Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 5 (2011) 866–873

Procedia Computer Science

The First International Workshop on Wireless Networked Control Systems (WNCS)

Reliable and Low Latency Transmission in Industrial Wireless Sensor Networks

Kan Yu[a], Mikael Gidlund[b], Johan Åkerberg[b], Mats Björkman[a]

[a]*Mälardalen University, Sweden*
[b]*ABB AB, Corporate Research, Sweden*

Abstract

The major advantages with *Industrial Wireless Sensor Networks* (IWSNs) in process automation are cable cost reduction, enhanced flexibility and enabling new emerging applications such as wireless control. However, transmission over the wireless channel is prone to noise and interference which causes packets to be erroneous received at the receiver node. To improve the link reliability in lossy channels, error correcting codes are commonly used. In this paper we discuss the use of forward error correction (FEC) codes in IWSN in order not only to improve the link reliability but also to reduce the number of retransmissions in harsh industrial environments. We propose a FEC scheme suitable for MAC level protection where the packet is divided into groups and encoded using systematic FEC codes. We have implemented different FEC codes in a typical IWSN chip to evaluate memory consumption and to ensure that we are not violating the strict timing rules for acknowledgment. Our results show that some FEC codes are suitable to be implemented in a typical IWSN node while several fails due to large memory footprint or to long encoding and decoding time.

Keywords: Industrial Wireless Sensor Network, ARQ, FEC, reliable, low latency

1. Introduction

Industrial wireless sensor networks (IWSNs) are foreseen to be deployed in harsh environments where the radio signal will be affected by random noise and channel fading which causes packet errors which may harm production, equipment and personnel. Reliability and latency are the primary requirements in IWSNs to guarantee deterministic real-time communication in industrial automation. A common method to improve the link reliability is to employ a *automatic repeat request* (ARQ) procedure (a retransmission procedure) and another approach is to apply *forward error correction* (FEC) strategies [1], reducing the bit error rate and consequently the number of retransmissions.

In most available wireless systems ARQ is used to tackle erroneous packets. However, in bad channel conditions it has been showed that ARQ cannot guarantee the strict latency requirements given by process industry [2]. The worst case is that a sender still fails to transmit the data by trying maximum retransmission times allowed by the standard. Assume a network with hundreds of sensor nodes working in this environment. A huge number of retransmissions could be triggered at the same time because of the harsh channel. Then the bandwidth resources will be temporary exhausted, since too many nodes try to transmit data at the same time. Measurements in [3] show that transmission failure does not only bring latency to the industrial application, but can also result in network congestion.

To solve this problem, forward error correction code is one of the appropriate approaches. In a FEC code, extra parity bits are imposed on the original message to recover the corrupted bits caused by the noisy wireless channel. FEC has been considered for WSN in previous works but focusing on energy consumption [4], [5], [6] without considering constraints on memory size and processing time. It should be mentioned that adding FEC into IWSN

Selection and/or peer-review under responsibility of Prof. Elhadi Shakshuki and Prof. Muhammad Younas.
doi:10.1016/j.procs.2011.07.120

may increase the energy consumption but this is still negligible considering other processing costs. Compared to the energy consumption, availability and reliability are much more important. A serious congestion or deadline misses in the wireless sensor network will stop the applications, which may result in economic loss or serious safety problems.

In this paper, we discuss how to increase the reliability and reduce the latency in IWSN by using FEC codes. The lack of access to the "silicon" makes it infeasible to employ FEC codes in the physical layer, therefore we propose a method to apply FEC codes at the MAC layer. In our proposed scheme the complete MAC layer packet, except for the FCS field, is divided into groups and encoded with systematic FEC codes. Then the added redundancy from each encoded group is collected and stored into a new FEC field in the tail of payload with an additional new flag in the header to indicate FEC code operation. The rationale of this proposed method is that when a sensor node receives a packet, it is able to check the packet using the FCS checksum. If no bit error is found, the node can directly filter the packet by the header without decoding the complete packet. We also investigate the execution time of different FEC codes to ensure that we are not violating the strict timing rules for acknowledgments. Therefore, we benchmark some different FEC codes such as Bose-Choudhary-Hocquenhem (BCH), Reed Solomon (RS) and Golay code on an embedded system to ensure that the processing time of FEC is within the execution time budget, as well that the FEC code requires a reasonable amount of memory.

