Error Correction on an Insertion/Deletion Channel Applying Codes From RFID Standards

Guang Yang[†], Ángela I. Barbero[‡], Eirik Rosnes[†], and Øyvind Ytrehus[†]
[†]Department of Informatics, University of Bergen, N-5020 Bergen, Norway
(e-mail:{guang.yang,eirik,oyvind}@ii.uib.no)
[‡]Department of Applied Mathematics, University of Valladolid, 47011 Valladolid, Spain
(e-mail: angbar@wmatem.eis.uva.es)

***Abstract*—This paper[1] investigates how to improve the performance of a passive RFID tag-to-reader communication channel with imperfect timing, by using codes mandated by international RFID standards.**

I. Shortcut

This brief section is intended for those who want to skip the practical motivation and jump directly to the theoretical problem setting.

Essentially, we have a binary channel which transmits information in terms of the length of runs of identical symbols. The valid runlengths are one or two, and if the receiver can determine exactly the time of each transition, she can also acquire the transmitted information sent. Due to a noisy process and with probability p, a given length-one run is detected as a length-two run, in which case a symbol has been inserted. Vice versa, with probability p, a given length-two run is detected as a length-one run, in which case a symbol has been deleted. Thus, this is a *special case* of an insertion/deletion channel.

The uncoded information is totally vulnerable to the noise of this channel. In order to protect the information, an error correction code is applied. In this paper, the error correcting code is actually a CRC-CCITT code, mandated by many international standard protocols (but intended for error detection).

Now, if you also know about cyclic redundancy check (CRC) codes, you can go to Section VI if you want to skip the introduction.

II. Introduction

Inductive coupling is a technique by which energy from one circuit is transferred to another without wires. This is a fundamental technology for near-field passive radio frequency identification (RFID) applications as well as lightweight sensor applications. In the passive RFID application, a *reader*, containing or attached to a power source, controls and powers a communication session with a *tag*; a device without a separate power source. The purpose of the communication session may be, for examples, object identification, access control, or acquisition of sensor data.

[1]This work was supported by NFR through the ICC:RASC project, and by the project MTM2010-21580-C02-02.

The operating range of a reader-tag pair is determined by communications requirements as well as by power transfer requirements. To meet the communications requirements, the reader-to-tag and the tag-to-reader communication channels satisfy specified demands on communication transfer rate and reliability. To meet the power transfer requirements, the received power at the tag must be sufficiently large as to provide operating power at the tag.

In [1], a discretized Gaussian shift channel is proposed as a modified bit-shift channel to model synchronization loss. In this paper, we will apply the same model to the tag-to-reader channel. In terms of coding, the practical difference is that the tag-to-reader channel allows more elaborate decoding schemes, especially since the volume of data transmitted and the transmission rates are modest.

We will investigate the performance of Manchester coding, which is a standardized modulation technique for RFID applications. As a stand-alone code this code was studied in [1]. Here, we will consider the performance when the Manchester code is used together with a CRC code, which is also mandated by many RFID standards for use in automatic-repeat-request (ARQ) protocols.

III. System Model of the Tag-To-Reader Channel

A coding strategy for the communication from a tag to a reader is depicted in Figure 1. The encoder structure of the tag is a serial concatenation of a CRC code as the outer code and a modulation code as the inner code. In more detail, an *information source* generates k bits of information $\mathbf{u} = (u_1, \ldots, u_k)$, which are first encoded by a CRC outer code to a codeword $\mathbf{v} = (v_1, \ldots, v_m)$. We use the CRC-CCITT code for the outer CRC code, since it is mainly used in RFID standards. The codeword $\mathbf{v} = (v_1, \ldots, v_m)$ of the outer code is then passed through the modulation encoder to produce a coded frame $\mathbf{c} = (c_1, \ldots, c_n)$ of the overall serially concatenated code. In this paper, we use the Manchester code as the inner modulation code, since it is popular in many communication protocols. The Manchester code is a very simple block code that maps 0 into 01 and 1 into 10. Since the Manchester code is a rate-$1/2$ code, it follows that $n = 2m$. Finally, the encoded frame $\mathbf{c}$ is transmitted through the discretized Gaussian shift channel. This channel model

