Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

armasuisse

Department of Information Technology
and Electrical Engineering of ETH Zürich
Computer Engineering and Networks Laboratory (TIK)
Communication Systems Group (CSG)

Master Thesis

"Ultra-fast and Accurate Wireless Link Quality Estimation and the Benefits it Provides for Detection of Reactive Jamming"

Michael Spuhler

Supervisor:	Prof. Dr. Bernhard Plattner (ETH)
Main advisor:	Dr. Vincent Lenders (Armasuisse)
Advisor:	Dr. Domenico Giustiniano (ETH)
Advisor:	Dr. Franck Legendre (ETH)
Handover Date:	21th of August 2012

Abstract

Link quality estimation is an important feature for wireless network protocols such as jamming detection, routing, rate switching, handover or topology control mechanisms. Existing link quality estimators (LQEs) tend to fail on specific link conditions: Packet statistic based LQEs are accurate in static scenarios but show poor performances due to long estimation times in dynamic environments such as mobile setups. Whereas signal strength based LQEs are fast but inaccurate.

We propose new LQEs relying on different properties of preamble symbols for direct sequence spread spectrum transceivers. Our approaches consider chip errors, the number of received preambles, the variance of the chip errors based LQEs, and hybrid concepts. These new LQEs are evaluated experimentally using software defined radios on IEEE 802.15.4 under four different link conditions such as cable, wireless line-of-sight, non-line-of-sight, and mobile environments.

We define CEPPS FWA (Chip Error per Preamble Symbol Filtered Weighted Average) as our best performing estimator based on preamble chip errors that weights and filters sequential estimations on a per-packet level. We show that CEPPS FWA performs in all wireless scenarios more accurately and faster than state-of-the-art estimators.

Moreover a novel approach to detect sophisticated reactive jamming attacks that target the start of frame delimiter during a packet transmission is presented. Fast chip error evaluations in the synchronization header allow to predict the link quality quick enough to accurately detect jamming attacks for links with packet delivery ratios greater than 40%.

Zusammenfassung

Die Schätzung der Kanalqualität ist ein integraler Bestandteil in drahtlosen Netzwerkprotokollen wie Störsendererkennung, Datenverkehrslenkung, Wahl der Übertragungsrate, Zugangspunktübergabe oder Netzstruktur Kontrollmechanismen. Bestehende Schätzer der Kanalqualität (SKQ) weisen unter bestimmten Kanalbedingungen Schwächen auf: Paketstatistik basierte SKQ arbeiten in statischen Szenarien genau, scheitern jedoch aufgrund zu langen Schätzzeiten in dynamischen Umgebungen wie beispielsweise in mobilen Netzwerken. Hingegen sind signalbasierte SKQ schnell, weisen aber ungenaue Schätzungen auf.

Wir präsentieren einen neuen SKQ basierend auf verschiedenen Eigenschaften der Präambelsymbole für Sende-Empfänger Strukturen, die Direktsequenzspreizspektren verwenden. Unser Ansatz berücksichtigt Chipfehler, die Anzahl Präambelsymbole, die Varianz von Chipfehlerbasierten SKQ und hybride Methoden. Die neuen SKQ werden experimentell mit computergestützten Programmen für drahtlose Datenübertragungen unter verschiedenen Kanalbedingungen wie Kabelverbindungen, drahtloser Sichtverbindung, verdeckter Sichtverbindung und mobilen Umgebungen ausgewertet.

Wir definieren CEPPS FWA (Chip Error per Preamble Symbol Filtered Weighted Average) als unseren besten SKQ, der auf der Auswertung von Chipfehlern basiert und aufeinanderfolgende Schätzungen auf Paketstufe gewichtet und filtert. Wir zeigen, dass CEPPS FWA in allen drahtlosen Datenübertragungszenarien schneller und genauer die Kanalqualität schätzt als andere SKQ auf dem letzten Stand der Forschung.

Zusätzlich präsentieren wir einen neuartigen Ansatz für die anspruchsvolle Erkennung von reaktiven Störsendern, die Synchronisationssymbole während der Paketübertragung stören. Schnelle Chipfehler Auswertungen während der Synchronisationsphase erlauben die Kanalqualität genügend schnell zu schätzen, um Störangriffe für Paketübertragungsraten über 40% zuverlässig zu erkennen.

Acknowledgment

I wish to express my gratitude to my supervisor, Prof. Dr. Bernhard Plattner for his support and all the contributing input.
Special thanks are due to Dr. Vincent Lenders from Armasuisse for this excellent supervision of this thesis, the great support and all the motivating discussions we had. Further I would like to thank Dr. Domenico Giustiniano and Dr. Franck Legendre for the coordination at the ETH Zürich and the interesting conversations.
Finally I would like to convey my best thanks to Pirmin Heinzer and Björn Muntwyler for their support and help during this work.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The rapid evolving wireless technologies like IEEE 802.15.4 or IEEE 802.11a/b/g/n gained tremendous popularity in the recent past. These types of networks supply a large amount of the increasing demand for wireless data traffic.
A fundamental problem of these network protocols relates to link quality estimation. The performance jamming detection, routing, rate selection, handover, or topology control mechanisms heavily depend on accurate and fast link quality estimators (LQEs). However, the task of estimating the effective link quality in real-life wireless networks remains a challenge. Particularly in dynamic link environments, where moving nodes or objects foster the unpredictable and location-sensitive nature of wireless channels. Especially in mobile environments, the link quality tends to be hard to estimate quickly and accurately with estimators that rely on signal strength, packet statistics, or hybrid approaches [5].
A related topic to LQE is jamming detection in wireless networks. Especially due to the growth of wireless sensor networks (WSNs) there imposes new requirements to detect jamming-style attacks. The highly shared medium upon WSNs are built makes it easy for adversaries to launch signal interference attacks to corrupt sent packets. Specific attacks as reactive jammers turned out to be very hard to detect [6] and lack new detection approaches.

1.2 Thesis Goals

The goal of this thesis is to develop a new wireless LQE that manages to quickly and accurately estimate the link quality in mobile networking environments. In particular, we are interested in estimating the probability that a packet may successfully be received at its destination. To achieve a rapid estimation, we develop and explore a novel approach that relies on the preamble of the physical layer. The goal is to estimate the link quality within milliseconds and an accuracy

of less than 5% absolute error.
Relying on the preamble symbol in direct sequence spread spectrum systems (DSSSs), we expect two important benefits. First, unsuccessful packets where the synchronization fails may be detected and hence used to model the packet delivery ratio (PDR). Second as we will show in this thesis, the estimation model requires only a few symbols in the preamble and should therefore allow to quickly estimate the PDR.
To validate this approaches, we implement new estimators and compare them under realistic wireless channel conditions to a set of existing LQEs. Candidates of the following four estimator classes were evaluated for comparison:

1. Packet statistics
2. Signal strength
3. Chip errors
4. Hybrid

Moreover the developed approaches are applied to a jamming detection scenario to show the benefits of the preamble based LQEs.

1.3 Thesis Structure

This thesis is structures as follows: Chapter 2 introduces to the topic of link quality estimations, focussing on the fundamentals, the related work and eventually points out the benefits of the preamble symbol based link quality estimation. Besides jamming detection is discussed with its fundamentals and related work.
Chapter 3 explains the experimental setup in detail and illustrates the different measurement scenarios. The next Chapter 4 documents the exploration of the preamble symbols and shows how the different LQEs were designed. The results revealing the performance of the LQEs are explained in Chapter 5 for different link conditions. Further the results of the jamming detection estimation is shown. Eventually the conclusions in Chapter 6 complete this thesis and give an outlook of the presented work.

2

Link Quality Estimation and Jamming Detection

2.1 Link Quality Estimation

This section introduces the topic of link quality estimation and further gives an overview of the related work which describes different estimation strategies.

2.1.1 Fundamentals of Link Quality Estimation

Link quality estimation is a crucial building block for higher layer protocols of low-power links. Essential network protocols rely on LQE. For instance routing protocols improve their efficiency by avoiding bad quality links. Also topology control mechanisms need link quality estimation to establish stable topologies that resist to link fluctuations [7]. Moreover the performance of rate selection, handover, or jamming detection algorithms heavily depend on accurate LQEs.
The fundamental approach is to evaluate a certain metric that is related somehow to the respective link quality. This metric is an arbitrary expression, e.g. signal strength, retransmission rate, or the number of symbols.
As the term "link quality" is not generally defined, we need to find a quantity that properly reflects the quality of a link between a sender and a receiver. For this purpose we define the random variable $\mathcal{X}$ that expresses if a packet was transmitted successfully ($\mathcal{X} = 1$) or got lost ($\mathcal{X} = 0$). The so called packet delivery ratio (PDR) can then be described as

$$\text{PDR} = P(\mathcal{X} = 1).$$

Henceforth in this thesis we refer to the link quality the probability mass, that a packet is successfully delivered, i.e. the PDR.
In experimental measurements it is not possible to determine $P(\mathcal{X} = 1)$ at any point of time to any accuracy due to changing link properties. Therefore the PDR has to be approximated in a proper way to provide meaningful information about the link quality. For this purpose

two important conditions have to be met. Suppose that a window of e consecutive events ($\mathcal{X}$ was 1 or $\mathcal{X}$ was 0) were obtained in the measurement. The first condition is that the considered amount of samples e is large enough, i.e. the mean of the PDR is converged. Second, the channel is supposed to be constant over all events e. The approximated PDR in a measurement within an observation window w can then be calculated as

$$\text{PDR}(w) = \frac{\text{Number of correctly decoded packets within } w}{\text{Number of sent packets within } w}.$$

The observation window w can be interpreted as w seconds or w received or sent packets depending on the context.
The mentioned second condition becomes crucial in measurement setup where the channel shows dynamic behavior, e.g. in mobile settings. How fast this changes occur is mainly dependent on the relative speed of the sender and the receiver v and the carrier frequency f_c. The greater v and the higher f_c, the more fluctuations of the PDR over time are expected. The so called coherence time T_c characterizes how fast the changes of the channel are. To be more precise, the channel is assumed to be constant over the coherence time T_c. It can be approximated as

$$T_c \approx \frac{1}{4D_{\text{spread}}},$$

where D_{spread} is the Doppler spread. The Doppler spread D_{spread} itself can be calculated as

$$D_{\text{spread}} = D_r - D_s,$$

with D_s the Doppler shift of the sender and D_r the Doppler shift of the receiver. These shifts are defined as

$$D_x = \frac{f_c v_x}{c}.$$

Where c is the speed of light ($c = 299'792'458$ m/s).

Therefore any window w that considers PDR evaluations has to be smaller than T_c to provide meaningful information over an amount of time, where the link quality was not changing.

The process until the final estimation of a LQE normally undergoes the following three steps [7]:

1. **Link monitoring:** A link can be monitored within an observation window w in an active, passive or hybrid way. If the node is monitoring the channel actively, he sends probe packets at a defined rate.
   Passive monitoring has been widely used in WSNs due to its energy-efficiency compared to active monitoring [8, 9]. Here the normal traffic of the link is exploit without incurring additional communication overhead. A disadvantage of passive link monitoring is if the network operates at unbalanced traffic or low data rate. This may cause a lack of up-to-date link measurements and the link quality could be estimated inaccurately.
   Hybrid link monitoring is a combination of active and passive monitoring. This mix allows to find a trade-off between energy-efficiency and up-to-date link measurements [10].

2. **Link measurements:** Link measurements collect the required data during the transmission to gain sufficient information for the calculation of the used estimator metric. This can be done at sender- or receiver-side. Receiver-side link measurements use information of received packets, such as sequence numbers, or time stamps. The computation of sender-side LQEs encounter e.g. packet retransmission count or other attributes.

3. **Metric evaluation:** The retrieved data from the link measurements were analyzed to calculate a metric that indicates the link quality, i.e. the PDR. This could be the average signal strength of a packet, that is mapped to the respective PDR, based on a known correlation between the signal strength and the PDR [11].

An accurate link quality estimator is characterized by several requirements. These requirements guarantee that a designed estimator is able to perform well under different conditions. Three of such important requirements [7] are described next.

1. **Accuracy:** A challenging task for LQEs is the ability to correctly reproduce the real behavior of the link. Since there is no metric that is used universally as a "real" measurement of the link quality and the fact, that it is described by different quantities[1], comparisons to a real picture of the link is very difficult.

2. **Reactivity:** This term refers to the property to react on intermediate-term changes[2] in the link quality. These changes could arise in a network setting where the sender, the receiver or both move. The ability of an estimator to react on such fluctuations is an essential quality in mobile settings.
   Reactivity depends on two factors: the link monitoring scheme and the estimation window w. Active monitoring and small w tend to be reactive LQEs. On the other hand large w are not able to capture short-time changes.

3. **Stability:** In contrast to reactivity this property expresses the ability to tolerate short-term fluctuations of LQEs. The stability of a LQE can be characterized by the variance of the mean. An easy approach to introduce more stability is to calculate a weighted moving average with a large smoothing factor as proposed in [9].

Generally the two requirements - stability and reactivity - are at odds. For instance consider a mobile scenario where the nodes move at a certain speed. Due to the speed and the small coherence time the channel is assumed to vary a lot over time. With the design goal to create a highly stable LQE we set the observation window w to a large value and use a weighted moving average with a large memory, i.e. a large smoothing factor. As a result the link quality estimations are highly averaged out over time and the estimator is unsusceptible to small-term changes of the link quality. This benefit will be at the cost of reactivity because this long-term estimator will not encounter link dynamics at a fine grain and will not perform well in dynamic scenarios. As a consequence an accurate and versatile LQE should provide a trade-off between reactivity and stability.

2.1.2 Related Work

The related work of LQEs can be divided into four groups: signal power, packet statistic, hybrid and error based link quality estimation. For each group different important LQE are presented

[1] Like packet reception rate, packet retransmission or packet delivery ratio as used in this thesis
[2] In IEEE 802.15.4 networks typically from a few milliseconds to a couple of seconds (see next chapter).

and discussed.

Signal Power based Link Quality Estimation

Received Signal Strength Indicator (RSSI): The RSSI LQE is a hardware-based approach that reads the received signal strength from an eight bit register in the radio chip. The first eight symbols, i.e. the preamble sequence which precedes every packet is considered to calculate the RSSI. For ZigBee applications the CC2420 [12] radio chip from chipson is usually integrated in the sensor nodes.
This approach is suitable to distinguish between poor link qualities ($0\% < \text{PDR} \leq 10\%$), intermediate links ($10\% < \text{PDR} \leq 90\%$) and good links ($90\% < \text{PDR} \leq 100\%$) [13].

Signal to Noise Ratio (SNR): This on hardware based approach as discussed in [11], encounters in comparison to RSSI the noise floor as well. It is widely used in theoretical channel modeling to calculate the expected bit error rate, which can be extrapolated to the packet error rate and then to the PDR [14]. In practice it is measured using oscilloscopes or read from the hardware-based RSSI values. The noise floor is determined by measuring the signal strength between two packets. This technique of including the noise floor is assumed to perform better than the related RSSI, that encounters only the pure received signal strength. As the noise floor is not identical at every different node SNR should be preferred over RSSI [13].

Link Quality Indicator (LQI): This as well by hardware (CC2420 [12]) provided LQE is not directly dependent on signal power but indicates the average correlation value of the eight preamble symbols. Unfortunately there is no exact algorithm presented and the LQI suffers trustworthiness due to variations of the correlation coefficient between LQI and PDR [15, 16].
Interesting is that the relatively high variance of the LQI estimator can be exploit for quality estimation. Boano et al. [17] were able to distinguish between good links at a low LQI variance and bad links at a high LQI variance. A mapping function is not presented for the conversion of the variance to the PDR. In our work we present a LQE approach which is based on the variance of a designed LQE including the mapping and the performance evaluation.

Kalman filter based quality estimator (KLE): The KLE [18] is used to overcome the poor reactivity of average-based LQEs. To allow an estimation after a single received packet, the RSS (Received Signal Strength) is extracted and injected to a Kalman filter. This filtering produces an estimation of the RSS. Further the noise floor is subtracted from the RSS in order to approximate the SNR. This SNR is then mapped to a pre-calibrated PDR-SNR curve to obtain the KLE link quality. In the studies [18] the accuracy of KLE was not examined. This performance would be dependent on the PDR-SNR curve, which was considered constant over time. From this point of view it seems that the promising results found by Senel et al. are due to the steady environment in the experimental scenario.

Conclusion: The advantage of hardware-based LQEs relying on the signal power is that they provide a fast and inexpensive way of a rough link quality classification into good or bad. Further they do not require any additional computation. However these LQEs don't provide accurate estimates as reported in previous studies [19, 20]. Mainly for the following three reasons [21]. First these metrics are measured based on the samples of the first eight symbols of a received packet and not over the whole packet. Second, these metrics are only measured for successfully received packets. If a radio link is poor, they may overestimate the link quality by not considering information of lost packets. Third, these hardware metrics are not capable to provide a fine-grained estimation of the link quality [22, 23]. Especially in the so called transitional region for PDRs between 10% and 90%. This region occupies a large interval in the possible values of PDR

and is characterized by high-variances in reception rates and asymmetric connectivity.
Finally G. Zhou et al. showed in [24] that the RSSI varies when changing the propagation direction from the sender. This irregularity leads to non isotropic PDRs. With other words, the PDR has not the same value in all directions from the source that complicates estimations based on the RSS.

Packet Statistic based Link Quality Estimation

Required Number of Packet transmissions (RNP): Cerpa et al. described in [25] this sender-side estimator that counts the average number of packet transmissions and retransmissions that are required for a successful reception. This metric is computed for each estimation window w as the total transmission and retransmission of packets divided by the number of successfully received packets, minus 1 to exclude the first packet transmission.

Expected Transmission Count (ETX): This receiver-side estimator, proposed by Coutor et al. [26] uses active monitoring where probe packets are sent periodically. ETX is calculated as the inverse of the product of the forward delivery ratio d_f and the backward delivery ratio d_b encountering link asymmetries. The value d_f refers to the PDR based on the received probe packets. d_b on the other hand is computed based on the received acknowledgments.

Window Mean with Exponentially Weighted Moving Average (WMEWMA): In [9] Woo and Culler describe this receiver-side LQE based on passive monitoring. WMEWMA uses an exponential weighted moving average (EWMA) filter to weight a history of computed PDR with previously observed ones. Proceeding like this the PDR estimation is smoothed and the result is a metric that averages out transient fluctuations.

Conclusion: These packet statistic based LQEs were widely used in mesh networks where the links are relatively stable over short times. Since they require a few packets to determine their estimation of the PDR (usually above 10 packets), packet statistics are not well suited for fast link estimation with low channel coherence times as found in mobile settings. But even in static scenarios where the wireless devices do not move, the properties of the wireless channel will typically change over time because of the movement in the area around the devices.

Hybrid based Link Quality Estimation

Four-Bit: This universal metric is presented in [22] and uses four bits of information from different layers. One is from the physical layer to rapidly identify good links on a packet level. The second bit from the link layer indicating the successful transmission of a packet. The last two bits are from the network layer and are used for a neighbor table replacement policy. The link quality is composed of two metrics through the EWMA filter. Namely the RNP and the inverse of the WMEWMA minus 1. It uses both, passive (data packet) and active (probe packets) traffic monitoring.

Fuzzy Link Quality Estimator (F-LQE): The benefit of F-LQE [21] is the link quality estimation on the basis of four link properties. It encounters smoothed packet reception ratio (SPRR), link stability factor (SF), link asymmetry level (ASL), and channel average signal to noise ratio (ASNR). ASNR defines the capacity of the link to successfully deliver data. ASL expresses the difference in connectivity between the uplink and the downlink. SF quantifies the variability level of the link and the ASNR reflects the degree of noise in the channel. The proposed fuzzy rule reads: "If the link has high packet delivery and low asymmetry and high stability and high channel quality, then it has high link quality." This quantity is calculated using membership

functions and the received score is further smoothed using the EWMA filter to provide stable link quality estimations.

Triangle Metric: This metric as described in [27] combines geometrically the information of PDR, LQI and SNR. The receiver calculates the mean of the LQI and SNR values over an estimation window of approximately 10 packets. Further the norm

$$n = \sqrt{{\text{SNR}_{\text{mean}}}^2 + {\text{LQI}_{\text{mean}}}^2}$$

is calculated as comparison to empirical-based thresholds. Then the channel can be quantified into four different link qualities.

Conclusion: Hybrid estimators combine several metrics to cope with weaknesses of single metric approaches. These estimators tend to be more accurate than approaches that rely on single metrics as signal power or packet statistics. But they still require a few packets for the packet statistic analysis and therefore don't offer a reliable and fast link quality estimation in dynamic scenarios as moving nodes or a fast changing environments. Moreover the sophisticated evaluation of different layers or the need of higher memory footprint and computation complexity (e.g. F-LQE) is in contradiction to the low-energy consumption of WSNs.

Error based Link Quality Estimation

Bit Errors: Studies as in [28–30] propose approaches to predict the packet error rate (PER) or bit error rate (BER) based on error considerations. This could be extended to the concept to estimate link qualities based on bit errors. Figure 2.1 illustrates the measured occurrence of bit errors in our testbed. The vertical axis shows the probability that we obtain a wrong frame check sequence (FCS) in a transmitted packet. A wrong FCS implies that at least one bit in the packet is erroneous. Note the logarithmic vertical axis and for a PDR of 0.5 we obtain already the probability of 0.05% that one or more bits are flipped. However this sparse bit errors would imply huge estimation windows to provide meaningful information about the link's PDR. Therefore these approaches are not suitable for our considerations. Further e.g. [28] relies on error estimating codes (EEC) that imply additional packet overhead that is no suitable for approaches based on simple network principles.

The authors in [31] suggest that the chip error rate might be a better channel quality indicator than signal power based metrics particularly in the presence of interference.

Chip Errors per Frame (CEPF): To the best of our knowledge there exists only one study - CEPF [5] which considers chip errors in DSSS based systems to estimate the link quality. They analyze chip errors in the payload on a packet level and map the obtained chip error rate to the PDR in a measured linear correlation curve of chip errors and PDRs. They evaluated various LQEs and showed that a LQE based on chip errors perform more accurate than signal strength approaches and much faster than packet statistic based LQEs. In contrast to our work, they did not analyze the synchronization header, i.e. the eight preamble symbols.
Conclusion: Due to sparsely occurring bit errors, this metric is not suitable for PDR estimations. Bit error based LQEs need especially for good links huge estimation windows that makes it impossible to follow fast link quality changes in dynamic setups as mobile environments.
Chip error based approaches show promising results in terms of fast channel estimation. But they lack accuracy in static scenarios in comparison to packet statistic based approaches. In dynamic settings they outperform some LQEs but don't show the desired superior performance in terms of accuracy. Obviously it still remains a challenge to exploit chip errors for accurate link quality estimation.

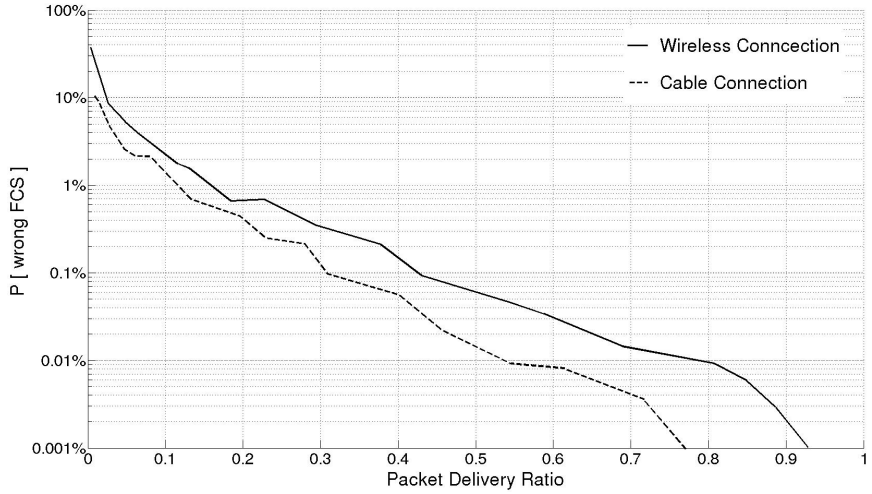


Figure 2.1: This graph shows the measured probability that a decoded packets has a wrong frame check sequence (FCS) for wireless and cable connections. A wrong FCS implies that there occurred one or more bit (or equally symbol) errors in a decoded packet. Note the logarithmic vertical axis, i.e. the probability that some bit errors happen decreases very fast for increasing link qualities. We obtain already for a PDR of 0.5 a probability that at least one bit will be corrupted of 0.05%.