The remaining part of the article is organized as follows. Section II describes industrial wireless channel conditions and the ARQ mechanism in the IEEE 802.15.4 standard in a more detailed fashion. Section III presents our solution to apply FEC at the data link layer. Section IV describes the experimental setup for benchmarking the different FEC codes and discusses the achieved results in terms of memory size and processing time. Section V contains detailed discussion about the framework and finally the article is concluded in Section VI.

2. IEEE 802.15.4 and Industrial Wireless Channel Conditions

It is common knowledge that the wireless channel condition is dynamic and its reliability is much lower than the wired counterpart, because signals could be easily interfered by many factors, such as attenuation, path loss, etc. Moreover, most WSNs for industrial purpose run at the frequency 2.4 GHz. The communication reliability will also suffer from interferences from others wireless systems such as IEEE 802.11 and Bluetooth working in the same Industrial, Scientific and Medical (ISM) band. In [7], Sikora and Groza pointed out the impacts from other wireless systems using ISM band on IEEE 802.15.4 may result in a timeout in the PHY layer or an enlarged packet error rate.

In order to guarantee reliable communication, the IEEE 802.15.4 standard provides an stop-and-wait ARQ mechanism. For direct transmission, a device that sends data with its acknowledgment request subfield set to one shall wait for at most *macAckWaitDuration* symbol periods for the corresponding acknowledgment frame to be received. If an acknowledgment frame is received within *macAckWaitDuration* symbol periods, the transmission is considered to be successful, no further action regarding retransmission is required. If an acknowledgment is not received within *macAckWaitDuration* symbol periods, the device shall conclude that the single transmission attempt has failed. Then the device shall repeat the process of transmitting the data and waiting for the acknowledgment, up to a maximum of *macMaxFrameRetries* times. If no acknowledgment is received after *macMaxFrameRetries* retransmissions, the MAC sublayer shall assume the transmission has failed and notify the next higher layer of the failure [8].

In order to exhibit the problem in a real industrial environment, measurements were done in [3] to investigate the channel conditions in a power plant where IWSN compliant to the IEEE 802.15.4 standard was used. One sensor node continuously sends data to another node with the distance of 30 m non-line-of-sight. Then the value of *received signal strength indicator* (RSSI) is measured on the receiver side. The results from [3] show that most of the signal strength concentrates between -65 dBm and -55 dBm, but still more than 10% of the signal strength is less than -68 dBm. It may be caused by deep fading and shadowing from the harsh wireless channel. If the receiver is not sensitive enough to pick up the weak signal, bit errors are introduced into the output data. Since an ARQ mechanism is applied at the MAC layer of IEEE 802.15.4, retransmission will be trigged. The results of the measurement also show that the packet error rate (PER) varies between 0.37 - 0.38. It is also notable that one successful transmission often involved several retransmissions. When hundreds of nodes exist in the same network, it does not only cause high latency, but also results in network congestion due to the exhaust of bandwidth resources. This is not acceptable for real-time industrial applications.

3. Applying Forward Error Correction (FEC) Technology in IWSN

The basic idea of forward error correction codes is to introduce redundancy into the original data in order to recover the original data from the corrupted data and is proven to be an efficient method to improve the reliability of noisy channels. Due to different operation modes, FEC codes are divided into two types: block codes and convolutional codes. A block code (n, k) means that the encoder divides the data into message blocks with k symbols length. Each block of information is treated independently. When a block is encoded, k symbol length messages are extended to n symbol length codewords. The $n - k$ redundant symbols are added to provide the final codeword, which can correct a certain length of erroneous bits caused by external interference. At the receiver end, the decoder can detect and correct erroneous bits. The code rate, $R = k/n$ ($R < 1$), indicates the transmission efficiency of the FEC. Hard-decision and soft-decision are two different methods to decode FEC codes. Hard-decision decoding methods can be used only when the signal strength, after demodulation, is quantized to two levels, 0 and 1. Otherwise only the soft-decision decoding method is available. Compared to the hard-decision method, the soft-decision decoding scheme obtains a 3 dB extra coding gain, but with an additional computational complexity.