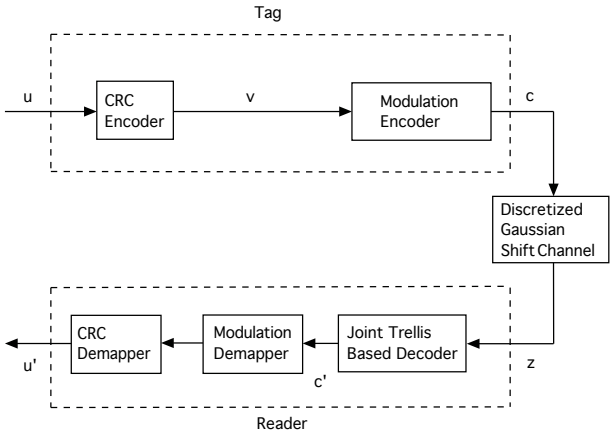

Fig. 1. System model.

was used to model synchronization errors in the reader-to-tag channel in a recent paper [1] and will be explained in detail in Section V below.

At the receiver side, the received binary sequence, denoted by $\mathbf{z}$, is decoded using a joint trellis for the overall serially concatenated code. In particular, the decoder uses a *stack algorithm* to estimate the most likely transmitted frame $\mathbf{c}'$ based on the joint trellis structure of the overall code. Finally, the most likely transmitted frame $\mathbf{c}'$ is mapped to an information sequence $\mathbf{u}'$ (the estimate of $\mathbf{u}$) using the encoder mapping of the Manchester code and that of the CRC code.

IV. The CRC-CCITT Code

CRC codes are shortened cyclic codes that, due to the existence of simple and efficient encoders, gained popularity and entrance into standard ARQ protocols, i.e., error detection. The CRC-CCITT code is used in HDLC (or ISO/IEC 13239), ISO 14443 (proximity RFID), and other RFID standards like ISO/18000-7 (DASH 7) and ISO 11784/5. In more detail, the CRC-CCITT code is a shortened cyclic code [2, p. 183] generated by the polynomial

$$g(x) = x^{16} + x^{12} + x^5 + 1.$$

For the theoretically inclined, the code is a shortened even-weight subcode of a cyclic Hamming code. The natural length N of the cyclic code corresponding to $g(x)$ is $2^{15} - 1$, but the CRC is usually used with much shorter block lengths (in which case it turns out [3, 4] that the generator polynomials are not the best possible with respect to the probability of undetected error).

The use of the CRC-CCITT code as an error correcting code on the binary erasure channel was considered in [5]. Here, we will use it for dealing with insertions/deletions.

V. The Discretized Gaussian Shift Channel

We introduced the discretized Gaussian shift channel in [1] in order to model a practical channel where performance is limited by incorrect timing. Such behavior has been observed in some inductively coupled channels.

Consider a binary channel with input $\mathbf{x} = (x_1, \ldots, x_L)$, where by assumption the value of x_1 is known to the receiver, and binary output $\mathbf{z} = (z_1, \ldots, z_{l'})$, where l' is related to but not necessary equal to l.

The binary input sequence $\mathbf{x}$ can be viewed as a sequence of phrases, where each phrase is a consecutive sequence of equal bits. Please observe that this parsing of $\mathbf{x}$ is done by the channel (and not by an encoder). Then, the integer sequence of phrase lengths $\tilde{\mathbf{x}}$ is transmitted over a channel. For instance, $\mathbf{x} = (0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1)$ is transformed into the integer sequence $\tilde{\mathbf{x}} = (2, 3, 3, 4)$ of phrase lengths.