2.1.3 Benefits of Preamble Symbol based Link Quality Estimation

State of the art LQE approaches still suffer from crucial disadvantages regarding universal usability (e.g. static versus dynamic environments). So far non of the presented LQE could cope with the challenge to fulfill the mentioned key requirements of a well performing LQE as energy efficiency, accuracy, reactivity and stability at the same time. Especially the contradiction of optimizing reactivity and stability at the same time exacerbate the design of universally well performing LQEs.
Our novel approach of considering preamble symbols comes with significant improvements in comparison to the above mentioned LQEs. Namely we experience enhancements in accuracy and reactivity. Further our approach realizes an optimization of the reactivity and stability at the same time.
By using the preamble symbols we are not dependent on a successful synchronization on a sent packet. An estimation can already be performed, although the receiver did not synchronize on the packet. This effect of an ultra-fast estimation becomes even more noticeable if the link quality is poor and the expected waiting time for a successful synchronization is large. Beside this time savings, our LQE approach extracts further advantages of being independent on synchronization. We are not evaluating chip errors based on this conditional event (i.e. a successful synchronization as in [5]), thus we don't risk to overestimate the channel if the radio link suffers from excessive packet losses. This allows to rely on a unbiased picture of the channel condition and leads to more accuracy in the link quality estimation.
Moreover our approach realizes an elegant method to end up with a good trad-off of the above considerations to meet the requirements of reactivity and stability. To consider reactivity, we

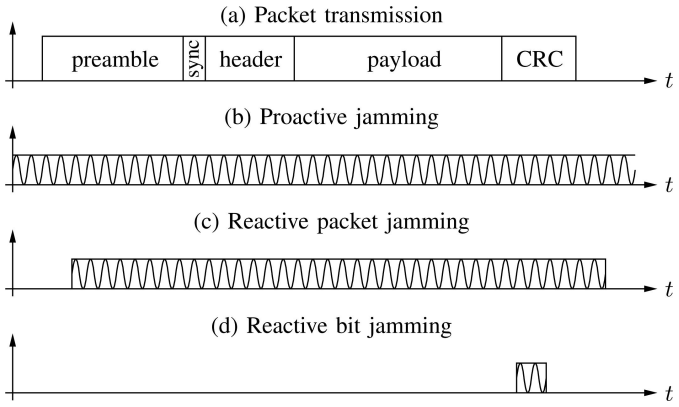


Figure 2.2: Three different jamming types [1].(b) Proactive jammers emit continually radio signals so that no transmissions [Fig.1(a)] are possible. If an ongoing transmission has been detected reactive jammers (c) in contrast jam only once, typically for the entire packet length. Reactive bit jammers (d) target their jamming signal at a specific part of the packet and keep the jamming duration to a minimum.

provide ultra-fast link quality estimations on symbol level [3]. To guarantee stability, this estimations are weighted over a window of successive estimations. Therefore we obtain a large amount of different link quality estimations within shortest time and are able to obtain an averaged value without increasing the estimation window to a large value at the cost of reactivity, i.e. ignoring intermediate-term changes of the link.

2.2 Jamming Detection

2.2.1 Fundamentals of Jamming Detection

Due to the fact that wireless networks are built upon a shared medium it makes it easy for adversaries to launch jamming-style attacks. Such jamming attacks are accomplished by emitting radio frequency signals that do not follow an underlying MAC protocol [6].
As WSNs gained tremendous popularity in the recent past, security and trustworthiness became crucial issues. Many wireless security threats may be addressed through appropriately designed network security architectures [32–36]. These studies are mainly modifications of traditional security services, namely authentication, confidentiality, and integrity to wireless domain. However wireless networks are susceptible to threats that are not addressed via cryptographic methods. Jamming attacks are an important class of such threats that cannot be fend off by conventional security mechanisms. From this point of view the detection of jamming is an integral part in seizure of security issues in WSNs. Moreover the ability of wireless devices to detect radio frequency attacks allow to identify regions of poor link conditions. This is a key factor in routing, handover, rate selection algorithms or restorative mechanisms, such as channel surfing or spatial retreats [37].
A wide range of different attacker strategies that a jammer can perform is known so far [6,38–41]. The three most important attacker models are described below. For a more detailed description,

[3] i.e. within less than a milliseconds in our testbed

please refer to [38].

1. **Proactive jammer:** The proactive jammer continually sends out random bits to the communication medium without following the MAC-layer etiquette. Traditional approaches for the detection of such jammers use the PDR and the received ambient signal strength as a decision criteria whether the link is proactively jammed or not. It is detected as soon the PDR and/or the ambient signal strength exceeds a predefined threshold [6, 42]. However these types of jammers are easily detected due to to long exposure time to the communication medium and furthermore not well suited for low-power WSNs concerning energy consumption.

2. **Random jammer:** This approach imitates the proactive jammer but attacks the network at random time slots and alternates between sleeping and jamming mode. Compared to the proactive jammer, this strategy takes energy conservation into account. By adjusting the sleep time and the jam time a trade-off can be realized between energy efficiency and jamming effectiveness. Nevertheless great jamming rates cannot be achieved due to restricted energy supply in WSNs. Further the detection of random jammers are covered by the same approaches as for the proactive jammer.

3. **Reactive jammer:** As illustrated in Figure 2.2(d) reactive jammers try to corrupt only a small fraction of a detected packet. By doing so, the jammer is only exposed to the communication medium for a very short time slot. This comes with two important benefits. On the one hand, this minimizes energy consumption on the other hand this exacerbate the jamming detection drastically. The above mentioned ambient signal jamming detection approaches are well-suited for mid- or long-term jamming. But they cannot cope with the challenge of detecting reactive jamming for essentially two reasons [1]: First, existing approaches rely only on the frame check sum (FCS) of a packet do decide if it was received correctly. In general it can not be distinguished between packet losses due to weak radio links and interference. Second, if only a small fraction of a packet is jammed, this does not necessarily result in a steady and high RSS [43–45]. The increase in the the effective RSS is thus kept to a very low value and can hence avoid of being detected with current approaches.

In order to evaluate our jamming detection approach, we will further focus on the reactive jammer. Moreover we will introduce a jammer attack model that is within the family of reactive jammers even the most challenging one. By setting the jammed fraction of the packet to the start of frame delimiter (SFD) that is responsible for the synchronization, the receiver will not be able to synchronize to any packet. This reduces the information provided at a sender side jamming detection significantly. Approaches that rely on this lost information as the payload or the FCS fail rigorously. Our approach does not suffer from such limitations (e.g. a successful synchronization). To the best of our knowledge no work is proposed that solves this problem of reactive jamming detecting in a far satisfying way and it still remains a challenge to cope with such short-term jamming attacks. There is one approach proposed by Strasser et al. [1] that allows to detect reactive jamming on a packet level. But this work imposes sever restrictions as predetermined information of a packet, overhead caused by error correcting/detecting codes, or limited node wiring. This approach is not suitable for versatile wireless scenarios since these restrictions may not be applicable. In the following section we will discuss this in more detail.

2.2.2 Related Work

Because our novel approach of jamming detection is focussing on reactive jamming, we will not list related works of other jammer families as e.g. proactive or random jammers. Furthermore we will explain why our novel approach can cope with the above mentioned jamming attacks on the SFD.

The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks [6]: Xu et al. propose the usage of PDR along with either signal strength at the receiver or location information as a consistency check for jamming detection. In the first case, jamming is detected if the PDR is low although the RSS is high. In the second case if the PDR is low although the senders are close. Unlike our work, their approaches are not able to detect reactive jamming that might affect only a few bits of a packet.
Detection of Reactive Jamming in Sensor Networks [1]: Strasser et al. proposed the first jamming detection scheme for sensor networks that enables the detection of reactive (single-bit) jamming on a per-packet basis. The main idea of their approach is to identify the cause of individual bit errors within a packet and to deduce therefrom whether the packet was jammed or just sent over a weak link. If the node receives a packet it records the RSS for each received bit of the packet. The intuition behind this process is that if there was a bit error although the RSS value was high, this indicates external interference; if the error was due to a weak signal, the RSS value should be low.
This process of the identification of bit errors can be accomplished based on three different restrictions: Predetermined knowledge of the sent packet, error correcting codes, or limited node wiring.
Obviously they rely on a successful synchronization of the receiver and a sent packet. Thus this approach will not be able to perform jamming detection in our described attacker's model, because no decoded symbols of the MPDU (MAC Protocol Data Unit) are available at the receiver side due to prevented synchronizations. Common WSNs detection algorithms are not robust enough to resist jamming attacks of the synchronization header. Strasser et al. proposes that the sender applies error correcting codes to the header and shuffles the encoded bits according to a pseudo random sequence based on a secret key shared by the sender and the receiver. This packet detection algorithm with the additional added overhead of error correcting codes increase the header length and thus require more energy for a packet transmission and reduce the data rate. In contrast to our approach, we don't relay on a successful synchronization and don't suffer from these limitations.
A further challenge is to localize the bit errors within the received packet. Note that on simple DSSS systems as used in WSNs there exist no error localization or correction mechanisms on a symbol or bit level. One possibility to overcome this problem it the predetermined knowledge of the sent bit stream as suggested in [1]. Having this sever restriction, the transferred information in a packet is therefore limited to at best a few bits. As our approach is not relying on any error position localization, we don't experience this restriction. Strasser et al. proposed to overcome these limitations by means of error detecting/correcting codes: These codes allow for detecting bit errors in arbitrary messages. By comparing the received and the recovered data bit errors can be localized. Again this technique cause additional overhead and the packet length (and thus the energy required for its transmission) might be several times higher than the length of the original packet. In addition if a code word can be identified as being faulty but cannot be corrected, all bits in the work might equally be wrong and not correctable packet can't be used for the jamming detection.
They show a third way to acquire the error position based on limited, short-range sensor node

wiring in the form of wired node chains. With the introduced redundancy by transmitting information over wireless links and cable connections, different versions of RSS-sequences are combined into an error sample and with further comparisons the errors within a packet can be localized. Again this sophisticated method of error localization causes the expensive evaluation of this algorithm and further implies cable connections within the network that is in most scenarios unwanted and moreover imposes sever restriction on node mobility.
Conclusion: Even Strasser et al. propose a working jamming detection algorithm with very promising results concerning detection rates. But the mentioned restrictions introduce sever limitation which are contradicting the policies of WSNs and in a wide range of scenarios not applicable. Our novel approach does not suffer at all from these restrictions. We experience great benefits of not being dependent on a successful synchronization or any bit error localization.

3
Experimental Setup

The scope of our investigations of the preamble symbols is only limited to DSSS systems. Therefore our concept can be extended to any communication systems that show the key characteristic of DSSS where a sequence of bits are spread to a higher rate sequence of so called chips. The signal can than be modulated by any appropriate modulation scheme.
To find a suitable platform, our measurements rely on one possible instance of a communication standard that employs DSSS, namely IEEE 802.15.4. This standard allows relatively simple communication protocols and facilitates to deal with the respective signal processing blocks.

3.1 IEEE 802.15.4 Standard

The IEEE 802.15.4 standard [2] defines a communication layer in the OSI (Open System Interconnection) model as shown in Table 3.1. It specifies the physical layer and the media access control (MAC) for low-rate wireless personal area networks (WPANs). The intention of this standard is to provide a low-rate network protocol considering technological simplicity, without sacrificing flexibility or generality. The benefits as low power consumption or low manufacturing costs are suitable for ubiquitous devices in many applications.

On top of the IEEE 802.15.X, there established a product-oriented open global standard, called ZigBee [46]. ZigBee is a group comprised of international technology companies that work together to enable reliable, cost-effective, low-power, wirelessly networked, monitoring and control products.
In this standard, the higher layers in the OSI model are specified that enables the interoperability of different applications. It is used in a wide range of products, e.g. building automation, remote control, energy management, health care, home automation or retail services [47]. The concrete benefits of e.g. home automation provided by the mentioned technology are described by an online magazine [48] like this: "You arrive home from a long day at work. As soon as you use your digital key to unlock the door, your house adjusts the lighting, heat, and window blinds to your liking and puts on your favorite CD in the kitchen. While you were at work, the

No.	Layer	Data Unit
7.	Application	Data
6.	Presentation	Data
5.	Session	Data
4.	Transport	Segments
3.	Network	Datagram
2.	Data Link - LLC Sublayer - MAC Sublayer	Frame
1.	Physical	Bit

Table 3.1: The OSI Model

house fed the cat, turned off the space heater your kids accidentally left on in the basement, and recorded motion-triggered video from security cameras around the property. Your refrigerator detected an almost empty milk carton and added a gallon of two percent to the shopping list that it will e-mail to you on Friday." This could be a description of a home environment using ZigBee technology where a bunch of network nodes collect and exchange data between each other or even on the world wide web to please the users needs.

The IEEE 802.15.4 operates on one of three possible unlicensed frequency bands:

- 868.0 - 868.6 MHz (Europe)
- 902 - 928 MHz (North America)
- 2.4000 - 2.4835 GHz (Worldwide)

Among these, our measurements rely on the most widely-used 2.4 GHz band. It is divided into 16 channels each spaced 5 MHz apart with a spectral window of 2 MHz. The channel numbers 0 to 10 are allocated in the two lower bands. Therefore the lowest channel in the 2.4 GHz band starts at the number 11 and ends at the channel number 26. The center frequency f_c of each channel number k is defined as:

$$f_c(k) = 2405 + 5(k - 11) \text{ MHz}, \ k = 11, 12, \ldots, 26$$

In the 2.4 GHz band, data is transmitted at a rate of 250 kbit/s. As a modulation technique, a 16-ary quasi-orthogonal modulation based on DSSS (Direct Sequence Spread Spectrum) is employed. This modulation spreads a low rate sequence of bits to a higher rate sequence of so called chips. The binary source data is divided into groups of 4 bits (later referred to as symbols) and mapped to a nearly orthogonal 32-chip pseudo-noise sequence $(b_0, b_1, b_2, b_3) \mapsto (c_0, c_1, c_2, c_3, \ldots, c_{31})$, resulting in a chip rate of 2 MChips/s. For example the mapping of the zero symbol: $(b_0, b_1, b_2, b_3) = (0, 0, 0, 0)$ to the respective chip sequence $(c_0, c_1, c_2, c_3, \ldots, c_{31})$ is

$$(0, 0, 0, 0) \mapsto (1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0).$$

The entire predefined conversion according to the IEEE 802.15.4 standard from these 16 different symbols to the respective chip sequences can be found in Table B.1.

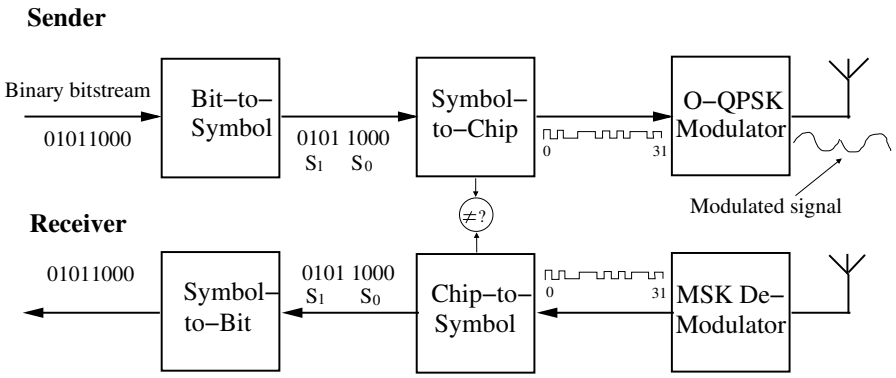


Figure 3.1: Modulation and Spreading Functions in IEEE 802.15.4 [2]

At the sender side the resulting chip sequences are concatenated and the aggregated string consisting of zeros and ones are modulated onto the carrier using minimum shift keying (MSK), which is equivalent to offset quadrature phase shift keying (O-QPSK) with half-sine pulse shaping. A simple schema to convert MSK to O-QPSK is described in [49] by J.Notor et al.. The chip sequence is O-QPSK modulated to a baseband transmission waveform and eventually emitted by the sender's antenna, see Figure 3.1.
At the receiver side, the signal is demodulated according to the modulation scheme using O-QPSK or MSK as O-QPSK modulation with half-sine pulse shape. Further the demodulated signal is decoded using a correlator to map the received 32 chip sequences back to symbols.
Due to the characteristics of a wireless channel, the received chips may contain errors caused by fading or interference. A specific received (and maybe corrupted) chip sequence $R_j, j = 1, 2, \ldots, 2^{32}$ is compared to the 16 predefined chip sequences $C_i, i = 1, 2, \ldots, 16$. The receiver chooses the best match, i.e. the C_i, such that $h(R_j, C_i)$ is minimized, where $h(\cdot, \cdot)$ is the hamming distance (number of positions containing different chips) between the two arguments. For example $h((01101), (01110)) = 2$ because the two vectors vary in the last two bits which are flipped.

However if too many chips are flipped, the expression $h(R_j, C_i)$ may be minimized for the wrong i, such that the receiver interprets the received chip sequence as a wrong symbol. The receiver will detect these symbol errors in an incorrect frame check sum (FCS). Note that there exists no error correction on a bit or symbol level in the IEEE 802.15.4 standard. Therefore a packet containing a false FCS is simply discarded and has to be retransmitted.

The PHY protocol data unit (PPDU) frame is structured as illustrated in Figure 3.2. It consists of a preamble sequence (four bytes of 0x00), a start of frame delimiter (SFD), a frame length and the MAC protocol data unit (MPDU). The MPDU itself is divided into the frame control field (FCF), the data sequence number, the address information, the frame payload and finally two bytes of frame check sequence (FCS). Depending on the type of frame, the address can be left out or consists of up to 20 bytes. The FCS is the cyclic redundancy check of a 16-bit checksum of the MPDU. The checksum uses the polynomial $p(x) = x^{16} + x^{12} + x^5 + x$.

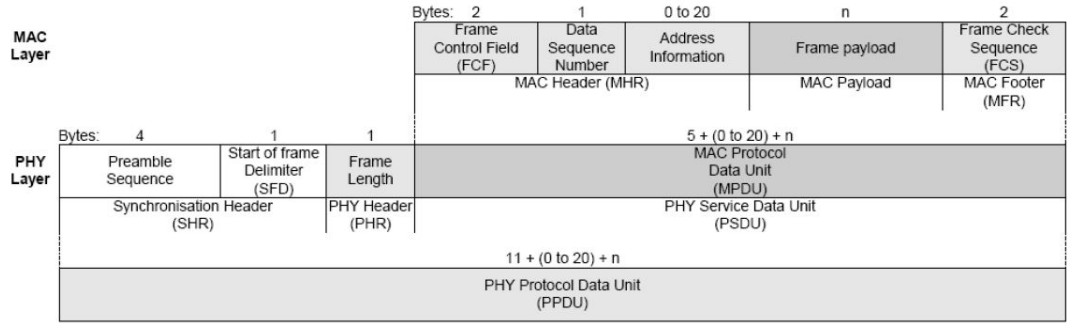


Figure 3.2: Frame Layout of an IEEE 802.15.4 Packet [2]

3.2 Hardware and Software Platform

3.2.1 Software Defined Radio

Software defined radio (SDR) is a system which uses software for modulation and demodulation of communication signals. The goal of such an implementation is to use as few hardware as possible. Replacing hardware by software comes with significant benefits concerning flexibility. Since the 90s, SDRs have been around and started out in the analog modem industry, where the implementation of modems was done in software. This allowed to easily upgrade the modulation schemes when a new standard came out, without changing the hardware. This was an incredible flexibility improvement, though one pays for it with more computing power [4].
SDR offers an efficient and comparatively inexpensive solution to the problem of traditional hardware based radio devices that limit cross-functionality and can only be modified through physical intervention [50]. Nowadays SDR found important utility for the military or cell phone services due to the possibility to change radio protocols in real time.
The main reason to use SDR in our experimental setup is the flexibility of changing parameters in the whole flow of signal processing. The quickness of compiling and loading software allows researchers much higher cycle rate in terms of iterative development.
Further the modularity of SDR that enables to build new projects upon existing code has been exploit in this work by using the GNU Radio community.

3.2.2 GNU Radio 802.15.4 En- and Decoding

GNU Radio [51] is a collection of open source software that allows to implement radios with low-cost external radio frequency (RF) hardware and commodity processors. GNU Radio provides a combination of signal and data processing blocks to easily develop further to powerful modulation, demodulation, or more complex signal processing systems. These applications are primarily written in the python [52] programming language. The computationally expensive signal processing primitives are programmed in C++ [53]. An interface compiler SWIG (Simplified Wrapper and Interface Generator) [54] integrates C++ into scripting language. Further provides GNU Radio a simple interface to the signal processing blocks from python.
Various applications written in GNU Radio range from decoding HDTV pictures, which can re-

(a) USRP1 (b) USRP2

Figure 3.3: The USRP1 and the more powerful USRP2 [3]

ceive and send AM/FM broadcast radio to implementations of packet radio system using Gaussian minimum shift keying (GMSK) modulation and demodulation to transmit packets from one host to another.
GNU Radio supports several hardware platforms [55], like sound cards, and multiple RF frontends to receive different bands of the RF spectrum. The most commonly used one in the universal software radio peripheral (USRP).

USRP [3] was developed by Matt Ettus as a low-cost and flexible platform for SDRs. It consists of one motherboard which holds the analog digital converter (ADC), digital analog converter (DAC), and a field programmable gate array (FPGA). An antenna is connected to the daughterboard. The function of the daughter-board is to down-convert a specific band from higher frequency into an intermediate frequency which can be handled by the ADCs. In the most of our measurements the USRP1 has been used. But in order to reproduce results from related work [5, 56] partially the USRP2 was utilized to guarantee identical experimental conditions. Furthermore the used platform, as proposed in [57] for the jamming scenario, which will be described in the following section, uses the USRP2. In both versions of the USRPs, we used the XCVR2450 2.4 GHz - 2.5 GHz daughter-board by Ettus Research [3]. One of the major difference between the USRP1 and the USRP2 is the connection between the device and the running host computer. In the other USRP1 it is realized by an USB 2.0 link in contrast to the USRP2, where a gigabit Ethernet (GigE) connection links the device to the computer. GigE has a maximum transfer rate of 120 MB/s that is equivalent to 30 MS/s. On the other hand the USB 2.0 link is limited to 32 MB/s, i.e. 8 MS/s. Moreover the user can implement custom functions in the FPGA, or in the on-board 32-bit RISC softcore on the USRP2.

To implement a whole IEEE 802.15.4 standard based on SDR and the GNU Radio we adapted the UCLA (University of California, Los Angeles) ZigBee project. The UCLA ZigBee framework proposed by Thomas Schmid [4] implements the missing processing blocks necessary for modulation and demodulating of the IEEE 802.15.4 standard.