In order to apply FEC on top of the IEEE 802.15.4 PHY layer, we need access to the "silicon". Therefore, we focus our work on how to apply FEC on top of the MAC layer in an efficient manner. The feasibility of applying FEC codes at the MAC layer depends on two important issues, namely, the way to properly apply FEC codes to the MAC frame, and the timing requirement of the FEC encoding/decoding in order to meet the overall IEEE 802.15.4 acknowledgment timing requirement.

3.1. Applying FEC Codes at the MAC Layer

In this section, we use the 802.15.4 data frame at the MAC layer as shown in Figure 1 as an example during our discussion. As can be seen in Figure 1, the MAC sublayer data frame consists of three main parts: the MAC header, the payload and the frame check sequence (FCS).

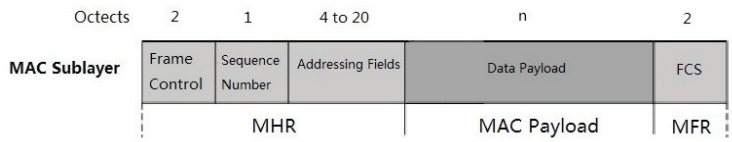

Figure 1: IEEE 802.15.4 Data Frame structure

The ideal approach is to encode the complete packet with a FEC code. However, we propose that the FCS field should not be encoded since the introduction of FEC codes will change the packet structure. The FCS field, which carries a cyclic redundancy check (CRC), plays a similar role as the FEC code, but it can only detect bit errors rather than to correct them. The FCS field should be excluded from the FEC coding for two reasons. The first reason is the compatibility concern of the IEEE 802.15.4. In the standard, the FCS value is calculated over the complete packet. If the FCS field is encoded with a FEC code, the FCS value is not the correct checksum value of the packet, since the packet will be changed after FEC encoding. Any node, which have no FEC scheme implemented, will calculate a different FCS value and consider the packet as corrupted. The second reason is the convenience for a receiver to only decode packets if the FCS check fails. If the FCS is out of the protection range of the FEC code, the FCS value can be used first to check the correctness of the packet since the computation of the FCS is much faster than the FEC decoding time. Furthermore, if no error is detected it is unnecessary to decode the packet. The receiver can filter incoming packets more efficiently. However, the benefit of this depends on the encoded packet structure as we will discuss later.

For completeness, instead of encoding the packet without the FCS field, there are two other alternatives of encoding range: only to encode the header, or only to encode the payload. The introduction of a FEC scheme results in more energy consumption and longer packet processing time, whereas a shorter range of error correction obviously reduces the negative impact. There is another advantage of only encoding the payload. If the correctness of the header is trusted, then the nodes do not have to decode every received packet in order to check the header information. In other words, packets can be filtered much more efficiently with lower energy consumption. The drawback of these

two alternatives is also obvious. When unprotected fields are corrupted, the packets needs to be retransmitted and additional latency cannot be avoided. As mentioned before, the energy consumption is not the most important concern today for industrial automation. In order to achieve the most benefit of the FEC code, we propose that both the header and the payload should be encoded.

