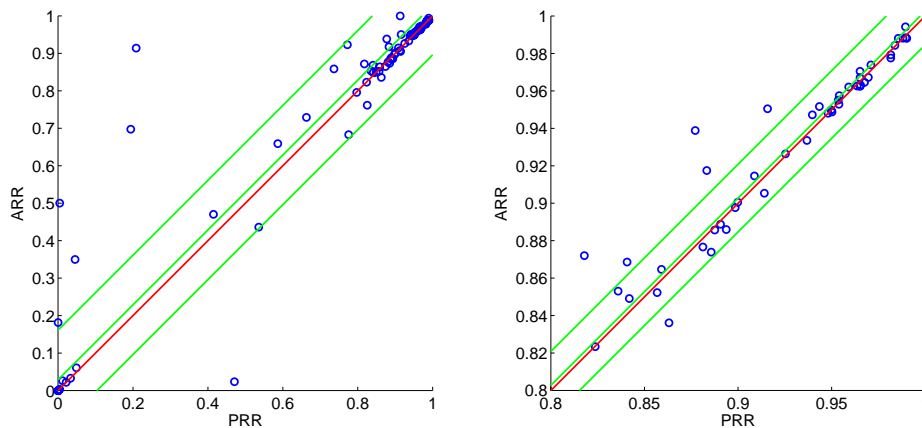


Figure 5-19

These figures show the Acknowledgments packet Reception Rates (APP) versus the PRR of the same link and the same direction. The left graph includes all links while in the right one only the links with PRRs greater than 80% are plotted. The green lines indicate the average and the standard deviations of the measurements.

significantly greater than the PRR.

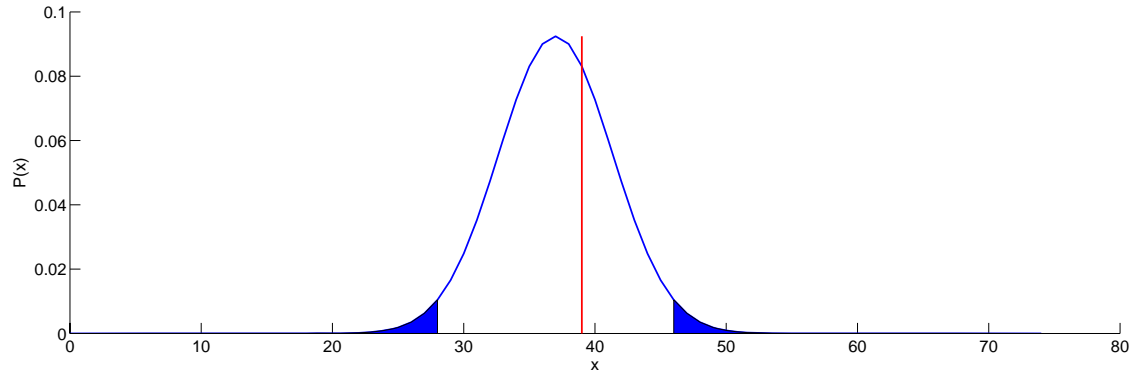


Figure 5-20

The figure shows the result of the sign test. The blue filled surfaces illustrate the two-tailed 5% interval of the binomial distribution with $N=74$. The red line is the number $X=39$, we calculated for the sign test. With this result, we cannot reject the null hypothesis.

6

Link Estimation

In this chapter, we investigate wireless link estimation. As mentioned in Section 1.1, we designed non-adaptive link estimation algorithms for wireless sensor networks. Since sensor nodes have very limited resources, the power consumption of our algorithm should be considered. With just a few packets, a reasonable estimation of a wireless link should be done.

In the first section, we specify the evaluation method we were using for the comparison of different algorithms. Furthermore, we discuss various algorithms that estimates wireless link qualities and performed detailed evaluations of these algorithms.

6.1 Evaluation Method

This section describes in detail the process of the evaluation of different estimation algorithms, which turned out to be a quiet challenging task. We designed an automated tool chain shown in Figure 6-1, which evaluates an algorithm based on our link measurements. In the following, a brief overview of this tool chain is given. More detailed information about the individual tools of the tool chain is given in Sections 6.1.1, 6.1.2, 6.1.3, and 6.1.4. In the last section, we show how to evaluate and compare different estimation algorithms with this tool chain.

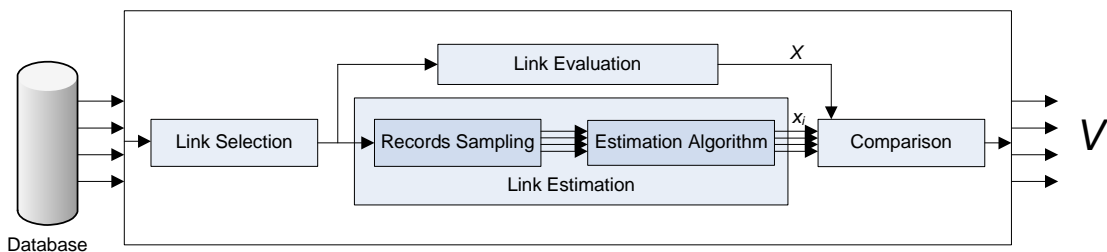


Figure 6-1
This figure shows the tool chain of the evaluation of the estimation algorithms.

On the left side, we start with the LET database, in which all data of our measurements are stored. For each of the 656 measured links, we evaluate the tool chain individually. First, the link selection tool filters the unstable links. This means that if a link is unstable, we cancel further evaluations with this link. The filtering of unstable links has been performed due to the fact that these links can hardly be estimated correctly and would have unpredictable effects on the evaluation of our estimation algorithms.

If a link passes the link selection, it is going to be evaluated as shown in the upper path of the tool chain. This means that we calculate the PRR X of the link.

On the bottom path, we estimate the PRR of the link with small samples of the data. By using a sliding window technique for the record sampling, the link estimation tool makes an estimation of the link quality for each record $(x_1 \dots x_n)$.

The last tool in the chain compares the estimations with the evaluation by calculating the variance. This results in a single value for each stable link.

Finally, we repeat this procedure with all links in our database and combine the resulting values of the tool chain to the variance vector V , which represents the evaluation of a given estimation algorithm.

6.1.1 Link Selection

This section describes the method we were using to select the unstable links. As shown in Section 5.3.1, we measured a couple of links with non-deterministic behavior.

However, we wanted to get rid of them since they might influence the results. The standard deviation of the temporal PRR (refer to Section 5.3.1) turned out to be a good indication to select and filter these unstable links and is discussed in details. First, we analyzed the standard deviation of all our links. We varied the numbers of packets m , which we are using for the aggregation to calculate the temporal PRR and achieved the results shown in Figure 6-2.

The data seems to have a lower bound with functional dependency to the PRR. Therefore, we also determined the standard deviations of the temporal PRR if a normal distribution of the packet losses is assumed. Other than in Chapter 5, we did not simulate this evaluation, but we calculated the expected distribution. Our traces, which consist of only '1's (packet received) and '0's (packet not received) can be described with a Bernoulli process $B(PRR)$. In our case, the probability for a success in a Bernoulli experiment corresponds to the PRR. The aggregation of m packets leads to a binomial distribution for the temporal PRR.

$$\frac{1}{m}Bin(PRR, m) \tag{6.1}$$

If we consider the characteristics of the binomial distribution and of the variance, we can evaluate the distribution for the standard deviation of the temporal PRR.

$$\begin{aligned} VAR\left(\frac{1}{m}Bin(PRR, m)\right) &= \frac{1}{m^2}[VAR(Bin(PRR, m))] \\ &= \frac{1}{m^2}[mPRR(1 - PRR)] \end{aligned}$$

$$= \frac{1}{m}PRR(1 - PRR)$$

The standard deviation σ is equal to the square root of the variance ($\sigma(X) = \sqrt{VAR(X)}$). This results in the formula

$$\sigma = \sqrt{\frac{1}{m}PRR(1 - PRR)} \quad (6.2)$$

and leads to the red line shown in Figure 6-2.

It seems to give a good classification of stable and unstable links if we introduce a cycle modification factor γ to the Equation 6.2. We defined the standard deviation of the temporal PRR of a link for the aggregation size m as σ_m . Thus, the criteria for stable links is given with

$$\sigma_m < \gamma \sqrt{\frac{1}{m}PRR(1 - PRR)} \quad (6.3)$$

We tested different m and γ values in an experimental way. That means that we classified the links for different values and examined the plots of the temporal PRR for the links at the edge of the two classes. With our feeling, we result in two pairs of m and γ values that give a reasonable division between stable and unstable links.

$$\begin{aligned} m_1 &= 100 \quad , \quad \gamma_1 = 3.0 \\ m_2 &= 500 \quad , \quad \gamma_2 = 4.8 \end{aligned}$$

This leads to the final conditions, which each stable link needs to satisfy.

$$\begin{aligned} \sigma_{100} &< 3.0 \sqrt{\frac{1}{100}PRR(1 - PRR)} \\ \sigma_{500} &< 4.8 \sqrt{\frac{1}{500}PRR(1 - PRR)} \end{aligned} \quad (6.4)$$

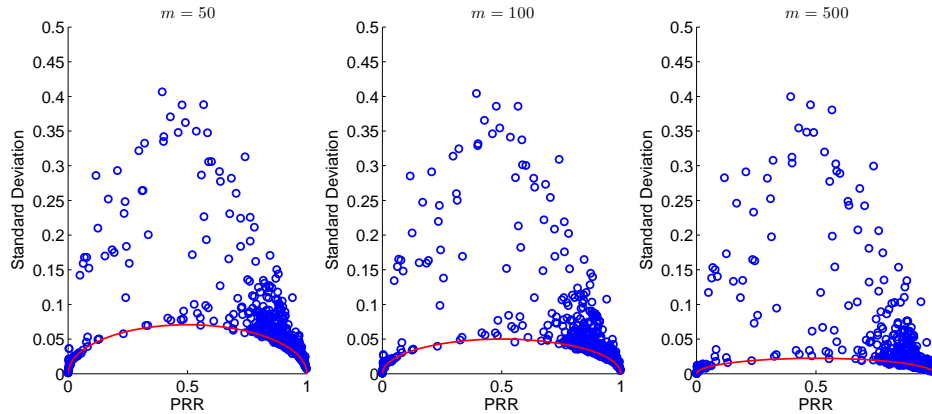


Figure 6-2

This figure shows the standard deviations of the temporal PRR. The evaluation is based on all measured links. For the three different figures, three different numbers of packets for the aggregations are used: 50, 100, and 500, respectively. The red line is the expected value if a normal distribution of the packet losses is assumed (see Equation 6.2).

The resulting selection is shown in Figure 6-3. In our measurement, about 10.7% of all links are unstable, based on these conditions. To give an intuition of the relation between the temporal behavior and the standard deviation, Figure 6-4 plots the standard deviations for the unstable links shown in Figure 5-5.