An overview of the modulation is depicted in Figure 3.4 as a block diagram. As a first step, the messages are received from the python application and are put into a queue. Then the packet source generates a stream of bytes which are sent into the *ieee802.15.4_mod* block. This block translates bytes into chips (Figure 3.1), i.e. it spreads the symbols according to the IEEE 802.15.4 protocol. The next block "Chips to Symbols" translates the integers to another sequence of integers where each bit in the integer is represented as a 0 or a 1. The 0 and 1 integers are

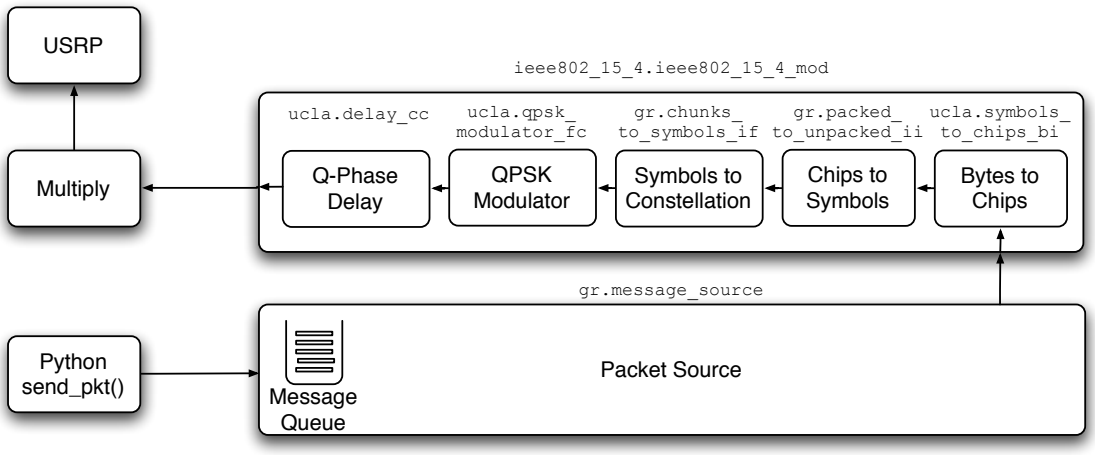


Figure 3.4: IEEE 802.15.4 modulation in GNU Radio [4]

mapped to the constellation points according to $0 \mapsto -1$ and $1 \mapsto 1$.
The generated sequence of -1 and 1 gets into the QPSK modulator which outputs the complex baseband QPSK signal. To generate the O-QPSK signal, the Q-Phase is delayed by two samples. Finally before the signal is sent to the USRP, it gets multiplied by the factor 8000 to scale it to the full range of the 14-bit DAC. In the USRP, the signal will be modulated onto the defined carrier frequency f_c.
The demodulation process at the receiver side is illustrated in Figure 3.5. In order to avoid unnecessary attempts to decode noise, the received signal is first passed through a squelch filter that passes only signals which have a certain dB strength. The "FM Demod" box decodes the MSK signal and a Mueller and Müller discrete-time error-tracking synchronizer recovers the clock. The last step in the receiving path is the packet sink which is implemented in the C++ object *ucla.iee802_15_4_packet_sink*. This object stands for the physical frame detector and with the help of the packet length field decodes the whole MPDU. Once a complete MPDU is found, it is added to a message queue. In the original implementation the packet sink does not allow any errors in the decoding process, therefore it needed to find all four 0x00 preamble bytes. This has been changed in our implementation where the sender synchronizes on a packet, even if less then four preamble bytes were detected. This improved the reception rate considerably and is discussed in more detail in Chapter 4.
The message queue is observed by an external python thread in the packet sink. This thread starts, as soon as there are messages queuing and starts to call a callback function which process the MPDU further.

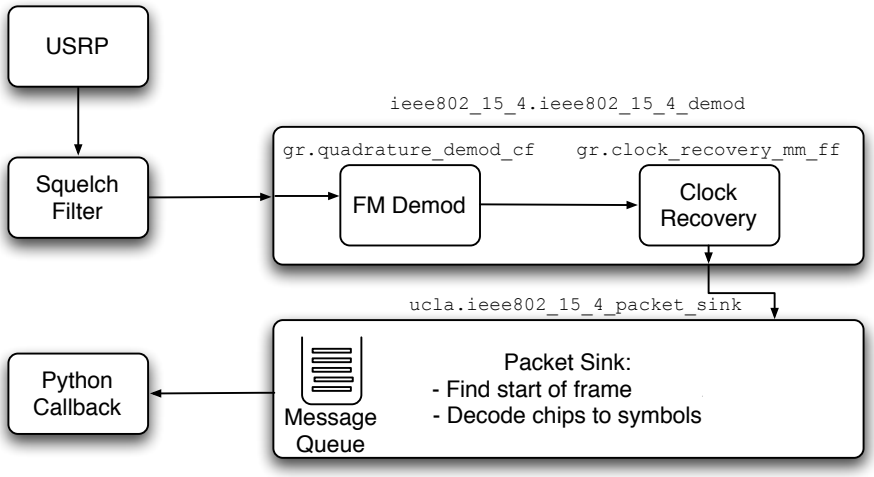


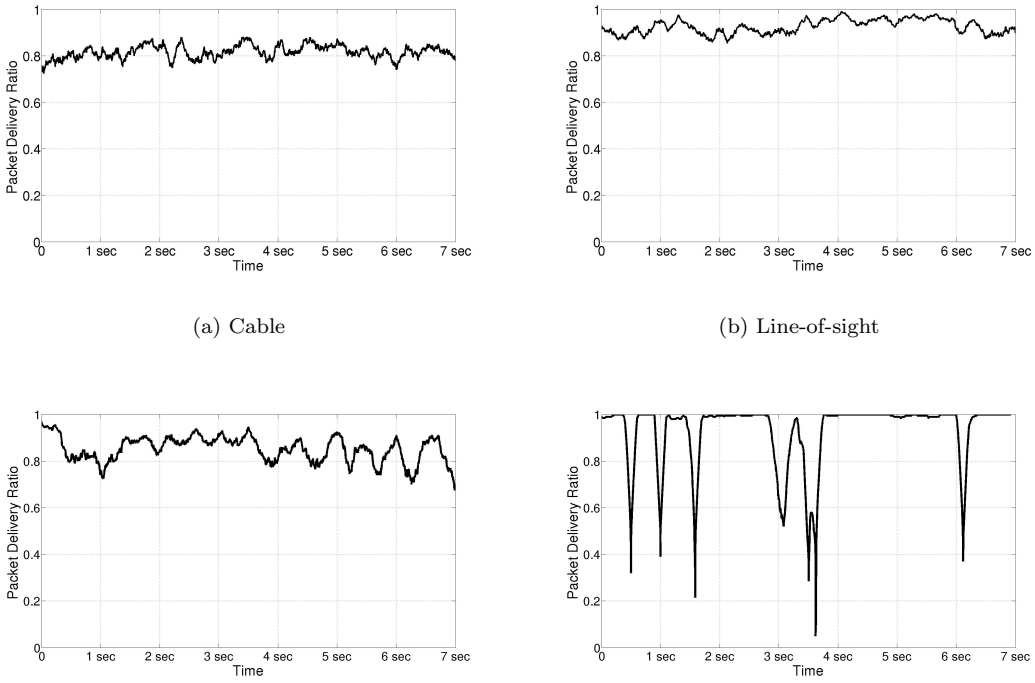
Figure 3.5: IEEE 802.15.4 demodulation in GNU Radio [4]

3.3 Scenarios

3.3.1 Link Quality Estimation

In order to capture effects of different channel conditions such as fading, shadowing, or mobility on link quality estimation, we set up our testbed in four different types of connectivity which are described in detail below.
A least biased and environment independent scenario is the attenuated cable connection. Even our investigations are not focussing on static wired links, it serves as a most reproducible reference model excluding wireless channel phenomenons. Further we investigated experiments having an indoor line-of-sight and non-line-of-sight link between sender and receiver. Various fading effects are affecting the data transmission. Large-scale fading as path loss and shadowing are realized by varying the transmit power and blocking the line-of-sight path by obstacles. Small-scale fading is achieved by an indoor office environment where reflections from walls, the floor, the ceiling or other objects cause multi-path propagation where delayed replicas of the sent signals arrive at the receiver. As a last, for link quality estimators most challenging scenario, an indoor mobile scenario was set up. A lot of LQE tend to fail in terms of an accurate short term estimation of the link at the point where the relative speed of the sender and the receiver cause an unpredictable and fast changing behavior of the channel.Figure 3.6 gives an impression of the different channel behavior in the four scenarios at a certain transmit power in a time interval of seven seconds.

To evaluate the LQEs in the whole range of channel qualities, i.e. from very poor channels of a PDR of about 5% to perfect channels of PDR of 100%, the transmit power of the sender was adjusted. The packets were transmitted on channel 16 ($f_c = 2.43$ GHz) at 250 kbit/s, a common data rate of IEEE 802.15.4 networks [2].
To gain the information to implement different LQEs based on various link metrics, e.g. signal strength, chip errors or number of received symbols, every sent and received symbol was logged on a chip and symbol level with a corresponding time stamp. At the sender side the information about the sent symbols were obtained in the "symbols-to-chip" block of the modified UCLA ZigBee implementation, that can be found in Figure 3.4. Correspondingly, the receiver logged



(a) Cable

(b) Line-of-sight

(c) Non-line-of-sight

(d) Mobile

Figure 3.6: Typical measured behavior of the packet delivery ratio over time for a certain transmit power

the information on the chip and symbol level in the packet sink block as well a step before in the demodulation block to build the digital SNR metric.Figure 3.1 shows in the symbol-to-chip and chip-to-symbol boxes where we tap chip-level information.
The sender and the receiver code are executed on separate host computers which are connected to the USRP hardware radio frontends through GigE connection or USB 2.0 links.

Attenuated Cable Connection

The sender and the receiver were connected by a shielded 60 cm coaxial cable with a 30dB attenuator as depicted in Figure 3.7 in box (a).

Indoor Line-of-Sight

This indoor deployment with line-of-sight (LOS) connectivity and omni-directional antennas (Ettus VERT 2450 [3]). The devices were placed on a table two meters apart from each other with perpendicular oriented antennas with respect to the table area. The nodes were static during the experiments and only moved between different measurement series. Different link condition are emulated by changing the transmit power of the sender.

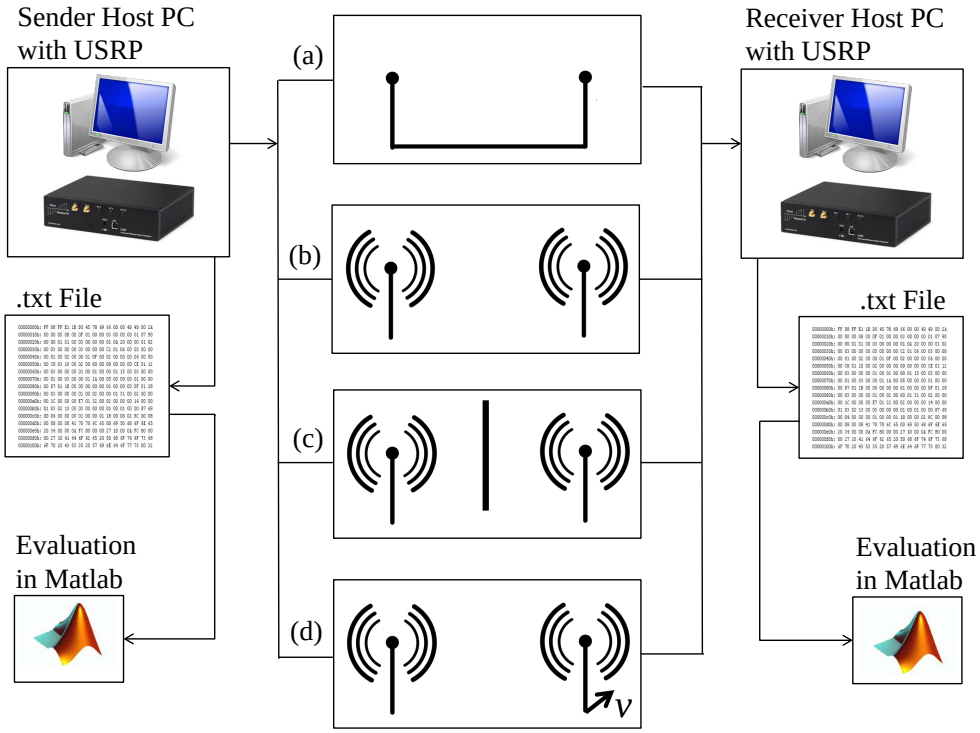


Figure 3.7: Experimental setup: The transmitted packets were generated at the sender PC and send via the USRP to the receiver under four different link conditions: (a) cable, (b) line-of-sight, (c) non-line-of-sight, and (d) mobile. The received packets were recorded on chip and symbol level and written to a text file. The evaluation of the logged data was done with Matlab routines.

Indoor Non-Line-of-Sight

The third scenario is identical to the indoor LOS setting beside that the direct LOS component is blocked by a heavy metal cupboard. The nodes were as well static during the transmissions.

Indoor Mobile

By moving the receiver node, we introduce a faster changing channel characteristic. The sender is placed again on the table and the receiver is on another table equipped with wheels which has been moved at a constant speed.
As this setup is not a static scenario, it is assumed that the PDR will vary during a measurement run. In order to meet the derived requirements for meaningful evaluations from Section 2.1.1, the measurement run was divided into slices of the length of the coherence time T_c and the receiver velocity was set to an appropriate value further discussed in Chapter 5.

3.3.2 Jamming Detection Estimation

In contrast to the previous link quality estimation scenarios we need for the jamming detection estimation setup a third node. This new node will be used to influence the data transmission of the other two nodes. The jamming detection measurement runs have been conducted under indoor office NLOS conditions. Sender and receiver were placed seven meters away from each other. The LOS path was blocked by table boards. During the experiment non of the nodes were moved. The different link conditions were again realized by adjusting the transmit power of the sender. We analyzed two different types of jammers:

- **Continuous reactive jammer**. The goal of this jammer is to destroy every packet that is sent from the sender to the receiver node. As the jammer targets the SFD (start of frame delimiter) and therefore prevents the receiver from successfully decoding the synchronization header - the receiver is not even able to synchronize on any sent packet.
- **Random reactive jammer**. The random reactive jammer destroys a packet with a certain probability and generates with his jamming rate between 80% and 100% PDRs of 6%. This jammer targets as well the SFD and prevents the sender as well from synchronizing on the most packets.

To realize such sophisticated jamming mechanisms, we use a system called WiFire, proposed by Matthias Wilhelm et al [57]. WiFire detects and analyzes packets during their transmission, checking their contents against a set of rules. It relies on reactive jamming techniques to selectively block undesired communication. It is implemented on the USRP2 software defined platform for IEEE 802.15.4 radios.
As a measurement of the performance of the jamming detection estimation, we evaluated the

$$\text{false negatives} = P(\text{"Guess Jamming off"} \mid \text{"Jamming on"})$$

and the

$$\text{false positives} = P(\text{"Guess Jamming on"} \mid \text{"Jamming off"}).$$

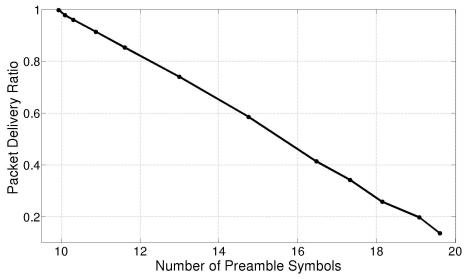
4

Estimator Design based on Preamble Symbols

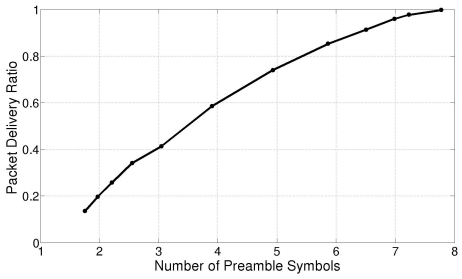
4.1 Exploration of Preamble Symbols

4.1.1 Exploration of Preamble Symbol Occurrence

To start with the simplest approach to explore the behavior of the preamble symbols we focus on the analysis of the number of decoded preamble symbols. We suggest the assumption that the number of decoded preamble symbols correlates with the PDR. I.e. receiving a small number of preamble symbols indicates a poor link quality, whether the reception of all eight preambles would provide an indication of a very good link.
As a first approach we analyzed how many preambles per successfully decoded packet were detected at the receiver in average. The results based on the attenuated cable setup are illustrated in Figure 4.1(a) in case of a cable connection. The horizontal axis shows the number of obtained preambles for a decoded packet. This almost linear relation between the PDR and the number of detected preambles could be a promising basis of a LQE. Note that this curve shows a shift of about two symbols to the right. At the point of a PDR of 100% we observe 10 decoded preambles at the receiver. But there are only eight preambles transmitted according to the IEEE 802.15.4 standard. This additional two symbols occur because of accidentally decoded preamble symbols on noise during two consecutively sent packets.
Figure 4.1(b) shows the correlation on a per packet level. There we count the average number of consecutive decoded preambles before a packet got successfully decoded or the synchronization was lost. This correlation as well shows nice linear behavior and it is assumed that even on a packet level there could be made some meaningful predictions of the link quality.



(a) Average number of decoded preamble symbols per successfully delivered packet

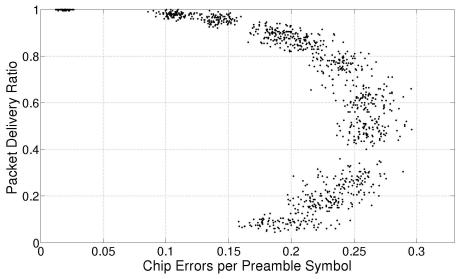


(b) Average number of consecutive decoded preamble symbols per packet

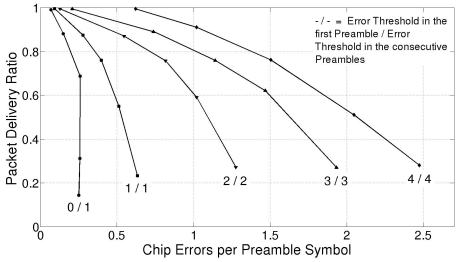
Figure 4.1: Cable Connection: Number of received preamble symbols (a) per successfully decoded packet and (b) per sent packet.

4.1.2 Exploration of Preamble Chip Errors

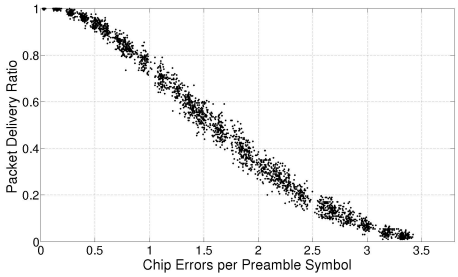
Analyzing the chip errors per symbol (CEPS) as shown in [5] for a random generated payload turned out to be a promising approach to estimate the link quality. The observed linear reciprocal dependence of the CEPS to the PDR served as the basis for a fast and accurate LQE. Assuming the same behavior in the synchronization part of a packet, we investigated similar measurements on the preamble symbols.
As a first measurement we evaluated the correlation of the chip errors per preamble symbol (CEPPS) to the PDR in the cable scenario. The results are shown in Figure 4.2(a). Every point belongs to the mean over a window of 200 sent packets. The curve shows an interesting behavior: the maximum is reached at a PDR of about 50%. For even worse links the chip error rate decreases again. This peculiarity is due to the implementation of the receiver in the original implementation. If the receiver is in the search state waiting for the first preamble symbol, only preambles which were decoded with a zero chip error tolerance were accepted. Contrary, consecutive symbols after the first one were accepted with one chip error per symbol. As for poor-quality links a loss of synchronization is more probable, the receiver spends more time in the search state and is therefore decoding using a zero error tolerance. This causes that the obtained chip errors per preamble symbol decrease again for bad links.
For the design of a LQE that guesses the PDR based on the observed CEPPS, this resulting sickle shaped curve is undesirable. To realize more promising dependency of the CEPPS on the PDR, we varied the error tolerance for the preamble as illustrated in Figure 4.2(b). Note that this originally implemented error threshold in the UCLA framework is not defined by the IEEE 802.15.4. However this value was set to any value and was not optimized so far considering e.g. data rates. For this purpose the chip error threshold has been changed between zero and four. The benefit of increasing the chip error tolerance is the greatly enlarged interval of CEPPS for the whole range of link qualities. The maximum of the CEPPS interval is obtained for an error threshold of four. Further increases of result only in shifts of the curve to larger error values and not in a bigger interval of CEPPS.
Remarkable as well is the increase of the system performance. Especially for poor-quality links we obtain a huge increase in the PDR of almost 100% from zero to one error. In the range of



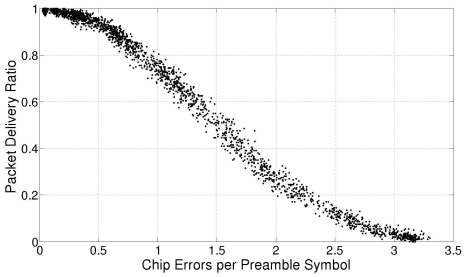
(a) First evaluation of the correlation between the PDR and the CEPPS in the cable scenario. Due to the original implementation of the decoder, the curve is bent in a sickle shape that makes a unique mapping of the CEPPS to the PDR impossible.



(b) Modification of the decoder chip error threshold. The error tolerance of originally 0 chip errors in the first preamble and 1 chip error in the consecutive preambles (0/1) has been modified to 1/1, 2/2, 3/3, and 4/4.



(c) Resulting correlation of the cable scenario using a constant chip error tolerance of 4 chip errors in all preamble symbols.



(d) Resulting correlation in a wireless scenario using a constant chip error tolerance of 4 chip errors in all preamble symbols.

Figure 4.2: Modification of the decoder: The originally implemented error tolerance (in the UCLA framework) of 0 chip errors in the first decoded preamble and 1 chip error in the consecutive preambles (a) has been changed (b) in order to obtain a correlation that allows unique mappings from CEPPSs to PDRs, (c) and (d). Note that the originally implemented error threshold (0/1) is not defined by IEEE 802.15.4. However this error threshold has been optimized in our work that leads to a suitable correlation curve and to higher possible data rates. Further we didn't notice any drawback caused by these settings.

one to four chip errors, the PDR does not change considerably.

4.2 Link Quality Estimation

To derive benefits from the findings in the last section, we describe different approaches of LQEs separated into two groups: Fast link quality estimation and ultra-fast link quality estimation. Different approaches are proposed based on the number of preambles, chip errors, and hybrid

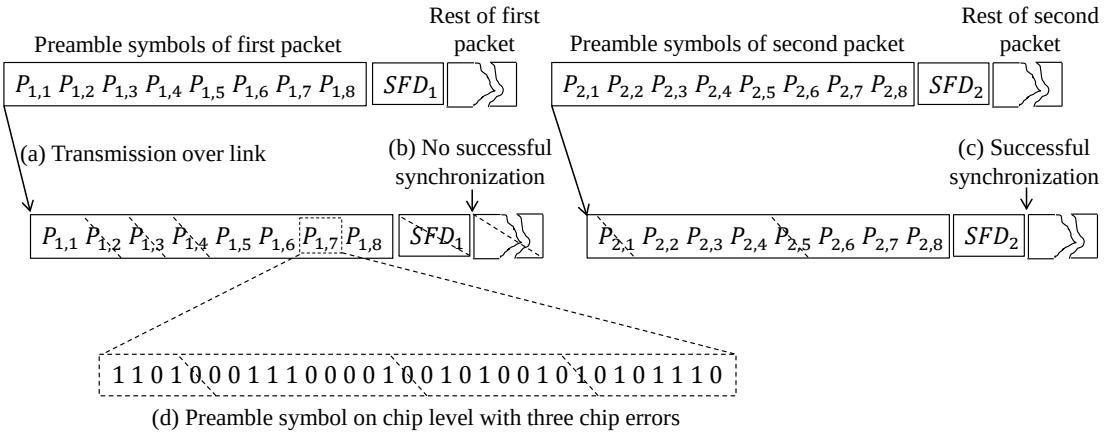


Figure 4.3: Data transmission of two packets: (a) the sender starts to transmit the preamble sequence, the SFD (start of frame delimiter) and the corresponding part of the packet (named here as the rest of packet). During the transmission of the eight preamble symbols of the first packet, $P_{1,2}$, $P_{1,3}$, $P_{1,4}$ could not be decoded correctly due to too much chip errors. E.g. $P_{1,7}$ was transmitted successfully because as shown in (d) only three chips were flipped during the transmission and the maximum error threshold of four chip errors is not exceeded. Due to a corrupted bit in the SFD_1 the synchronization of the first packet fails and the receiver is not able to decode this packet. Contrary to the first packet, the second packet is transmitted successfully (c) and only the preambles $P_{2,1}$ and $P_{2,5}$ were lost.

implementations that combine both concepts. An overview of all LQEs can be found in Table 4.1. Note that the following correlation curves in this section between the different metrics and the PDR are measured for all scenarios and each time shown in the same figure (see receptive legends). Every correlation point for a specific PDR is then averaged over a whole measurement run of 40'000 sent packets that correspond to a measurement time of 40 seconds each point. In case of the mobile setting, the average was each calculated over one coherence time T_c. The environments were the same as described in Chapter 3. By changing the respective USRP devices we obtained no changes in the curves. Moreover a change of the measurement room did not affect the curve. It just has to be considered as already described in the last section, that the decoder error threshold changes the course of the curve and need to be unchanged after the calibration.