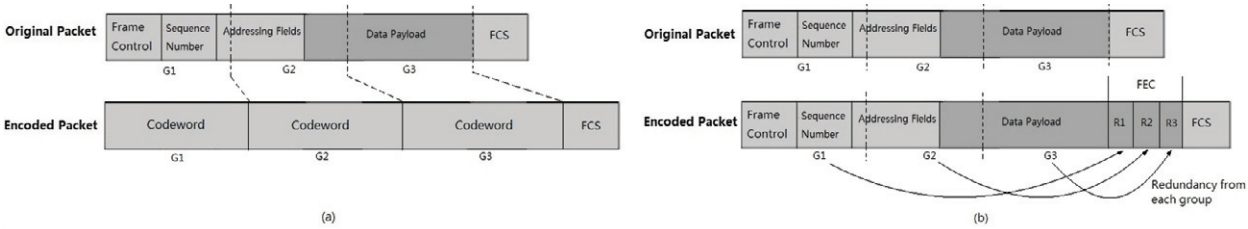

Figure 2: FEC Encoding Scheme at the MAC Layer (a) Scheme I, (b) Scheme II

There are two approaches to encode a packet using FEC at the MAC layer. The first approach is to divide the complete packet, except for the FCS field, into several groups. The group length is derived from the data length of the block code. If the last group length does not equal the block code length, a series of zeros are padded at the end of the last group to meet the length requirement. Each group is individually encoded into a codeword using the FEC algorithm. Then all the codewords are kept in the same order as original data groups. As shown in Figure 2 (a), as an example, a packet is divided into three groups (G1, G2 and G3) and is encoded into three different groups of codewords. This straightforward approach does not require any additional operations, but the drawback of this method is that the syntax of the packet is completely changed. Other nodes which do not apply FEC scheme cannot understand the content of the encoded packets.

Algorithm 1 Decoding and Error Correction Procedure

```
1:  if calculated FCS = received FCS then
2:      Successful receiving, return
3:  else if FEC flag set in Frame Control Field then
4:      Calculate the total block number and FEC field offset
5:      Parse the payload into the data field and FEC field, pad zeros in the data field if necessary
6:      for all i such that 0 ≤ i ≤ total block number -1 do
7:          Rebuild FEC block i by combining data block i with corresponding redundant data in FEC field
8:          Decode FEC block i
9:          if decoding fail, errors cannot be corrected then
10:             Failed receiving, return
11:         end if
12:     end for
13:     Successful receiving, return
14: else
15:     Failed receiving, return
16: end if
```

In the second approach, the packet is also divided into several groups, padding zeros at the end if necessary. The difference compared to the first mentioned approach is that systematic block code is used to encode each group. When a message is encoded with a systematic code, parts of the codeword are the original data. Thus the codeword is divided into two parts: original data and the additional redundancy. Then, the redundant data is stored in a separate location in the payload, leaving the original packet unchanged, as illustrated in Figure 2 (b). Then the redundant data R1, R2 and R3 from each encoded group are placed into the new FEC field. In this way, it is possible to parse the packet directly after successful verification of the FCS. The length of the FEC field depends on the total redundancy length of the FEC code. Furthermore, adding a flag in the Frame Control field in the header, where reserved bits exist, can indicate whether or not this packet is encoded with FEC. This will increase the efficiency when receiving a packet

Table 1: Definitions[8]

Parameter	Definition	Value (symbol)
aUnitBackoffPeriod	The number of symbols forming the basic time period used by the CSMA-CA.	20
aTurnaroundTime	RX-to-TX or TX-to-RX maximum turnaround time	12
phySHRDuration	The duration of the synchronization header for the current PHY.	10
phySymbolsPerOctet	The number of symbols per octet for the current PHY.	2
macAckWaitDuration	The maximum number of symbols to wait for an acknowledgment.	Equation (1)

from nodes that do not apply FEC code. There is another advantage with this approach, as we previously proposed that the FCS field is out of encoding range of FEC. When a node receives a packet, it is able to verify the packet using the FCS checksum. If no bit error is found, the node can directly parse the packet's header without decoding anything. The reason why the FEC field is placed in the tail of the payload is that usually after MAC header, the upper layer header is followed, such as network layer header for ZigBee. The action taken by the receiver using the second approach to decode the message and correct errors is summarized in Algorithm 1. Successful receiving means that the received message contains no error, or errors have been successfully corrected. Failed receiving means that the received message still contains errors and a retransmission is required.

3.2. Timing Requirement

Usually, the combination of a FEC code and an ARQ mechanism are used, known as Hybrid ARQ. As described in Section II, a timing limitation of acknowledgment is defined in the standard. We first consider the simplest scenario (Scenario I): the sender encodes the data frame and the receiver does not encode the acknowledgment frame. For the receiver, it has to complete decoding within at most *macAckWaitDuration* symbols time. According to the 802.15.4 standard, the *macAckWaitDuration* can be calculated as

$$\begin{aligned} macAckWaitDuration &= aUnitBackoffPeriod + aTurnaroundTime \\ &+ phySHRDuration + \lceil 6 \times phySymbolsPerOctet \rceil \end{aligned} \quad (1)$$

