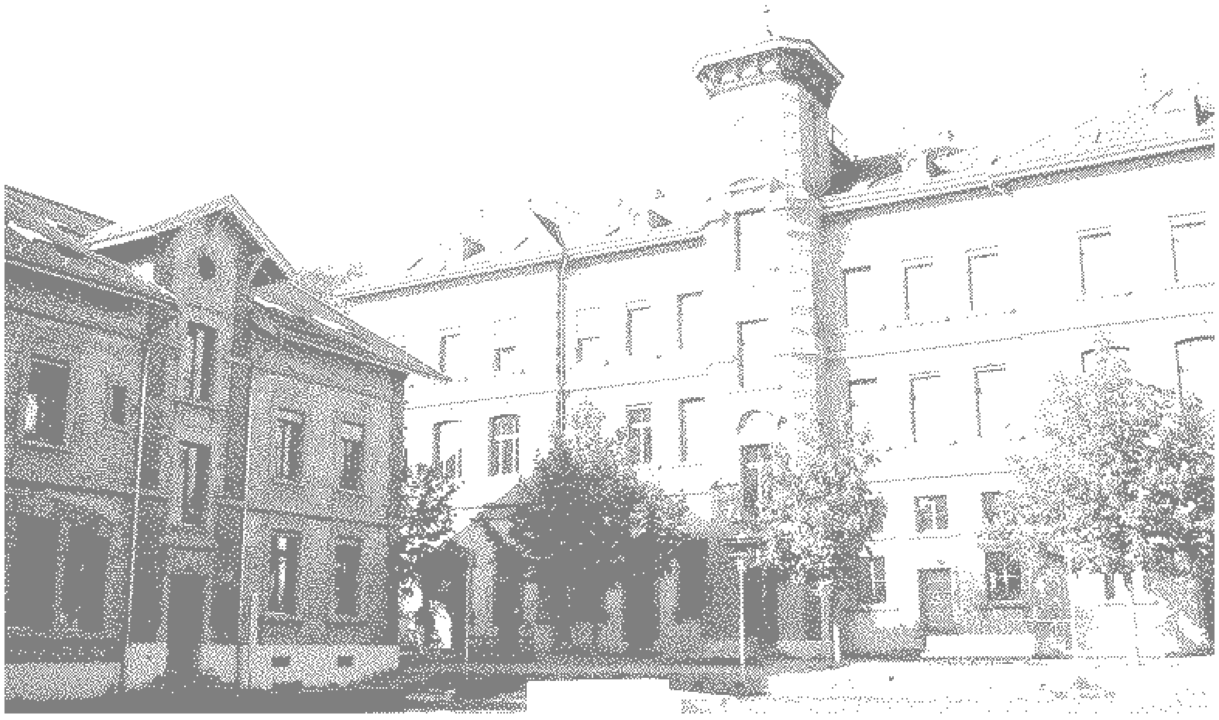
b
UNIVERSITÄT
BERN

Link-Quality Aware Run-Time Adaptive Forward Error Correction Strategies in Wireless Sensor Networks

P. Hurni, T. Braun

Technischer Bericht IAM-11-003 vom 22.11.2011

Institut für Informatik und angewandte Mathematik, www.iam.unibe.ch



Link-Quality Aware Run-Time Adaptive Forward Error Correction Strategies in Wireless Sensor Networks

Philipp Hurni, Torsten Braun

Technischer Bericht IAM-11-003 vom 22.11.2011

CR Categories and Subject Descriptors:
C.2.1 [Computer-Communication Networks]: Network Architecture and Design; C.2.2 [Computer-Communication Networks]: Network Protocols; C.2.3 [Computer-Communication Networks]: Network Operations; C.2.4 [Computer-Communication Networks]: Distributed Systems

General Terms:
Wireless Sensor Networks, Forward Error Correction

Additional Key Words:
Run-time Parameter Adaptation

Institut für Informatik und angewandte Mathematik, Universität Bern

Abstract

This paper investigates the potential of Forward Error Correction (FEC) mechanisms and dynamic/run-time adaptive FEC variants in Wireless Sensor Networks. We implemented eight different Error Correction Codes (ECCs), ranging from simple bit-repetition schemes over Hamming codes to complex Bose-Chaudhuri-Hocquenghem codes, and three run-time adaptive FEC schemes which adapt the correctional power of ECCs to the current link quality. The paper evaluates the computational costs and the resulting benefits of the FEC schemes under real-world conditions in a distributed testbed environment.

Link-Quality Aware Run-Time Adaptive Forward Error Correction Strategies in Wireless Sensor Networks

Abstract. This paper investigates the potential of Forward Error Correction (FEC) mechanisms and dynamic/run-time adaptive FEC variants in Wireless Sensor Networks. We implemented eight different Error Correction Codes (ECCs), ranging from simple bit-repetition schemes over Hamming codes to complex Bose-Chaudhuri-Hocquenghem codes, and three run-time adaptive FEC schemes which adapt the correctional power of ECCs to the current link quality. The paper evaluates the computational costs and the resulting benefits of the FEC schemes under real-world conditions in a distributed testbed environment.

1 Introduction

Wireless Sensor Networks (WSNs) are growing in popularity for various applications: they are increasingly used in healthcare, in business automation and logistics, the automotive industries or as central technology in various research projects of the natural sciences. A key factor for the proliferation of WSN technologies is the reliability of the communication: it is crucial for many applications that the sensed data is delivered quickly and reliably across the network. The low-power wireless channel is, however, often prone to many hard-to-predict wireless phenomena. Transmission errors are likely to occur due to a variety of reasons, ranging from multipath propagation effects, fading, scattering and reflection to interferences with other ongoing transmissions. Often, the channel exhibits a timely and spatially variable bit error rate (BERs) in the range of 10^{-4} or even higher, which results in packet loss rates ranging from less than 1% to sometimes far more than 10%-20%. In wired networks, BERs are usually several magnitudes lower (e.g., DSL networks [1] exhibit BERs of maximum 10^{-7}, but usually in the range of 10^{-9}).

Automatic Repeat reQuest vs. Forward Error Correction: High error rates on the link level inevitably lead to a higher rate of corrupted packets, rendering the retrieved data unusable. The simplest and most naive way to deal with transmission errors on the link layer is to retransmit the same packet again until it has been correctly received or a maximum retry count is reached. RFC 3366 [2] describes different *Automatic Repeat reQuest (ARQ)* schemes that are used today in different kinds of networks, ranging from various wireless networks to wireline and optical networks. In ARQ, the sender appends a cyclic redundancy checksum (CRC) [3] to the transmitted packet and waits for the acknowledgement (ACK) from the receiver. In order to reliably determine the integrity of the packet, the receiver calculates the CRC across the received payload again and compares it to the received checksum. If both CRCs match, the receiver confirms the successful reception to the sender with an ACK. If the sender does not receive an ACK within a certain time window, it assumes that the transmission attempt has failed and invokes a retransmission.