4.2.1 Ultra-fast Link Quality Estimation

The design goal of the ultra-fast link quality estimation is to realize an estimation within a short amount of time. The idea behind this ultra-fast estimation is to set the estimation window w to a small as possible value. To meet this requirement, we evaluate the first received symbols of a packet, i.e. the preambles. This results then in a maximum estimation window w of eight preambles[1] and a minimal w of one preamble. Note that even for a lost packet, where e.g. the synchronization failed or a bit error occurred during the transmission, this LQE works on a

[1]That corresponds in our test bed to reaction time of 0.26 ms

per-packet level and doesn't rely on a successful packet transmission.
Figure 4.3 illustrates schematically the transmission of two packets, where the first packet is lost due to a synchronization error and the second packet is transmitted successfully. The ultra-fast estimator makes a first estimation of the link quality at the point (b) where the synchronization fails based on the sent preambles $P_{1,1}$, $P_{1,2}$, $P_{1,3}$, $P_{1,4}$, $P_{1,5}$, $P_{1,6}$, $P_{1,7}$, $P_{1,8}$ as

$$\widehat{PDR_1} = ultrafast(\{P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4}, P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8}\}),$$

and a second estimation at (c) after receiving the preambles of the second packet $P_{2,1}$, $P_{2,2}$, $P_{2,3}$, $P_{2,4}$, $P_{2,5}$, $P_{2,6}$,$P_{2,7}$,$P_{2,8}$:

$$\widehat{PDR_2} = ultrafast(\{P_{2,1}, P_{2,2}, P_{2,3}, P_{2,4}, P_{2,5}, P_{2,6}, P_{2,7}, P_{2,8}\}).$$

Or more generally let $\mathcal{S}_k$ denote the set of all *correctly decoded* preamble symbols in the k-th packet with the preamble numbering j of the elements in $\mathcal{S}_k$ as $j = 1, 2, \ldots, 8$. Then the PDR estimation is computed as

$$\widehat{PDR_k} = ultrafast(\{P_{k,1}, P_{k,2}, P_{k,3}, P_{k,4}, P_{k,5}, P_{k,6}, P_{k,7}, P_{k,8}\}).$$

The function $ultrafast(\cdot)$ is not specified so far but will be discussed in the next two small sections. First based on the occurrence of the preamble symbols and second, based on chip errors.

Ultra-fast Link Quality Estimation based on the Occurrence of Preambles

In Section 4.1 we obtained a correlation between the occurred number of preamble symbols $|\mathcal{S}|$ per packet and the PDR. The metric mapping function $ultrafast(\cdot)$ is defined as $ultrafast(\cdot) := count(|\cdot|)$. Where the function $count(\cdot)$ simply maps the number of decoded preamble symbols to the PDR, and $|\cdot|$ denotes the cardinality. Note that preambles who could not be detected due to transmission errors (e.g. $P_{1,2}$, $P_{1,3}$, $P_{1,4}$ of the first packet or $P_{2,2}$, $P_{2,5}$ of the second packet) are considered as empty sets and cannot be used in the estimation process.
According to the example in Figure 4.3 we can calculate the PDR based on the first packet as

$$PDR_{\,Preamble\ Count\ Ultra\text{-}fast,\ 1} = ultrafast(\{P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4}, P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8}\}),$$

and therefore

$$PDR_{\,Preamble\ Count\ Ultra\text{-}fast,\ 1} = count(|\{P_{1,1}, P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8}\}|) = count(5).$$

For the second packet it follows

$$PDR_{\,Preamble\ Count\ Ultra\text{-}fast,\ 2} = ultrafast(\{P_{2,1}, P_{2,2}, P_{2,3}, P_{2,4}, P_{2,5}, P_{2,6}, P_{2,7}, P_{2,8}\}),$$

and

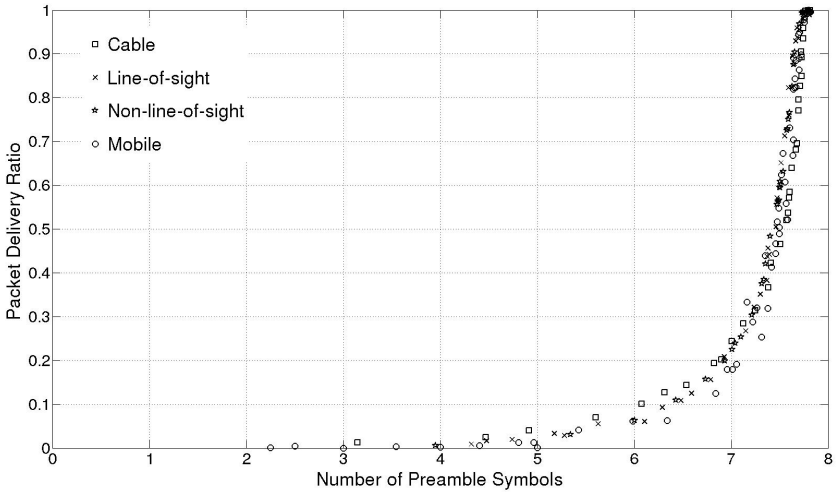


Figure 4.4: Ultra-fast estimation: Correlation of the observed number of preamble symbols and the packet delivery ratio for the cable, line-of-sight, non-line-of-sight, and the mobile scenario.

$$PDR_{\,Preamble\ Count\ Ultra\text{-}fast,\,2} = count(|\{P_{2,2}, P_{2,3}, P_{2,4}, P_{2,6}, P_{2,7}, P_{2,8}\}|) = count(6).$$

Or more generally

$$PDR_{\,Preamble\ Count\ Ultra\text{-}fast,\,k} = count(|\mathcal{S}_k|).$$

The measured correlation of the four different scenarios are depicted in Figure 4.4. Note that this curve shows a different shape compared to Figure 4.1(b). This es due to the modification of the receiver chip error threshold that has been increased. The fit of the measured data points is shown in Figure 4.5. In order to find a reasonable trade-off between the goodness of the fit and the computational overhead, a 5th degree rational fit has been used for the approximation. The mapping from the number of preamble symbols $|\mathcal{S}_k|$ to the respective PDR is

$$count(p) = \frac{\mathrm{p}_0\ p^5 + \mathrm{p}_1\ p^4 + \mathrm{p}_2\ p^3 + \mathrm{p}_3\ p^2 + \mathrm{p}_4\ p + \mathrm{p}_5}{p^5 + \mathrm{q}_0\ p^4 + \mathrm{q}_1\ p^3 + \mathrm{q}_2\ p^2 + \mathrm{q}_3\ p + \mathrm{q}_4}.$$

The polynomial coefficients can be found in Table C.1.

This type of LQE is later referred to as *Preamble Count Ultra-fast*.

Ultra-fast Link Quality Estimation based on Preamble Chip Errors

In the case of a ultra-fast link quality estimator based on chip errors the metric function $ultrafast(\cdot)$ is defined arbitrary for the k-th received packet as

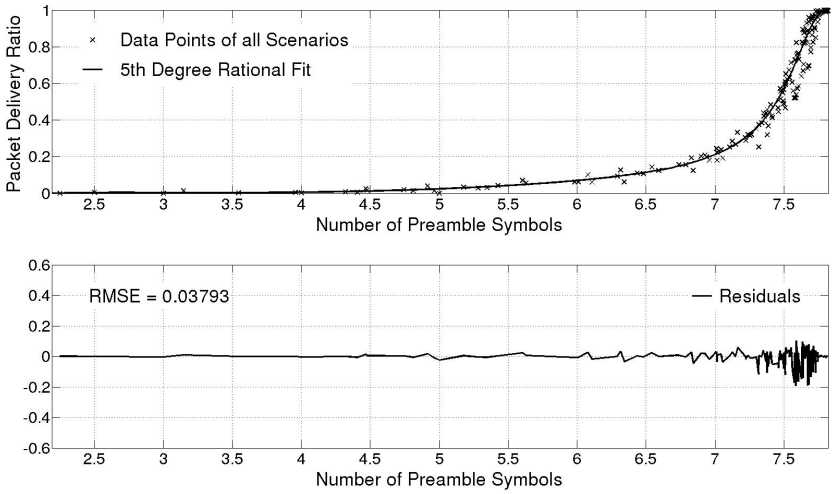


Figure 4.5: 5th degree rational fit for the ultra-fast LQE based on the number of preamble symbols per packet.

$$PDR_{\,CEPPS\ Ultra\text{-}fast,\ k} = ultrafast(\{P_{k,1}, P_{k,2}, P_{k,3}, P_{k,4}, P_{k,5}, P_{k,6}, P_{k,7}, P_{k,8}\}) =$$

$$chiperror\left(\frac{\sum_{i=0}^{31}\sum_{j\in\mathcal{S}_k}\left(P_{k,j}[i]\oplus P[i]\right)}{|\mathcal{S}_k|}\right).$$

Where $P_{k,j}[i]$ is a row vector containing the 32 chips of the j-th received preamble symbol of the k-th packet for $i = 0, 1, 2, \ldots, 31$. $P[i]$ denotes the know correct preamble symbol. Therefore the error vector $e_{k,j}[i]$ of every received preamble symbol can be calculated as $e_{k,j}[i] = P_{k,j}[i] \oplus P[i]$, where $\oplus$ is the exclusive or and $e_{k,j}[i] = 1$ if and only if the i-th chip is false. This becomes more clear if we write this out for the first transmitted packet in our example:

$$PDR_{\,CEPPS\ Ultra\text{-}fast,\ 1} = ultrafast(\{P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4}, P_{1,5}, P_{1,6}, P_{1,7}, P_{,8}\}) =$$

$$chiperror\left(\frac{\sum_{i=0}^{31}\left(P_{1,1}[i]\oplus P[i] + P_{1,5}[i]\oplus P[i] + P_{1,6}[i]\oplus P[i] + P_{1,7}[i]\oplus P[i] + P_{1,8}[i]\oplus P[i]\right)}{|\{P_{1,1}, P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8}\}|}\right).$$

With these definitions the input argument of $chiperror(\cdot)$ is simply the average chip errors per preamble symbol of the considered preambles $P_{1,1}, P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8}$.

And in the same notation for the second packet the packet delivery ratio is calculated as

$$PDR_{\,CEPPS\ Ultra\text{-}fast,\ 2} = ultrafast(\{P_{2,1}, P_{2,2}, P_{2,3}, P_{2,4}, P_{2,5}, P_{2,6}, P_{2,7}, P_{2,8}\}) =$$

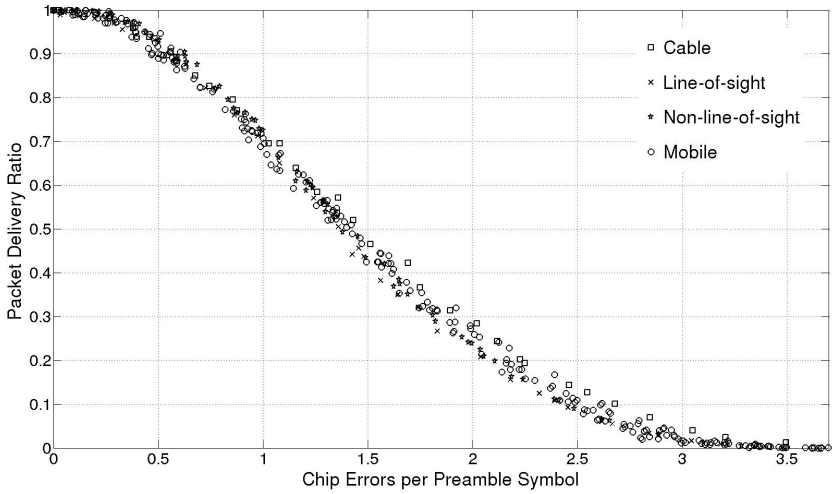


Figure 4.6: Correlation of the observed chip errors per preamble symbol and the packet delivery ratio for the cable, line-of-sight, non-line-of-sight, and the mobile scenario.

$$chiperror\left(\frac{\sum_{i=0}^{31}\left(P_{2,2}[i]\oplus P[i]+P_{2,3}[i]\oplus P[i]+P_{2,4}[i]\oplus P[i]+P_{2,6}[i]\oplus P[i]+P_{2,7}[i]\oplus P[i]+P_{2,8}[i]\oplus P[i]\right)}{|\{P_{2,2},P_{2,3},P_{2,4},P_{2,6},P_{2,7},P_{2,8}\}|}\right).$$

To derive the mapping function of the chip error based LQE we need to fit the measured correlation between the CEPPS and the PDR for all four link connections. Figure 4.6 shows this curve. Because the correlation curve behaves like a simple polynomial, we approximated the data points with a 5th degree polynomial,

$$chiperror(p) = \mathrm{p}_0\ p^5 + \mathrm{p}_1\ p^4 + \mathrm{p}_2\ p^3 + \mathrm{p}_3\ p^2 + \mathrm{p}_4\ p + \mathrm{p}_5.$$

Again the precise values of the polynomial coefficients can be looked up in Table C.2.
Figure 4.8 shows how the absolute estimation error of this LQE behaves as a function of time.

This type of LQE is later referred to as *CEPPS Ultra-fast* (Chip Error per Preamble Symbol Ultra-fast).

Ultra-fast Link Quality Estimation based on Filtered and Weighted Average of Preamble Chip Errors

This modification of the above introduced LQE is motivated to increase the stability of the single metric ultra-fast LQE (see Figure 4.8). As the concept of this new statistical strategy relies as well on the fast link quality estimation approach, we will explain it in detail in the next section about fast link quality estimation. However the very first estimation of the link quality $PDR_{\,CEPPS\ FWA,\,1}$, is the same as for *CEPPS Ultra-fast*. Therefore this estimator provides an estimation with the same reaction time and in therefore categorized and listed here as ultra-fast

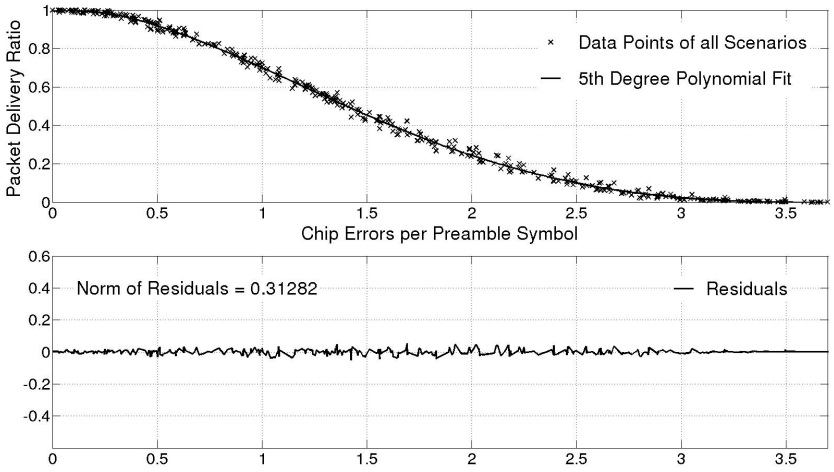


Figure 4.7: 5th degree rational fit for the ultra-fast LQE based on chip errors per preamble symbol

LQE.

This type of LQE is later referred to as *CEPPS FWA* (Chip Error per Preamble Symbol Filtered Weighted Average).

Ultra-fast Link Quality Estimation based on Estimator Variance

As we will see in Chapter 5, the chip error based LQEs have their worst performance at PDRs of about 50% and their minimal errors towards 100% and 0%. Therefore it is assumed that the variance of these estimators is symmetrically distributed around 50% and can thus be used as well to estimate the PDR. For this purpose we analyzed the second moment of the mentioned chip error based estimator. The correlation of this variance and the PDR is illustrated in Figure 4.9. Note that based on this correlation characteristics it is not possible to map a fixed variance to a unique PDR. From this perspective a case differentiation has to be done to decide whether the estimation region is below the vertex and the dashed fit has to be used or the estimation region is above the vertex and the solid line fit has to be used. This information is provided by the estimation of the chip error based LQE. Note that variances exceeding the vertical limit subtending the vertex cannot be allocated to any fit and are therefore in the estimation algorithm not used.
After the reception of the preambles of the first packet, the metric function $ultrafast(\cdot)$ is defined in this case as

$$ultrafast(\{P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4}, P_{1,5}, P_{1,6}, P_{1,7}, P_{,8}\}) := variance(\mathrm{var}_1),$$

where

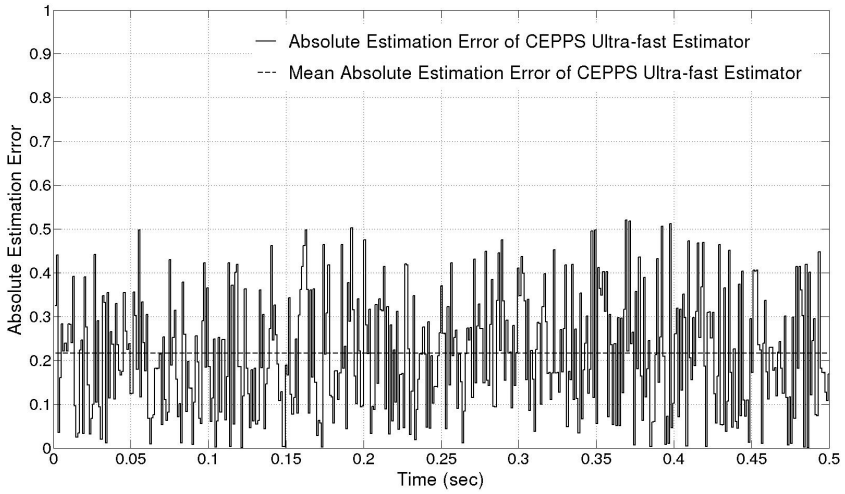


Figure 4.8: Ultra-fast preamble chip error based estimator: Error tracking of the absolute estimation over sequential estimations. This example is taken from the line-of-sight measurement with a PDR of 50%. There the chip error based estimators show the largest error variance. To increase the stability, i.e. to reduce the fluctuations we propose a filtered and weighted average estimator modification.

$$\mathrm{var}_1 = \mathrm{Var}\left(\mathit{chiperror}\left(\frac{\sum_{i=0}^{31} (P_{1,1}[i] \oplus P[i])}{|\{P_{1,1}\}|} \right), \ldots, \mathit{chiperror}\left(\frac{\sum_{i=0}^{31} (P_{1,8}[i] \oplus P[i])}{|\{P_{1,8}\}|} \right) \right),$$

and by inserting 1 for the cardinality of the one element sets we obtain

$$\mathrm{var}_1 = \mathrm{Var}\left(\mathit{chiperror}\left(\sum_{i=0}^{31} (P_{1,1}[i] \oplus P[i]) \right), \ldots, \mathit{chiperror}\left(\sum_{i=0}^{31} (P_{1,8}[i] \oplus P[i]) \right) \right).$$

with $\mathrm{Var}(X) = \mathrm{E}\left[(X - \mathrm{E}[X])^2 \right]$ where X is a random variable and has the mean $\mathrm{E}[X]$. The function $\mathit{variance}(\cdot)$ maps then finally var_1 to the respective PDR according to Figure 4.9.
The final guess is then calculated as the mean of the first and second order estimation values. If var_1 exceeds the maximum value that can be used, the final link quality estimation is then simply the first order estimation.
The extension of this approach to the k-th received packet is

$$\mathit{ultrafast}(\{P_{k,1}, P_{k,2}, P_{k,3}, P_{k,4}, P_{k,5}, P_{k,6}, P_{k,7}, P_{k,8}\}) = \mathit{variance}(\mathrm{var}_k),$$

with

$$\mathrm{var}_k = \mathrm{Var}\left(\mathit{chiperror}\left(\sum_{i=0}^{31} (P_{k,1}[i] \oplus P[i]) \right), \ldots, \mathit{chiperror}\left(\sum_{i=0}^{31} (P_{k,\max j \in \mathcal{S}_k}[i] \oplus P[i]) \right) \right).$$

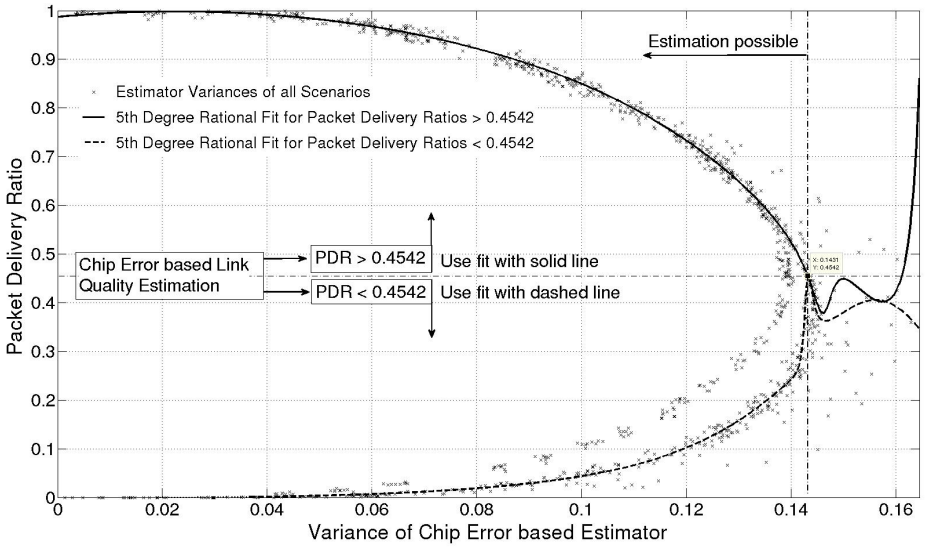


Figure 4.9: Correlation of the variance according to the chip error based LQE and the packet delivery ratio. Because the merged two fitting curves don't allow an unique mapping from a fixed variance to a PDR, a case differentiation has to be done to know if the solid line fit or the dashed line fit has to be used. This information is provided by the estimation of the chip error based LQE. Is this estimation above the vertex of the two fits, the lower fit is used, if it is below this value the upper fit is used. Note that variances exceeding the vertical limit subtending the vertex cannot be allocated to any fit and can therefore not be used.

This type of LQE is later referred to as *Variance Ultra-fast*.

Hybrid ultra-fast Link Quality Estimation

In order to combine the different metrics to cope with the weaknesses of single metric LQEs, we introduce an ultra-fast hybrid LQE based on the concepts of counting the preamble symbols and on chip errors. The resulting estimation is simply the mean of the two estimators. This is calculated as

$$PDR_{\,Hybrid\ Ultra\text{-}fast} = \frac{1}{2}(PDR_{\,CEPPS\ Ultra\text{-}fast} + PDR_{\,Preamble\ Count\ Ultra\text{-}fast}).$$

This type of LQE is later referred to as *Hybrid Ultra-fast*.