where each parameter is defined in Table 1. Hence totally, $macAckWaitDuration = 20 + 12 + 10 + 6 \times 2 = 54$ symbols time. The data rate in 2.4 MHz is 250 kbps, which equals 62500 symbols per second. Then, $macAckWaitDuration = 54/62.5 = 0.864$ ms. Thus, the time duration should be less than 0.864 ms when decoding the FEC code, otherwise, it is not compatible with the standard. In a more complicated scenario (Scenario II), when both the sender and the receiver encode the packet, the sum of encoding and decoding time should shorter than *macAckWaitDuration* to achieve the requirement.

Until now, we have proposed an efficient FEC scheme at the MAC layer. Finally, the duration of FEC encoding and decoding is crucial in order to meet the timing requirement of IEEE 802.15.4 acknowledgments. This becomes the key point to utilize the proposed FEC scheme for IWSN, and is the starting point of our evaluation of FEC codes in the next section.

4. Performance Evaluation

The purpose of our evaluation is to identify possible FEC algorithms and primarily measure their processing time. Since IWSN nodes are embedded devices with limited memory resources, the evaluation of footprint of the FEC codes is also an important part. The energy savings and communication quality evaluations are out of the scope of this paper, since the evaluation of memory footprint and processing time is the first step to show the feasibility of FEC codes used in IWSN.

4.1. Selection of FEC codes for evaluation

Compared to convolutional codes, block codes are more suitable for the data link layer due to their memoryless property. Since the computational complexity of soft-decision decoding methods is much more complex than hard-decision method, we focus on block codes using hard-decision decoding method. Golay (23, 12, 7) code is one of the block codes we will evaluate. It is a perfect binary error-correcting code that is able to correct three or fewer error bits in 23 bit positions. In our evaluation, both the encoding and decoding scheme are based on look-up tables. The tables can be pre-calculated and stored in the memory. This approach apparently decreases the computing complexity at the cost of memory. Binary Bose-Choudhary-Hocquenhem (BCH) is the second type of FEC code in our evaluation. BCH code is an important subclass of cyclic codes, based on a strict algebraic structure. Specific BCH can be easily created according to the error correction requirement. We chosen to evaluate BCH (31, 21) and BCH (48, 36). Reed-Solomon (RS) code is the last type of FEC code we will evaluate. The RS code is one type of non-binary BCH codes, and is considered to be efficient. It has burst error correcting capabilities, which is extremely suitable for wireless channels. We chose to evaluate RS (15, 11) and RS (128, 96). RS (15, 11) is much less complex than RS (128, 96), at the cost of a reduced error correction capability. RS (128, 96) is a shorten RS code from RS (255,223) and they have the same error correction capabilities.

4.2. Experimental Setup

We use a development kit from ST Microelectronics to evaluate the FEC algorithms, with 32-bit high-performance and industry-leading RISC core ARM Cortex-M3 operating at maximum 72 MHz frequency (1.25 DMIPS/MHz using Dhrystone 2.1) [9] which performances at 0 wait state memory access for highly deterministic real-time applications. 32 bit ARM micro controllers or similar is often found in industrial devices within the automation domain. It is commonly accepted that the cost of processing power and memory decrease over time, therefore we use a more powerful micro controller compared to what is commonly found today in order to meet the future needs of IWSN.

The results consist of two parts: memory footprint and processing time. The IAR Embedded Workbench v5.50 compiler supports 3 levels of optimization in terms of size or speed. In order to achieve the fastest processing time, we only show the results and comparisons using speed optimization. All the implementations are based on pure software using C language. All implementations of FEC codes are not optimal with respect to performance, so they could be further optimized to reduce the footprint and the execution time in order to meet the requirements for less powerful micro controllers.

4.3. Evaluation Results

Usually the maximum message length of IWSN is 128 bytes in IEEE 802.15.4. In order to find the maximum latency introduced by FEC, we process the maximum data length using the identified algorithms previously presented. The encoded data length should be less than 128 bytes. Therefore the original data length must be even shorter. The maximum length of original message depends on the FEC code rate, $L = 128 \cdot R$. The original messages are divided into groups with the length according to the required length of the FEC code, and then individually encoded by each block code. The code rates and error correction capabilities of all candidates are summarized in Table 2.