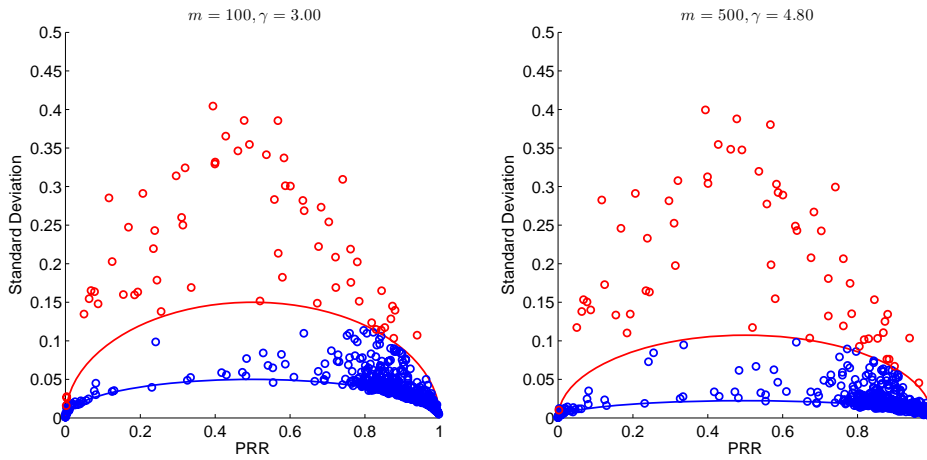


Figure 6-3
 These figures show the standard deviations of the temporal PRR with aggregation size $m_1 = 100$ and $m_2 = 500$ of the unstable links (red). The red lines define the maximum deviation for a stable link with given γ as shown in Equation 6.3.

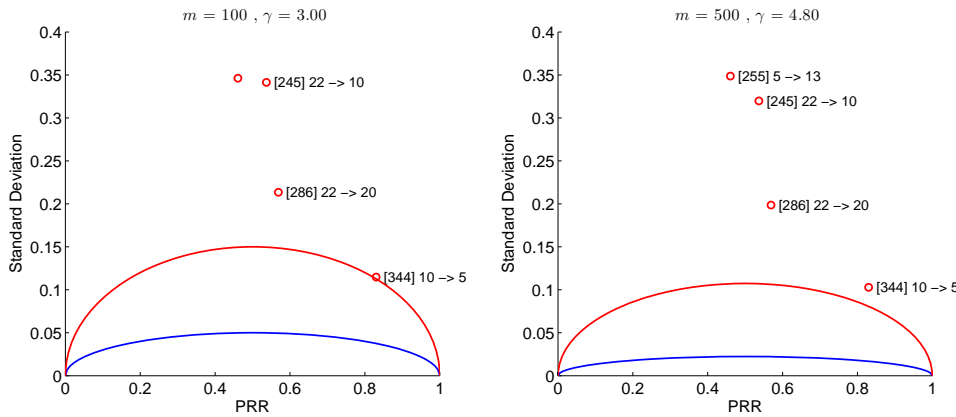


Figure 6-4
 These figures show the standard deviations of the temporal PRR of the unstable links shown in Figure 5-5.

6.1.2 Link Evaluation Tool

For the evaluation of a link in our tool chain, the PRR seems to be a good metrics for the quality of a link. This results in a value X in the range of $0 \dots 1$.

6.1.3 Record Sampling and Link Estimation Tools

A link estimation algorithm defines the record sampling and the link estimation tools. The sampling of the data is based on a pattern. With a sliding window mech-

anism and a given window step size, this pattern is slid through the whole dataset. For each step, the link estimation tool estimates the link quality with the information out of the sampled data. Mostly, the estimation is done by calculating the PRR over this little data.

In Figure 6-5, an example of this procedure is given.

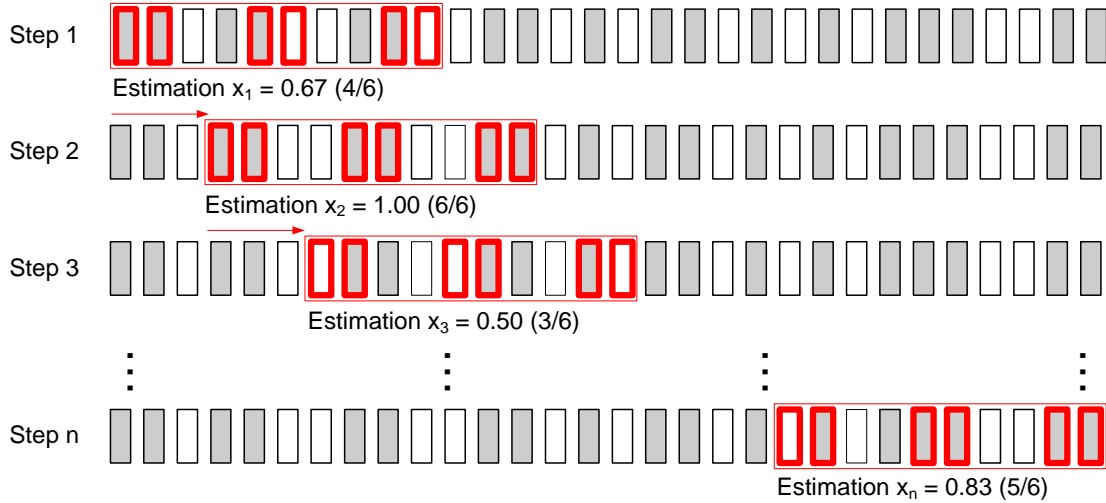


Figure 6-5

This illustration shows an example of how we sample records out of the data with a sliding window technique. Each row of bars represents the full dataset of the same link. The bars itself stand for the packets. If the bar is filled, the packet has been received, otherwise it got lost. The red pattern is sliding through the data with a window step size of 3. In each step, an estimation, based on the red marked packets, is done. For instance, for the first step: 4 out of 6 bars are filled, so the estimated packet reception rate is $\frac{4}{6} = 0.67$.

6.1.4 Comparison Tool

The last step in our tool chain is the comparison tool. It compares the different estimations ($x_1 \dots x_n$) with the evaluation (X) of the link, by calculating the variance.

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (X - x_i)^2 \quad (6.5)$$

If we evaluate the whole tool chain for all stable links ($N = 588$) in the database, we result in a vector V with N elements.

$$V = \begin{pmatrix} \sigma_{Link1}^2 \\ \sigma_{Link2}^2 \\ \sigma_{Link3}^2 \\ \vdots \\ \sigma_{LinkN}^2 \end{pmatrix}$$

6.1.5 Evaluation of different Estimation Methods

The goal of this tool chain is to compare different link estimation algorithms. This can be done by exchanging the estimation algorithm tool in the chain and comparing the resulting variance vectors. For the comparison of two such vectors (V^A , V^B), we standardized them and computed then the average of its elements. The smaller this value is, the better performance has an algorithm. For the standardization, we divided the elements of both vectors by the elements of V^A and obtained V'^A and V'^B . Thus, the resulting vector V'^A consists of 1 only.

$$v_i'^A = \frac{v_i^A}{v_i^A} = 1 \quad , \quad v_i'^B = \frac{v_i^B}{v_i^A} \quad (6.6)$$

This standardization is done to have a weighting of 1 for each link. Without the standardization, the whole evaluation would mainly base on the links with high variances. To demonstrate this effect, we make an example: Consider two estimation algorithms with their variance vectors V^A and V^B (to simplify matter, we set $N = 4$, which means that we evaluated the algorithm on only 4 links)

$$V^A = \begin{pmatrix} 0.2 \\ 0.1 \\ 0.3 \\ 5.0 \end{pmatrix} \quad , \quad V^B = \begin{pmatrix} 0.3 \\ 0.3 \\ 0.6 \\ 4.0 \end{pmatrix}$$

With these values, it seems that algorithm A performs better, because the variance is in 3 out of 4 links smaller than the variance of algorithm B. We calculated the average of the elements.

$$\begin{aligned} \overline{V^A} &= \frac{1}{4} \sum_{i=1}^4 v_i^A = 1.4 \\ \overline{V^B} &= \frac{1}{4} \sum_{i=1}^4 v_i^B = 1.3 \end{aligned}$$

With this evaluation, algorithm B is the better algorithm because the average variance is smaller than for algorithm A. However, as mentioned before, we expect that algorithm A has a better rating. So we standardize the vectors as defined in Equation 6.6 and calculated the mean value of the standardized vectors.

$$V'^A = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad , \quad V'^B = \begin{pmatrix} 1.5 \\ 3.0 \\ 2.0 \\ 0.8 \end{pmatrix}$$

$$\begin{aligned} \overline{V'^A} &= 1 \\ \overline{V'^B} &= 1.825 \end{aligned}$$

These results seem to represent the performance of the estimation algorithms A and B. They denote that, in average, algorithm B has a 1.8 times greater variance than A and is therefore the inferior algorithm.

6.2 Link-Quality Estimation Algorithms

In this section, the different link-quality estimation algorithms we analyzed are described and evaluated. The evaluation of these algorithms is based on the tool chain specified in section 6.1. First, we introduce the standard algorithm, which is the reference for the evaluation of all our algorithms. Then, we compare different algorithms with the standard algorithm and try to improve the quality of the estimation. We first change the number of packets, than we introduce patterns, and finally we use RSSI and LQI values for our algorithms. In Figure 6-2 at the end of this section, an overview of the evaluation of all our algorithms is given.

6.2.1 Standard Algorithm

For the comparison of two different algorithms, we need to standardize the variance vectors V (refer to Equation 6.6). Therefore, we define a standard algorithm, which is going to be used for the standardization in all our evaluations ($V^{SA} = 1$).

Standard Algorithm (SA) The standard algorithm sends 20 packets in a row with a frequency of 5 packets/sec. The estimated value is the calculated PRR over these 20 packets.

In addition to the evaluation with the tool chain, we present for several evaluations a more detailed statistic in which the different link qualities are analyzed separately. Therefore, we defined 6 link quality levels. The numbers in brackets indicate the number of links each quality level contains in our measurements.

$$Q_1 : 0 \leq PRR < 0.5 \quad (48)$$

$$Q_2 : 0.5 \leq PRR < 0.8 \quad (48)$$

$$Q_3 : 0.8 \leq PRR < 0.85 \quad (78)$$

$$Q_4 : 0.85 \leq PRR < 0.9 \quad (126)$$

$$Q_5 : 0.9 \leq PRR < 0.95 \quad (133)$$

$$Q_6 : 0.95 \leq PRR \leq 1 \quad (155)$$

An example of how this detailed analysis looks like can be seen in Figure 6-6, which represents the detailed analysis of the SA. The better an algorithm is the more accumulated along the diagonal the values are. An optimal algorithm would achieve zero values for all fields expect for the ones in the diagonal, on which the values would be 100%.

6.2.2 Algorithm with different Numbers of Packets

In this section, we analyze how the number of packets impacts the performance of a link-quality estimation algorithm.