4.2.2 Fast Link Quality Estimation

Figure 4.10 shows the mean absolute estimation error based on $chiperror(\cdot)$ as a function of the number of considered preamble symbols. This motivates to develop a LQE, that analyzes even more than eight preambles (as in the best case of *CEPPS Ultra-fast*) as the error still decreases

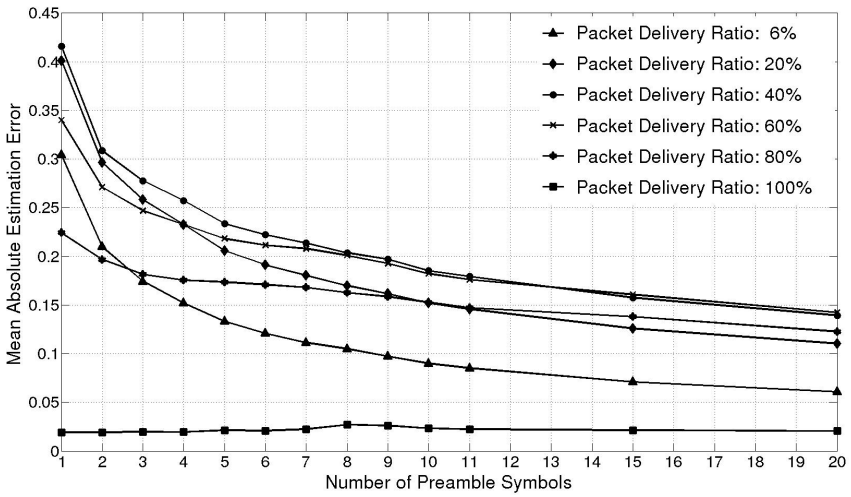


Figure 4.10: Line-of-sight link: Mean absolute estimation error based on $chiperror(\cdot)$ as a function of the number of considered preamble symbols. An increase of the estimation window (number of preamble symbols) results in a smaller estimation error.

after eight symbols. Therefore for the fast LQE we will increase the observation window w to obtain a LQE that is more accurate at the cost of the estimator reactivity.
To explain the concepts of the fast LQEs we rely again on the example of Figure 4.3.
As the ultra-fast estimator set his estimation window w to the preamble group of a packet, no matter if there happened a successful synchronization after the last preamble. However the rule of the fast LQEs is that they extend their observation window w until a packet got successfully decoded. Therefore the PDR is estimated based on the received preambles $P_{1,1}$, $P_{1,2}$,$P_{1,3}$,$P_{1,4}$,$P_{1,5}$, $P_{1,6}$, $P_{1,7}$, $P_{1,8}$ of the first packet and on the preambles of the second packet $P_{2,1}$,$P_{2,2}$, $P_{2,3}$, $P_{2,4}$, $P_{2,5}$,$P_{2,6}$,$P_{2,7}$,$P_{2,8}$. It follows

$$\widehat{PDR_1} = fast(\{P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4}, P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8}, P_{2,1}, P_{2,2}, P_{2,3}, P_{2,4}, P_{2,5}, P_{2,6}, P_{2,7}, P_{2,8}\}).$$

It is clear that the estimation window w converges to the one of the ultra-fast estimation window for PDRs towards 100%. To keep the consistency of our before introduced notation let k denote the k-th successfully decoded packet and let the set $\mathcal{S}_k$ be all decoded preamble symbols between the k-th and the $(k-1)$-th successfully decoded packet with the preamble numbering j of the elements in $\mathcal{S}_k$ as $j = 1, 2, \ldots, \max j \in \mathcal{S}_k$. This leads to

$$\widehat{PDR_k} = fast(\{P_{k,1}, P_{k,2}, \ldots, P_{k,\max_j \in \mathcal{S}_k}\}).$$

Fast Link Quality Estimation based on the Occurrence of Preambles

As in the previous case of the ultra-fast estimation we define the metric mapping function $fast(\cdot)$ as $fast(\cdot) := count(|\cdot|)$. Note that this function $count(\cdot)$ differs from the one obtained in the ultra-fast approach. This becomes clear if we consider the resulting correlation between the

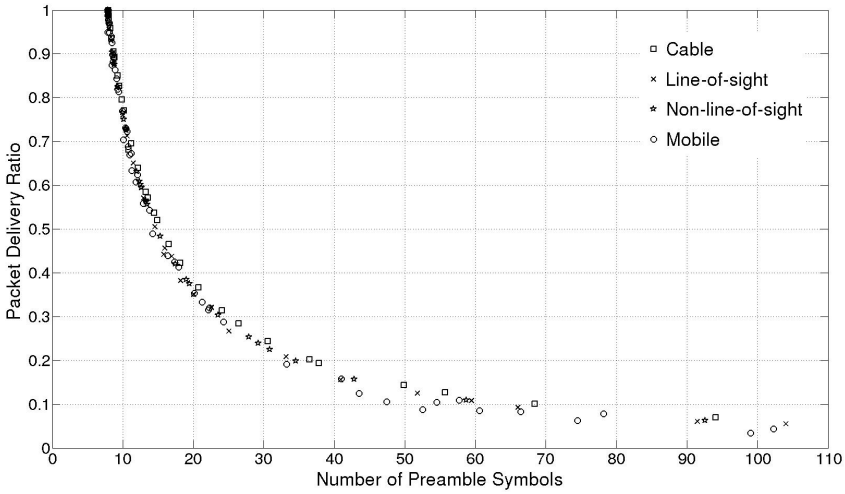


Figure 4.11: Fast estimation: Correlation of the observed number of preamble symbols and the packet delivery ratio for the cable, line-of-sight, non-line-of-sight, and the mobile scenario.

occurred numbers of preamble symbols and the PDR as shown in Figure 4.11. The mapping function $count(\cdot)$ is shown in Figure 4.12 as a 5th degree rational approximation,

$$count(p) = \frac{\mathrm{p}_0\ p^5 + \mathrm{p}_1\ p^4 + \mathrm{p}_2\ p^3 + \mathrm{p}_3\ p^2 + \mathrm{p}_4\ p + \mathrm{p}_5}{p^5 + \mathrm{q}_0\ p^4 + \mathrm{q}_1\ p^3 + \mathrm{q}_2\ p^2 + \mathrm{q}_3\ p + \mathrm{q}_4}.$$

The values of the coefficients are appended in Table C.3.

The estimation of the PDR holds

$$PDR_{\ Preamble\ Count\ Fast,\ 1} = fast(\{P_{1,1}, P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8}, P_{2,2}, P_{2,3}, P_{2,4}, P_{2,6}, P_{2,7}, P_{2,8}\}),$$

and thus

$$\begin{aligned} PDR_{\ Preamble\ Count\ Fast,\ 1} &= count(|\{P_{1,1}, P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8}, P_{2,2}, P_{2,3}, P_{2,4}, P_{2,6}, P_{2,7}, P_{2,8}\}|) \\ &= count(11). \end{aligned}$$

Or more generally

$$PDR_{\ Preamble\ Count\ Fast,\ k} = count(|\mathcal{S}_k|).$$

This type of LQE is later referred to as *Preamble Count Fast*.

Fast Link Quality Estimation based on Preamble Chip Errors

In the case of the fast link quality estimation based on chip errors we do not need do define a new metric function $chiperror(\cdot)$ because since we only average over a larger observation window w of

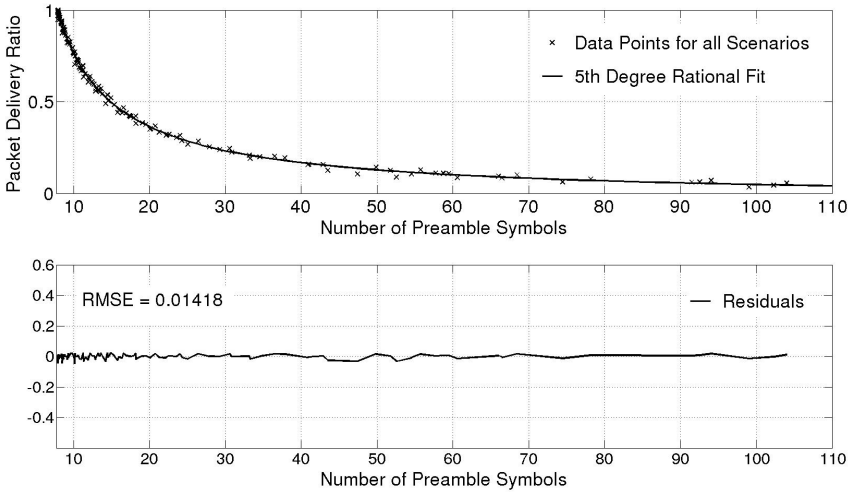


Figure 4.12: 5th degree rational fit for the fast LQE based on the number of preamble symbols per packet.

preamble symbols. The correlation between the chip error and the PDR as depicted in Figure 4.7 does not change. With the extension of the notation of $\mathcal{S}_k$ to all preambles between to successful packets and not restricted to a preamble group of one packet it holds

$$PDR_{\,CEPPS\ Fast,\ k} = fast(\{P_{k,1}, P_{k,2}, \ldots, P_{k,\max j \in \mathcal{S}_k}\}) =$$

$$chiperror\left(\frac{\sum_{i=0}^{31}\sum_{j\in\mathcal{S}_k}(P_{k,j}[i] \oplus P[i])}{|\mathcal{S}_k|}\right)$$

as well.

This type of LQE is later referred to as *CEPPS Fast* (Chip Error per Preamble Symbol Fast).

Fast Link Quality Estimation based on Estimator Variance

The concepts of the ultra-fast link quality estimation based on the variance apply in the same way to the fast estimation. Again by $\mathcal{S}_k$ to all detected preambles between to successful packets, it holds in the same way:

$$fast(\{P_{k,1}, P_{k,2}, \ldots, P_{k,\max j \in \mathcal{S}_k}\}) = variance(\mathrm{var}_k),$$

where

$$\mathrm{var}_k = \mathrm{Var}\left(chiperror\left(\sum_{i=0}^{31}(P_{k,1}[i] \oplus P[i])\right), \ldots, chiperror\left(\sum_{i=0}^{31}(P_{k,\max j \in \mathcal{S}_k}[i] \oplus P[i])\right)\right).$$

again with $\mathrm{Var}(X) = \mathrm{E}\left[(X - \mathrm{E}[X])^2\right]$ where X is a random variable and has the mean $\mathrm{E}[X]$. The function $variance(\cdot)$ maps then finally var_k to the respective PDR according to Figure 4.9.

This type of LQE is later referred to as *Variance Fast*.

Ultra-fast Link Quality Estimation based on Filtered and Weighted Average of Chip Errors

As mentioned in Section 4.2.1, we will discuss now the ultra-fast LQE that shows increased stability. As discussed in section 2.1.1 about the fundamentals of link quality estimation, stability is one of the key requirements of good LQE. It can be characterized by the variance of the mean. A classical approach to increase stability is to weight the sequential link quality estimations in form of a weighted moving average as shown in [9] for estimators based on packet statistics. We will apply a similar approach in order to smooth consecutive estimations. For this reason we perform a low pass filtering of the weighted average over a window w of consecutive link estimations. Suppose $\mathcal{R}_k$ is the set of the past $l = |\mathcal{R}_k|$ estimations of the original CEPPS fast estimator at the position k. Let further $r_{k-l}, r_{k-l+1}, \ldots, r_{k-2}, r_{k-1} \in \mathcal{R}_k$ be the past l estimations. Then the weighted average wa_k over these recent l estimations at the position k is calculated as

$$wa_k = \sum_{m=0}^{l} \beta_m r_{k-m},$$

with the weighting factor β such that

$$\sum_{m=0}^{l} \beta_m = 1.$$

And it further holds

$$wa_k = \sum_{m=0}^{l} \beta_m \; chiperror\left(\frac{\sum_{i=0}^{31}\sum_{j\in\mathcal{S}_k}\left(P_{k-m,j}[i] \oplus P[i]\right)}{|\mathcal{S}_{k-m}|}\right)$$

Using this weighted average wa_k we compute now the output of the low pass filter fwa_k (filtered weighted average) as

$$fwa_k = \alpha \; fwa_{k-1} + (1-\alpha)\left(\frac{1}{wa_k} - 1\right),$$

where $\alpha \in [0, 1]$ controls the smoothness. A small factor α enables to give more importance to current link behavior and vice versa.
Finally the k-th link quality estimation based on this statistical manipulations is obtained as

$$PDR_{\,CEPPS\;FWA,\,k} = \frac{1}{1 + fwa_k}.$$

Estimator	Input	Window (#Packets)	Calibration
ETX	packet statistics	10	No
WMEWMA	packet statistics	≥ 10	No
Four-Bit	hybrid	≥ 5	Yes
SNR	signal strength	1	Yes
Preamble Count Fast	Number of received preambles	1	Yes
Preamble Count Ultra-fast	Number of received preambles	0.1	Yes
CEPF	chip errors	1	Yes
CEPPS Fast	chip errors	1	Yes
CEPPS Ultra-fast	chip errors	0.1	Yes
CEPPS FWA	chip errors	≥ 0.1	Yes
Hybrid Fast	signal strength and chip errors	1	Yes
Hybrid Ultra-fast	signal strength and chip errors	0.1	Yes

Table 4.1: Overview of different LQEs.

The benefits of more stability is illustrated in Figure 4.13. The three figures on the left side show the estimation error of the originally implement CEPPS fast estimator for three different PDRs. The respective figures on the right side show the resulting estimation error after filtering and weighting the estimations for $l = 6$ and $\beta_0 = 0.3, \beta_1 = 0.2, \beta_{2,\ldots,6} = 0.1$. With the parameter l the estimation window can be changed in order to allow more or less estimator reactivity.

This type of LQE is later referred to as *CEPPS FWA* (Chip Error per Preamble Symbol Filtered Weighted Average).

Hybrid fast Link Quality Estimation

Similarly to the ultra-fast estimation, the hybrid fast link quality estimation weights the estimated PDR of the chip and preamble counting based estimators equally as

$$PDR_{\,Hybrid\ Fast} = \frac{1}{2}(PDR_{\,CEPPS\ Fast} + PDR_{\,Preamble\ Count\ Fast}).$$

This type of LQE is later referred to as *Hybrid Fast*.

4.3 Jamming Detection Estimation

As it will become apparent in the performance evaluation of the various LQEs, the preamble chip errors propose a very promising metric to estimate the link quality. This finding will be exploit the design of the jamming detection estimation.
The basic idea is to monitor the link traffic and to check continuously two metrics. The first metric we observe is the PDR at the receiver, simply recording how many packets arrive within an observation window w_o, with the resulting PDR_o. The second metric are the preamble chip errors within an estimation window w_e and the respective estimation of the PDR $\widehat{PDR_e}$. It

should be clear that if the two windows w_o and w_e are smaller then the coherence time of the link it should hold $PDR_o \approx \widehat{PDR}_e$. Apparently every estimator induces an estimation error and therefore PDR_o and $\widehat{PDR}_e$ cannot be set equal to each other. The average estimation error can be calculated as $\left|PDR_o - \widehat{PDR}_e\right|$.
Let's now investigate the case if a jammer tries to prevent the communication by emitting interference signals. In the case of the introduced reactive jammer that destroys the two symbols of the start of frame delimiter (SFD), thus non of the sent packets arrive at the receiver because of the synchronization errors. This is effecting the two expressions PDR_o and $\widehat{PDR}_e$ in the way that PDR_o will tend to zero (as implemented in the continuous reactive jammer - or to 6% as in the case of the random reactive jammer) but $\widehat{PDR}_e$ should not be affected because the preamble symbols are unaffected by the jammer. This occurring discrepancy between PDR_o and $\widehat{PDR}_e$, namely $\Delta = \left|PDR_o - \widehat{PDR}_e\right|$ if the jammer is active will be exploit for the jamming detection. The challenge is now to find a suitable error threshold ϵ for the decision rule if the jammer is active or not. If the difference between PDR_o and $\widehat{PDR}_e$ exceeds ϵ, it is assumed that a jammer is active.
Basically two design goals try to optimize ϵ. On the one hand side the false positives rate that tries to push ϵ to a large value and on the other hand the false negatives rate which tries to minimize ϵ. It is clear that ϵ is optimized for a PDR of 100% to a much larger value than for instance for a PDR of 10%. Therefore we introduce an error threshold ϵ that depends on $\widehat{PDR}_e$. Further let's define the null hypothesis H_0 and the alternative hypothesis H_1 as

$$H_0 : \text{"Normal transmission"},$$

$$H_1 : \text{"Jammed transmission"}.$$

The sum Λ of probabilities of the the false positives and false negatives can then be calculated as

$$\Lambda = P(\text{accept } H_0 \mid H_1 \text{ is valid}) + P(\text{reject } H_0 \mid H_1 \text{ is valid}).$$

In order to obtain the dependency of ϵ on $\widehat{PDR}_e$ under the condition that Λ has to be minimized, we sweeped for every value of PDR over $\epsilon \in [0, 1]$. The optimized ϵ shows a linear dependency, i.e.

$$\underset{\epsilon = f(\widehat{PDR}_e)}{\operatorname{argmin}} \quad \Lambda = \frac{1}{2}\widehat{PDR}_e.$$

This dependency of $\epsilon = \frac{1}{2}\widehat{PDR}_e$ supports as well theoretical reflections, where the error threshold is assumed to lie in the geometric center line of the decision region.

The estimator rule can then be stated under the hypotheses H_0 and H_1 for $\Delta = \left|PDR_o - \widehat{PDR}_e\right|$ as

$$\text{accept } H_0, \text{ if } \Delta > \epsilon,$$

$$\text{reject } H_0, \text{ if } \Delta < \epsilon,$$

and for $\Delta = \epsilon$ a random guess is performed.

Let's assume $\mathcal{S}_k$ is the set of all detected preambles during the estimation window w_e with the preamble numbering j of the elements in $\mathcal{S}_k$ as $j = 1, 2, \ldots, \max j \in \mathcal{S}_k$.
Therefore $\widehat{PDR}_e$ can be calculated as the k-th estimation

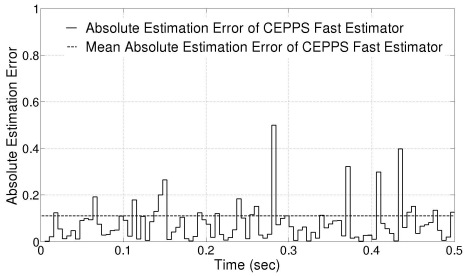
$$\widehat{PDR}_{e,k} = chiperror\left(\frac{\sum_{i=0}^{31}\sum_{j\in\mathcal{S}_k}(P_{k,j}[i]\oplus P[i])}{|\mathcal{S}_k|}\right),$$

and it follows for the jamming detection estimation rule for the k-th decision

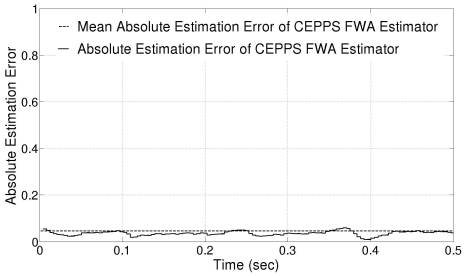
$$\text{accept } H_0, \text{ if } \left|PDR_o - chiperror\left(\frac{\sum_{i=0}^{31}\sum_{j\in\mathcal{S}_k}(P_{k,j}[i]\oplus P[i])}{|\mathcal{S}_k|}\right)\right| > \epsilon,$$

$$\text{reject } H_0, \text{ if } \left|PDR_o - chiperror\left(\frac{\sum_{i=0}^{31}\sum_{j\in\mathcal{S}_k}(P_{k,j}[i]\oplus P[i])}{|\mathcal{S}_k|}\right)\right| < \epsilon.$$

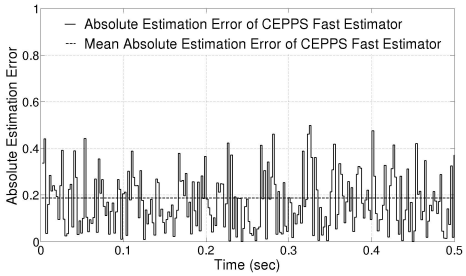
Clearly, the function $chiperror(\cdot)$ needs not to be defined again, since the mapping from a value of chip error to the respective guess of the PDR is still valid.



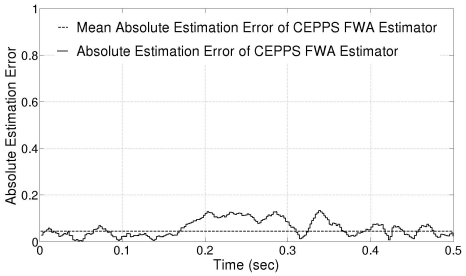
(a) PDR = 20%: Tracking of the absolute estimation error of the preamble chip error based LQE.



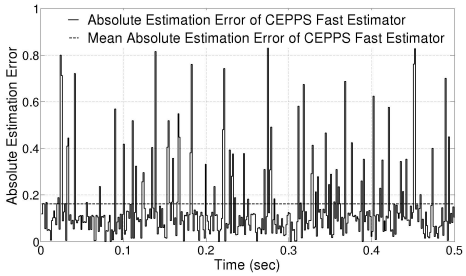
(b) PDR = 20%: Stability improvement of LQE (a) due to filtering and weighting moving average.



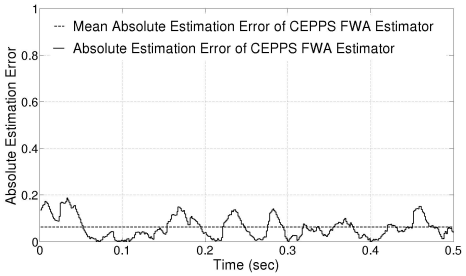
(c) PDR = 50%: Tracking of the absolute estimation error of the preamble chip error based LQE.



(d) PDR = 50%: Stability improvement of LQE (c) due to filtering and weighting moving average.



(e) PDR = 80%: Tracking of the absolute estimation error of the preamble chip error based LQE.



(f) PDR = 80%: Stability improvement of LQE (e) due to filtering and weighting moving average.

Figure 4.13: Tracking of the absolute estimation error of the preamble chip error based LQE in the line-of-sight setup. The figures (a),(c), and (e) on the left side show for the PDR of 20%, 50%, and 80% the estimation error of the originally CEPPS fast estimator on consecutive estimations within 0.5 seconds. The stability improvement of the CEPPS FWA (filtered and weighted average) estimator is shown on the right side for the same PDRs. Note as well how the estimation window shrinks for good quality links (i.e. from top down over the sub figures). This is due to the fact that the fast estimator estimates the channel after the reception of a successfully decoded packet - this event is more probable for greater link qualities. Further note that the mean absolute estimation error (dashed lines) is not the mean with respect to the first 0.5 seconds but on the entire measurement run.

5 Results

In this chapter we show how the designed LQEs perform under different link conditions. To assess the performance of the LQEs, we compare them to competitive estimators from related works. The LQEs that were used for the comparison are described in the next section. The last two sections show the performance of the LQEs with respect to their estimation error, respectively to the false positive and false negative rates.

5.1 Selection of Competitive Estimators

To provide comparisons to most diverse LQEs we chose estimators based on packet statistics, signal strength, hybrid, and chip errors.
In the case of the packet statistic based LQEs we implement two different estimators, *ETX* and *WMEWMA*. For the signal strength LQE we chose the best performing *SNR* based estimator. *Four-Bit* was elected from the hybrid estimators and for the error based LQE, *CEPF* that considers chip errors in the payload.

1. ETX [26]: *ETX* serves as a standard packet statistic based LQE that is widely used in WSNs. By sending broadcast probes at an average period τ within a window of w seconds each node can estimate the probe reception rate. The PDR can then be calculated as

$$PDR_{\text{ETX}} = \frac{probes(t-w,t)}{w/\tau},$$

   where $probes(t-w,t)$ is the number of probe packets received during the window w, and w/τ is the number of probes that should have been received. We set $\tau = 1$ second and $w = 10$ seconds as proposed in [26].

2. WMEWMA [9]: Compared to *ETX*, *WMEWMA* is statistically more sophisticated and is

an important LQE that encounters stability and reactivity by using low pass filtering and calculating weighted moving averages over sequential estimations. It is calculated as

$$PDR_{\text{WMEWMA},\,i} = \alpha\ PDR_{\text{WMEWMA},\,i-1} + (1-\alpha)\ \text{WMEWMA}_{\,i}.$$

As proposed by Baccour et at. in [19] we set the smoothing factor $\alpha = 0.6$ and calculate the instantaneous PDR over a window size of five received packets to obtain WMEWMA.

3. SNR [11]: This single metric LQE represents the group of the fast signal strength based LQE as it is assumed to perform better than RSSI [7]. Here a predefined correlation between the average signal strength of the packet and the PDR is used to estimate the link quality. This correlation is approximated by a linear fit and we found for our testbed the parameters:

$$PDR_{\text{SNR}} = 0.12\ dSNR - 1.7,$$

where $dSNR$ is the average signal power of a packet divided by the noise floor around the packet.

4. Four-Bit [22]: *Four-Bit* is closely related to *WMEWMA* but applies a filtering step more. For unicast transmissions the value $FourBit$ can be calculated as

$$FourBit_{\,i} = \alpha\ FourBit_{\,i-1} + (1-\alpha)\ \left(\frac{1}{WMEWMA_{\,i}} - 1\right),$$

and finally the PDR is obtained as

$$PDR_{\text{Four-Bit},\,i} = \frac{1}{1 + FourBit_{\,i}}.$$

5. CEPF [5]: This estimator considers chip errors in the whole payload. In our testbed the payload consists of 26 symbols. As proposed in [5] the correlation between the chip errors and the PDR was approximated by a linear fit. In this case the PDR is calculated as

$$PDR_{\text{CEPF}} = 1 - \frac{CEPS}{Chiplimit},$$

where $CEPS$ are the chip errors per symbol and $Chiplimit$ was set to 3.44, as obtained in the linear fit in our measurements. For $CEPS > Chiplimit$, PDR_{CEPF} is set to zero.