Table 2: Code Rates and Error Correction capability

FEC	Code Rate	Error Correction capability
Golay(23,12,7)	0.522	3/23 = 0.130
RS (15, 11)	0.733	2/15 = 0.133
RS (128, 96)	0.75	16/128 = 0.125
BCH(31,21)	0.677	2/31 = 0.065
BCH(48,36)	0.75	2/48 = 0.042

The footprint of each FEC algorithm is shown in Figure 3. The memory usage is shown in three different regions: read-only code memory, read-only data memory and read-write data memory. Read-only code memory represents the

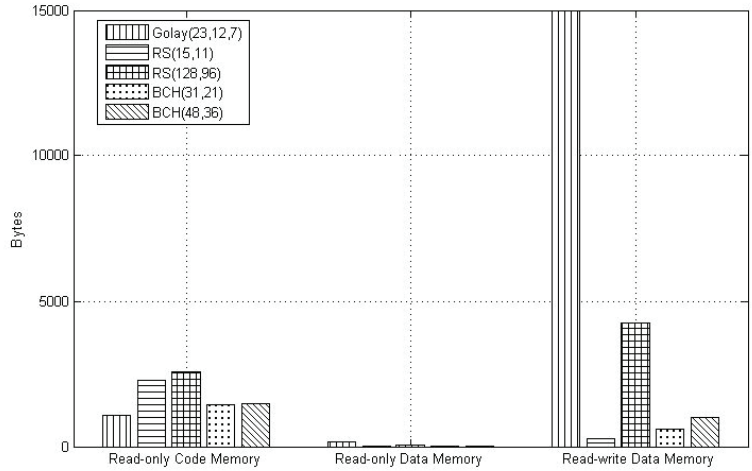

Figure 3: Memory Footprint

Table 3: Processing Time

FEC code	Encode (*ms*)	Decode with no error (*ms*)	Decode with max errors (*ms*)
Golay(23,12,7)	0.036±0.002	0.140±0.022	0.146±0.019
RS(15,11)	0.322±0.020	0.540±0.025	1.061±0.030
RS (128, 96)	1.340±0.030	2.810±0.090	5.467±0.120
BCH(31,21)	1.353±0.120	1.287±0.122	2.228±0.082
BCH(48,36)	1.754±0.115	1.293±0.098	2.531±0.085

size of the FEC executable binary. Read-only data memory represents the footprint of the initialized constant values. Read-write data memory represents the RAM size required by the algorithms.

The results of the execution time, based on CPU cycles, of each FEC algorithm are shown in Table 3. Since we let the ARM Cortex M3 micro controller run at 72 MHz, the execution time can easily be calculated into milliseconds. The execution time required for decoding messages with no errors and errors are significantly different, therefore we present them separately. Decoding with maximum errors, means that the message contains the maximum number of error bits the corresponding FEC algorithm can correct.

5. Analysis and Discussion

As previously mentioned, in order to achieve a successful transmission, the timing requirement of acknowledgment from the standard should be strictly followed. Therefore, the aim of the evaluation is to investigate whether FEC codes from different types using software implementations are fast enough to meet the timing requirement. The memory footprint measurement shows the feasibility of FEC codes for memory constrained embedded devices. Confirming energy consumption or transmission quality improvement by using FEC is not the purpose of this paper.

The evaluation result of memory footprint is shown in Figure 3 and time consumption is shown in Table 3. It is notable that the processing time changes slightly both when encoding and decoding packets. The reason for this is due to the different input data. The encoding time will slightly change when encoding different messages. As for decoding time, the error bits in the message could influence the execution time. When there is no error or few errors, the decoder performs faster compared to when the error bits increase. The reason is that it takes some extra time for the decoder to detect the error bits and correct them.