Algorithms N-X These algorithms send X packets in a row with a frequency of 5 packets/sec. The estimation is done by calculating the PRR over these packets. The algorithm N-20 is equal to the SA.

The evaluation shows that using more packets improves the estimation clearly, as pictured in Figure 6-7. The relation between the number of packets X and the resulting value \bar{V}' is almost indirectly proportional. This means that if we double the number of packets, the variance of the estimation is cut in half. However, this result needs to be qualified in regard to the power and time efficiency of the algorithms. For example, algorithm N-200 requires ten times more energy and time than the SA.

In Figure 6-8 the detailed analysis of the Algorithms N-30, N-100, and N-300 is shown.

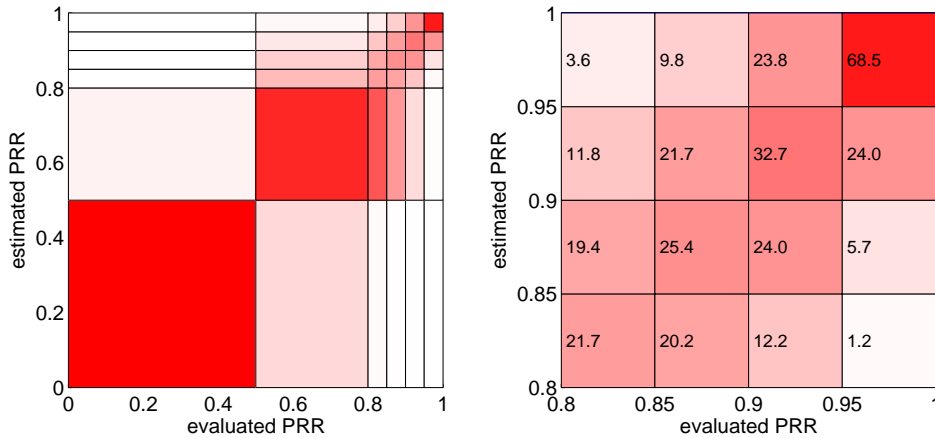


Figure 6-6

These figures show the evaluation of our basic algorithm. For each link quality ($Q_1 \dots Q_6$) the distribution of the estimations is given. The right picture is an extract of the upper right part of the left figure. The colors and the numbers show the percentage that belongs to the corresponding field (the darker the higher the percentage). For example the field in the top right means that the links with quality Q_1 are in 68.5% of all samples estimated as a Q_1 -link. The field underneath state that the Q_1 -links are in 24.0% of all samples estimated as Q_2 -links.

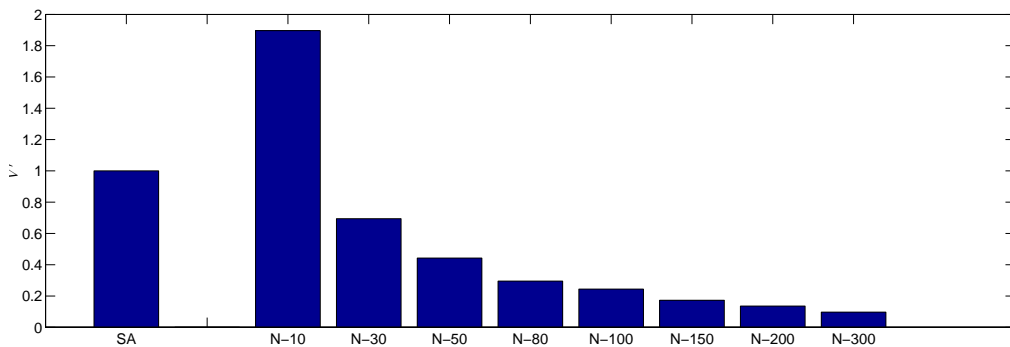
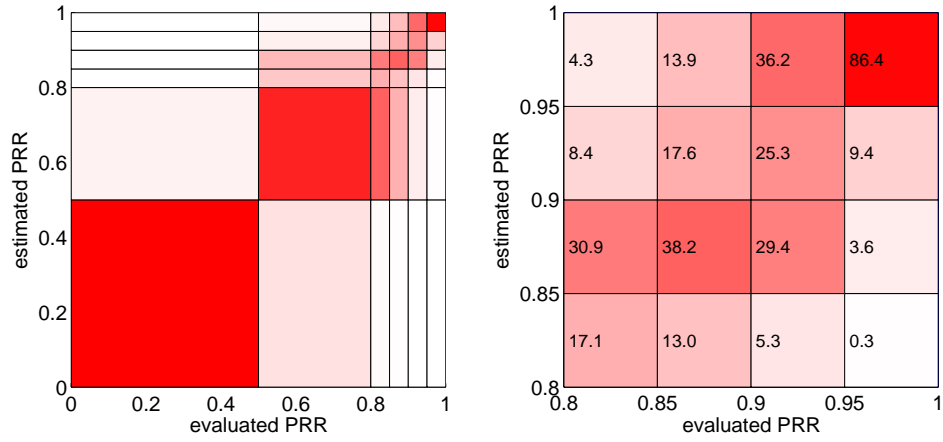
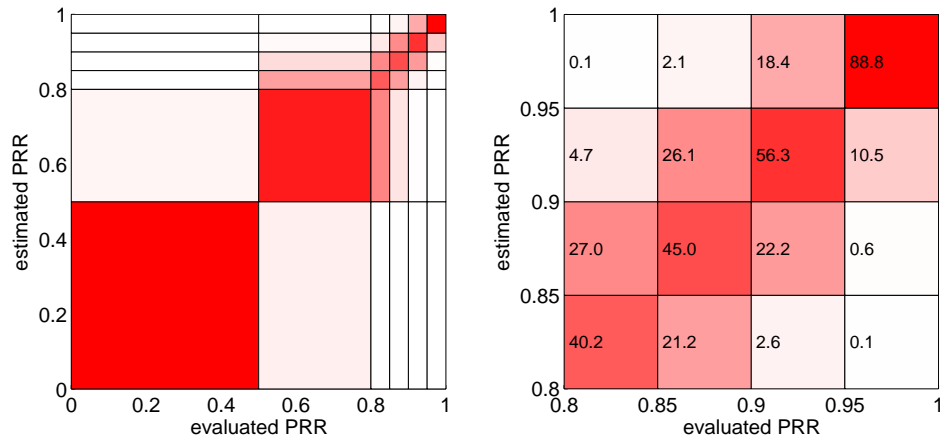


Figure 6-7

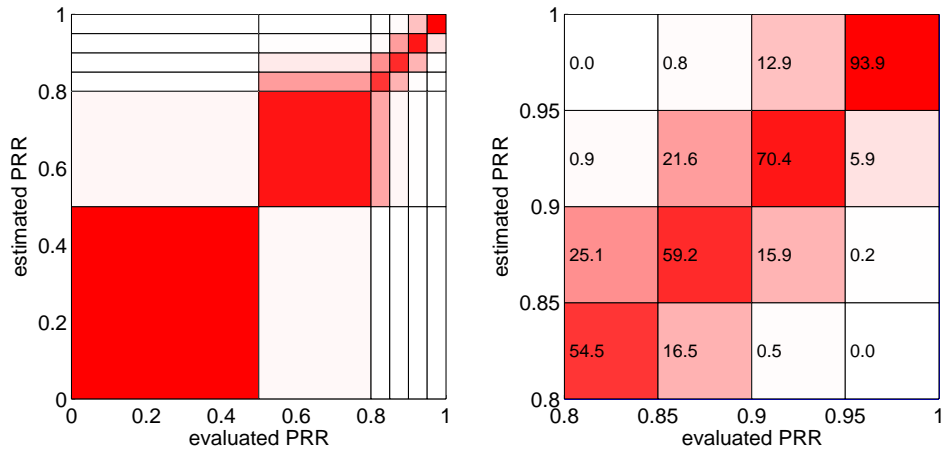
This figure shows the evaluation of the algorithms N-X with different X values. The results are standardized with the variance vector of the standard algorithm (SA).



(a) N-30



(b) N-100



(c) N-300

Figure 6-8
 These figures show a detailed analysis of the algorithms with different numbers of packets.

6.2.3 Algorithm using a Pattern for the Packet Acquisition

In this section, we analyze algorithms, which are using a pattern for the packet acquisition. First, we look at very simple patterns, which basically just change the packet frequencies. After that, we discuss more complicated patterns. In this section, all algorithms are using 20 packets, i.e. the same amount as the SA requires.

Algorithms P1-X These algorithms estimate the link with the PRR out of 20 packets. The value X indicates the packet frequency of acquiring one packet. For instance P1-1s has a packet frequency of 1 packet/sec and P1-200ms is equal to the SA.

The evaluation of these algorithms with our tool chain is visualized in Figure 6-9. It can be seen that a lower packet frequency improves the estimation. However, the enhancement remains almost static for X values higher than 10s. This might have a relation to the burst size of the packet losses we evaluated in Section 5.3.2. If such a burst occurs during the estimation, and the estimation is mainly based on packets of this burst, the resulting estimations might be incorrect. Therefore, a frequency that is bigger than the majority of the burst sizes can improve the quality of an estimation.

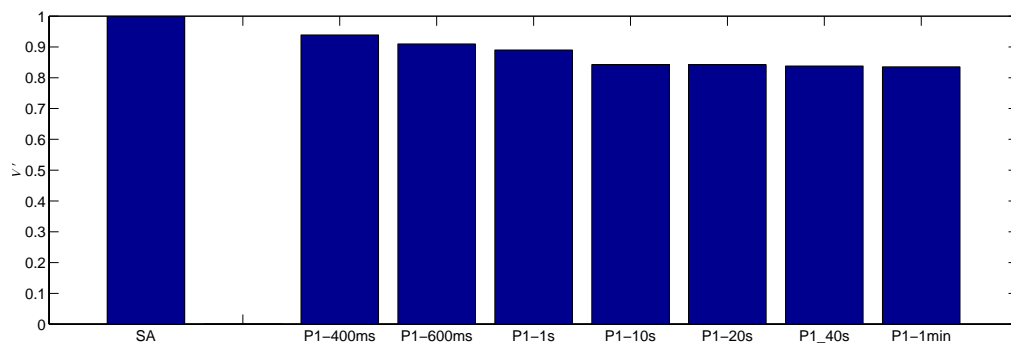


Figure 6-9

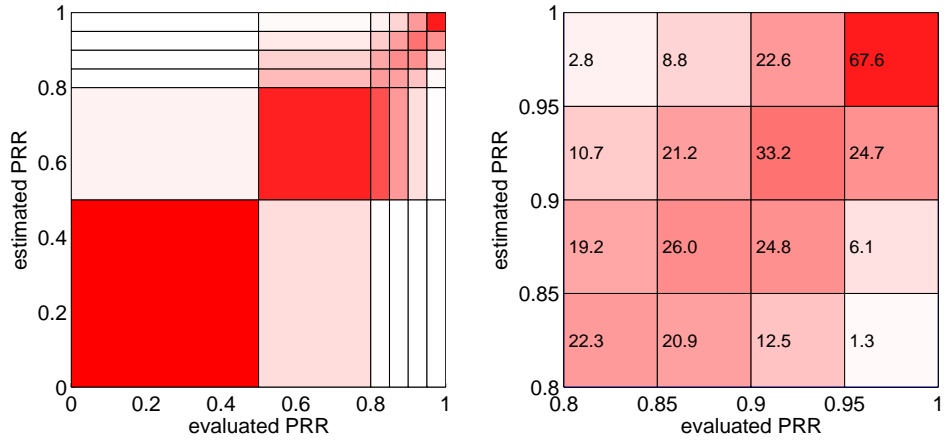
This figure shows the result of the algorithms P1-X for different X values. The evaluations are based on our tool chain and standardized with the SA.