A sophisticated mechanism to cope with packet corruption is the concept of *Forward Error Correction (FEC)* [4]. FEC is used in a wide range of commercial and industrial products where data is transmitted over erroneous channels and where, henceforth, bit errors are likely to occur. FEC is based upon *Error Correcting Codes (ECCs)*, which can detect and correct a certain amount of bit errors in a sequence of bits. In FEC, the sender computes parity information according to the applied ECC over the data bits and adds this redundant information to the payload. At the receiver, the decoder of the applied ECC checks the received data bits for errors by taking this parity information into account. FEC schemes hence generally introduce an overhead with respect to computation (encoding/decoding) and the number of bits that need to be transmitted. This overhead can however pay off with an increased packet delivery rate (PDR) and a reduction of the (re)transmission overhead and latency, since in case an error occurs, the correction can immediately take place after packet reception.

Towards Dynamic Run-time Adaptive FEC Strategies: WSNs are typically configured for their intended deployment scenario at compile-time, which can lead to suboptimal parameter settings if the discovered network conditions deviate significantly from the expectations prior to network deployment. In practice, it is rather impossible to predict the frequency and severity of signal distortions, and hence, the probability of bit errors and the patterns with which they occur. Therefore, it is also impossible to choose an adequate ECC code which exhibits "just enough" correctional power in advance of network deployment. The application of powerful ECCs makes sense when the channel exhibits a high BER, it however constitutes a waste of resources in case the network's link qualities are good. By facing the challenge of selecting the *best* ECC for a given link and channel in our own indoor WSN testbed, we found that the application of run-time adaptive FEC mechanisms for WSNs operating under timely and spatially variable channel conditions has generally not been studied at all. Dynamic FEC schemes allocating the correctional power of ECCs in an *on-demand* manner could be a viable alternative to static FEC with network-wide ECC configuration, where the link conditions are not taken into account.

The contributions of this paper are threefold: first, we study the performance of a wide range of ECCs preconfigured in a statical manner in several real-world WSN experiments. Second, we evaluate whether run-time adaptive ECC selection strategies can cope with the particularities of a distributed WSN, where link qualities are expected to exhibit temporary short-lived variations of the link quality, as well as variability across the different links of the network. We propose three run-time adaptive FEC strategies, which allocate the correctional power of ECCs in an *on-demand* manner. Third, we make the library of ECC codes *libECC* [5] tailored for the most frequent microcontroller on WSN platforms, the MSP430 [6], publicly available, since such a library has to date been missing.

Section 2 introduces into recent work in this research area and relates it to this paper's contribution. Section 3 discusses the selected implemented ECCs and their properties. Section 4 illustrates the proposed adaptive FEC schemes, which are - together with the static ECC schemes - evaluated in different real-world scenarios in Section 5. Section 6 concludes the paper.

2 Related Work

A recent analysis of related work and literature on FEC mechanisms and adaptive schemes in particular conveyed that the related work in this field is rather limited. The performance of a small set of rather simple ECC codes has been investigated with real-world hardware platforms and different radio transceivers, e.g., the Chipcon CC1000 [7] in [8], or the RFM TR1001 [9] in [10], both using a frequency around 868 MHz and on-off-keyed (OOK) modulation. In many studies, the examined topologies were far from real-world environmental conditions, e.g., the topology used in [10] consisted of 16 sensor nodes in a line of sight with one sender. Similarly, in [11] the distance between any two nodes was fixed to only 30 cm. Often enough, crucial wireless phenomena were not taken into account, e.g., signal attenuation through concrete walls and floors. The study [8] conveyed that the most frequently occurring error patterns in indoor experiments were 1-bit and 2-bit errors across one packet. The entire analysis however was conducted in the absence of any other ongoing concurrent traffic. The authors conclude that in their case, complex ECCs with a high correctional power are hence not necessarily required, but that this trade-off needs to be investigated in environments with higher error rates. The authors of [10] come to a similar conclusion. As in [8] however, the effect of concurrent transmissions or signal attenuation due to concrete walls and floors, which have a crucial impact in real-world WSNs, are not studied at all, since in the scenario of [10], only one node is broadcasting packets which are then logged by the remaining 15 nodes. The remaining nodes are all arranged in a line of sight with the sender.

To the best of our knowledge, extensive real-world experiences with run-time adaptive FEC schemes applied in WSNs do not exist to date. A simulation-based study on adaptive FEC schemes was conducted in [12]. The study proposes so-called FEC-level adaptation (FECA), in which the parameters of one ECC driving the amount of parity bits are adapted at run-time. In FECA, the parameters n and k of BCH are adapted in three steps, in order to increase and decrease the amount of parity bits depending on the channel conditions. FEC adaptations are activated by either a packet loss or the timeout of a back-off timer. FECA [12], however, is not particularly designed for sensor networks, but rather for IEEE 802.11-based wireless mobile ad hoc networks. The authors use the network simulator ns-2 and apply a generic wireless channel error model. They conclude that FECA performs better than the application of statically configured FEC mechanisms, given that the error rates do not oscillate too rapidly.

In the recent past, the application of simulation tools have been identified as a general drawback of many ad hoc and sensor network studies. Inappropriate parameter settings, unrealistic channel, traffic and error models of many simulation studies have led to an erosion in trust in simulation results [13] [14]. The trend in research on WSNs has clearly shifted towards experimental feasibility studies of protocols on real-world devices. This paper clearly distinguishes from the mentioned related work by its real-world analysis of eight different ECCs and several adaptive FEC strategies in realistic deployment topologies.

3 Forward Error Correction Library libECC

Our library of ECCs *libECC* [5] implemented during this study consists of eight different ECCs from four basic classes, each with different degrees of redundancy and different block sizes. Figure 1 illustrates the block size, the correctional power per block and the amount of redundancy of these ECCs. The parameters in brackets define the particular ECC configurations, but their semantics may differ among the classes (e.g., payload/parity bits, repetition factor, etc.):

- The Repetition Code [15] is the most simple and naive method to introduce an error correction capability, since it simply stretches the number of input bits and then decodes using majority logic decoding. We refer to our implementation of the repetition code as REP(3,8) throughout this chapter, because it stretches the input bits by a factor of 3 and operates on a block unit of one byte. The advantage of REP(3,8) is the low computational overhead, which comes at the cost of the resulting large portion of parity information, in particular when comparing with more complex codes.
- The Hamming code is a linear error-correcting code invented by Richard Hamming in 1950 [16]. This simple code can be seen as a cornerstone for the development of modern ECCs. *libECC* contains the popular Hamming(7,4) code, which encodes 4 data bits into 7 encoded bits.
- The Double Error Correction Triple Error Detection (DECTED) proposed in [17] is similar to Hamming. DECTED is able to correct up to two bit errors and can detect up to three adjacent bit errors per block. The DECTED(16,8) code implemented in *libECC* takes 8 bits as input and creates an encoded word of 16 bits, hence exhibiting comfortable block size units of 1 byte for the raw data and 2 bytes for the encoded data.