Suppose the tag transmits a run of $\tilde{x}$ consecutive equal symbols (or bits). This corresponds to an amplitude modulated signal of duration $\tilde{x}$. At the reader, we assume that this is detected (according to the reader's internal clock) as having duration

$$\tilde{y} = \tilde{x} \cdot K$$

where K is a random variable with, in general, a Gaussian distribution $\mathcal{N}(\alpha, \varepsilon^2)$ with mean α and variance ε^2. Consecutive samplings of K are assumed to be independent. If $\alpha \neq 1$, it means that the reader has a systematic drift, which may affect the reader's ability to function at all. Thus, we will focus on the case $\alpha = 1$. With this definition, the input to the demodulator will be a sequence of alternating runs of high and low amplitude values; the detected duration $\tilde{y}$ of each run being a *real-valued* number.

As a simplification, and to deal with the fact that $\tilde{y}$ may become negative (K has a Gaussian distribution), which of course does not have any physical interpretation, the timing is discretized and K is truncated. The optimal choice for the quantization thresholds, i.e., the thresholds when mapping the real-valued numbers $\tilde{y}$ to *positive* integers $\tilde{z}$, will depend on the code under consideration.

In general, let $\mathcal{Q}(\mathcal{A}, \mathcal{T})$ denote a quantization scheme with quantization values $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$, where $1 \leq a_1 < \cdots < a_{|\mathcal{A}|} \leq L$, and L is some positive integer, and quantization thresholds $\mathcal{T} = \{t_2, \ldots, t_{|\mathcal{A}|}\}$, where $a_l \leq t_{l+1} \leq a_{l+1}$, $l = 1, \ldots, |\mathcal{A}| - 1$. The quantization scheme works in the following way. Map a received real-valued number $\tilde{y}$ to an integer $\tilde{z}$ in $\mathcal{A}$ using quantization thresholds in $\mathcal{T}$, i.e., if the received real-valued number $\tilde{y}$ is in the range $[t_l, t_{l+1})$, $l = 2, \ldots, |\mathcal{A}| - 1$, map it to $\tilde{z} = a_l$, if it is in the range $[t_{|\mathcal{A}|}, \infty)$, map it to $\tilde{z} = a_{|\mathcal{A}|}$, and, otherwise, map it to $\tilde{z} = a_1$. Now, we define the discretized Gaussian shift channel with quantization scheme $\mathcal{Q}(\mathcal{A}, \mathcal{T})$ as the cascade of the Gaussian shift channel and the quantization scheme $\mathcal{Q}(\mathcal{A}, \mathcal{T})$, where the quantization scheme $\mathcal{Q}(\mathcal{A}, \mathcal{T})$ is applied to the real-valued sequence at the output of the Gaussian shift channel.

In [1], two different quantization schemes were proposed, denoted by $\mathcal{Q}_{\text{rounding}}$ and $\mathcal{Q}(A)$. The quantization scheme $\mathcal{Q}_{\text{rounding}}$ is based on rounding the received values to the nearest positive integer values, while the second quantization scheme has quantization thresholds $t_l = 2a_{l-1}a_l/(a_{l-1} + a_l)$, $l = 2, \ldots, |\mathcal{A}|$. Here, $\mathcal{A}$ will be the positive integers. The reason to choose this quantization scheme will be evident later

on, during the proof of Theorem 1.

A. The Discretized Gaussian Shift Channel as an Insertion/Deletion Channel

When Manchester encoding with hard-decision decoding is applied on top of the discretized Gaussian shift channel, we get a special case of an insertion/deletion channel. In the following, we will use the quantization scheme $\mathcal{Q}([1,2])$ with the Manchester code. Then, received runlengths are either 1 or 2, and a single quantization threshold of $4/3$ is used. The specialization occurs in that the information is transmitted in terms of the times of transitions between runs of zeros and runs of ones. Thus, an insertion of a symbol happens only when a run of length one is detected as having length two, and a deletion of a symbol happens only when a run of length two is detected as having length one.