We also tested other, more complicated pattern. For instance we took 5 packets with a frequency of 1 packet/sec, then we waited for 20 seconds and we took another 5 packets with a frequency of 1 packet/sec and so on until we acquired 20 packets. These pattern improved the estimation, but in the same scale as the algorithms P1-X. Hence, the basic characteristic for such algorithms is the temporal extension of the link estimation and the underlying pattern.

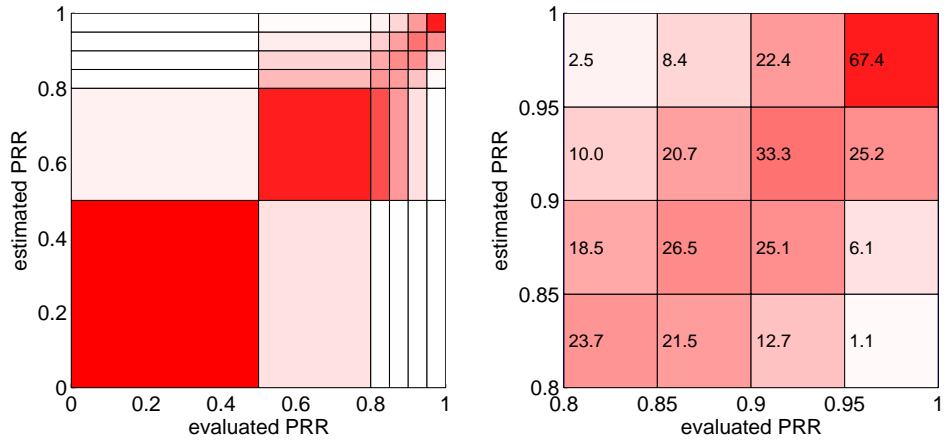
6.2.4 Algorithm based on RSSI and LQI values

This section describes and evaluates estimation algorithms that are based on RSSI and LQI values. As stated in Section 5.3.3, these values, especially the RSSI value, have a correlation with the packet reception rate. The goal of the algorithms presented in this section is to benefit from these findings.

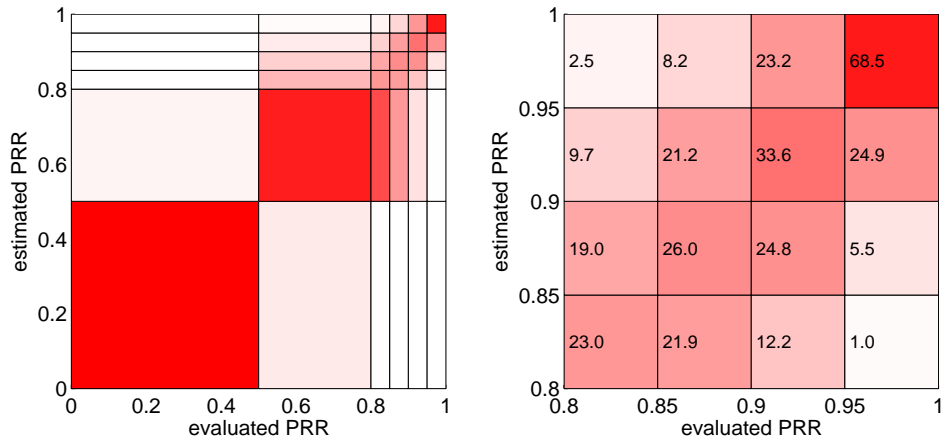
First, we analyzed the minimum RSSI values for a link similar to the evaluations done in Section 5.3.3. Figure 6-11(a) shows the result. The envelope of this graph



(a) P1-1s



(b) P1-10s



(c) P1-1min

Figure 6-10
 These figures show a detailed analysis of the algorithms that acquire packets at different frequencies.

RSSI Range [dBm]	Smallest PRR	RSSI Range [dBm]	Average PRR
-65...0	99.3%	-65...0	99.3%
-70...0	98.5%	-70...-65	99.0%
-75...0	95.1%	-75...-70	98.7%
-80...0	83.2%	-80...-75	95.7%
-85...0	0.3%	-85...-80	92.2%

Table 6-1: These tables present the smallest possible and the average PRR for a given signal strength in our measurements. The data are gathered from the Figure 6-11(a).

locks similar to the envelope based on the average RSSI values. Therefore, we can determine several thresholds for high PRR (refer to Table 6-1). For example if we analyze a packet with an RSSI value of -68dBm, we can be sure that the link quality is $\geq 98.5\%$ in our measurements.

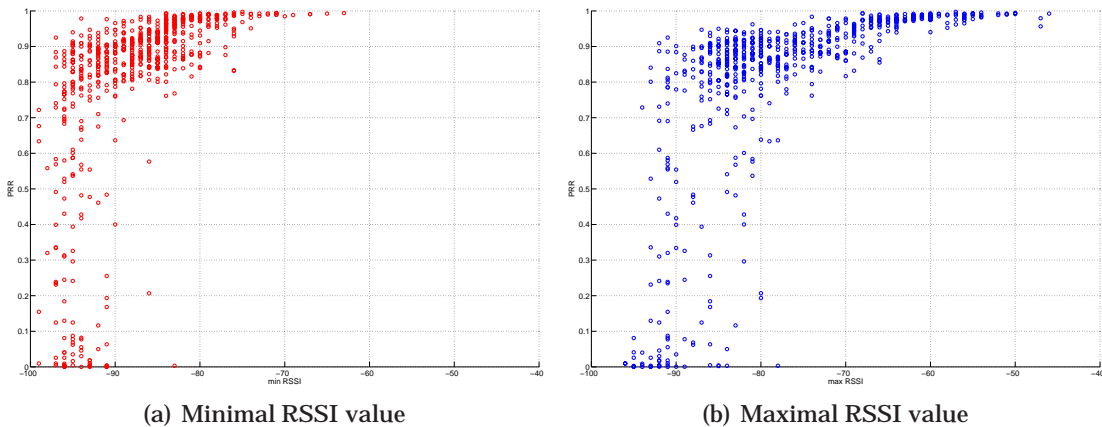


Figure 6-11

These figures show the minimal and maximal RSSI values of each link of our measurements.

We tried to find similar thresholds for low quality links with the maximal RSSI value (refer to Figure 6-11(b)), but the low RSSI values are homogeneous distributed over all link qualities. At a maximum RSSI value of -93dBm of the packets of one link, we still have packet reception rates of more than 80%. Only if a link has a maximum RSSI value of less than -95dBm, we can conclude, that the link has a packet reception rate of less than 8%.

By means of these findings, we designed an algorithm as followed.

Algorithms R The algorithm R is based on a pattern that takes 1 packet/sec up to maximum of 31 packets. The algorithm is divided into three rounds:

1. Only the first packet is considered. If it did not receive, the algorithm jumps to the second round. Otherwise, it checks if the RSSI value falls within

one of the following ranges and returns the corresponding estimation:

$$\begin{aligned} -65dBm < RSSI & \Rightarrow PRR = 0.993 \\ -70dBm < RSSI \leq -65dBm & \Rightarrow PRR = 0.990 \\ -75dBm < RSSI \leq -70dBm & \Rightarrow PRR = 0.987 \end{aligned}$$

If the packet has been qualified, the algorithm stops. Otherwise, it continues with the next round.

2. The algorithm considers another 15 packets. If the PRR of these 16 packets is less than 90%, more than one packet got lost, the algorithm jumps to the last round. Otherwise, it evaluates the minimum RSSI value of these packets and checks if this value falls within the following range and returns the calculated PRR of these packets:

$$-80dBm < RSSI \leq -75dBm$$

If this condition is fulfilled, the algorithm stops. Otherwise, it continues with the last round.

3. The algorithm returns the PRR of all 31 packets.

For the evaluation of this algorithm with our data, the average number of packets it uses is about 19 packets. With less packets than the SA, this algorithm achieves significantly better results as shown in Figure 6-12. If we look at the detailed evaluation in Figure 6-13, we can see that this algorithm enhances the estimation primarily for the high quality links.

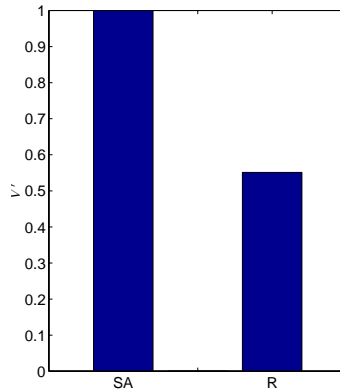


Figure 6-12

This figure shows the comparison of the algorithm R that is based on RSSI values and the SA.

Furthermore, we analyzed the maximal and the minimal LQI values for a link, similar to the minimal and maximal RSSI values. Figures 6-14(a) and 6-14(b) show the relation between these values and the PRR. If we examine thresholds for the minimum LQI, we detect that if the LQI is ≤ 90 , we already have some links with $PRR < 1\%$. Similar to the maximal RSSI values, it is hard to detect any thresholds out of the maximal LQI value of a link. We extended our algorithm R with some few LQI thresholds but could not achieve better results.

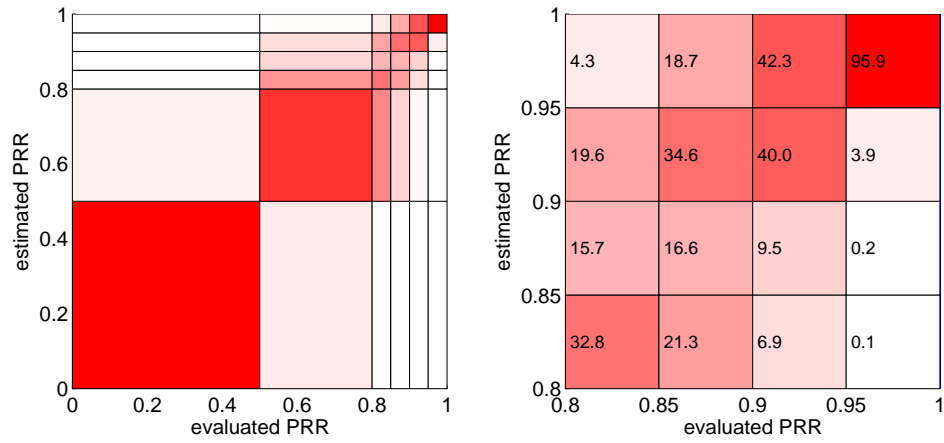


Figure 6-13
This figure shows a details analysis of the algorithm R that is based on RSSI values.