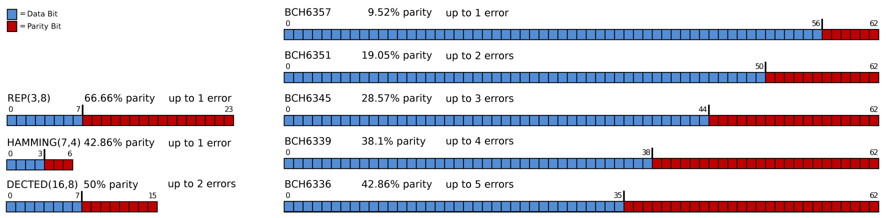

= Data Bit
= Parity Bit
REP(3,8) 66.66% parity up to 1 error
0 7 23
HAMMING(7,4) 42.86% parity up to 1 error
0 3 6
DECTED(16,8) 50% parity up to 2 errors
0 7 15
BCH6357 9.52% parity up to 1 error
0 56 62
BCH6351 19.05% parity up to 2 errors
0 50 62
BCH6345 28.57% parity up to 3 errors
0 44 62
BCH6339 38.1% parity up to 4 errors
0 38 62
BCH6336 42.86% parity up to 5 errors
0 35 62

Fig. 1: libECC Codes: Block Size, Correctional Power, Redundancy

- The Bose-Chaudhuri-Hocquenghem (BCH) codes, invented in 1959 by Hocquenghem [18] and independently in 1960 by Bose and Chaudhuri [19], are the most complex ECCs within *libECC*. BCH codes belong to the class of *cyclic* block codes. BCH(n, k) encodes k data bits into n encoded bits. *libECC* implements BCH codes with 63 bits block size, allocating 64 bits for practical reasons, since 64 bit blocks nicely fit into the `long long` datatype. BCH(63,57), BCH(63,51), BCH(63,45), BCH(63,39), and BCH(63,36) implemented in *libECC* can correct 1,2,3,4 or 5 errors per block.

Memory Footprint: WSN nodes are heavily restrained with respect to the available computational power and memory resources. Having this restriction in mind, we cautiously kept the memory footprint of *libECC* as low as possible. Although many of the implemented ECCs require large matrices, syndrome value lookup tables and polynomials, the current version of *libECC* consumes only roughly 10 KBytes for the text segment and 326 bytes for the data segment. *libECC* allocates one statically allocated data structure, where detailed information about the decoding and correction procedures is stored after each encoding and decoding operation, e.g., the number of corrected errors, the size of the decoded payload, the duration of the decoding. *libECC* uses the basic data types of the mspgcc compiler [20], but could easily be ported to other platforms. We have successfully tested the library on different MSP430-based WSN platforms and operating systems, such as the MSB430 [21] using ScatterWeb^2 OS [22] and TelosB [23] using Contiki OS [24].

4 Adaptive Forward Error Correction

WSN nodes are typically preconfigured with the most crucial parameter settings at compile-time, hence much before the actual network deployment. As pointed out in Section 1, this can result in suboptimal performance in case the actually encountered environment differs much from the conditions expected during planning. When deciding to apply FEC, a crucial design question consists in selecting the appropriate ECC. While a too weak code might not be able to correct many errors, a too strong code would waste precious time and energy for encoding/decoding and transmitting the additional parity data. With energy and processing power being limited resources in WSNs, this decision is of particularly high importance. The channel quality is almost certain to exhibit variations over time and can differ heavily across the different links, which renders the choice of a network-wide "optimal" ECC prior to network deployment impossible.

With our proposed adaptive FEC approaches, we address this crucial design problem of choosing an appropriate ECC code by adaptively selecting the ECC codes for each individual link at run-time instead of applying network-wide settings prior to network deployment. In each of the adaptive FEC schemes we introduce in the following, the sender decides which ECC to use based on past transmissions to the target node, since the channel quality and transmission success probability is usually tied to a particular link. Optimally, the selected

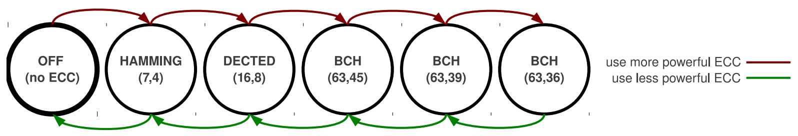


Fig. 2: Adaptive Forward Error Correction: ECC Selection Sequence

ECC adds as little overhead as possible, but provides just enough correctional power to overcome the encountered error patterns. Figure 2 illustrates the finite-state-machine based concept of selecting different ECCs at run-time. The states are kept in a table for each neighbor and denote the current ECC that is used on the link to that neighbor. The ECCs contained in this chain of states are increasing in their correctional power from the leftmost to the rightmost state, and similarly with respect to computational and parity overhead. In the default *OFF* state, the node does not encode the payload at all. Hamming(7,4), can correct one error per block, DECTED(16,8) up to two errors per block, and BCH(63,45), BCH(63,39), and BCH(63,36) can correct up to 3,4, and 5 errors per block, respectively. The ability to correct multiple adjacent errors per block is a crucial advantage of the BCH variants, because random bit errors tend to occur temporally correlated, e.g., during the transmission of the same byte.

Stateful Adaptive FEC (SA-FEC): SA-FEC is the most simple adaptive FEC mechanism thinkable. It selects the ECC for the next transmission to the specified destination according to the success of the last one. This means that if the last transmission of an ECC packet has been successful, SA-FEC selects the next less powerful ECC. If not, SA-FEC selects the next more powerful ECC, c.f. Figure 2. The decision depends only on the success of the last transmission, i.e., whether a subsequent ACK has been received or not. Since SA-FEC only takes the recent past into account, it reacts quickly to link quality changes.