The general *Levenshtein* distance [6] between two sequences of symbols over the same alphabet is defined as the number of insertions and/or deletions required to transform one sequence into the other. Decoding using the Levenshtein distance metric has been considered in several papers in the literature, for instance, [7] and [8], which address trellis based decoding approaches. Also, in [9], the performance of linear and cyclic codes under insertion/deletion channels has been considered. In this paper, however, we will use a slightly different approach.

VI. Decoding Strategy

The obvious decoding strategy is to decode to the modulated codeword with smallest Levenshtein distance to the received sequence. In order to do this efficiently, we shall need a metric table for the Manchester code on the insertion/deletion channel.

A. Metric Table for the Manchester Code

The reader receives a sequence $\mathbf{z} = (z_1, \ldots, z_{l'})$. This sequence is a channel corrupted version of the transmitted sequence of bit pairs 01 and 10. Due to the discretized Gaussian shift channel, $\mathbf{z}$ may contain some insertion/deletion bits. For example, the transmitted sequence 01 10 10 is received as 01 10 $\underline{0}$1 0 if there is an insertion after the fourth bit, while if a deletion happens at the third bit, 01_01 0 will be received. The decoder works in the following manner. It checks a previous bit and estimates a received pair to be either 01 or 10. In particular, if the previous bit is 1 and 01 is received, 01 could have been transmitted with no insertion or deletion. On the other hand, 10 could also have been transmitted and because of a deletion, 1 01 is received (next bit pair must be 10). The decoder processes the whole received frame in this manner. In particular, for each bit pair, the two possible decoding results lead to different conditions (previous bit and time offset) for the next bit pair decoding. Thus, combining this decoding scheme with the outer code's trellis structure can produce an efficient decoding procedure.

received	decoded	new point in received	advance	weight
1.01X	1.01 1.10	101.X 10.1X	2 1(+D)	0 errors 1 deletion
1.100	1.10 1.	110.0 11.00	2 1(-I)	0 errors 1 insertion
1.101	1.10 1.01	110.1 1101.	2 3(-I)	0 errors 1 insertion
1.001	1.10 1.01	10.01 1001.	1(+D) 3(-I)	1 deletion 1 insertion

Fig. 2. Metrics for computing the Levenshtein distance, assuming that the previous symbol was a 1. The case of 0 as the previous symbol is symmetric and is omitted. Colour green represents the received symbols that we have already past decoding. Blue represents the symbols in the decoded sequence that were received with insertion. Red represents the symbols in the decoded sequence that were received with deletion. X means that it does not matter whether the next symbol is 1 or 0.

VII. Stack Decoder

A. Exhaustive Decoding

The goal of the exhaustive decoder is to pick the legal codeword with the smallest Levenshtein distance to the received sequence. This can be achieved by computing the Levenshtein distance between the received sequence and all codewords, and then choose the one corresponding to the smallest value. Such a decoder is a maximum-likelihood (ML) decoder.

Theorem 1: In the discretized Gaussian shift channel with quantization scheme $\mathcal{Q}([1,2])$ and quantization threshold $4/3$, ML decoding corresponds to picking the legal codeword with the smallest Levenshtein distance to the received sequence.

Proof: As stated in Section V, a binary sequence $\mathbf{x}$ can be transformed into the integer sequence $\tilde{\mathbf{x}}$ of phrase lengths.

In this particular case, a sequence produced by the Manchester code will have only two possible values of phrase lengths, namely 1 or 2, and the only possible errors are either insertions, in the case of a phrase length 1 that transforms into one of length 2, or deletions in case a run of length 2 is perceived as one of length 1. Hence, we can represent the vector of errors as a binary vector, where 0 represents that the corresponding runlength in $\tilde{\mathbf{x}}$ has not been modified, while 1 represents an insertion in case the corresponding element in $\tilde{\mathbf{x}}$ was 1, but was received as 2, or a deletion in case the corresponding element in $\tilde{\mathbf{x}}$ was 2, but was received as 1.