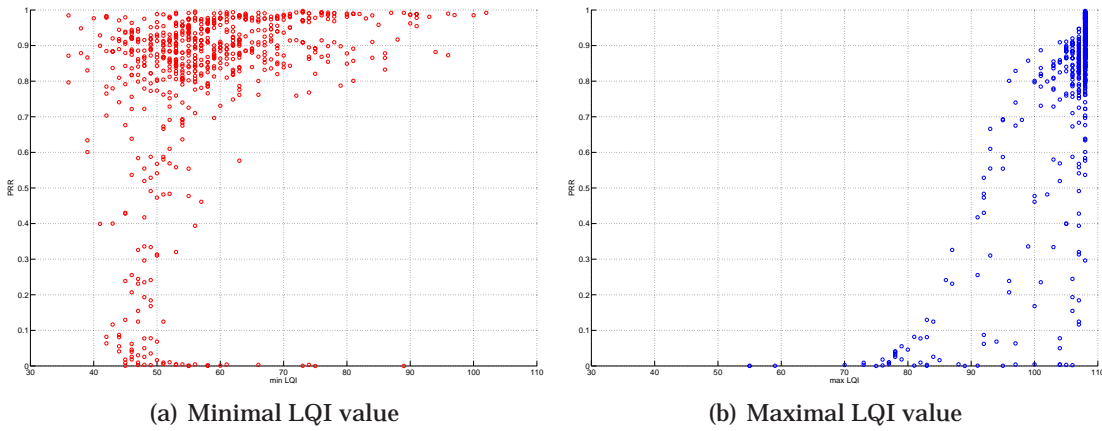


Figure 6-14
These figures show the minimal and maximal LQI values of each link of our measurements.

Algorithm	\bar{V}	Number of Packets	Duration [s]
SA	1	20	3.8
N10	1.8966	10	1.8
N30	0.6945	30	5.8
N50	0.4426	50	9.8
N80	0.2953	80	15.8
N100	0.2441	100	19.8
N150	0.173	150	29.8
N200	0.1357	200	39.8
N300	0.0968	300	59.8
P1-400ms	0.9388	20	7.6
P1-600ms	0.9097	20	11.4
P1-1s	0.8899	20	19
P1-10s	0.8427	20	190
P1-20s	0.8423	20	380
P1-40s	0.838	20	760
P1-1min	0.8355	20	1140
R	0.5512	18.95	17.95
		33% of the estimations with 1 packet	
		14% of the estimations with 16 packets	
		53% of the estimations with 31 packets	

Table 6-2: This table provides a summary of the evaluations of all our algorithms.

There are other, similar algorithms that could be designed and evaluated. However, the benefit of these algorithms and the found parameters is limited to our environment. In other surroundings, other thresholds need to be determined. With these algorithms, we showed that efficient link estimation is usually possible.

7

Conclusion and Outlook

7.1 Conclusion

The following conclusions have been drawn on extensive link-quality measurements performed in an office-like environment, i.e. with 18 Tmote Sky nodes distributed over multiple rooms with an extent of 65m.

- About ten percent of all the links showed to be unstable in terms of a time period of about 30 minutes. Moreover, some of the links have degradations over a longer period, i.e. a week or a month. This impacts the upper level routing protocol, requiring them to do over-provisioning in terms of number of links or alternatively to continuously check for new links.
- The occurrence of packet-failure bursts differs from a simulation with normal distributed data. The consequence of this result has been seen in the evaluation of estimation algorithms, based on short patterns. Therefore, an estimation algorithm should include packets with a certain time-lag.
- Comparing the data of two links with the same sender showed that the packet failures on these links are slightly more correlated than in an experiment with normal distributed data. Based on this finding, multi path routing does not seem to be meaningful. This means that if a packet on one path gets lost, the probability of a packet loss on the other path has increased.
- The measurements confirmed the existence of the 'gray area' [22]. However, this area covers more than 80% of the communication range of the Tmote Sky. Therefore, link estimation based on the distance information only, is hardly conceivable.
- Wireless links seem to have some asymmetries. Furthermore, based on a sign test, an immediate acknowledge has not improved the PRR. This means that the acknowledge packet has the same PRR as a normal packet sent over this link.

- The analysis of the RSSI and the LQI shows that wireless links have a correlation between these values and the PRR. While this correlation is highly pronounced for the RSSI values of the good quality links, it is not that obvious for the LQI values.
- Based on the link analysis, especially the RSSI values, different link-quality estimation algorithms have been designed and evaluated. This evaluation shows that an estimation
 - based on more packets performs better,
 - using a wide-stretched pattern improves the result,
 - that takes the RSSI values into account is much more efficient.

However, the parameters used for the estimations based on RSSI values are likely to depend on the environment and might need to be adapted to other environments and sensor devices.

7.2 Contribution

The contribution of this master thesis includes four main issues:

- A test bed for wireless link measurements has been implemented. This test bed, based on the DSN and specified for the Tmote Sky, allows performing link tests in various locations.
- With this test bed, extensive data has been gathered from link tests in an office-like environment.
- A script base to analyze the behavior of wireless links has been created. These scripts allow performing an in-depth analysis of the link measurements.
- An evaluation tool chain for link estimation algorithms has been designed. Evaluation of various estimation methods showed that an adaptive algorithm combined with RSSI values seems to be the most promising approach.

7.3 Outlook

The results of this master thesis are based on measurements within a specific environment. Therefore, similar tests should be performed in other surroundings and perhaps with artificial lineups of the sensor nodes. Using our portable test bed and the designed script base allows in a simple way to perform such measurements in other locations.

Furthermore, our estimation algorithms should be implemented in a routing protocol and the performance should be analyzed. In the context of evaluations, the gain of our algorithms for the efficiency of a routing protocol has not been analyzed.

A

Task Description



MASTERARBEIT

für
Tobias Rein

Betreuer: Andreas Meier
Ausgabe: 2. Oktober 2006
Abgabe: 1. April 2007

Energy and Time Efficient Link Estimation for WSNs

Einleitung

Ein drahtloses Sensor Netzwerk (WSN—Wireless Sensor Network) besteht aus einer Vielzahl von kleinen ressourcenbeschränkten Knoten welche mit Funkmodul und Sensoren bestückt sind. Diese werden in der Umwelt (z.B. in einem Haus) verteilt und erstellen möglichst autonom ein Netzwerk. Ein solches Netz ermöglicht den Knoten Sensor-Messungen auszutauschen und diese Daten gemeinsam zu verarbeiten. Nach einer Vision von Stankovic et al. [1] soll dies die 'nahtlose Integration von Rechner mit der Umwelt mit Hilfe von Sensoren und Aktoren ermöglichen'.

In verschiedenen Projekten [2, 3] konnte in den vergangenen Jahren Erfahrungen mit Sensor-Netzwerken gemacht werden. Dabei wurde festgestellt, dass eine Vielzahl der Pakete nicht bei ihrer Destination (z.B. Senke) angekommen sind. Ein Grund dafür liegt im kabellosen und daher unsicheren und schwer einschätzbaren Übertragungskanal, dessen Linkqualität infolge Interferenz und Fading sehr stark variieren kann. In verschiedenen Forschungsgruppen wurde dieses Verhalten untersucht [4, 5]. Dabei wurde zum Beispiel festgestellt, dass in einem gewissen Abstand, der sogenannten Grey Area, die Linkqualität bei sehr kleiner Änderung der Position stark variieren kann.

Für viele Applikationen und Netzwerkprotokolle ist eine gewisse Linkqualität notwendig um eine zuverlässige Funktionalität zu gewährleisten. Oft ist man mit der vorteilhaften Situation konfrontiert, dass man aus einer Vielzahl von Links die Zuverlässigsten Auswählen kann. Dabei ist es wünschenswert Links zu benutzen die auch über längere Zeit stabil sind. In dieser Arbeit soll ein Protokoll entwickelt werden, das erlaubt solche zuverlässigen Links auszuwählen. Im Hintergrund soll eine Applikation stehen, die über einen lange Zeitraum (d.h. mehrere Wochen/Monate) mit niedriger Datenrate Nachrichten verschickt. Die Linkauswahl soll dabei möglichst wenig Zeit und insbesondere wenig Energieresourcen in Anspruch nehmen.

Beim Aufbau eines WSNs ist man mit der Problematik konfrontiert, dass man nicht genau weiss was in den einzelnen Knoten geschieht. Es ist zwar prinzipiell möglich zusätzliche Information über das WSN verschicken, jedoch wird dies höchstwahrscheinlich das Verhalten des Netzwerkes verändern. Zudem kann es auch gut sein, dass die Kommunikation noch nicht zuverlässig funktioniert, und es deshalb gar nicht erst möglich ist, Zugang zum Netzwerk und somit den Knoten zu erhalten. Eine Möglichkeit für das Fehlersuchen, die Datenerfassung wie auch das Softwareupdates ist, ein so genanntes 'Deployment Support Network' (DSN) [6] zu benutzen. Ein DSN ist ein kabelloses Sekundärnetzwerk und ermöglicht die Entwicklung, das Testen, und die Validierung von Sensor-Applikationen. Dazu werden die WSN/DSN Knotenpaare gebildet welche mit einem kurzen Kabel verbunden sind. Die DSN Knoten bauen eigenständig ein drahtloses Netzwerk auf und ermöglichen somit, wie in Abbildung 1 dargestellt, einen einfach Zugang zu den angehängten WSN Knoten.

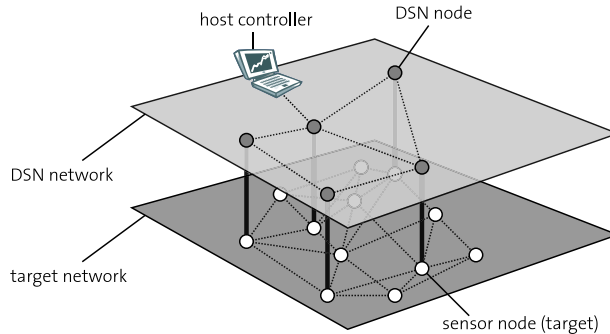


Abbildung 1: Die Knoten des Target Netzwerkes sind per Kabel mit den Knoten des ‘Deployment Support Network’ (DSN) verbunden. Dieses Sekundärnetz erlaubt einen einfachen zugriff auf die Targets, insbesondere erlaubt es Firmware updates, loggen von Nachrichten sowie das Versenden von Commands and die Targets.