Stateful History Adaptive FEC (SHA-FEC): SHA-FEC is similar to SA-FEC, but maintains a variable denoting the currently used ECC *and* a history of entries representing the recent past transmissions, which is manipulated in a FIFO-manner. In case a transmission succeeds and a MAC-level acknowledgement (ACK) is received, SHA-FEC stores an integer value representing the next *lower* ECC into the history, since it assumes that a lower ECC would have sufficed as well. In case the transmission fails and no ACK is received, SHA-FEC stores a value representing the next *higher* ECC, assuming that more correctional power is necessary. We sticked to a history size of five entries throughout all our evaluations. The selection of the ECC used for the next transmission is based on calculating the rounded average of the values stored in the history. As soon as the majority of entries represents the *lower* ECC, the node switches back one step in the state chain depicted in Figure 2. SHA-FEC reacts less quickly to link quality changes than SA-FEC, but avoids fast oscillation effects of selecting different ECCs for each transmission. The history-based selection mechanism provides a means to cope with longer-term interferences, since the mechanism does not immediately fall back to a less powerful ECC after one successful transmission, but waits until a couple of transmission have succeeded and then only stepwise shifts back in the state chain illustrated in Figure 2.

Stateful Sender Receiver Adaptive FEC (SSRA-FEC): SSRA-FEC extends SHA-FEC by taking into account an additional history containing *receiver* information into the ECC selection process. The entries in this new history denote the maximum number of corrected errors per block, counted by the receiver node across the last packet, which exactly equals the number of errors per block that can be corrected by the ECCs in the state chain in Figure 2. This number is transmitted in the ACK in case the frame could be correctly decoded. To determine the ECC for the next transmission, the sender computes the rounded average of the unrounded averages of both histories. SSRA-FEC hence attempts to implement a *closed-loop* parameter adaptation, where not only the *sender knowledge* is taken into account, but also *feedback* from the receiver. The mechanism involving two histories was designed to find a "suitable" ECC quickly by integrating receiver feedback and therefore having more indications to rely upon.

5 Experimental Evaluation

We evaluated the different ECCs of *libECC* and the three adaptive mechanisms SA-FEC, SHA-FEC and SSRA-FEC on the MSB430 [21] platform with the ScatterWeb^2 OS [22]. The MSB430 has a CC1020 [25] byte-level radio operating in the 804-940 MHz ISM band with a transmission rate of 19.2 kbit/s and on-off keyed modulation. We used the default IEEE 802.11-like ScatterWeb^2 OS CSMA layer, which does not duty-cycle the radio in any form. We explicitly chose a non duty-cycled MAC in order to safely exclude the potential influence of radio duty-cycling on the resulting PDRs of the different ECCs.

5.1 Computational Complexity

We first evaluated the ECC implementations of *libECC* with respect to computational complexity. We therefore measured the time needed for encoding different payload sizes, ranging from 3 to 52 bytes of data and for decoding the corresponding code words. Figure 3 depicts the encoding and decoding times for the given payload sizes. The figures display the mean values and standard deviations of 1000 encoding/decoding operations for each code in *libECC*. Obviously,

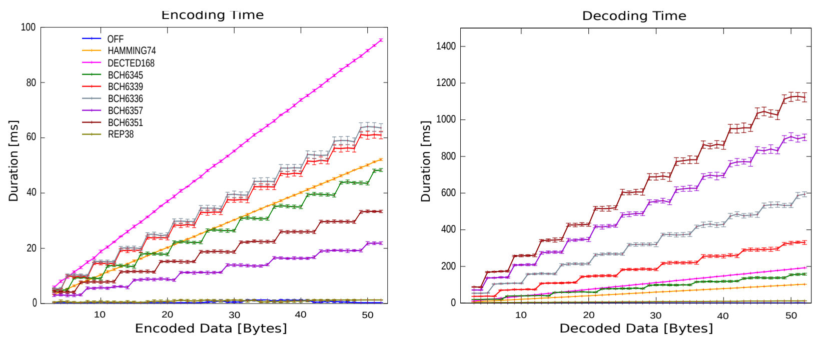


Fig. 3: Encoding and Decoding Time vs. Payload Bytes

the encoding and decoding durations grow linearly with the amount of payload bytes for each ECC examined. The figures, however, clearly display a trade-off between the correctional power of the ECCs and the required computational complexity. The repetition code REP(3,8) for example is by far the fasted ECC, since it requires few computation and mainly consists of memory copying operations. The hidden costs of the repetition code, however, lies in the resulting large size of the parity data, which can significantly impact on the energy consumption when transmitting the frame. The BCH code exhibits a characteristic step-shaped pattern in the encoding and decoding times, which can be explained by the blockwise encoding and decoding process of the BCH code and the large block size units. The complex BCH variants with higher correctional power which add more parity information clearly exhibit longer encoding/decoding times.

5.2 Energy Cost Estimation of Forward Error Correction

In order to be able to quantify the energy cost of FEC, we measured the current draw of the MSB430 node with the CPU in the two operation modes of the MSP430 [6] chip used by ScatterWeb[2] OS, namely the *fully active mode* and the *Low Power Mode 1 (LPM1).* In LPM1, the CPU and master clock are disabled, but timers and peripheral interrupts are still enabled. We measured the node's current draw to account to 3.75 mA for the fully active operation mode and 1.92 mA for LPM1. This difference might vary by a few percent dependent on the node chosen for the measurement, since different nodes can exhibit small variations in the power consumption [26]. With the measured current draws, we derive a cost function to quantify the energy costs of the implemented ECCs, taking into account the time it takes to encode or decode a payload. We define $P_{Default}(t)$ as the power consumption function of the node without using FEC, and $P_{FEC}(t)$ as the respective function of the node applying FEC. We define the *cost* of the application of FEC as the *additional power* consumed while encoding and decoding. The power cost function $P_{cost}(t)$ can then be denoted as

$$P_{cost}(t) = P_{FEC}(t) - P_{Default}(t)$$

Integrating the measured values of the MSB430 into the equation yields:

$$P_{cost}(t) = (I_{FEC} - I_{Default}) \cdot U_{supply} = 1.83\ mA \cdot 4\ V = 7.3\ mW$$

I_{FEC} corresponds to the average current used with the CPU in the active mode, while $I_{Default}$ is the average current used in LPM1. The encoding and decoding durations depicted in Figure 3 can hence be linearly mapped using the cost function $P_{cost}(t)$ to corresponding energy cost functions. For example, the energy cost $E_{enc/dec}$ of an encoding and decoding operation at sender and receiver of, e.g., a BCH(63,45) encoded payload of 32 bytes, which takes roughly $T_{enc} = 30\ ms$ for encoding and $T_{dec} = 100\ ms$ for decoding, calculates as

$$E_{enc/dec} = E_{enc} + E_{dec} = T_{enc} \cdot P_{cost} + T_{dec} \cdot P_{cost} = 0.95\ mJ$$