For example, if the transmitted sequence is $\mathbf{x} = (0,1,1,0,1,1,0,0,1,0,0)$, then $\tilde{\mathbf{x}} = (1,2,1,2,2,1,2)$ and an error vector $\mathbf{e} = (0,1,1,0,0,1,0)$ will give as a result a received sequence of runlengths $\tilde{\mathbf{z}} = (1,1,2,2,2,2,2)$, or equivalently, the received sequence is $\mathbf{z} = (0,1,0,0,1,1,0,0,1,1,0,0)$, meaning the second run of bits experienced a deletion, the third run an insertion, and the sixth run an insertion.

In this context, the Levenshtein distance between the sent and the received sequences equals the Hamming weight of the binary error vector $\mathbf{e}$.

In order to prove the theorem we will establish that the probability of a given error vector decreases with its weight, hence the error vectors with smaller weight are more probable.

We need to introduce some notation. Let $P_I(\varepsilon)$ be the probability of insertion (which depends of course on the variance ε^2), that is, the probability that a runlength $\tilde{x}_i = 1$ is perceived as a runlength $\tilde{z}_i = 2$. Since $\tilde{y}_i = \tilde{x}_i \cdot K$ with K following a Gaussian distribution $\mathcal{N}(\alpha, \varepsilon^2)$, and the quantization threshold is $t_2 = 4/3$, we have

$$\begin{aligned} P_I(\varepsilon) &= P(\tilde{z}_i = 2|\tilde{x}_i = 1) = P(\tilde{y}_i \geq 4/3|\tilde{x}_i = 1) \\ &= P(\tilde{y}_i = 1 \cdot K \geq 4/3) = P(K \geq 4/3). \end{aligned}$$

Note that since consecutive samplings of K are independent, the insertion probability does not depend on the index i.

In the same way, let $P_D(\varepsilon)$ be the probability that a given runlength $\tilde{x}_i = 2$ experiences a deletion. Again, it can be computed as

$$\begin{aligned} P_D(\varepsilon) &= P(\tilde{z}_i = 1|\tilde{x}_i = 2) = P(\tilde{y}_i < 4/3|\tilde{x}_i = 2) \\ &= P(\tilde{y}_i = 2 \cdot K < 4/3) = P(K < 2/3). \end{aligned}$$

Because of the symmetry of the normal distribution we have $P_I(\varepsilon) = P_D(\varepsilon) = p < 1/2$.

Finally,

$$P(\text{error vector} = \mathbf{e}) = p^{w(\mathbf{e})}(1-p)^{l-w(\mathbf{e})}$$

where l is the length of the runlength sequence $\tilde{\mathbf{x}}$ and $w(\cdot)$ is the Hamming weight of its argument.

Hence, given a received sequence, ML decoding corresponds to picking the most likely codeword which, according to the last last formula, will be the codeword at shortest Levenshtein distance from the received sequence (error vector $\mathbf{e}$ with smallest Hamming weight). ■

Please observe that the proof uses the fact that $P_D(\varepsilon) = P_I(\varepsilon)$, which is guaranteed by the choice of the quantization threshold. A different choice of threshold will unbalance the two probabilities and the result might not be true. For example, suppose $P_I(\varepsilon) = 0.49$ and $P_D(\varepsilon) = 0.01$ and the received sequence is $\mathbf{z} = (0, 0, 1, 1, 0)$, so that the quantized runlength sequence is $\tilde{\mathbf{z}} = (2, 2, 1)$. Suppose also that $\mathbf{x} = (0, 1, 0)$ and $\mathbf{x}' = (0, 0, 1, 1, 0, 0)$ are two legal codewords.

In this case

$$\begin{aligned} P(\text{sent} = \mathbf{x}|\text{received} = \mathbf{z}) &= P_I \cdot P_I \cdot (1 - P_I) \\ &= 0.122 \\ P(\text{sent} = \mathbf{x}'|\text{received} = \mathbf{z}) &= (1 - P_D) \cdot (1 - P_D) \cdot P_D \\ &= 0.009. \end{aligned}$$

Thus, ML decoding will prefer $\mathbf{x}$ even when its Levenshtein distance to the received sequence is larger than the distance from the received sequence to the codeword $\mathbf{x}'$.