Im ersten Teil dieser Arbeit soll eine Plattform aufgebaut werden, die ein möglichst autonomes Messen und Analysieren der Linkqualität des CC2420 [7] auf dem TMote Sky [8, 9] erlaubt. Dafür soll die Messplattform DSNAnalyser [10] ausgebaut werden, welche mittels Deployment Support Netzwerk automatisierte Linktests ermöglicht. Insbesondere ist auf dem TMote Sky und TinyOS 2.x [11, 12] eine Applikation zu entwickeln, welche die Befehle des DSNAnalysers versteht und entsprechend ausführt.

Im zweiten Teil der Arbeit soll die Messplattform untersucht und validiert werden. So ist es zum Beispiel denkbar, Linktests in einer EMV-Zelle durchzuführen um jegliche Interferenzen von Aussen auszuschliessen.

Im dritten Teil der Arbeit soll ein Protokoll entwickelt werden, dass die Qualität eines bzw. mehrerer Links abschätzt. Dabei soll der Energiebedarf und der Zeitaufwand optimiert werden. Um die charakteristischen Kenngrößen der Linkqualität zu bestimmen, werden auf [4, 5, 10] aufbauend weiterführende Messungen notwendig sein. So könnte es beispielsweise Sinn machen, das Übertragungsverhalten auf verschiedenen Kanälen zu betrachten.

Aufgabenstellung

1. Erstellen Sie einen Projektplan und legen Sie Meilensteine sowohl zeitlich wie auch thematisch fest [13]. Erarbeiten Sie in Absprache mit dem Betreuer ein Pflichtenheft.
2. Machen Sie sich mit den relevanten Arbeiten im Bereich Sensornetze, Systeme, Linkqualitätsmessungen sowie Linkqualitätsabschätzung vertraut. Führen Sie eine Literaturrecherche durch. Suchen Sie auch nach relevanten neueren Publikationen. Vergleichen Sie bestehende Konzepte anderer Universitäten. Prüfen Sie welche Ideen/Konzepte Sie aus diesen Lösungen verwenden können.
3. Die Applikation soll auf dem TmoteSky [8] und TinyOS 2.x [11] entwickelt werden. Arbeiten Sie sich in die Softwareentwicklungsumgebung der Knoten ein. Machen Sie sich mit den erforderlichen Tools vertraut und benutzen Sie die entsprechenden Hilfsmittel (Versionskontrolle, Bugtracker, online Dokumentation, Mailinglisten, Application Notes, Beispielapplikationen). Schauen Sie dazu insbesondere den TinyOS Programming Guide [12] an.
4. Nehmen Sie das JAWS Deployment-Support Network [6, 14] auf einigen Knoten in Betrieb und testen Sie dieses auf Zuverlässigkeit und Leistung.
5. Machen Sie sich mit dem DSNAnalyser [10] vertraut. Nehmen sie dazu die Testumgebung mit Siemens A80 Knoten sowie Adapterboard und BTnode in Betrieb.
6. Erstellen Sie ein Konzept für die Applikation auf dem TmoteSky, welche mit Hilfe des DSNAnalysers automatisierte Linktests durchführt.

7. Setzen Sie dieses Konzept um, d.h. implementieren Sie die Applikation auf dem TmoteSky.
8. Validieren Sie die erstellte Applikation und Messplattform.
9. Führen sie aufbauend auf existierenden Resultaten [4, 5, 10] Linkmessungen durch und indentifizieren sie Kenngrößen welche die Linkqualität charakterisieren.
10. Erstellen sie ein Protokoll das die Linkqualität mehrerer Links auf effiziente Art und Weise bestimmt. Dabei soll insbesondere der Tradeoff Güte der Abschätzung versus Zeit- und Energiebedarf untersucht werden.
11. Dokumentieren Sie Ihre Arbeit sorgfältig mit einem Vortrag, einer kleinen Demonstration, sowie mit einem Schlussbericht.

Durchführung der Semesterarbeit

Allgemeines

- Der Verlauf des Projektes Semesterarbeit soll laufend anhand des Projektplanes und der Meilensteine evaluiert werden. Unvorhergesehene Probleme beim eingeschlagenen Lösungsweg können Änderungen am Projektplan erforderlich machen. Diese sollen dokumentiert werden.
- Sie verfügen über einen PC mit Linux/Windows für Softwareentwicklung und Test. Für die Einhaltung der geltenden Sicherheitsrichtlinien der ETH Zürich sind Sie selbst verantwortlich. Falls damit Probleme auftauchen wenden Sie sich an Ihren Betreuer.
- Stellen Sie Ihr Projekt zu Beginn der Semesterarbeit in einem Kurzvortrag vor und präsentieren Sie die erarbeiteten Resultate am Schluss im Rahmen des Institutskolloquiums.
- Besprechen Sie Ihr Vorgehen regelmässig mit Ihren Betreuern.
- Sie führen ein Researchtagebuch in welchem sie die Fortschritte täglich protokollieren.

Abgabe

- Geben Sie vier unterschriebene Exemplare des Berichtes, das Researchtagebuch sowie alle relevanten Source-, Object und Konfigurationsfiles bis spätestens am 1. April 2007 dem betreuenden Assistenten oder seinen Stellvertreter ab. Diese Aufgabenstellung soll im Bericht eingefügt werden.
- Räumen Sie Ihre Rechnerkonten soweit auf, dass nur noch die relevanten Source- und Objectfiles, Konfigurationsfiles, benötigten Directorystrukturen usw. bestehen bleiben. Der Programmcode sowie die Filestruktur soll ausreichen dokumentiert sein. Eine spätere Anschlussarbeit soll auf dem hinterlassenen Stand aufbauen können.

Literatur

- [1] J. A. Stankovic, I. Lee, A. K. Mok, and R. Rajkumar, "Opportunities and obligations for physical computing systems.," *IEEE Computer*, vol. 38, no. 11, pp. 23–31, 2005.
- [2] 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.
- [3] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macrocope in the redwoods," in *c-sensys05*, (New York, NY, USA), pp. 51–63, ACM Press, 2005.
- [4] 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.

-
- [5] 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), Oct. 2004.
- [6] J. Beutel, M. Dyer, L. Meier, and L. Thiele, "Scalable topology control for deployment-sensor networks," in Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05), pp. 359–363, IEEE, Piscataway, NJ, Apr. 2005.
- [7] Chipcon, CC2420, Single Chip 2.4 GHz RF Transceiver for IEEE 802.15.4 and ZigBee, 2003.
- [8] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05), pp. 364–369, IEEE, Piscataway, NJ, Apr. 2005.
- [9] "Moteiv." <http://www.moteiv.com/>.
- [10] P. Oehen, "DSNAnalyzer: Backend for the Deployment Support Network," Master's thesis, Computer Engineering and Networks Lab, ETH Zürich, Switzerland, Sept. 2006.
- [11] 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.
- [12] P. Levis, "Tinyos programming." <http://csl.stanford.edu/pal/pubs/tinyos-programming-1-0.pdf>, June 2006.
- [13] E. Zitzler, "Studien- und Diplomarbeiten, Merkblatt für Studenten und Betreuer." Computer Engineering and Networks Lab, ETH Zürich, Switzerland, Mar. 1998.
- [14] "BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks." <http://www.btnode.ethz.ch>.
- [15] "NCCR-MICS: Swiss National Competence Center on Mobile Information and Communication Systems." <http://www.mics.org>.
- [16] TIK, "Notengebung bei Studien- und Diplomarbeiten." Computer Engineering and Networks Lab, ETH Zürich, Switzerland, May 1998.

B

Overview of the Tests

In this appendix, all the information about our tests with its parameters is given. First, the data from the table called 'test' in the LET database, secondly, the table 'participant', and finally the table 'setting'.

For the data of the 'test' table, we abbreviated some of the headings. The meaning of these abbreviations are given in the following:

Sc = Scenario

Se = Sender

TP = Transmission Power

Nb = Number (of packets)

Fq = Frequency (of sending packets)

L = Length (of a packet)

LP = Low Power mode

Tid	Sid	Pid	Sc	Se	TP	Packet			LP		CCA CCA
						Nb	Fq	L	TX	RX	
120	2	3	1	22	15	1000	200	12	0	0	1
121	2	3	1	21	15	1000	200	12	0	0	1
122	2	3	1	7	15	1000	200	12	0	0	1
123	2	3	1	5	15	1000	200	12	0	0	1
124	2	3	1	11	15	1000	200	12	0	0	1
125	2	3	1	20	15	1000	200	12	0	0	1
126	2	3	1	9	15	1000	200	12	0	0	1
127	2	3	1	13	15	1000	200	12	0	0	1
128	2	3	1	10	15	1000	200	12	0	0	1
129	2	3	1	2	15	1000	200	12	0	0	1
130	2	3	1	22	20	1000	200	12	0	0	1
131	2	3	1	21	20	1000	200	12	0	0	1
132	2	3	1	7	20	1000	200	12	0	0	1
133	2	3	1	5	20	1000	200	12	0	0	1
134	2	3	1	11	20	1000	200	12	0	0	1
135	2	3	1	20	20	1000	200	12	0	0	1
136	2	3	1	9	20	1000	200	12	0	0	1
137	2	3	1	13	20	1000	200	12	0	0	1
138	2	3	1	10	20	1000	200	12	0	0	1
139	2	3	1	2	20	1000	200	12	0	0	1
140	2	3	1	22	25	1000	200	12	0	0	1
141	2	3	1	21	25	1000	200	12	0	0	1
142	2	3	1	7	25	1000	200	12	0	0	1
143	2	3	1	5	25	1000	200	12	0	0	1
144	2	3	1	11	25	1000	200	12	0	0	1
145	2	3	1	20	25	1000	200	12	0	0	1
146	2	3	1	9	25	1000	200	12	0	0	1
147	2	3	1	13	25	1000	200	12	0	0	1
148	2	3	1	10	25	1000	200	12	0	0	1
149	2	3	1	2	25	1000	200	12	0	0	1
150	2	3	1	22	31	1000	200	12	0	0	1
151	2	3	1	21	31	1000	200	12	0	0	1
152	2	3	1	7	31	1000	200	12	0	0	1
153	2	3	1	5	31	1000	200	12	0	0	1
154	2	3	1	11	31	1000	200	12	0	0	1
155	2	3	1	20	31	1000	200	12	0	0	1
156	2	3	1	9	31	1000	200	12	0	0	1
157	2	3	1	13	31	1000	200	12	0	0	1
158	2	3	1	10	31	1000	200	12	0	0	1
159	2	3	1	2	31	1000	200	12	0	0	1