5.2 Evaluation Methodology

To ensure an efficient and meaningful evaluation, we recorded the entire transmitted packets down to chip level. This logged data were analyzed in Matlab scrips. This allows the performance evaluation of all LQEs on the same tracked data and it doesn't arise the problem of computation bottlenecks due to simultaneous real time evaluations of several estimators. Further no measurement needs to be reproduced because the logged data are stored and anytime available. Furthermore the observation window to calculate the effective PDR during the measurement can be set to $\pm$ 100 sent packets and half the window considers future values and

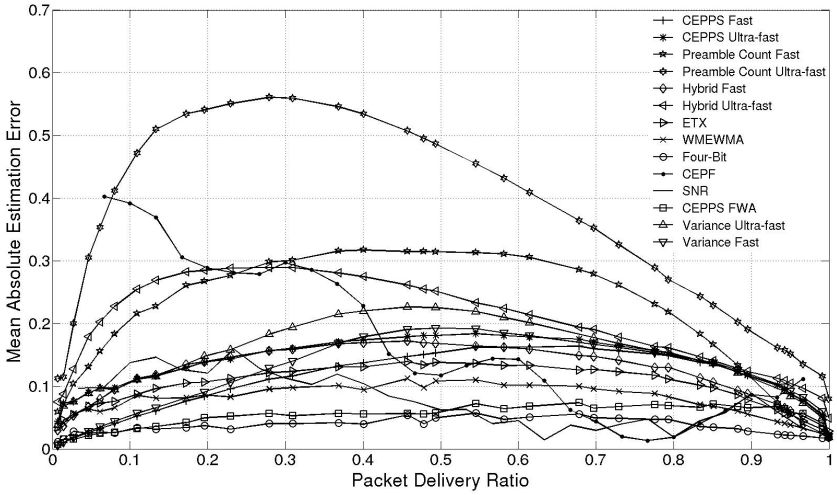
the other half of the window past values. An instantaneous link quality estimation of a tested estimator lies therefore in the middle of the estimation window and provides best accuracy of the ground truth. If we had to calculate the real PDR in real time, only past transmissions could be analyzed and the instantaneous estimation of a LQE lies then at the end of the estimation interval of the real PDR.
The applied sliding observation window w of $\pm$ 100 sent packets meets the requirement to be smaller than the coherence time and meets the discussed conditions to accurately assess the ground truth of the PDR since this value was not changing considerably anymore for larger w.
One measurement run for a discrete PDR value of one scenario consisted 40'000 transmitted packets of 13 bytes payload. The absolute estimation error of a LQE is simply calculated as the absolute value of the difference between the observed PDR and the estimated PDR.

5.3 Performance of Link Quality Estimation

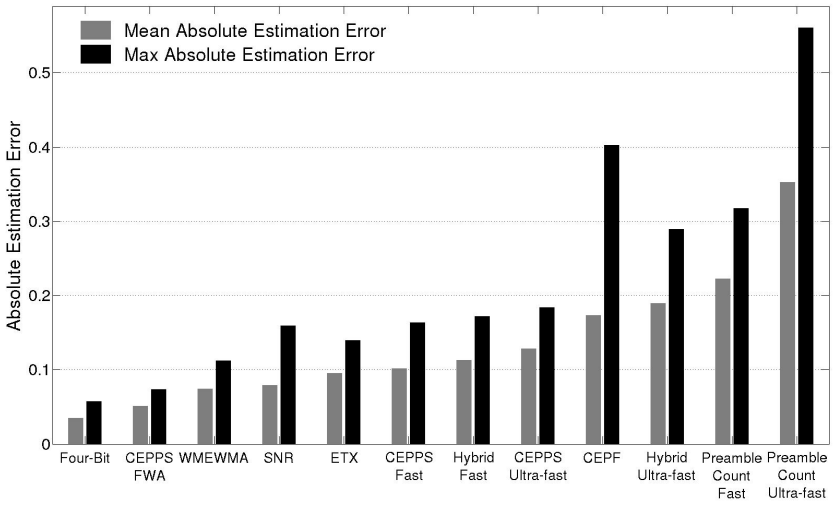
We show in the following subsections for every scenario the performance of the mentioned LQEs with respect to the mean absolute estimation error in the whole range of PDR.
Further for every scenario a ranking of the estimators is provided with the mean absolute estimation error over all different link qualities and the respective maximal obtained estimation error. Moreover we summarize the LQE performances in an average case where the estimator performances are weighted over the measured scenarios. Finally the convergence of the estimation error as a function of time is shown for selected estimators.
A detailed evaluation of all LQE can be found in Appendix D.

5.3.1 Cable Scenario

The results of the attenuated cable scenario are depicted in Figure 5.1. The cable scenario is characterized as a very stable link because unpredictable and uncertain factors (e.g. small scale fading) as observed for wireless links do not occur. Therefore this should lead to suitable link conditions for packet statistics estimators. This is confirmed by our measurements: These LQEs perform under wireless link conditions in average 67% worse than with a cable link. The best performance shows *Four-Bit* as a double filtered packet statistic based estimator with an absolute estimation error of 3.5%.
Remarkable is the performance of *CEPPS FWA* that shows smaller errors than packet statistic approaches as *WMEWMA* or *ETX*. Nevertheless the packet statistic estimators are implemented with realistic parameters that are widely used in related works, it has to be mentioned that the performance heavily depends on these values.
SNR is supposed to show promising results as well due to no interference. With an average estimator error of 7.9%, *SNR* ranked as the 4-th best estimator. Further shows *CEPPS Ultra-fast* compared to *CEPPS Fast* a 26% larger error. This is due to the fact, that the slower implementation considers more preamble symbols and can rely on more statistically relevant information as showed in Figure 4.10. The hybrid approaches, *Hybrid Fast* and *Hybrid Ultra-fast* are not able to outperform one of the single metrics they're based on. Obviously this is due to the bad performance of the estimators that only count the preambles: *Preamble Count Fast* and *Preamble Count Ultra-fast*. The slower estimator again is able to reduce the error of the ultra-fast implementation about 37%.
All our designed estimators show arch-shaped curves with vanishing errors towards the PDRs of 100% and 0%. This shape is assumed if we consider Figure 4.9 at least for the chip error based



(a) Evaluation of the LQE with respect to the absolute estimation error.



(b) Average and maximum estimation error over all link qualities

Figure 5.1: Cable Scenario: (a) Performance evaluation with respect to the mean absolute estimation error and (b) shows the average and maximum estimation error over all link qualities.

approaches, where the maximum variance is reached around PDRs of 50% and almost very small variances towards 100% and 0%.
The shapes of *CEPF* and *SNR* don't show this behavior. This can be explained that the fit for the correlation approximation between the respective metric and the PDR is a linear fit. Therefore some PDR regions over all different scenarios that are not perfectly covered by this linear fit cause irregularly errors potentially spread somewhere in the entire interval of PDRs. This statement can be extended to all scenarios because there is always used just one universal fit for all scenarios.
Note that the evaluation of the variance based LQEs are only shown for the cable scenario in Figure 5.1(a). The additional information of the second moment (LQE *Variance Fast* and *Variance Ultra-fast*) does not lead to a smaller estimation error compared the the respective single metric first moment LQE *CEPPS Fast* and *CEPPS Ultra-fast*. Especially in the critical decision region around a PDR of 50% (Figure 4.9) obviously a lot of confusions occur mistaking the correct fit and therefore induce relatively large errors. Further the decision rule with very specific values (i.e. the PDR threshold of 0.4542) and the unhomogeneous correlation between the variance of the estimator and the PDR suggest that this LQE is sensitive to parameter settings (e.g. PDR threshold of 0.4542, or the concatenation of two high order rational fits) and not reliable enough for a versatile estimator. Since only the correlation in the cable scenario is used for this fit (and already downgrade the results), there is no benefit in extending the approach to all four scenarios. An approximation of the correlation based on all setup points would induce even more errors. For that reason it is no longer considered for the remaining scenarios.

5.3.2 Line-of-Sight Scenario

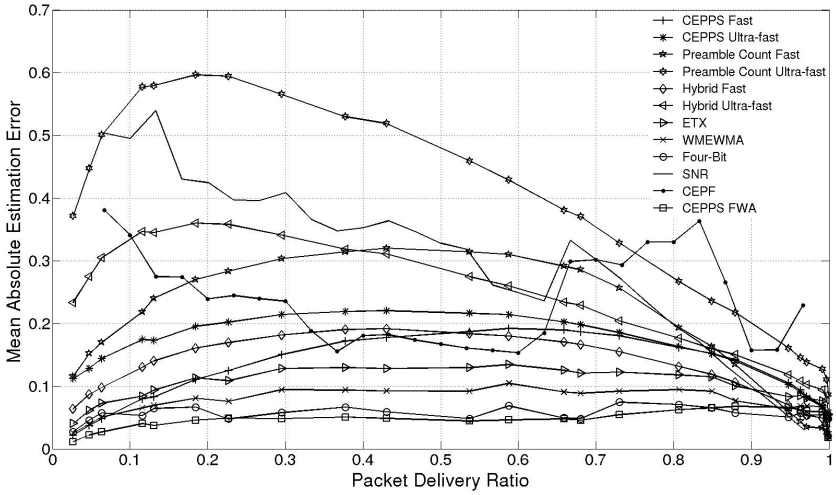
Turning to the first wireless scenario with a line-of-sight link component, we obtain a new order of the LQE with respect to the performance, see Figure 5.2. *CEPPS FWA* shows the smallest error of 4.7%. Since the link can still be assumed as static, the packet statistic based approaches should perform further on well. The ranking of these LQE reflect this as the 2nd, 3rd and the 4th best estimator are *Four-Bit*, *WMEWMA* and *ETX* with estimation errors below 10%. Again the hybrid implementations don't show improvements compared to both of the single metric implementations. *CEPF* still does not show superior results compared to the preamble chip error based estimators.

5.3.3 Non-Line-of-Sight Scenario

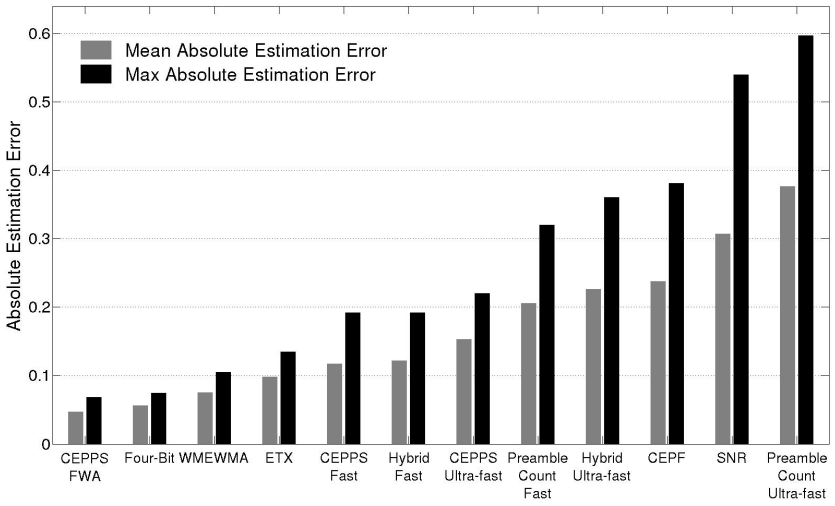
Compared to the line-of-sight scenario, the results in the non-line-of-sight setup don't reveal significant differences. The ranking is almost the same as well as the mean absolute estimation errors. Obviously the lack of the line-of-sight component seems not to influence the performance of the packet statistic and chip error based approaches. This results are shown in Figure 5.3.

5.3.4 Mobile Scenario

To measure enough data within a coherence time T_c we set the receiver speed to a maximum value of $v = 1$ cm/s in order to meet the requirements derived in Section 2.1.1. This results in a coherence time T_c of about 4 seconds in our setup. This time with our predefined sending data rate according to the IEEE 802.15.4 standard of 250kbit/s [2] in approximately 500 packets per

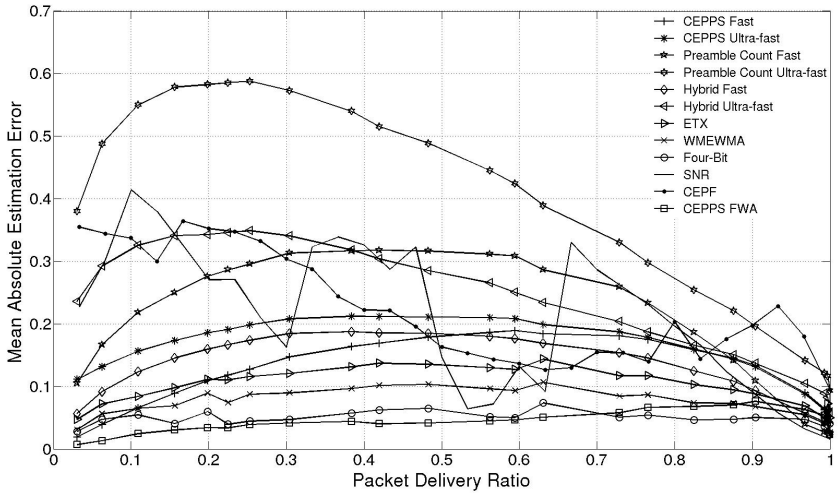


(a) Evaluation of the LQE with respect to the absolute estimation error.

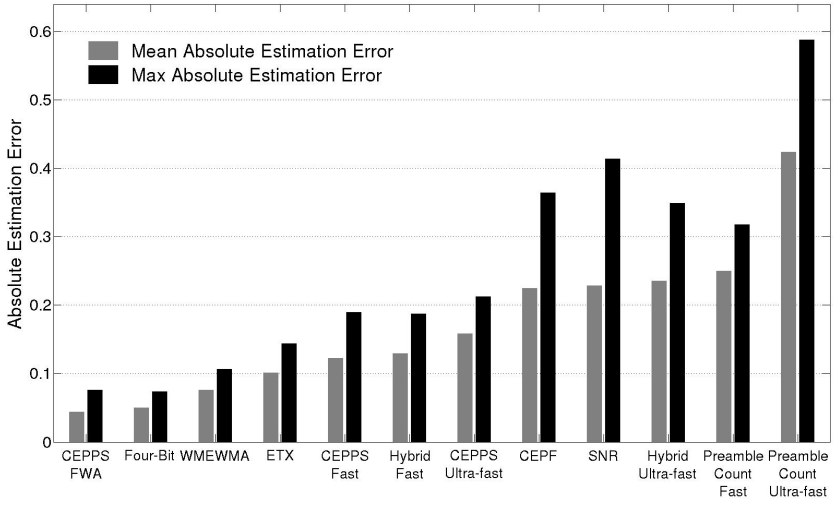


(b) Average and maximum estimation error over all link qualities

Figure 5.2: Line-of-sight scenario: (a) Performance evaluation with respect to the mean absolute estimation error and (b) shows the average and maximum estimation error over all link qualities.



(a) Evaluation of the LQE with respect to the absolute estimation error.



(b) Average and maximum estimation error over all link qualities

Figure 5.3: Non-line-of-sight scenario: (a) Performance evaluation with respect to the mean absolute estimation error and (b) shows the average and maximum estimation error over all link qualities.

T_c for a poor link quality of a PDR = 10%. This can be considered as statistically significant with a converged mean.

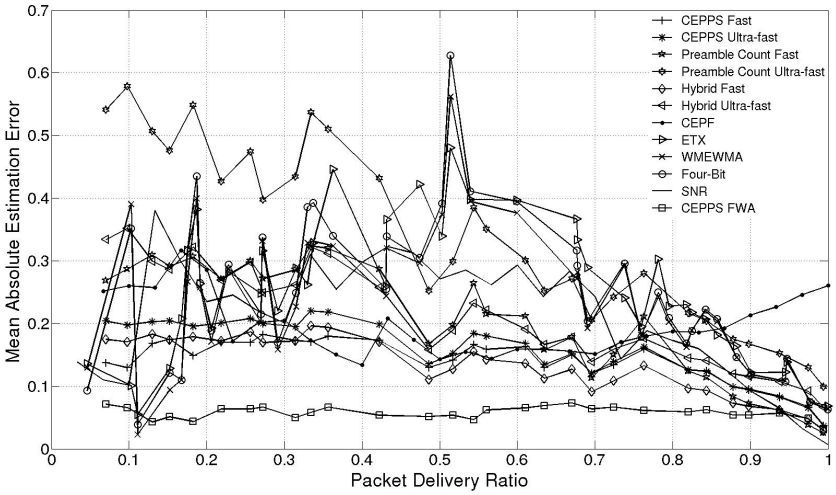
The mobile scenario realizes a non static channel behavior. This means that a good performing estimator needs to be reactive in order to take short term variations of the channel quality into account. Therefore it is assumed that the packet statistic based LQE are not able to capture such changes and are too strong relying on their estimation history and fail in dynamic setups. This assumption is confirmed if we consider the results of the mobile scenario in Figure 5.8. Except for *Preamble Count Ultra-fast*, *ETX*, *Four-Bit*, and *WMEWMA* show the worst performance. Compared to the static scenarios, they have in average an almost 300% estimation error increase with mean absolute estimation errors between 23% and 26%. Whereas the preamble chip error based LQE only show a slight increase in the error performance compared to the static setups of 17%. *CEPPS Ultra-fast* actually only increases his estimation error of 8.2%, what expresses the high reactivity of this estimator since the link quality is estimated on a per packet level.
Again as in all wireless setups, *CEPPS FWA* is the leader of all LQEs with an mean absolute estimation error of 5.7%. Despite the weighted average over the last six estimations at the cost of reactivity, this LQE is still reactive enough to beat all the other estimators. Further it has to be considered that even this estimator weights over an estimation window much larger than *CEPPS Ultra-fast* and *CEPPS Fast*, this estimation window remains small (compared to the coherence time) because the sequential estimations are as well almost on a per packet level.
Also *Hybrid Fast* beats the first time his single metric estimator based on chip errors by a narrow margin of 5% less error. After the preamble chip error based LQE *CEPF* and *SNR* perform as well better than the packet statistics with a mean estimator error of 19.7% and 20.1%.

5.3.5 Average Case

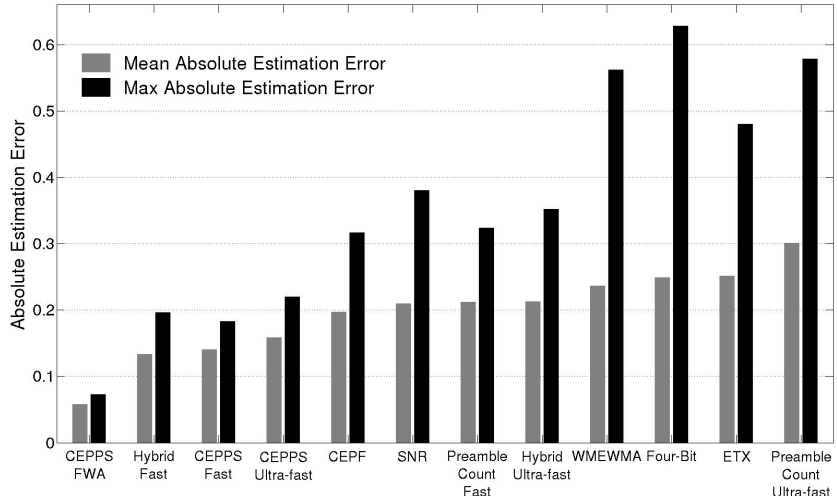
Normally in a real scenario we are not obtaining link conditions as cable connection and wireless line-of-sight at the same time. This average case tries to encounter a potentially real environment. Here we suppose that in half of the time the nodes are moving and in the remaining time there exists a static setup with no movements. Further half of the time exists a line-of-sight path and the other half the direct component between sender and receiver is blocked. These results are shown in Figure 5.5. Again the leading estimator *CEPPS FWA* shows an average absolute estimation error of 5.2% followed by *Hybrid Fast*, *CEPPS Fast* with errors of 12.9% and 13.0%. The packet statistic based estimators perform in this average case are characterized by large maximum estimation error that occur during the mobility of the nodes.

5.3.6 Error Convergence

To illustrate the behavior of the estimation error convergence of the different LQEs a typical example of the line-of-sight scenario with a PDR of 50% is depicted in Figure 5.6. The subfigures (a), (b), and (c) show how the error of the fast estimators shrink within the first couple of ms after the start of a transmission. The huge difference in the speed of the convergence can be seen in subfigure (d) where e.g. *ETX* needs 10 seconds to reach his converged estimation error. Note that the time needed for an update (a step in the plot) of the chip error based and signal strength based LQEs depend on the PDR. As for a higher PDR the expected time for a successful packet reception decreases, coincidentally the respective estimation times shrink as well. The opposite happens if the PDR gets poor, the average waiting time for a packet reception increases. The only estimators that don't rely on the event of a successful packet reception is the *CEPPS*



(a) Evaluation of the LQE with respect to the absolute estimation error.



(b) Average and maximum estimation error over all link qualities

Figure 5.4: Mobile scenario: (a) Performance evaluation with respect to the mean absolute estimation error and (b) shows the average and maximum estimation error over all link qualities.

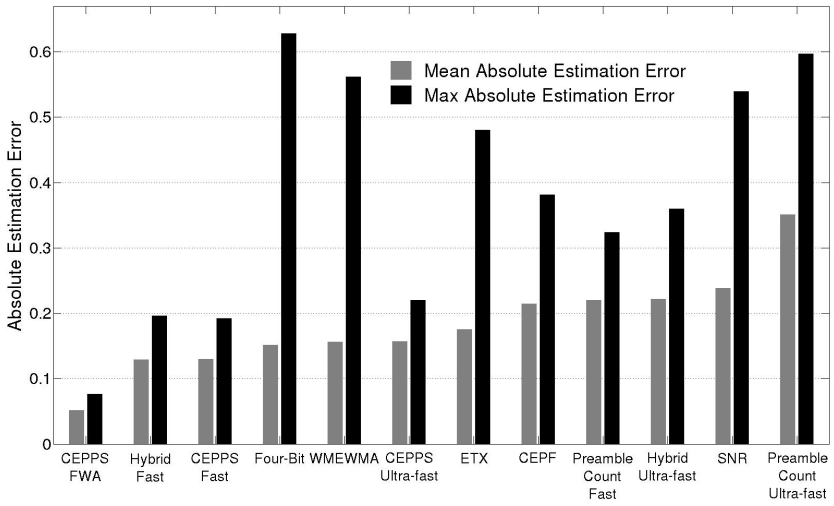


Figure 5.5: Average Case: Evaluation of the LQE with respect to the absolute estimation error. In this case we assumed that in 50% of the time, the nodes are moving, while in the other 50% the nodes are static with half the time line-of-sight and half the time non-line-of-sight conditions.

Ultra-fast (and the *CEPPS FWA*), since the probability is very high that even for bad links some preambles are successfully decoded per sent packet.

5.3.7 Discussion

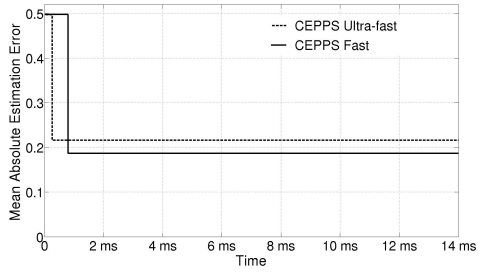
With the evaluated four experimental scenarios we can confirm the findings from related works concerning the performance of various LQEs:

1. Signal Power based: The considered estimator *SNR* provides fast estimations but with an average estimation error over all scenarios of 20.6%, this LQE lacks accuracy. However they are suitable for a rough link quality classification into good or bad.

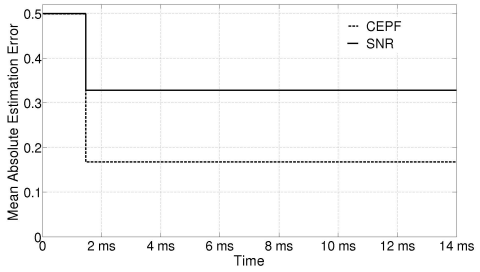
2. Packet Statistic based: *ETX*, *WMEWMA*, and *Four-Bit* show small estimation errors in stable link environments as cable, or static wireless links (errors $< 10\%$). But due to the fact that they heavily depend on a large observation window of the past transmissions, they fail in mobile network settings (errors $> 23\%$).

3. Payload Chip Error based: *CEPF* is suitable for mobile environments because this LQE shows approximately the same performance as in the static setups ($\pm$ 7%). In mobile settings, *CEPF* outperforms all packet statistic and signal based LQE, but with an average estimation error of 19.7% *CEPF* cannot compete with the the hybrid and preamble chip error based estimators (e.g. error of *CEPPS FWA* $< 6\%$).

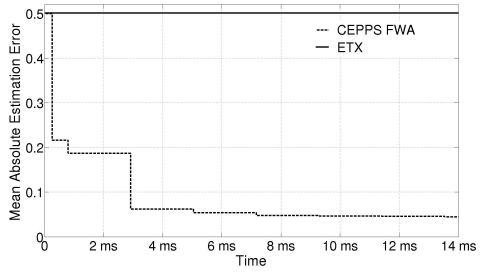
4. Preamble Chip Error based: This LQEs provide very promising performances in all scenarios. CEPPS WMA shows the best performance over all scenarios with an average estimation error of 5.0%. Because of their very small estimation window they are extremely suitable



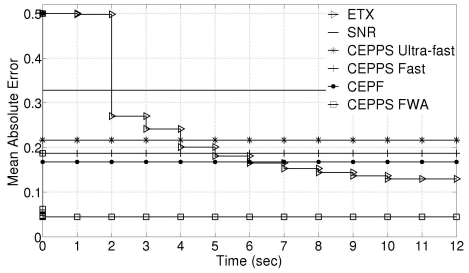
(a) *CEPPS Ultra-fast* estimates the link quality after the reception of the first preamble group. This leads to an average estimation time of 0.26 ms. In the case of the *CEPPS Fast* estimator that waits until a successful packet reception, the estimation time is 0.72 ms.