In Scenario I, when only the sender encode the packet using FEC, the receiver should complete the FEC decoding less than *macAckWaitDuration* symbol periods, namely 0.864 ms. From Table 3, Golay code using look-up table is the fastest, perfectly meeting the requirement. It is fast enough with a decoding time less than 0.16 ms that can even be applied in Scenario II, when the acknowledgment frame is also encoded by the receiver. However, it requires a huge amount of memory, nearly 24K bytes in RAM. In our platform, the maximum RAM size is only 64K bytes. It is impractical to use for industrial purpose. Another drawback of Golay code is low code rate, as shown in Table 2, representing less efficient transmission.

The second fastest FEC code, RS (15, 11) requires much less memory compared to the Golay code. When the received message contains no error or few errors, RS(15,11) is fast enough for Scenario I, around 0.5 ms to finish decoding within *macAckWaitDuration* symbol periods. When the worst case happens, RS (15, 11) requires around 1 ms for decoding, which is a little beyond the timing boundary. However, this worst case only happens when the maximum message is transmitted and 13.3% of this message is corrupted, which rarely occurs in practical applications. Even if it happens, high latency will not be introduced. The reason is that the desired message, after decoding, has been already correctly delivered to the next layer before the receiver sends back the delayed acknowledgement. Another advantage of RS (15, 11), compared to Golay code, is the higher code rate and the better error correction capability, 2 error symbols out of 15. Thus, 8 successive error bits can be corrected by RS (15, 11). In a noisy wireless channel, bursts of error bits often appears, so RS(15, 11) has excellent performance compared to the Golay code in wireless communication.

The decoding time of RS (128, 96) and both of the two BCH codes are far to slow, especially the RS (128, 96). It required more than 5 ms for decoding, nearly 5 times longer than the requirement. Perhaps further optimized implementation will make them feasible, but it is not what the present paper is intended to cover.

6. Conclusions and Future Work

The most important requirements within industrial wireless sensor networks are reliability and latency. Due to the dynamic nature of the wireless link, it is difficult to provide link reliability and to guarantee deterministic real-time communication. In this article we have proposed a method where FEC codes are introduced at the MAC layer in order to improve reliability and to reduce the latency, e.g., the number of retransmissions in the wireless network. We have also benchmarked different FEC codes with respect to memory and execution time since these are limited resources on available RF chips. Our results show that both Golay (23,12,7) and RS (15,11) is fast enough to be used in IWSN. Nevertheless, Golay code requires a look-up table which allocates a significant amount of memory. Therefore, we recommend that RS(15,11) is used since it fulfills the requirements given in process automation. The evaluation and analysis of network performance by using FEC codes are not included in this paper, and is considered as future work.

7. References

[1] R. H. Morelos-Zaragoza, The Art of Error Correcting Coding, Second Edition, JOHN WILEY and SONS, LTD, 2006.
[2] J. Åkerberg, M. Gidlund, M. Björkman, Future research challenges in wireless sensor and actuator networks targeting industrial automation, to appear in IEEE 9th International Conference on Industrial Informatics (INDIN'11) (2011) 154 –159.
[3] J. Åkerberg, M. Gidlund, F. Reichenbach, M. Björkman, Measurements on an industrial wireless hart network supporting profisafe: A case study, to appear in IEEE Conference on Emerging Technologies and Factory Automation (ETFA'11) (2011) 1–8.
[4] S. Kim, R. Fonseca, D. Culler, Reliable transfer on wireless sensor networks, Sensor and Ad Hoc Communications and Networks, 2004 (2004) 449.
[5] L. Li, R. Maunder, B. Al-Hashimi, L. Hanzo, An energy-efficient error correction scheme for ieee 802.15.4 wireless sensor networks, Circuits and Systems II: Express Briefs, IEEE Transactions on 57 (3) (2010) 233 –237.
[6] M. Vuran, I. Akyildiz, Cross-layer analysis of error control in wireless sensor networks, Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on 2 (2006) 585 –594.
[7] A. Sikora, V. Groza, Coexistence of ieee802.15.4 with other systems in the 2.4 ghz-ism-band, Instrumentation and Measurement Technology Conference, 2005. IMTC 2005. Proceedings of the IEEE 3 (2005) 1786 –1791.
[8] 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 Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003) (2006) 1 –305.
[9] S.Microelectronics, Stm3210e-eval evaluation board.[online] (2010).
URL `http://www.st.com/stonline/books/pdf/docs/14220.pdf`