Tid	Sid	Pid	Sc	Se	TP	Packet			LP		CCA CCA
						Nb	Fq	L	TX	RX	
160	2	3	1	22	15	1000	200	12	0	0	1
161	2	3	1	21	15	1000	200	12	0	0	1
162	2	3	1	7	15	1000	200	12	0	0	1
163	2	3	1	5	15	1000	200	12	0	0	1
164	2	3	1	11	15	1000	200	12	0	0	1
165	2	3	1	20	15	1000	200	12	0	0	1
166	2	3	1	9	15	1000	200	12	0	0	1
167	2	3	1	13	15	1000	200	12	0	0	1
168	2	3	1	10	15	1000	200	12	0	0	1
169	2	3	1	2	15	1000	200	12	0	0	1
170	2	3	1	22	20	1000	200	12	0	0	1
171	2	3	1	21	20	1000	200	12	0	0	1
172	2	3	1	7	20	1000	200	12	0	0	1
173	2	3	1	5	20	1000	200	12	0	0	1
174	2	3	1	11	20	1000	200	12	0	0	1
175	2	3	1	20	20	1000	200	12	0	0	1
176	2	3	1	9	20	1000	200	12	0	0	1
177	2	3	1	13	20	1000	200	12	0	0	1
178	2	3	1	10	20	1000	200	12	0	0	1
179	2	3	1	2	20	1000	200	12	0	0	1
180	2	3	1	22	25	1000	200	12	0	0	1
181	2	3	1	21	25	1000	200	12	0	0	1
182	2	3	1	7	25	1000	200	12	0	0	1
183	2	3	1	5	25	1000	200	12	0	0	1
184	2	3	1	11	25	1000	200	12	0	0	1
185	2	3	1	20	25	1000	200	12	0	0	1
186	2	3	1	9	25	1000	200	12	0	0	1
187	2	3	1	13	25	1000	200	12	0	0	1
188	2	3	1	10	25	1000	200	12	0	0	1
189	2	3	1	2	25	1000	200	12	0	0	1
190	2	3	1	22	31	1000	200	12	0	0	1
191	2	3	1	21	31	1000	200	12	0	0	1
192	2	3	1	7	31	1000	200	12	0	0	1
193	2	3	1	5	31	1000	200	12	0	0	1
194	2	3	1	11	31	1000	200	12	0	0	1
195	2	3	1	20	31	1000	200	12	0	0	1
196	2	3	1	9	31	1000	200	12	0	0	1
197	2	3	1	13	31	1000	200	12	0	0	1
198	2	3	1	10	31	1000	200	12	0	0	1
199	2	3	1	2	31	1000	200	12	0	0	1

Tid	Sid	Pid	Sc	Se	TP	Packet			LP		CCA CCA
						Nb	Fq	L	TX	RX	
200	2	3	1	22	15	1000	200	12	0	0	1
201	2	3	1	21	15	1000	200	12	0	0	1
202	2	3	1	7	15	1000	200	12	0	0	1
203	2	3	1	5	15	1000	200	12	0	0	1
204	2	3	1	11	15	1000	200	12	0	0	1
205	2	3	1	20	15	1000	200	12	0	0	1
206	2	3	1	9	15	1000	200	12	0	0	1
207	2	3	1	13	15	1000	200	12	0	0	1
208	2	3	1	10	15	1000	200	12	0	0	1
209	2	3	1	2	15	1000	200	12	0	0	1
210	2	3	1	22	20	1000	200	12	0	0	1
211	2	3	1	21	20	1000	200	12	0	0	1
212	2	3	1	7	20	1000	200	12	0	0	1
213	2	3	1	5	20	1000	200	12	0	0	1
214	2	3	1	11	20	1000	200	12	0	0	1
215	2	3	1	20	20	1000	200	12	0	0	1
216	2	3	1	9	20	1000	200	12	0	0	1
217	2	3	1	13	20	1000	200	12	0	0	1
218	2	3	1	10	20	1000	200	12	0	0	1
219	2	3	1	2	20	1000	200	12	0	0	1
220	2	3	1	22	25	1000	200	12	0	0	1
221	2	3	1	21	25	1000	200	12	0	0	1
222	2	3	1	7	25	1000	200	12	0	0	1
223	2	3	1	5	25	1000	200	12	0	0	1
224	2	3	1	11	25	1000	200	12	0	0	1
225	2	3	1	20	25	1000	200	12	0	0	1
226	2	3	1	9	25	1000	200	12	0	0	1
227	2	3	1	13	25	1000	200	12	0	0	1
228	2	3	1	10	25	1000	200	12	0	0	1
229	2	3	1	2	25	1000	200	12	0	0	1
230	2	3	1	22	31	1000	200	12	0	0	1
231	2	3	1	21	31	1000	200	12	0	0	1
232	2	3	1	7	31	1000	200	12	0	0	1
233	2	3	1	5	31	1000	200	12	0	0	1
234	2	3	1	11	31	1000	200	12	0	0	1
235	2	3	1	20	31	1000	200	12	0	0	1
236	2	3	1	9	31	1000	200	12	0	0	1
237	2	3	1	13	31	1000	200	12	0	0	1
238	2	3	1	10	31	1000	200	12	0	0	1
239	2	3	1	2	31	1000	200	12	0	0	1

Tid	Sid	Pid	Sc	Se	TP	Packet			LP		CCA CCA
						Nb	Fq	L	TX	RX	
241	2	3	1	7	19	100000	200	12	0	0	1
242	2	3	1	10	19	100000	200	12	0	0	1
243	2	3	1	20	19	100000	200	12	0	0	1
244	2	3	1	2	19	100000	200	12	0	0	1
245	2	3	1	22	19	100000	200	12	0	0	1
246	2	3	1	22	19	10000	200	12	0	0	1
247	2	3	1	7	19	10000	200	12	0	0	1
248	2	3	1	10	19	10000	200	12	0	0	1
249	2	3	1	20	19	10000	200	12	0	0	1
250	2	3	1	2	19	10000	200	12	0	0	1
251	2	3	1	21	19	10000	200	12	0	0	1
252	2	3	1	9	19	10000	200	12	0	0	1
253	2	3	1	13	19	10000	200	12	0	0	1
254	2	3	1	11	19	10000	200	12	0	0	1
255	2	3	1	5	19	10000	200	12	0	0	1
256	2	10	2	22	19	10000	200	12	0	0	1
257	2	10	2	7	19	10000	200	12	0	0	1
258	2	11	2	22	19	10000	200	12	0	0	1
259	2	11	2	10	19	10000	200	12	0	0	1
260	2	12	2	22	19	10000	200	12	0	0	1
261	2	12	2	20	19	10000	200	12	0	0	1
262	2	13	2	22	19	10000	200	12	0	0	1
263	2	13	2	2	19	10000	200	12	0	0	1
264	2	14	2	7	19	10000	200	12	0	0	1
265	2	14	2	10	19	10000	200	12	0	0	1
266	2	15	2	7	19	10000	200	12	0	0	1
267	2	15	2	20	19	10000	200	12	0	0	1
268	2	16	2	7	19	10000	200	12	0	0	1
269	2	16	2	2	19	10000	200	12	0	0	1
270	2	17	2	10	19	10000	200	12	0	0	1
271	2	17	2	20	19	10000	200	12	0	0	1
272	2	18	2	10	19	10000	200	12	0	0	1
273	2	18	2	2	19	10000	200	12	0	0	1
274	2	19	2	20	19	10000	200	12	0	0	1
275	2	19	2	2	19	10000	200	12	0	0	1

Tid	Sid	Pid	Sc	Se	TP	Packet			LP		CCA CCA
						Nb	Fq	L	TX	RX	
276	2	3	1	22	19	10000	200	12	0	0	1
277	2	3	1	7	19	10000	200	12	0	0	1
278	2	3	1	10	19	10000	200	12	0	0	1
279	2	3	1	20	19	10000	200	12	0	0	1
280	2	3	1	2	19	10000	200	12	0	0	1
281	2	3	1	21	19	10000	200	12	0	0	1
282	2	3	1	9	19	10000	200	12	0	0	1
283	2	3	1	13	19	10000	200	12	0	0	1
284	2	3	1	11	19	10000	200	12	0	0	1
285	2	3	1	5	19	10000	200	12	0	0	1
286	2	3	1	22	19	10000	200	12	0	0	1
287	2	3	1	7	19	10000	200	12	0	0	1
288	2	3	1	10	19	10000	200	12	0	0	1
289	2	3	1	20	19	10000	200	12	0	0	1
290	2	3	1	2	19	10000	200	12	0	0	1
291	2	3	1	21	19	10000	200	12	0	0	1
292	2	3	1	9	19	10000	200	12	0	0	1
293	2	3	1	13	19	10000	200	12	0	0	1
294	2	3	1	11	19	10000	200	12	0	0	1
295	2	3	1	5	19	10000	200	12	0	0	1
300	2	20	2	7	19	10000	200	12	0	0	1
301	2	20	2	21	19	10000	200	12	0	0	1
302	2	21	2	7	19	10000	200	12	0	0	1
303	2	21	2	13	19	10000	200	12	0	0	1
304	2	22	2	7	19	10000	200	12	0	0	1
305	2	22	2	5	19	10000	200	12	0	0	1
306	2	23	2	7	19	10000	200	12	0	0	1
307	2	23	2	9	19	10000	200	12	0	0	1
308	2	24	2	7	19	10000	200	12	0	0	1
309	2	24	2	11	19	10000	200	12	0	0	1
310	2	25	2	10	19	10000	200	12	0	0	1
311	2	25	2	21	19	10000	200	12	0	0	1
312	2	26	2	10	19	10000	200	12	0	0	1
313	2	26	2	13	19	10000	200	12	0	0	1
314	2	27	2	10	19	10000	200	12	0	0	1
315	2	27	2	5	19	10000	200	12	0	0	1
316	2	28	2	10	19	10000	200	12	0	0	1
317	2	28	2	9	19	10000	200	12	0	0	1
318	2	29	2	10	19	10000	200	12	0	0	1
319	2	29	2	11	19	10000	200	12	0	0	1

Tid	Sid	Pid	Sc	Se	TP	Packet			LP		CCA CCA
						Nb	Fq	L	TX	RX	
320	2	30	2	20	19	10000	200	12	0	0	1
321	2	30	2	21	19	10000	200	12	0	0	1
322	2	31	2	20	19	10000	200	12	0	0	1
323	2	31	2	13	19	10000	200	12	0	0	1
324	2	32	2	20	19	10000	200	12	0	0	1
325	2	32	2	5	19	10000	200	12	0	0	1
326	2	33	2	20	19	10000	200	12	0	0	1
327	2	33	2	9	19	10000	200	12	0	0	1
328	2	34	2	20	19	10000	200	12	0	0	1
329	2	34	2	11	19	10000	200	12	0	0	1
330	2	3	1	22	19	10000	200	12	1	1	1
331	2	3	1	7	19	10000	200	12	1	1	1
332	2	3	1	10	19	10000	200	12	1	1	1
333	2	3	1	20	19	10000	200	12	1	1	1
334	2	3	1	2	19	10000	200	12	1	1	1
335	2	3	1	21	19	10000	200	12	1	1	1
336	2	3	1	9	19	10000	200	12	1	1	1
337	2	3	1	13	19	10000	200	12	1	1	1
338	2	3	1	11	19	10000	200	12	1	1	1
339	2	3	1	5	19	10000	200	12	1	1	1
340	2	5	1	2	31	10000	200	12	0	0	1
341	2	5	1	5	31	10000	200	12	0	0	1
342	2	5	1	7	31	10000	200	12	0	0	1
343	2	5	1	9	31	10000	200	12	0	0	1
344	2	5	1	10	31	10000	200	12	0	0	1
345	2	5	1	11	31	10000	200	12	0	0	1
346	2	5	1	13	31	10000	200	12	0	0	1
347	2	5	1	20	31	10000	200	12	0	0	1
348	2	5	1	21	31	10000	200	12	0	0	1
349	2	5	1	22	31	10000	200	12	0	0	1
350	2	5	1	30	31	10000	200	12	0	0	1
351	2	5	1	31	31	10000	200	12	0	0	1
352	2	5	1	32	31	10000	200	12	0	0	1
360	2	3	1	22	19	10000	200	12	0	0	1
361	2	3	1	7	19	10000	200	12	0	0	1
362	2	3	1	10	19	10000	200	12	0	0	1
363	2	3	1	20	19	10000	200	12	0	0	1
364	2	3	1	2	19	10000	200	12	0	0	1
365	2	3	1	21	19	10000	200	12	0	0	1
366	2	3	1	9	19	10000	200	12	0	0	1
367	2	3	1	13	19	10000	200	12	0	0	1
368	2	3	1	11	19	10000	200	12	0	0	1
369	2	3	1	5	19	10000	200	12	0	0	1