(b) *CEPF* and *SNR* are based in the chip errors in the payload, respective the average SNR in the payload. They estimate as well after a packet reception and are therefore estimating on the same packet as *CEPPS Fast*. They have an average estimation time of 1.46 ms. The estimation time difference of 1.46 ms - 0.72 ms = 0.74 ms compared to *CEPPS Fast* is due to the time that laps away until the last symbol of the payload is decoded.



(c) ETX needs to wait for 1 second until the reception of the first probe packet. Therefore the absolute estimation error during the first second is 0.5. *CEPPS FWA* makes his first estimation after the reception after the first preamble group similarly as *CEPPS Ultra-fast*. After the successful reception of the first packet the second estimation is further updated in the same way as *CEPPS Fast* after 0.72 ms. Then it further updates the estimation according to the weighting and filtering rules after every packet reception in intervals of 2.12 ms until 13.54 ms when the steady state is reached.



(d) This figure illustrates on a larger time scale up to 12 seconds how the estimation error of *ETX* decreases after every second due to the sent probe packets. Because *ETX* considers an estimation window of 10 seconds, the error converges after this amount of time.

Figure 5.6: Example of the estimation error convergence of the LQEs: *ETX*, *SNR*, *CEPPS Fast* and *CEPPS Ultra-fast*, *CEPF*, and *CEPPS FWA*. Figures (a),(b), and (c) show how the absolute estimation error converges within the first 14 ms as in the example of the line-of-sight scenario for a PDR of 50%. Figures (d) illustrates the convergence on a larger scale in the first 12 seconds of a transmission of the same setup. If there is no estimation available then the error is assumed to be 0.5, i.e. a random guess.

for mobile settings, where they outperform all the other LQEs. But also in the static wireless setups consistently they show better performances as *CEPF* or *SNR*. The benefits of analyzing the preamble symbols becomes clear if we compare *CEPF* and *CEPPS Ultra-fast*. *CEPPS Ultra-fast* has an estimation error 28.2% less over all scenarios than *CEPF* while only maximum eight symbols are considered compared to *CEPF* that analyzes the whole payload, i.e. 26 symbols.

5. Preamble Counting based: The single metric LQE as *Preamble Count Fast* and *Preamble Count Ultra-fast* are very simple estimation approach since the estimation rule is only based on counting the received preamble symbols. However with an average estimation error over all scenarios of 22.2% and 36.4% for the fast and the ultra-fast implementations, the clearly fail due to estimation accuracy. One can extract advantages from these LQEs if an extremely simple estimation rule is needed and only distinctions from very poor and perfect links are needed.

6. Hybrid (4.&5.) based: The hybrid approaches don't show the desired effect of a superior performance of the respective single metric LQEs. This is due to the rather bad performance of the single metric LQEs *Preamble Count Fast* and *Preamble Count Ultra-fast*. Only in the mobile setting *Hybrid Fast* slightly outperforms *CEPPS Fast*.

7. Estimator Variance based: As already mentioned in the section about the cable scenario, *Variance Ultra-fast* and *Variance Fast* don't show a performance improvement compared to *CEPPS Fast* and *CEPPS Ultra-fast*. Since the correlation fit based only on the cable data points don't even show benefits, the approach was not extended to all four scenarios since the additional error caused by the more imprecise fit would worsen the absolute estimation error even more.

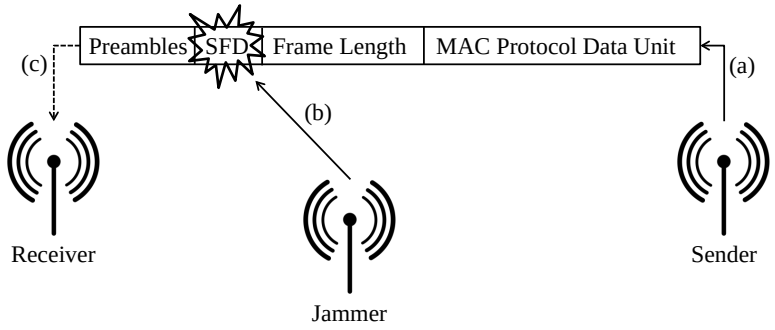


Figure 5.7: Jamming Scenario: A transmitted packet from the sender (a) is reactively jammed (b) on the start of frame delimiter (SFD) and prevents the receiver to successfully decode the packet. The sole information the receiver detects are the preamble symbols - then the synchronization fails due to the corrupted SFD and the receiver is not able to start decoding the frame length and the MAC Protocol Data Unit.

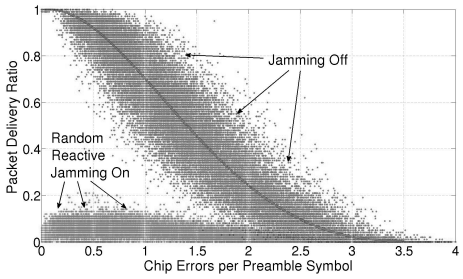
5.4 Performance of Jamming Detection Estimation

For the performance evaluation of the continuous reactive jammer and the random reactive jammer we set the observation window w_o to 100 packets and the estimation window w_e to 10 packets. This relatively small w_e allows to detect changes in the operation of a jammer (currently active or sleeping) within approximately 10 sent packets.
To explore the performance of the designed jamming detection estimation over all link qualities, we adapted the transmit power and sent for a specific PDR 10'000 packets that were analyzed to obtain the false positive and false negative rates:

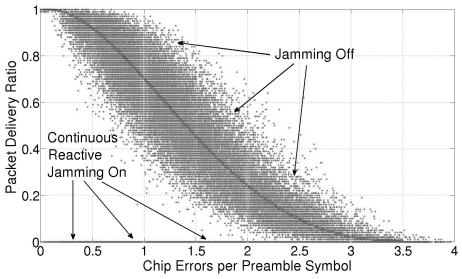
$$\text{false negatives} = P(\text{"Guess Jamming off"} \mid \text{"Jamming on"})$$

$$\text{false positives} = P(\text{"Guess Jamming on"} \mid \text{"Jamming off"})$$

The jamming scenario is illustrated in Figure 5.7 where a sent packet is reactively jammed on the SFD (start of frame delimiter) and prevents the receiver to successfully decode the packet. The sole information of the packet that reaches the receiver are the uncorrupted symbols of the preamble sequence.
To gain insight of the impact of the two different jammers, Figure 5.9 shows the affected PDRs. Subfigure (a) shows the effect of the random reactive jammer. The diagonal cloud (dark points) shows the measured PDRs when the jammer was turned off, i.e. a normal transmission. The dark small curve in the middle of this cloud belongs to the already in the link quality estimation measured correlation between the preamble chip errors and the PDR. As expected if the transmission is not affected by the jammer, the points are spread around this correlation curve. If the random reactive jammer is active, the position of these points will change. The light grayish points that amount to the horizontal cloud just above the horizontal axis show the affected PDRs. As the reactive jammer does not destroy every packet, we still obtain PDR up to 20%. This observation changes in the subfigure (b) where the situation for the continuous reactive jammer is depicted. This jammer corrupts every packet and if the jammer is turned on the PDRs tend to zero, i.e. the observed PDR points coextensive with the horizontal axis. The diagonal could again stands for the measured correlation points if the jammer was sleeping. It is obvious that



(a) Random Reactive Jamming: Comparison of the measured PDRs if there is no Jamming (diagonal cloud) and if the transmission is jammed by a random reactive jammer (horizontal could on the bottom).



(b) Continuous Reactive Jamming: Comparison of the measured PDRs with no Jamming (diagonal cloud) and with continuous reactive jamming (line on the bottom belonging to PDR = 0%)

Figure 5.8: Comparison of measured PDRs. The random reactive jammer in (a) and the continuous reactive jammer (b). The respective diagonal cloud shows the PDR distribution without any jamming

a jamming detector will struggle more in the case of the more sophisticated random reactive jammer, because the clouds tend to overlap for poor link qualities below a PDR of 15% and the points cannot dedicated clearly to the cases where the jammer was turned on or was sleeping.
To obtain the performance of these two jamming scenarios we measured the false positives and false negatives as depicted in Figure 5.9. All the evaluated curves show error probabilities below 5% from perfect links to PDR of 50%. Below PDRs of 40% the sophisticated random reactive jammer causes false negatives over 10% and constantly increasing for worse links. The false negatives of the continuous reactive jammer show smaller error probabilities that exceed 10% for PDRs below 20%. The false positives rate for both jammers stays as well for good links very small and exceeds the error threshold of 10% for PDR below 35% and then increases as well for worse link qualities.
This general observation of increasing false positive and false negative rates in both jamming scenarios is due to the fact that the measured PDRs if the jammer was active or turned off are overlapping. A PDR obtained in poor link environments can hardly be assigned to a jammed poor link quality situation or a ordinary poor link quality. But it has to be considered that the benefit in detecting jammers in poor link qualities conditions is not that crucial because the data flow is maybe anyway rerouted around this link. For good links with PDRs $> 50\%$, an accurate jamming detection is more valuable. In this region even for the sophisticated random reactive jammer shows no error rates below 10%. For PDRs $> 58\%$ even below 5%.
To the best of our knowledge there is no work known so far that can provide a jamming detection in our two proposed reactive jammer scenarios where only the transmission of the SFD is destroyed. In the work of Strasser et al. [1] they do not present their evaluation of the false positives and the false negatives over the whole range of PDRs. This error rates are only presented for a strong and a weak link (not revealing the respective PDRs) for the three different restrictions they impose. They obtain for these two links a false positive rate of 0% and a false negative rate for the strong link between 0% and 15.1% (depending on how many bits were jammed). And for weak links between 14.8% for two jammed bits and 0% for more than 16 jammed bits. Note

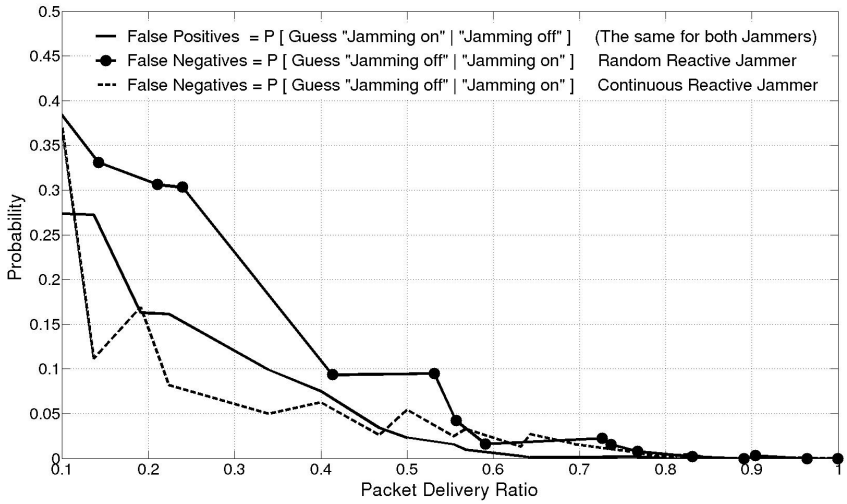


Figure 5.9: Performance evaluation of the random and the continuous reactive jammer with respect to the false positives and false negatives rate. The false positives are not affected by the jamming type.

that the performance comparison of the work in [1] and our evaluation have to be considered carefully because the results presented of Strasser et al. belong to setups that suffer from sever restrictions as discussed in Chapter 2. The situation proposed in this work is more general and imposes no restrictions. The jamming estimation approach in [1] would not work in our jamming scenario.
A further approach could be to analyze the signal strength in the preamble symbols instead of the chip errors. This approach would then work similarly as presented here but relying on a SNR-PDR correlation curve instead of a CEPPS-PDR calibration. Eventually affecting the performance of the estimator with respect to false positive and false negative rates is the ability for an accurate link quality estimation. But as shown in this work and discussed in several other studies [5, 19, 20] these LQEs do not provide accurate estimations.

6
Conclusions and Outlook

6.1 Conclusions

In this thesis we explored experimentally the performance of different LQEs considering various properties of preamble symbols. Further we showed their benefits in a jamming detection scenario.
With a software defined radio based implementation of IEEE 802.15.4 we analyzed chip errors, the number of received preambles, the variance of the chip error based LQEs, and hybrid approaches of the stated estimators. We showed that chip errors in the preamble symbols serve as a good indicator to predict a link's PDR.
As our best performing estimator we defined *CEPPS FWA*, a chip error based estimator that weights and filters sequential estimations on a per-packet level. Our LQE proved to be three times faster than state-of-the-art rapid estimators as *CEPF* and more accurate than other LQEs under different wireless channel conditions as mobile, line-of-sight, or non-line-of-sight links. In average *CEPPS FWA* shows half the absolute estimation error (5%) of the best performing packet statics based LQE and four times less error than the signal strength and the payload chip error based estimators. Especially in dynamic environments like mobile scenarios, our ultra-fast preamble chip error based LQE is particularly advantageous due to the ability to match up to the fast changing link qualities. *CEPPS FWA* fills the gap of an universal LQE that offers superior performance under different wireless link conditions.
We further proposed a novel approach to detect sophisticated reactive jamming attacks that target the start of frame delimiter (SFD) during a packet transmission. Analyzing chip errors in the synchronization header of the physical layer allows to predict the link quality before the synchronization between sender and receiver fails due to the jamming attack. This undistorted estimation of the link quality is then compared to the observed PDR at the receiver side that is affected by the jammer. This resulting discrepancy in case the jammer was active is explored to make a decision whether the packets were jammed or not.
We showed that we can detect the two different jammer attack approaches (continuous and ran-

dom) with a precision above 90% for PDRs > 40%. For moderate and good links with a PDR > 60% we reach a decisional accuracy above 97.3%. This novel approach relying on preamble chip errors offers high detection rates of reactive jammers for moderate and good links.

6.2 Outlook

Using software defined radio related to new approaches in link quality estimation and jamming detection turned out to be a promising approach to gain insight into the channel behavior based on real measurements. Interesting next steps would be to integrate the ZigBee implementation on top of the IEEE 802.15.4 standard to determine the benefits of the new designed LQEs in real applications and moreover in ad hoc network, opportunistic networks, or vehicular networks.
More specific our proposed LQE *CEPPS FWA* could be further developed in the sense of an adaptive estimation window. This window would then adapt to the current coherence time of the channel. Therefore the link quality would be estimated based on a maximum amount of information provided over a time where the channel is supposed to be constant. This would lead to more accurate link quality estimations in static scenarios, where the estimation time would be rather large. Whereas in dynamic setting as mobile environment the estimation window would shrink in order to capture the fast changing behavior of the link. The coherence time could be approximated by the approach of *CEPPS Ultra-fast*, where consecutive averaged estimations capture changes on a per-packet level and allow an ultra-fast reaction time on link quality changes.
The reactive jamming detection estimator could be extended to a two metric estimator. In addition to the preamble chip errors, the SNR could be determined. Especially in the poor performing region for bad link qualities, the single-metric approach could benefit from the second metric. Because in general the jammer will increase the received signal strength in case of bad links significantly. This could be exploit to reach better jamming detection rates for bad link qualities.

A

CD-ROM Content

Below there is a list of the files provided in the CD-ROM.
The folder *Master Thesis Report* contains the pdf of this report. The folder *Adapted UCLA ZigBee Versions* includes the adapted source code of the UCLA ZigBee implementation of [4]. It embodies the sender and the receiver side of the initial experiments and the versions of the scenario measurements. In the *GNU Radio 3.2.2 Source* folder you find the openly available GNU Radio software platform used in this thesis. The *LaTeX Source* folder provides you with the necessary source files of the report.
Matlab was used to analyze and extract information out of the log files generated by the software defined radio implementation. The *Matlab Source* folder contains the Matlab functions used to analyze and evaluate the .txt files.

It follows the content of the CD-ROM provided in addition to the document:

```
- Master Thesis Report
    |-- MaterThesis.pdf

- Adapted UCLA ZigBee Versions
    |-- gr_quadrature_demod adaptions
    |-- ZigBee_Receiver
    |-- ZigBee_Sender

- GNU Radio 3.2.2 Source
    |-- gnuradio-3.2.2.zip

- LaTeX Source
    |-- Thesis

- Matlab Source
    |-- scenario data
    |
    |-- append_error_vector.m
    |-- avg_preamble_symbols_per_packet.m
    |-- calculate_instantaneous_CEPS_all_preambles.m
    |-- calculate_instantaneous_CEPS_same_packet.m
    |-- calculate_instantaneous_pdr_all_packet.m
    |-- calculate_instantaneous_pdr_all_packet_hybrid.m
    |-- calculate_instantaneous_pdr_same_packet.m
    |-- calculate_instantaneous_pdr_same_packet_hybrid.m
    |-- calculate_PDR_window.m
    |-- calculate_PDR_window_MOBILE.m
    |-- calculate_PDR_window_MOBILE_prob.m
    |-- CEPS.m
    |-- CEPS_including_preamble_1.m
    |-- chip_error_distribution.m
    |-- correct_time.m
    |-- count_avg_preamble_symbols_per_delivered_packet.m
    |-- count_avg_preamble_symbols_per_delivered_packet_
        for_recovered_d.m
    |-- count_avg_preamble_symbols_per_successful_packet_
        decoding.m
    |-- count_delivered_packets.m
    |-- count_dropped_packets.m
    |-- count_errors_for_chip_position.m
    |-- count_preamble_groups.m
    |-- count_received_packets.m
    |-- count_rec_packets.m
    |-- count_sender_preambles.m
    |-- Data_Points.m
    |-- delete_1_preamblesnotfollowedby2.m
    |-- delete_all_No_One_Preambles.m
    |-- estimate_CEPPS.m
    |-- estimate_fast_preamble.m
    |-- estimate_ultra_fast_preamble.m
    |-- estimate_var_highPDR.m
    |-- estimate_var_lowPDR.m
    |-- ETX.m
    |-- ETX_real.m
    |-- evaluate_inter_preamble_symbol_time.m
    |-- evaluate_max_eof_packet_sign_to_next_after_
        preambel_break.m
    |-- evaluate_max_inter_packet_break.m
    |-- evaluate_max_inter_preamble_symbol_break.m
    |-- evaluate_min_eof_packet_sign_to_next_after_
        preambel_break.m
    |-- evaluate_min_inter_packet_break.m
    |-- evaluate_min_inter_preamble_symbol_break.m
    |-- evaluate_packet_delivery_time.m
    |-- evaluate_packet_time.m
    |-- evaluate_preamble_time.m
    |-- evaluate_rest_time.m
    |-- evaluation_estimator.m
    |-- find_0_Preamble_1_combinations.m
    |-- find_optimal_error_threshold.m
    |-- get_error_vector.m
    |-- get_indices_of_slicing.m
    |-- get_No_One_Preambles.m
    |-- get_vector_on_which_preamble_no_sync.m
    |-- give_1_a_dummy_time.m
    |-- give_lost_packets_time_stamp.m
    |-- herrorbar.m
    |-- jamming.m
    |-- jamming_real.m
    |-- LOS_Data_Points.m
    |-- map_delta_after_10_10_combination.m
    |-- map_delta_after_eop_symbol.m
    |-- map_delta_after_preamble_symbol.m
    |-- max_preamble_count_between_packets.m
    |-- mobile_PDR_CEPPS.m
    |-- NLOS_Data_Points.m
    |-- number_of_symbols_precision.m
    |-- plot_avg_max_preambles_count.m
    |-- plot_avg_preambles_got_sync.m
    |-- plot_avg_preambles_per_packet.m
    |-- plot_chip_error_distribution.m
    |-- plot_first_preamble_error_evolution.m
    |-- plot_jamming_boarder.m
    |-- plot_jamming_performance.m
    |-- plot_jamming_performance_real.m
    |-- plot_PRR_CEPS.m
    |-- plot_PRR_sum_preamble_symbols.m
    |-- preamble_nr_on_which_synchronisated.m
    |-- preamble_number_fast.m
    |-- preamble_number_ultra_fast.m
    |-- probability_packetdecoded_CRCwrong.m
    |-- probability_packetlost_syncmissed.m
    |-- PRR.m
    |-- recover_and_map.m
    |-- recover_lost_packets.m
    |-- remove_end_of_packet_signs.m
    |-- remove_noise_start_and_end.m
    |-- remove_packet_delivered_sign.m
    |-- remove_preamble_cnt.m
    |-- search_end_of_packet_indices.m
    |-- search_start_end_entire_data.m
    |-- show_deltas_preamble_serie_too_long.m
    |-- slice_into_Tc.m
    |-- slice_into_Tc_prob.m
    |-- sliding_window_CEPS.m
    |-- sliding_window_PDR.m
    |-- Tc_indices.m
    |-- time_between_two_packets.m
    |-- variance_analyze.m
    |-- WMEWMA.m
    |-- WMEWMA_real.m
```

B

IEEE 802.15.4 Symbol to Chip Sequence Conversion

Data Symbol (decimal)	**Data Symbol (binary)**	**Chip Values** $(c_0, c_1, c_2, \ldots, c_{31})$
0	0 0 0 0	11011001110000110101001000101110
1	1 0 0 0	11101101100111000011010100100010
2	0 1 0 0	00101110110110011100001101010010
3	1 1 0 0	00100010111011011001110000110101
4	0 0 1 0	01010010000101110110110011100011
5	1 0 1 0	00110101001000101110110110011100
6	0 1 1 0	11000011010100100010111011011001
7	1 1 1 0	10011100001101010010001011101101
8	0 0 0 1	10001100100101100000011101111011
9	1 0 0 1	10111000110010010110000001110111
10	0 1 0 1	01111011100011001001011000000111
11	1 1 0 1	01110111101110001100100101100000
12	0 0 1 1	00000111011110111000110010010110
13	1 0 1 1	01100000011101111011100011001001
14	0 1 1 1	10010110000001110111101110001100
15	1 1 1 1	11001001011000000111011110111000

Table B.1: IEEE 802.15.4 Symbol to Chip Sequence Conversion

C

Estimator Polynomial Coefficients

	coefficients p_i	**coefficients** q_i
$i = 0$	0.0159757288599271	-33.2301201417265
$i = 1$	-0.331596563212197	433.702812499815
$i = 2$	2.40895333730046	-2751.11303185885
$i = 3$	-7.2561605873854	8329.56739738512
$i = 4$	8.82790181294683	9276.54695248582
$i = 5$	-3.24278075225798	-

Table C.1: Coefficients of ultra-fast estimator based on the occurrence of preamble symbols

	coefficients p_i
$i = 0$	0.00236380302431496
$i = 1$	-0.0416198664714094
$i = 2$	0.258573756450039
$i = 3$	-0.603901367933267
$i = 4$	0.0924097366813666
$i = 5$	0.995581418309675

Table C.2: Coefficients of ultra-fast estimator based on chip errors in the preamble symbols

	coefficients p_i	**coefficients** q_i
$i = 0$	-0.00786953210672406	4.06703385467094
$i = 1$	5.67053468215535	81.0202425124916
$i = 2$	88.3820148833864	-336.464097126432
$i = 3$	-21.5007638551088	-88.8583243895553
$i = 4$	40.4204251570841	-15.5506556589827
$i = 5$	11.334497504376	-

Table C.3: Coefficients of fast estimator based on the occurrence of preamble symbols

D

Detailed LQEs Evaluation

Cable	[0% ... 33%]	[33%...66%]	[66%...100%]	[0%...100%] (mean, max)
1	Four-Bit	Four-Bit	Four-Bit	Four-Bit (0.03491, 0.05698)
2	CEPPS FWA	CEPPS FWA	SNR	CEPPS FWA (0.051, 0.07376)
3	CEPPS Fast	SNR	CEPPS FWA	WMEWMA (0.07401, 0.1122)
4	WMEWMA	WMEWMA	WMEWMA	SNR (0.07896, 0.1595)
5	ETX	ETX	CEPF	ETX (0.09507, 0.1393)
6	Hybrid Fast	CEPPS Fast	ETX	CEPPS Fast (0.1018, 0.1638)
7	CEPPS Ultra-fast	CEPF	Hybrid Fast	Hybrid Fast (0.1128, 0.1722)
8	SNR	Hybrid Fast	CEPPS Fast	CEPPS Ultra-fast (0.1286, 0.1837)
9	Pramble Count Fast	CEPPS Ultra-fast	CEPPS Ultra-fast	CEPF (0.1732, 0.4028)
10	Hybrid Ultra-fast	Hybrid Ultra-fast	Hybrid Ultra-fast	Hybrid Ultra-fast (0.1897, 0.2895)
11	CEPF	Preamble Count Fast	Preamble Count Fast	Preamble Count Fast (0.2229, 0.3173)
12	Preamble Count Ultra-fast	Preamble Count Ultra-fast	Preamble Count Ultra-fast	Preamble Count Ultra-fast (0.3527, 0.5609)