The subsequent evaluations of this study exclusively focus on reliability measures (link-level PDRs and end-to-end PDRs) and leave aside the energy-efficiency aspect. Nevertheless, we briefly illustrate the potential *benefit* of applying ECCs with respect to the energy consumption, continuing the exemplary calculation of $E_{enc/dec}$ noted above: we assume that a retransmission consists in a 50 ms frame T_f and a 20 ms ACK transmission T_{ack}. Based upon values measured in [26] of the mean currents of a MSB430 with the radio receiving (I_{rcv} =23.53 mA) and transmitting (I_{tx} =37.48 mA), the energy costs E_{re} of a retransmission consisting of the costs E_{snd} at the sender and E_{rcv} at the receiver account to:

$$E_{re} = E_{snd} + E_{rcv} = (T_f \cdot (I_{tx} + I_{rx}) + T_{ack} \cdot (I_{tx} + I_{rx})) \cdot U_{supply} = 17.08\; mJ$$

This cost E_{re} of an *entire retransmission* is almost twenty times higher than the encoding and decoding operations in case a mediocre ECC is applied. In the presence of unreliable links with bit errors corrupting a significant share of the packets, the application of FEC may hence even make sense from an energy point of view alone. A general judgment on this question can, however, not be answered without an assumption about the channel quality and hence the frequency of bit errors, and would require substantial further investigations.

5.3 Single-Link Scenario - Indoor and Outdoor Links

This section evaluates the different ECCs of *libECC* in two single-link scenarios, which are displayed in Figure 4 as links $A \rightarrow B$ and $A \rightarrow C$. Nodes A and B are placed in the two most distant offices on the same floor of the same building. Node A is placed on the windowsill in one office, and the signal from its antenna has to pass five concrete walls in order to reach node B, with a distance of roughly 25m. The second link between A and C is an outdoor link with a line of sight, since both nodes are placed on the windowsills of two office buildings facing each other. The line of sight distance between nodes A and C is 48m.

In the single-link experiments, we study and compare the real-world performance of the different ECCs under comparable conditions. All measurements were captured during the night and on the weekends, in order to minimize interferences caused by people working in the building and using GSM cellphones, IEEE 802.11 devices, wireless headphones, microwave ovens, or other potential sources of interference, which might affect the channel quality and hence call the comparability of the results into question. Since fog or increased humidity absorb radio signals, the measurements of the outdoor link $A \rightarrow C$ were gained

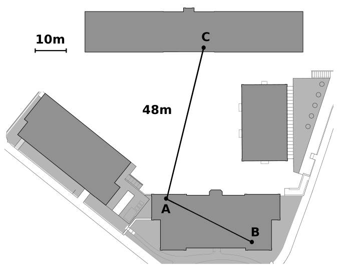


Fig. 4: Indoor and Outdoor Link

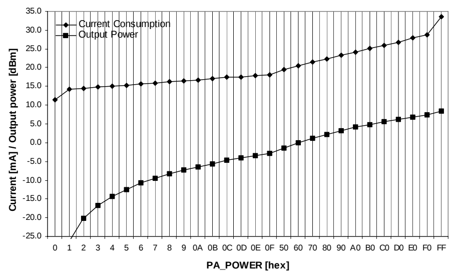


Fig. 5: CC1020 [25] Output Power

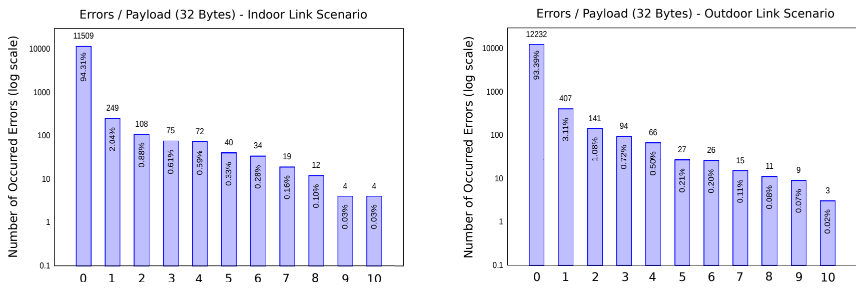


Fig. 6: Error Occurrence Patterns - Indoor and Outdoor Link

under dry weather conditions. We chose 1000 packets of 32 bytes (+2 bytes CRC) as the base configuration. Each three seconds, the sending node generates and encodes one packet and unicasts it to the receiver node B or C. We evaluated 15 different transmission power settings of the CC1020 chip, ranging from 1 tick ($\approx$ -25 dBm) to 15 ticks ($\approx$ -3.5 dBm), since we are interested in particular in the performance of the ECCs under different signal strengths. Figure 5 depicts the resulting output power of the CC1020 [25] with the different settings.

An issue that is closely linked to the resulting ECC performance is the question what error patterns actually occur, and whether there are substantial differences in the error patterns discovered on the indoor and outdoor link. We therefore counted the number of errors per packet and examined the probability distribution of the number of errors occurring across the 32 bytes payload. We achieved this by sending an unencoded payload consisting of a predefined random bit sequence of 32 bytes that is known to the sender and the receiver and bitwise checking for errors after reception. Figure 6 depicts the resulting histogram of the probability distribution of 0 to 10 errors per payload, calculated across all measured transmission power settings and displayed in logarithmic scale. As one can expect, most packets did not exhibit any errors (94.31% and 93.39% on the indoor and outdoor link). The most frequently encountered error patterns were 1-bit errors (2.04% and 3.11%) - i.e., 1 error across the entire 32 bytes payload. 2-bit errors were less than half as frequent (0.88% and 1.08%), and the probability of even more bit errors occurring across the 32-bytes payload was gradually decreasing on both links, which is well observable in Figure 6.

Figure 7 depicts the PDR for each examined ECC vs. the transmission power setting for the indoor and the outdoor link. When comparing against unencoded transmission (OFF), the PDR could be increased with every examined ECC. The impact of applying FEC, however, depends heavily on the transmission power: it has significant advantages in case of lossy links with low signal strengths where errors are more frequent to occur, but does not constitute a benefit in case the signal strength is strong enough to cope with minor interferences. We further observed that the most powerful ECCs did not necessarily result in a higher PDR. The most powerful codes within *libECC* did not significantly increase the PDR compared to the most simple ECCs. This indicates that the most frequently corrected errors were again 1-bit or probably 2-bit errors - an observation which was also made in [8] and [10] using a similar radio and the same modulation

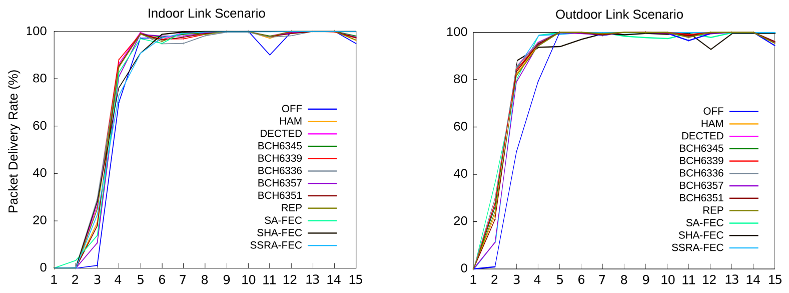


Fig. 7: Packet Delivery Rate vs. TX Power - Indoor and Outdoor Link

scheme in a comparable scenario, in particular also in the absence of other concurrent transmissions. A general observation is that the PDRs in the outdoor link with a line of sight connection are higher than those of the indoor link. This observation can be explained by the fact that the signal does not need to penetrate concrete walls and does not suffer from multipath propagation and reflection effects as in the indoor scenario.