Tid	Sid	Pid	Sc	Se	TP	Packet			LP		CCA CCA
						Nb	Fq	L	TX	RX	
500	3	100	1	2	31	10000	200	12	0	0	1
501	3	100	1	5	31	10000	200	12	0	0	1
502	3	100	1	10	31	10000	200	12	0	0	1
503	3	100	1	11	31	10000	200	12	0	0	1
504	3	100	1	13	31	10000	200	12	0	0	1
505	3	100	1	30	31	10000	200	12	0	0	1
506	3	100	1	31	31	10000	200	12	0	0	1
507	3	100	1	32	31	10000	200	12	0	0	1
508	3	100	1	40	31	10000	200	12	0	0	1
509	3	100	1	41	31	10000	200	12	0	0	1
510	3	100	1	42	31	10000	200	12	0	0	1
511	3	100	1	43	31	10000	200	12	0	0	1
512	3	100	1	44	31	10000	200	12	0	0	1
550	3	110	2	31	31	10000	200	12	0	0	1
551	3	110	2	30	31	10000	200	12	0	0	1
552	3	111	2	31	31	10000	200	12	0	0	1
553	3	111	2	32	31	10000	200	12	0	0	1
554	3	112	2	31	31	10000	200	12	0	0	1
555	3	112	2	40	31	10000	200	12	0	0	1
556	3	113	2	31	31	10000	200	12	0	0	1
557	3	113	2	2	31	10000	200	12	0	0	1
558	3	114	2	31	31	10000	200	12	0	0	1
559	3	114	2	13	31	10000	200	12	0	0	1
560	3	115	2	43	31	10000	200	12	0	0	1
561	3	115	2	41	31	10000	200	12	0	0	1
562	3	116	2	43	31	10000	200	12	0	0	1
563	3	116	2	40	31	10000	200	12	0	0	1
564	3	117	2	43	31	10000	200	12	0	0	1
565	3	117	2	42	31	10000	200	12	0	0	1
566	3	118	2	43	31	10000	200	12	0	0	1
567	3	118	2	44	31	10000	200	12	0	0	1
568	3	119	2	43	31	10000	200	12	0	0	1
569	3	119	2	32	31	10000	200	12	0	0	1
570	3	120	2	42	31	10000	200	12	0	0	1
571	3	120	2	40	31	10000	200	12	0	0	1
572	3	121	2	42	31	10000	200	12	0	0	1
573	3	121	2	41	31	10000	200	12	0	0	1
574	3	122	2	42	31	10000	200	12	0	0	1
575	3	122	2	44	31	10000	200	12	0	0	1
576	3	123	2	41	31	10000	200	12	0	0	1
577	3	123	2	44	31	10000	200	12	0	0	1
578	3	124	2	41	31	10000	200	12	0	0	1
579	3	124	2	40	31	10000	200	12	0	0	1

Pid	Nid	Channel
3	2	26
3	5	26
3	7	26
3	9	26
3	10	26
3	11	26
3	13	26
3	20	26
3	21	26
3	22	26

Pid	Nid	Channel
5	2	26
5	5	26
5	7	26
5	9	26
5	10	26
5	11	26
5	13	26
5	20	26
5	21	26
5	22	26
5	30	26
5	31	26
5	32	26

Pid	Nid	Channel
10	7	26
11	10	26
12	20	26
13	22	26
14	7	26
15	20	26
16	2	26
17	10	26
18	10	26
19	2	26
20	7	26
21	13	26
22	5	26
23	9	26
24	11	26
25	21	26
26	13	26
27	5	26
28	9	26
29	11	26
30	21	26
31	13	26
32	5	26
33	9	26
34	11	26

Pid	Nid	Channel
10	22	26
11	22	26
12	22	26
13	2	26
14	10	26
15	7	26
16	7	26
17	20	26
18	2	26
19	20	26
20	21	26
21	7	26
22	7	26
23	7	26
24	7	26
25	10	26
26	10	26
27	10	26
28	10	26
29	10	26
30	20	26
31	20	26
32	20	26
33	20	26
34	20	26

Pid	Nid	Channel
100	5	26
100	9	26
100	11	26
100	20	26
100	22	26
100	31	26
100	40	26
100	42	26
100	44	26

Pid	Nid	Channel
100	2	26
100	7	26
100	10	26
100	13	26
100	21	26
100	30	26
100	32	26
100	41	26
100	43	26

Pid	Nid	Channel
110	30	26
111	32	26
112	40	26
113	2	26
114	13	26
115	41	26
116	40	26
117	42	26
118	44	26
119	32	26
120	40	26
121	41	26
122	44	26
123	44	26
124	40	26

Pid	Nid	Channel
110	31	26
111	31	26
112	31	26
113	31	26
114	31	26
115	43	26
116	43	26
117	43	26
118	43	26
119	43	26
120	42	26
121	42	26
122	42	26
123	41	26
124	41	26

Sid	Nid	BTnode	Location	
			X	Y
2	2	00:04:3f:00:01:80	153	175
2	5	00:04:3f:00:01:c3	90	266
2	7	00:04:3f:00:00:e8	68	94
2	9	00:04:3f:00:00:d8	86	135
2	10	00:04:3f:00:01:c0	81	154
2	11	00:04:3f:00:01:6a	69	195
2	13	00:04:3f:00:00:e5	147	155
2	20	00:04:3f:00:01:41	13	207
2	21	00:04:3f:00:01:09	78	59
2	22	00:04:3f:00:01:77	33	10
2	30	00:04:3f:00:01:C2	322	184
2	31	00:04:3f:00:01:89	250	197
2	32	00:04:3f:00:01:AA	421	191
3	2	00:04:3f:00:01:80	210	219
3	5	00:04:3f:00:01:c3	90	266
3	7	00:04:3f:00:00:e8	68	94
3	9	00:04:3f:00:00:d8	86	135
3	10	00:04:3f:00:01:c0	81	154
3	11	00:04:3f:00:01:6a	69	195
3	13	00:04:3f:00:00:e5	147	155
3	20	00:04:3f:00:01:41	13	207
3	21	00:04:3f:00:01:09	78	59
3	22	00:04:3f:00:01:77	33	10
3	30	00:04:3f:00:01:C2	322	184
3	31	00:04:3f:00:01:89	250	197
3	32	00:04:3f:00:01:AA	421	191
3	40	00:04:3f:00:00:c8	488	144
3	41	00:04:3f:00:01:51	560	240
3	42	00:04:3f:00:01:86	650	237
3	43	00:04:3f:00:01:a1	658	145
3	44	00:04:3f:00:01:7d	710	309

Bibliography

- [1] BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks. <http://www.btnode.ethz.ch>.
- [2] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. *Mobi-Hoc'05*, 2005.
- [3] Chipcon. *CC2420, Single Chip 2.4 GHz RF Transceiver for IEEE 802.15.4 and ZigBee*, 2003.
- [4] The XML-RPC community. Xml-rpc. <http://www.xmlrpc.com/>.
- [5] D. Culler et al. Tinyos: An operating system for networked sensors. <http://webs.cs.berkeley.edu/tos>.
- [6] Matthias Dyer, Jan Beutel, and Lennart Meier. Deployment support for wireless sensor networks. *Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland*, 2005.
- [7] David Kotz, Calvin Newport, and Chip Elliott. The mistaken axioms of wireless-network research. *Dartmouth College Computer Science Technical Report TR2003-467*, July 18 2003.
- [8] L. Krishnanurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalmanger, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: Experiences from a semiconductor plant and the north sea. *SenSys'05*, 2005.
- [9] Dhananjay Lal, Aratu Manjeshwar, Falk Herrmann, Elif Uysal-Biyikoglu, and Abtin Keshavarzian. Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. *GLOBECOM '03*, 2003.
- [10] Philip Levis. Tinyos programming. <http://csl.stanford.edu/pal/pubs/tinyos-programming-1-0.pdf>, June 2006.
- [11] Roman Lim. Wireless fire sensor network demonstrator. Master's thesis, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2006.

- [12] Shan Lin, Tian He, Jingbin Zhang, Gang Zhou, Lin Gu, and John A. Stankovic. Atpc: Adaptive transmission power control for wireless sensor. *SenSys'06*, 2006.
- [13] Patrice Oehen. Dsn analyzer: Backend for the deployment support network. Master's thesis, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2006.
- [14] Niels Reijers, Gertjan Halkes, and Koen Langendoen. Link layer measurements in sensor networks, 2004.
- [15] Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. Experimental study of concurrent transmission in wireless sensor networks. *SenSys'06*, 2006.
- [16] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. Understanding the causes of packet delivery success and failure in dense wireless sensor networks. *SenSys'06*, 2006.
- [17] Kannan Srinivasan and Philip Levis. Rssi is under appreciated. *EmNets 06*, 2006.
- [18] Fred Stann, John Heidemann, Rajesh Shroff, and Muhammad Zaki Murtaza. Rbp: Robust broadcast propagation in wireless networks. *SenSys'06*, 2003.
- [19] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Tuner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. *SenSys'05*, 2005.
- [20] Andreas Willig and Robert Mutschke. Results of bit error measurements with sensor nodes and casuistic consequences for design of energy-efficient error control schemes. *EWSN*, 2006.
- [21] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. *SenSys'03*, 2003.
- [22] Jerry Zhao and Ramech Govindan. Understanding packet delivery performance in dense wireless sensor networks. *SenSys'03*, 2003.
- [23] Chipcon. <http://www.chipcon.com>.
- [24] Moteiv. <http://www.moteive.com>.
- [25] Zigbee alliance. <http://www.zigbee.org>.