LOS	[0% ... 33%]	[33%...66%]	[66%...100%]	[0%...100%] (mean, max)
1	CEPPS FWA	CEPPS FWA	CEPPS FWA	CEPPS FWA (0.04669, 0.06797)
2	Four-Bit	Four-Bit	Four-Bit	Four-Bit (0.05596, 0.07452)
3	WMEWMA	WMEWMA	WMEWMA	WMEWMA (0.07512, 0.1047)
4	ETX	ETX	ETX	ETX (0.0984, 0.1345)
5	CEPPS Fast	CEPF	Hybrid Fast	CEPPS Fast (0.1174, 0.1919)
6	Hybrid Fast	CEPPS Fast	CEPPS Fast	Hybrid Fast (0.1214, 0.1917)
7	CEPPS Ultra-fast	Hybrid Fast	CEPPS Ultra-fast	CEPPS Ultra-fast (0.1526, 0.2203)
8	Preamble Count fast	CEPPS Ultra-fast	Hybrid Ultra-fast	Preamble Count Fast (0.2058, 0.3202)
9	CEPF	Hybrid Ultra-fast	Preamble Count Fast	Hybrid Ultra-fast (0.2265, 0.3601)
10	Hybrid Ultra-fast	Preamble Count Fast	SNR	CEPF (0.2379, 0.3813)
11	SNR	SNR	Preamble Count Ultra-fast	SNR (0.3072, 0.5396)
12	Preamble Count Ultra-fast	Preamble Count Ultra-fast	CEPF	Preamble Count Ultra-fast (0.3764, 0.5968)

NLOS	[0% ... 33%]	[33%...66%]	[66%...100%]	[0%...100%] (mean, max)
1	CEPPS FWA	CEPPS FWA	Four-Bit	CEPPS FWA (0.04407, 0.07616)
2	Four-Bit	Four-Bit	CEPPS FWA	Four-Bit (0.05023, 0.07363)
3	WMEWMA	WMEWMA	WMEWMA	WMEWMA (0.07595, 0.1066)
4	CEPPS Fast	ETX	ETX	ETX (0.1009, 0.1438)
5	ETX	CEPPS Fast	Hybrid Fast	CEPPS Fast (0.1223, 0.1893)
6	Hybrid Fast	Hybrid Fast	CEPPS Fast	Hybrid Fast (0.1292, 0.1874)
7	CEPPS Ultra-fast	CEPF	CEPPS Ultra-fast	CEPPS Ultra-fast (0.158, 0.2121)
8	Preamble Count Fast	CEPPS Ultra-fast	Hybrid Ultra-fast	CEPF (0.2243, 0.3642)
9	SNR	SNR	CEPF	SNR (0.2283, 0.4142)
10	Hybrid Ultra-fast	Hybrid Ultra-fast	SNR	Hybrid Ultra-fast (0.2352, 0.3491)
11	CEPF	Preamble Count Fast	Preamble Count Fast	Preamble Count Fast (0.25, 0.3175)
12	Preamble Count Ultra-fast	Preamble Count Ultra-fast	Preamble Count Ultra-fast	Preamble Count Ultra-fast (0.4239, 0.5881)

Mobile	[0% ... 33%]	[33%...66%]	[66%...100%]	[0%...100%] (mean, max)
1	CEPPS FWA	CEPPS FWA	CEPPS FWA	CEPPS FWA (0.05772, 0.07309)
2	CEPPS Fast	Hybrid Fast	Hybrid Fast	Hybrid Fast (0.133, 0.1962)
3	Hybrid Fast	CEPPS Fast	Preamble Count Fast	CEPPS Fast (0.1401, 0.1826)
4	CEPPS Ultra-fast	CEPF	CEPPS Fast	CEPPS Ultra-fast (0.1586, 0.22)
5	ETX	CEPPS Ultra-fast	CEPPS Ultra-fast	CEPF (0.1971, 0.3163)
6	WMEWMA	Hybrid Ultra-fast	Hybrid Ultra-fast	SNR (0.2098, 0.3801)
7	Four-Bit	Preamble Count Fast	SNR	Preamble Count Fast (0.2117, 0.3239)
8	SNR	SNR	WMEWMA	Hybrid Ultra-fast (0.2124, 0.3518)
9	CEPF	Preamble Count Ultra-fast	Four-Bit	WMEWMA (0.2364, 0.5618)
10	Preamble Count Fast	WMEWMA	Preamble Count Ultra-fast	Four-Bit (0.2491, 0.6281)
11	Hybrid Ultra-fast	Four-Bit	CEPF	ETX (0.2515, 0.4802)
12	Preamble Count Ultra-fast	ETX	ETX	Preamble Count Ultra-fast (0.3011, 0.5783)

Overall [0%...100%] (mean, max)
CEPPS FWA (0.04987, 0.07616)
Four-Bit (0.09755, 0.6281)
WMEWMA (0.11537, 0.5618)
CEPPS Fast (0.1204, 0.1919)
Hybrid Fast (0.1241, 0.1962)
ETX (0.13647, 0.4802)
CEPPS Ultra-fast (0.14945, 0.2203)
SNR (0.20607, 0.5396)
CEPF (0.20813, 0.3813)
Hybrid Ultra-fast (0.21595, 0.3601)
Preamble Count fast (0.2226, 0.3239)
Preamble Count Ultra-fast (0.36352, 0.5968)

Average Scenario* [0%...100%] (mean, max)
CEPPS FWA (0.05155, 0.07616)
Hybrid Fast (0.12915, 0.1962)
CEPPS Fast (0.12998, 0.1919)
Four-Bit (0.1511, 0.6281)
WMEWMA (0.15597, 0.5618)
CEPPS Ultra-fast (0.15695, 0.2203)
ETX (0.17558, 0.4802)
CEPF (0.2141, 0.4028)
Preamble Count fast (0.2198, 0.3239)
Hybrid Ultra-fast (0.22163, 0.3601)
SNR (0.23877, 0.5396)
Preamble Count Ultra-fast (0.35062, 0.5968)

***** 50% mobile, 25% LOS, 25% NLOS

E

Master Thesis Task Assignment

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement für Verteidigung, Bevölkerungsschutz und Sport VBS

armasuisse
Wissenschaft und Technologie W+T

Master Thesis Task Assignment for Michael Spuhler (D-ITET)

Ultra-fast Wireless Link Quality Estimator for Mobile Networks

Main advisor:	Dr. Vincent Lenders (armasuisse)
Advisor ETH	Dr. Franck Legendre (ETH Zürich)
Supervisor	Prof. Dr. B. Plattner (ETH Zürich)
Start Date:	January 17, 2012
End Date:	August 21, 2012 (approved absence during 27.2-23.3 for military service)

1 Introduction

Wireless technologies like IEEE 802.15.4 or IEEE 802.11a/b/g/n have gained tremendous popularity in the recent past. A fundamental problem of these networks relates to link quality estimation. The performance of routing, rate selection, handover, or jamming detection algorithms heavily depend on such an accurate and fast estimator. However, estimating the effective link quality in real-life wireless networks is a quite challenging task given the unpredictable and location-sensitive nature of wireless channels. In particular, low quality links with a packet delivery rate (PDR) of 10%<PDR<75% tend to be hard to estimate quickly and accurately with estimators that rely on the SNR, packet statistics and/or chip errors. Nevertheless, low quality links remain important to exploit in mobile and challenged networks where the node positions cannot be optimized to yield better performance.

2 Thesis goals

The goal of this thesis is to develop a new wireless link quality estimator that manages to quickly and accurately estimate low quality links (10%<PDR<75%) in mobile networking environments. In particular, we are interested in estimating the **probability** that a packet may successfully be received at the receiver. To achieve a rapid estimation, the student should develop and explore a novel approach that relies on chips synchronization errors in the preamble of the physical layer. Chips are the smallest unit of transmitted data and represent bits of information in a direct sequence spread spectrum system (DSSS). For example in IEEE 802.15.4, four bits (one symbol) are represented as a series of 32 chips. At the beginning of each frame, the IEEE 802.15.4 transmitter sends a preamble of 8 “zero” symbols which allows a receiving device to synchronize and lock onto the bit stream.

Relying on chip error information from the preamble is expected to provide two important benefits. First, unsuccessful packets where the synchronization fails may be detected and hence used to model the PDR. Second, the estimation model requires only a few symbols in the preamble and should therefore allow to quickly estimate the packet delivery.

To validate this approach, the student should implement a new estimator and compare it under realistic wireless channel conditions to a set of existing estimators. At least one candidate estimator should be considered from the following classes for comparison:

- Packet statistics (e.g. [7, 13])
- SNR (e.g. [5])

- Bit errors (e.g. [8, 10, 11])
- Chip errors (e.g., [12])
- Hybrid (e.g. [15-16])

If time permits, the estimator should be applied to a routing, rate selection, handover, or jamming detection algorithm in order to quantify the potential gain of the approach in a real application context.

3 Tasks

The tasks of this thesis consist of:

1. Literature study of existing wireless link quality estimators proposed in the literature (e.g., [1-16]).
2. Experimental characterization of the chip and symbol error patterns in the preamble of IEEE 802.15.4 frames using measurements with the URSP/USRP2 software defined radio. Experiments should be conducted under diverse wireless channel conditions (line-of-sight vs. non-line-of-sight, mobile vs. static, etc) to understand the behavior in different fading environments.
3. Developing a model to predict the PDR using chip/symbol level error information from the preamble of IEEE 802.15.4 frames.
4. Implementing the new link quality estimator on the USRP/USRP2 software defined radio for IEEE 802.15.4.
5. Evaluating your estimator experimentally in environments with multi-path fading, node mobility, and interference/jamming. Compare the performance of your estimator with a packet statistics-, an SNR-, a bit error-, a chip error- and a hybrid-based model.
6. If time permits, showing the superiority of your proposed estimator compared to existing estimators in an application of your choice (e.g., rate switching, routing, base station selection or jamming detection).

4 Deliverables

- At the end of the second week, a detailed time schedule of the thesis must be given and discussed with the main advisors.
- At the end of the first month, the student has to propose a methodology and preliminary model answering the second and third bullet, respectively.
- At the end of the second month, a short discussion of 15 minutes with the supervisor and the advisors will take place. The student has to talk about the major aspects of the ongoing work using slides.
- At the end of month four, another meeting with the supervisor will take place . At this point, the student should already have a preliminary version of the written report or at least a table of content to hand in to the supervisor. This preliminary version should be brought along to the short discussion.
- At the end of the thesis, a presentation of 15 minutes must be given at armasuisse and at ETH (in English) during a CSG group meeting. The presentations should give an overview as well as the most important details of the work. If possible, a demonstrator should be presented (offline after the talk).
- The final report should be written in English but may be written in German. It must contain a summary written in both English and German, the assignment and the time schedule. Its structure should include an introduction, an analysis of related work, and a complete documentation of all used hardware/software tools. Four written copies of the final report must be delivered to the main advisor along with CD that includes developments undergone during the thesis.

References

[1] The β-factor: Measuring Wireless Link Burstiness: Kannan Srinivasan, Maria Kazandjieva, Saatvik Agarwal and Philip Levis, ACM Sensys 2008.

[2] Four-Bit Wireless Link Estimation: Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson and Philip Levis, ACM HotNets-VI.

[3] Link Estimation and Routing in Sensor Network Backbones: Beacon-based or Data-driven?: Hongwei Zhang, Anish Arora, Prasun Sinha, IEEE Transactions on Mobile Computing 2009.

[4] All Bits are Not Equal – A Study of IEEE 802.11 Communication Bit Errors: Bo Han, Lusheng Ji, Seungjoon Lee, Bobby Bhattacherjee and Robert Miller, INFOCOM 2009.

[5] Predictable 802.11 Packet Delivery From Wireless Channel Measurements: Daniel Halperin, Wenjun Hu, Anmol Sheth, David Wetherall, SIGCOMM 2010.

[6] Efficient Packet Error Rate Estimation in Wireless Networks: Bo Han and Seungjoon Lee, Tridentcom 2007.

[7] A High Throughput Path Metric for MultiHop Wireless Routing: Douglas S. J. De Couto Daniel Aguayo John Bicket Robert Morris, Mobicom 2003.

[8] Efficient Error Estimating Coding: Feasibility and Applications: Binbin Chen, Ziling Zhou, Yuda Zhao, Haifeng Yu, SIGCOMM 2010.

[9] Efficient Channel-aware Rate Adaptation in Dynamic Environments: Glen Judd, X. Wang, and Peter Steenkiste, MobiSys 2008.

[10] Evaluation of Packet Error Rate in Wireless Networks: Ramin Khalili and Kavé Salamatian, MSWiM 2004.

[11] A new analytic approach to evaluation of Packet Error Rate in the fading Channels: Ramin Khalili and Kavé Salamatian, CNSR 2005.

[12] Fast and Accurate Packet Delivery Estimation based on DSSS Chip Error Measurements: Pirmin Heinzer, Vincent Lenders and Franck Legendre, INFOCOM 2012.

[13] A. Woo and D. Culler, “Evaluation of efficient link reliability estimators for low-power wireless networks,” Tech. Rep., 2003.

[14] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, “Four bit wireless link estimation,” in HotNets VI, 2007.

[15] N. Baccour, A. Koubaa, H. Youssef, M. Ben Jamaa, D. do Rosario, M. Alves, and L. Becker, “F-lqe: A fuzzy link quality estimator for wireless sensor networks,” in Wireless Sensor Networks. Springer, 2010. pp. 240–255.

[16] C. Boano, M. Zuniga, T. Voigt, A. Willig, and K. R¨omer, “The triangle metric: Fast link quality estimation for mobile wireless sensor networks,” in ICCCN, 2010.

armasuisse
Science and Technology
C4I Networks

Dr. Vincent Lenders
Thun, January 17th 2012

F

Abbreviations

ADC	Analog-Digital Converter
AM	Amplitude Modulation
ASL	Asymmetry Level
ASNR	Average SNR
BER	Bit Error Rate
CEPF	Chip Errors per Frame
CEPS	Chip Errors per Symbol
CEPPS	Chip Errors per Preamble Symbol
CD-ROM	Compact Disc Read-Only Memory
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DAC	Digital-Analog Converter
dB	Decibel
DSSS	Direct Sequence Spread Spectrum
EEC	Error Estimation Codes
EWMA	Exponentially Weighted Moving Average
FCF	Frame Control Field
FCS	Frame Check Sequence
FM	Frequency Modulation
FPGA	Field Programmable Gate Array
FWA	Filtered Weighted Average
GMSK	Gaussian Minimum Shift Keying
GigE	Gigabit Ethernet
GNU	GNU's Not Unix!
IEEE	Institute of Electrical and Electronics Engineers
kbit/s	kilo bits per second
LOS	Line of Sight
LQ	Link Quality
LQE	Link Quality Estimator
LQI	Link Quality Indicator
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Media Access Control
MB	Megabyte
MFR	MAC Footer
MHz	Mega-Hertz
MPDU	MAC Protocol Data Unit
MS	Mega Samples
MSK	Minimum-Shift Keying
NLOS	None Line of Sight
O-QPSK	Offset Quadrature Phase-Shift Keying
OSI	Open Systems Interconnection
PAN	Personal Area Network
PC	Personal Computer
PER	Packet Error Rate

PHR	PHY Header
PHY	Physical Layer
PN	Pseudo-Random Noise
PPDU	PHY Protocol Data Unit
PDR	Packet Delivery Ratio
RF	Radio Frequency
RNP	Required Number of Packet Retransmissions
RSS	Received Signal Strength
Q-Phase	Quadrature-Phase
QPSK	Quadrature Phase-Shift Keying
SDR	Software Defined Radio
SF	Stability Factor
SFD	Start-of-Frame Delimiter
SHR	Synchronization Header
SNR	Signal to Noise Ratio
SPRR	Smoothed Packet Reception Ratio
SWIG	Simplified Wrapper and Interface Generator
UCLA	University of California, Los Angeles
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
WMEWMA	Window Mean Exponentially Weighted Moving Average
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Bibliography

[1] M. Strasser, B. Danev, and S. Čapkun, "Detection of reactive jamming in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 7, no. 2, p. 16, 2010.

[2] *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE Computer Society Std.

[3] M. Ettus, "Ettus research llc." [Online]. Available: www.ettus.com

[4] T. Schmid, "Gnu radio 802.15. 4 en-and decoding," UCLA NESL, Tech. Rep., 2005.

[5] P. Heinzer, V. Lenders, and F. Legendre, "Fast and accurate packet delivery estimation based on dsss chip errors," in *INFOCOM 2012*. Orlando, Florida, USA: IEEE, 2012.

[6] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2005, pp. 46–57.

[7] N. Baccour, L. Mottola, Z. Niga, Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: a survey," *ACM Trans. Sens. Netw*, 2012.

[8] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin, "Statistical model of lossy links in wireless sensor networks," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, ser. IPSN '05. Piscataway, NJ, USA: IEEE Press, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1147685.1147701

[9] A. Woo, Alec and A. Culler, David, "Evaluation of efficient link reliability estimators for low-power wireless networks," 2003. [Online]. Available: http://techreports.lib.berkeley.edu/accessPages/CSD-03-1270.html

[10] H. Zhang, L. Sang, and A. Arora, "Unravelling the subtleties of link estimation and routing in wireless sensor networks," Tech. Rep., 2008.

[11] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 packet delivery from wireless channel measurements," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 159–170, August 2010. [Online]. Available: http://doi.acm.org/10.1145/1851275.1851203

[12] Chipcon, *CC2420 - 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*, 2009. [Online]. Available: www.chipcon.com

[13] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "An empirical study of low-power wireless," *ACM Trans. Sen. Netw.*, vol. 6, no. 2, pp. 16:1–16:49, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1689239.1689246

[14] M. Zamalloa and B. Krishnamachari, "An analysis of unreliability and asymmetry in low-power wireless links," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 2, p. 7, 2007.

[15] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 1–13. [Online]. Available: http://doi.acm.org/10.1145/958491.958493

[16] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "Complex behavior at scale: An experimental study of low-power wireless sensor networks," Citeseer, Tech. Rep., 2002.

[17] C. Boano, T. Voigt, A. Dunkels, F. Osterlind, N. Tsiftes, L. Mottola, and P. Suarez, "Poster abstract: Exploiting the lqi variance for rapid channel quality assessment," in *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*. IEEE Computer Society, 2009, pp. 369–370.

[18] M. Senel, K. Chintalapudi, D. Lal, A. Keshavarzian, and E. Coyle, "A kalman filter based link quality estimation scheme for wireless sensor networks," in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*. IEEE, 2007, pp. 875–880.

[19] N. Baccour, A. Koubaa, M. Ben Jamaa, H. Youssef, M. Zuniga, and M. Alves, "A comparative simulation study of link quality estimators in wireless sensor networks," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, sept. 2009, pp. 1 –10.

[20] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*. Ieee, 2005, pp. 364–369.

[21] N. Baccour, A. Koubaa, H. Youssef, M. Ben Jamaa, D. do Rosario, M. Alves, and L. Becker, "F-lqe: A fuzzy link quality estimator for wireless sensor networks," in *Wireless Sensor Networks*, ser. Lecture Notes in Computer Science, J. Silva, B. Krishnamachari, and F. Boavida, Eds. Springer Berlin / Heidelberg, 2010, vol. 5970, pp. 240–255.

[22] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, "Four-bit wireless link estimation," in *Proceedings of the Sixth Workshop on Hot Topics in Networks (HotNets VI)*, 2007.

[23] C. Gomez, A. Boix, and J. Paradells, "Impact of lqi-based routing metrics on the performance of a one-to-one routing protocol for ieee 802.15. 4 multihop networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, p. 6, 2010.

[24] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 125–138. [Online]. Available: http://doi.acm.org/10.1145/990064.990081

[25] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin, "Temporal properties of low power wireless links: modeling and implications on multi-hop routing," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '05. New York, NY, USA: ACM, 2005, pp. 414–425. [Online]. Available: http://doi.acm.org/10.1145/1062689.1062741

[26] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "a high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, pp. 419–434, 2005. [Online]. Available: http://dx.doi.org/10.1007/s11276-005-1766-z

[27] C. Boano, M. Zu?niga, T. Voigt, A. Willig, and K. Ro?mer, "The triangle metric: Fast link quality estimation for mobile wireless sensor networks," in *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, aug. 2010, pp. 1 –7.

[28] B. Chen, Z. Zhou, Y. Zhao, and H. Yu, "Efficient error estimating coding: feasibility and applications," in *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4. ACM, 2010, pp. 3–14.

[29] R. Khalili and K. Salamatian, "A new analytic approach to evaluation of packet error rate in wireless networks," 2005.

[30] R. Kave, R. Khalili, and K. Salamatian, "Evaluation of packet error rate in wireless networks," 2004.

[31] Y. Qin, Z. He, and T. Voigt, "Towards accurate and agile link quality estimation in wireless sensor networks."

[32] Y. Hu, A. Perrig, and D. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wireless Networks*, vol. 11, no. 1-2, pp. 21–38, 2005.

[33] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad hoc networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[34] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," in *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, vol. 31. San Antonio, TX, 2002, pp. 193–204.

[35] B. Potter, "Wireless security's future," *Security & Privacy, IEEE*, vol. 1, no. 4, pp. 68–72, 2003.

[36] L. Zhou and Z. Haas, "Securing ad hoc networks," *Network, IEEE*, vol. 13, no. 6, pp. 24–30, 1999.

[37] W. Xu, T. Wood, W. Trappe, and Y. Zhang, "Channel surfing and spatial retreats: defenses against wireless denial of service," in *Proceedings of the 3rd ACM workshop on Wireless security*. ACM, 2004, pp. 80–89.

[38] M. Çakiroglu and A. Özcerit, "Jamming detection mechanisms for wireless sensor networks," in *Proceedings of the 3rd international conference on Scalable information systems*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 4.

[39] Y. Law, M. Palaniswami, L. Hoesel, J. Doumen, P. Hartel, and P. Havinga, "Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, p. 6, 2009.

[40] Y. Law, P. Hartel, J. den Hartog, and P. Havinga, "Link-layer jamming attacks on s-mac," in *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*. IEEE, 2005, pp. 217–225.

[41] A. Wood, J. Stankovic, and G. Zhou, "Deejam: Defeating energy-efficient jamming in ieee 802.15. 4-based wireless networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on.* IEEE, 2007, pp. 60–69.

[42] A. Wood, J. Stankovic, and S. Son, "Jam: A jammed-area mapping service for sensor networks," in *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE.* IEEE, 2003, pp. 286–297.

[43] *Modern Communications Jamming Principles and Techniques.* Artech House, 2004.

[44] *EW101: A First Course in Electronic Warfare.* Artech House, 2001.

[45] G. Noubir and G. Lin, "Low-power dos attacks in data wireless lans and countermeasures," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 29–30, 2003.

[46] Z. Alliance, "Zigbee specification," *ZigBee document 053474r06, version*, vol. 1, 2005.

[47] "The zigbee alliance." [Online]. Available: http://www.zigbee.org/

[48] "First glimpse, home awareness systems." [Online]. Available: http://www.firstglimpsemag.com

[49] J. Notor, A. Caviglia, and G. Levy, "Cmos rfic architectures for ieee 802.15. 4 networks," *Cadence Design Systems, Inc*, 2003.

[50] T. S. D. R. Inc., "What is software defined radio?" [Online]. Available: www.sdrforum.org

[51] B. E. et. al., "Gnu radio." [Online]. Available: www.gnuradio.org

[52] "Python programming language - official website." [Online]. Available: http://www.python.org

[53] "The c++ resources network, 2011." [Online]. Available: http://www.cplusplus.org

[54] "Simplified wrapper and interface generator." [Online]. Available: http://www.swig.org

[55] "Supported gnu radio hardware." [Online]. Available: http://comsec.com/wiki?GnuRadioHardware.

[56] P. Heinzer, "Wireless link quality estimation in mobile networks," Master's thesis, ETH Zürich, 2011.

[57] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders, "Wifire: a firewall for wireless networks," in *Proceedings of the ACM SIGCOMM 2011 conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 456–457. [Online]. Available: http://doi.acm.org/10.1145/2018436.2018518