Figure 8 depicts the PDR of the different ECCs of the outdoor link with the transmission power set to 3 ticks (≈ -17dBm) in more detail, in order to allow for a more fine-grained analysis. Again, the more powerful ECCs did not necessarily result in a higher PDR - the increment in PDR of applying FEC compared to transmitting unencoded packets, however, is significant. Figures 7 and 8 clearly convey that the adaptive FEC mechanisms designed in Section 4 performed astonishingly well, since for every transmission power setting, the three proposed mechanisms achieved comparable PDR values as the static ECCs - although they only apply FEC in an *on demand* manner. Figure 9 depicts the share of packets which are successfully received for each ECC scheme in the three adaptive approaches. The bars hence depict the results of only the three adaptive approaches of Figure 7 with the indoor link in the top graphs and the outdoor link on the bottom graphs. One can clearly see that the selection pattern of the three adaptive approaches are similar on both examined links, but that they differ among each other in the share of the applied different ECCs. SA-FEC reacts to packet loss quite quickly by changing to more powerful ECCs, which explains its larger share of Hamming(7,4) across all transmission power settings. SA-FEC changes to a stronger code as soon as it does not receive an ACK for the last packet, whereas in SHA-FEC and SSRA-FEC, one or more histories of items

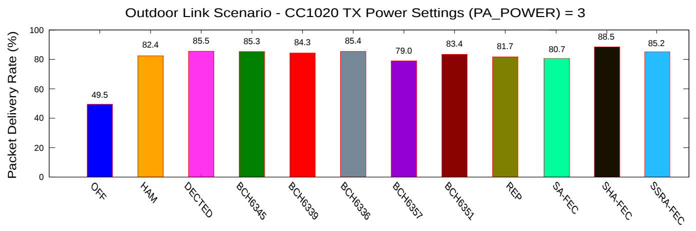


Fig. 8: Outdoor Link: PDRs per ECC with TX Power = 3 (≈ -17dBm)

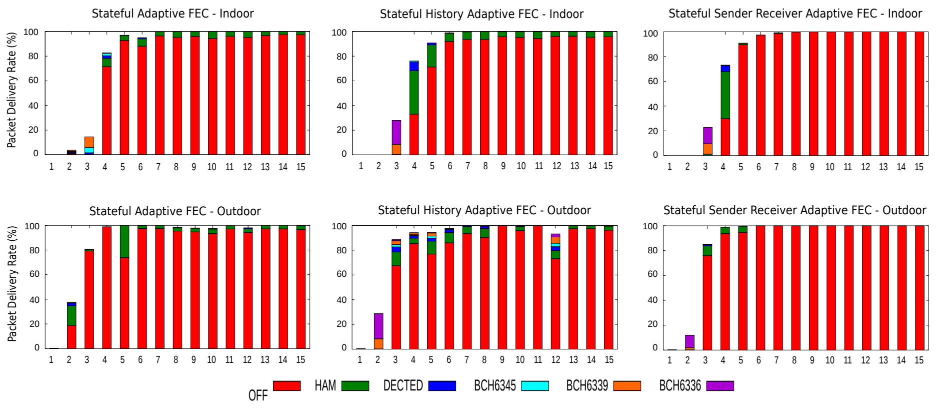


Fig. 9: ECC Selection of the Adaptive FEC Mechanisms: PDRs vs. TX Power

representing the transmission failures with the current ECC has to be filled before the code is changed. Since in SSRA-FEC, the decision to change to a more powerful ECC also depends on the reception of a subsequent acknowledgement indicating the amount of errors, the SSRA-FEC does not switch the code as quickly as SA-FEC and SHA-FEC, which makes its ECC selection more robust against oscillation. In general, Figure 9 conveys that the targeted behavior of the adaptive approaches has been accomplished: the proposed run-time adaptive ECC selection schemes apply FEC more when the link is lossy (low power settings $\leq$ 6 ticks) and less when the link is strong (high power settings $\geq$ 7 ticks). For values above 7 ticks, the vast majority of packets is sent unencoded, and few energy and time is wasted for unnecessary encoding and decoding.

5.4 Distributed Multi-Hop Scenario

In order to examine the different ECCs and the adaptive FEC approaches in an environment that comes close to real-world conditions, we evaluated each static ECC setting and each adaptive FEC scheme in our distributed testbed of seven MSB430 nodes in a multi-hop topology. We used our fully automated testbed and experiment management system [27] for reprogramming the sensor nodes and collecting the results. The examined network topology is depicted in Figure 10. The nodes are distributed across four floors in one building, forming a V-shaped network with the sink node 3 in the top left corner. The evaluated scenario has been chosen to examine the impact of a somewhat elevated traffic in a network, which is, however, still far away from being congested: each sensor node except the sink itself generates and sends packets to its gateway node towards the sink node. For every link, the same transmission power settings (5 ticks $\approx$ -12.5 dBm) are used. Static routes have been set in the beginning of the experiment, the respective links are depicted in Figure 10. We evaluated 1000 packets generated on each node for each run of the ECCs set in a static and network-wide manner, and for each run examined with the adaptive FEC. The delay between two generated packets follows uniform random distribution between 5 and 7 seconds.

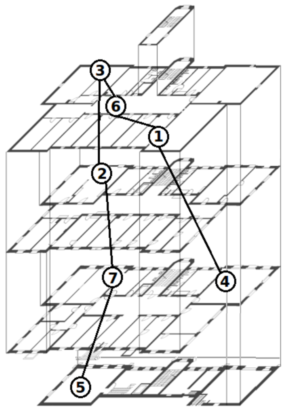


Fig. 10: Multi-Hop Scenario

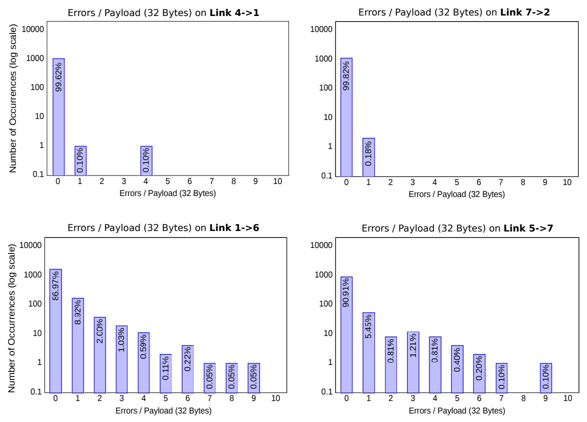


Fig. 11: Error Patterns: *Strong* Links (top) and *Weak* Links (bottom)

Since packets generated at nodes 5, 7, 1 and 4 are transmitted over multiple hops to the sink node 3, the total amount of transmissions within the network amounts to 12 transmissions every 6 seconds, or 2 transmissions per second. As the raw transmission time of one packet and the subsequent ACK takes roughly 50 ms + 20 ms (depending on the utilized ECC), we obtain a channel utilization of roughly 14% across the entire testbed. With this level of channel utilization, interferences due to other ongoing transmissions are likely to occur, which may render the application of ECC to be a valuable countermeasure.

We investigated the error patterns in the multi-hop scenario in the same manner as in the single-hop indoor and outdoor link evaluation. 1-bit and 2-bit errors were again the most frequently occurring errors in the multi-hop case. Figure 11 depicts the histograms of the probability distribution of 0 to 10 errors of the packets arriving at nodes 1,2 and 6,7 when applying no ECC. The figure clearly shows that the number of errors and the error pattern differs from link to link. Packets arriving at node 6 and 7 generally contain more errors and are more likely to contain more than 1-bit and 2-bit errors than those arriving at nodes 1 and 2. The most probable explanation for this observation is that the links 5$\rightarrow$7 and 1$\rightarrow$6 are prone to a higher absorption of the signal through walls and floors, since node 5 is in the basement of the building and has to penetrate a thicker floor than the links 4$\rightarrow$1 and 7$\rightarrow$2. The link 1$\rightarrow$6 even has to penetrate 5 concrete walls. The figure clearly shows that the error patterns differ on each link, and that as a consequence, the decision whether an ECC should be applied should be taken on a per-link basis on the node itself. Clearly, pinpointing which links would turn out to be weak and error-prone and which ones would be strong and reliable would have been impossible in advance of network deployment.

Figure 12 depicts the PDR bars of the examined ECC and the adaptive FEC approaches, leaving out the ECC codes that are never selected in the adaptive approaches (c.f. Figure 2). The figure depicts for each examined setting the PDR's of the different links and the overall source-to-sink PDR. As one can clearly see in the top-left corner in the case of unencoded transmissions, the links 1$\rightarrow$6 and

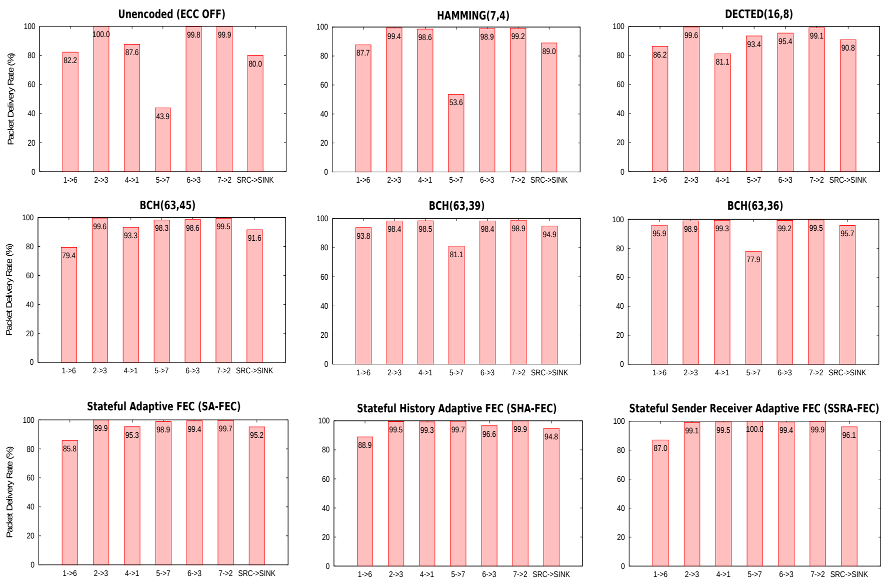


Fig. 12: PDRs per Link and Overall Source-to-Sink PDR

$5 \to 7$ have a much lower success rate, which confirms the findings of Figure 11. When comparing with the case of unencoded transmissions (top left corner of Figure 12), one can clearly see that the application of FEC made transmissions along the error-prone links more reliable, especially on $5 \to 7$. The application of FEC has clearly paid off with respect to alleviating the deteriorating impact of the lossy links $1 \to 6$ and $5 \to 7$: the improvement in PDR reaches up to 15% of the total generated packets (cf. Figure 12, DECTED, BCH).

Comparing the results of Hamming(7,4), DECTED(16,8) and the BCH-variants with the adaptive approaches, we can conclude that the adaptive FECs achieved astonishingly good results. The three strategies SA-FEC, SHA-FEC and SSRA-FEC outperformed almost every other static and network-wide setting of any of the implemented ECC codes. Since the adaptive schemes have only employed FEC on weak links and in periods of elevated BER, the majority of packets could be sent unencoded, which may also have led to fewer interference due to shorter transmission times, compared to static FEC settings. The major advantage of the adaptive approaches is the *on-demand* nature of using the correctional power of ECCs: with simple state-based concept, the adaptive approaches have managed to reach the same or better PDRs. Since the application of ECC comes at the cost of time and hence energy spent for encoding and decoding, ECCs should be limited to weak and error-prone links and/or time periods where the link quality suffers from deteriorating influences. Yet, the obtained performance of the three approaches SA-FEC, SHA-FEC and SSRA-FEC with respect to the achieved PDR were in the same range, and hence determining a “winner” is difficult. Taking into account the energy consumption however, the history-based approaches should be favored, since they are less prone to oscillation and do not immediately apply FEC after a single transmission failure.

6 Conclusions

In wireless networks, transmission errors appearing as bit flips in the received frames are more likely to occur than in traditional wireline networks, due to reasons ranging from signal absorption through concrete walls and floors, signal attenuation due to long distances, reflection from obstacles or interference by other ongoing wireless transmissions. In this paper, we have explored the potential of (adaptive) FEC techniques in the context of WSNs with lossy links. We have implemented eight different ECC codes in our library *libECC* and have proposed three run-time adaptive forward error correction schemes SA-FEC, SHA-FEC and SSRA-FEC, which react to deteriorating link quality by allocating the correctional power of ECC codes in an *on demand* manner.

We have analyzed the different codes in experiments conducted on single indoor and outdoor links and in a multi-hop network topology in a real-world sensor network testbed and have gained valuable insight into the occurrence patterns of bit errors. We chose to rely our entire investigation on real-world experiments, since simulation tools are inherently unreliable when dealing with wireless phenomena, which are as tightly linked to channel characteristics as FEC schemes. The results conveyed that 1-bit and 2-bit errors across one transmitted frame were the most frequently encountered error patterns, but that frames containing more errors were also likely to occur on weak and lossy links. The occurrence pattern of transmission errors has hence turned out to be a rather local and spatially and temporally variable issue, which in turn should be tackled on a per-link and not a network-wide basis. By analyzing the results of the single and multi-hop experiments discussed in this paper, we come to the conclusion that our proposed adaptive FEC schemes have the following three major advantages:

- Adaptive FEC approaches qualify to cope with *timely variable* error rates, since they adapt the level of correctional power based on the success of the recent past transmissions. In case of timely correlated interferences (e.g. because of a device which is temporarily using the same channel), adaptive FEC approaches can temporally switch to more powerful ECCs, and switch back again if the channel quality suffices to transmit packets unencoded.
- The occurrence pattern of transmission errors is a rather *local phenomenon* which can differ heavily from link to link. Link qualities in turn are almost impossible to predict in full depth before network deployment. Our proposed adaptive FEC approaches adapt their correctional power used for each link individually by switching between simple and complex codes. Statically selecting the strongest ECC code in a network-wide manner may achieve a high success rate, but do also introduce the highest latencies, since transmissions across several hops require multiple encoding and decoding operations.
- The energy aspect favors the adaptive approaches: our evaluations of SA-FEC, SHA-FEC and SSRA-FEC have conveyed that when using these simple state-based adaptive FEC schemes, the major share of packets was still sent unencoded. The mechanisms succeeded well in deciding *when* and *where* to apply FEC in a totally distributed manner. In contrast to network-wide

application of ECCs, precious resources for useless encoding and decoding operations on strong and reliable links can be saved, which likewise limits the overall energy overhead.

References

1. Thomas Starr, John M. Cioffi, P.J.S.: Understanding Digital Subscriber Line Technology. Prentice Hall, Upper Saddle River, USA (1999)
2. Fairhurst, G., Wood, L.: Advice to Link Designers on Link Automatic Repeat reQuest (ARQ). Request for Comments RFC 3366 (August 2002)
3. Peterson, W., Brown, D.: Cyclic Codes for Error Detection. Proceedings of the Institute of Electrical and Electronic Engineers (IRE) **49**(1) (January 1961)
4. Shannon, C.E.: A Mathematical Theory of Communication. SIGMOBILE Mobile Computer Communications **5** (January 2001) 3–55
5. Hurni, P., Barthlomé, S., Braun, T.: libECC: An Open-Source Library of Error Correcting Codes (ECCs) for the MSP430 Microcontroller http://rvs.unibe.ch/research/software.html.
6. Texas Instruments: 16-Bit Ultra-Low Power MSP430 Microcontrollers
7. Texas Instruments CC1000: Single Chip Very Low Power RF Transceiver
8. Jeong, J., Ee, C.T.: Forward Error Correction in Sensor Networks, International Workshop on Wireless Sensor Networks (WWSN), Morocco (June 2007)
9. RF Monolithics Incorporated: TR1001 868.35 MHz Hybrid Transceiver
10. Busse, M., Haenselmann, T., King, T., Effelsberg, W.: The Impact of Forward Error Correction on Wireless Sensor Network Performance, ACM Workshop on Real-World Wireless Sensor Networks (REALWSN), Uppsala, Sweden (June 2006)
11. Willig, A., Mitschke, R.: Results of Bit Error Measurements with Sensor Nodes and casuistic Consequences for Design of Energy-Efficient Error Control Schemes, European Workshop on Sensor Networks (EWSN), Zurich, Switzerland (2006)
12. Ahn, J.S., Heidemann, J.: An Adaptive FEC Algorithm for Mobile Wireless Networks, Technical Report ISI-TR-555, University of Southern California - Information Sciences Institute, California, USA (March 2002)
13. Kurkowski, S., Camp, T., Colagrosso, M.: MANET Simulation Studies: the Incredibles. ACM Mobile Computer Communications, New York, USA **9**(4) (2005)
14. Andel, T., Yasinsac, A.: On the Credibility of MANET Simulations. IEEE Computer Magazine, Los Alamitos, CA, USA **39** (July 2006)
15. Morelos-Zaragoza, R.: The Art of Error Correcting Coding. John Wiley (2006)
16. Hamming, R. Error Detecting and Error Correcting Codes **26**(2) (April 1950)
17. Gulliver, T.A., Bhargava, V.K.: A Systematic (16,8) Code for Correcting Double Errors and Detecting Triple-Adjacent Errors. IEEE Trans. on Computers **42** (1993)
18. Hocquenghem, A.: Codes Correcteurs d'Erreurs. Chiffres (Paris) 2 (1959)
19. Bose, R., Ray-Chaudhuri, D.: On a Class of Error Correcting Binary Group Codes. Information and Control Journal, San Diego, CA, USA **3**(1) (March 1960) 68 – 79
20. mspgcc - A port of the GNU tools to the Texas Instruments MSP430
21. Baar, M., Koeppe, E., Liers, A., Schiller, J.: The ScatterWeb MSB-430 Platform for Wireless Sensor Networks, SICS Contiki Workshop, Kista, Sweden (March 2007)
22. ScatterWeb^2 Operating System - Freie Universität Berlin & ScatterWeb GmbH
23. Polastre, J., Szewczyk, R., Culler, D.: Telos: Enabling Ultra-Low Power Wireless Research, International Conference on Information Processing in Sensor Networks (IPSN), Los Angeles, USA (April 2005) 364–369
24. Dunkels, A., Groenvall, B., Voigt, T.: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors, IEEE Workshop on Embedded Networked Sensors (EmNets), Tampa, Florida (2004)

25. Texas Instruments CC1020: Single-Chip FSK/OOK CMOS RF Transceiver
26. Hurni, P., Braun, T.: On the Accuracy of Software-based Energy Estimation Techniques, European Conference on Wireless Sensor Networks (EWSN), Bonn, Germany (February 2011) 49–64
27. Hurni, P., G.Wagenknecht, Anwander, M., Braun, T.: A Testbed Management System for Wireless Sensor Network Testbeds (TARWIS), European Conference on Wireless Sensor Networks (EWSN) Demo Session, Coimbra, Portugal (February 2010) 33–35