B. Viterbi Decoding

Inspired by [7–10], we could try to design a Viterbi decoder for this decoding problem. However, the states of such a decoder would have to be labelled by 1) the trellis state corresponding to the CRC code, 2) the previous symbol at the depth of a path (to determine current runlengths), and 3) the offset into the received sequence (determined by the number of assumed preceding insertions-deletions for each path). A trellis decoder for the CRC code alone has 2^{16} states, so the total trellis complexity can be very very large (maybe because of 3) it will be just as bad as exhaustive search).

A further problem with the Viterbi approach is that, at least in a straightforward implementation, it requires expanding all states at a given depth before we can proceed to the next one. This creates a problem if, e.g., 1.001 is received (see the metric table in Figure 2).

C. Reduced Complexity Stack Decoder

Start with state 0. At each time step, maintain a set of states with information 1), 2), and 3) above. Expand all of these as determined by the full trellis (or limited by other constraints). Heuristic step: Discard those with metrics value exceeding some predetermined D. This will be a stack decoder, with $D + 1$ stacks (one for each metrics value). We proceed by explaining the algorithm (although some details are omitted).

Let state S be described by 1) a CRC trellis state $S.crc$, 2) an input symbol (to determine current runlengths) $S.p$, 3) an offset into the received sequences (determined by the number of preceding insertions-deletions) $S.o$, 4) a path metric value $S.m$, 5) a trellis depth $S.d$, 6) a back pointer $S.b$, and 7) a stack pointer $S.s$.

Algorithm 1 Stack Decoder

1: /∗ Stack decoder for a systematic code on the discretized Gaussian shift channel with Manchester inner code. ∗/
2: Start with initial state S and put it on top of stack 0. All other stacks are empty.
3: **while** there is a nonempty stack **do**
4: Let S be the first element of the stack with lowest number which is not empty. Remove it from the stack.
5: **if** $S.i =$ the information block length **then**
6: Add the tail according to $S.crc$, and compute the overall Levenshtein metric as $S.m$ + $L(\text{tail})$, where $L(\text{tail})$ is the additional Levenshtein distance between the tail and the remaining part of the received sequence. If better than the record, save it for later. If less than D, reduce D.
7: **else**
8: Create two new states S_0 and S_1 according to: $S_0.crc = S.crc$ and $S_1.crc = S.crc$ xored with the remainder of the division $x^{S.d+16}/g(x)$, $S_i.p = i$, $i = 0, 1$, $S_0.o$ and $S_1.o$ are updated to $S.o$ in addition to extra offset from metric table, $S_0.m$ and $S_1.m$ are updated to $S.m$ plus $0, 1, 2$ as given by the metric tables, $S_i.d = S.d + 1$, $i = 0, 1$, $S_i.p = S$, $i = 0, 1$, and if $S_i.m \leq D$, store in stack number $S_i.m$ by pushing S_i onto that stack.
9: **end if**
10: **end while**

In order to simplify the description, the sketch of the

algorithm does not include the case of received sequence 1.100 or 0.011 (see Figure 2). If these cases are not implemented, they will be the predominant cause of error, so it is essential to include them in the implementation.

VIII. Discussion: Decoding Performance, Complexity, Choice of Codes, and Other Issues

The CRC code and the Manchester code are not sophisticated code constructions, neither is the concatenation of them. However, the codes are mandated by standard protocols, so it is interesting to see how they perform.

Theorem 2: The coded system under study is single error correcting for any valid information block length.

Proof: Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be two different codewords in the CRC code, and let $\mathbf{c}_1 = (c_{1,1}, \ldots, c_{1,n})$ and $\mathbf{c}_2 = (c_{2,1}, \ldots, c_{2,n})$ be their respective Manchester encoded versions. Assume that there exists a single error that will transform $\mathbf{c}_1$ into some received vector $\mathbf{z}$, and another single error that will transform $\mathbf{c}_2$ into the same received vector $\mathbf{z}$. In this case, if $\mathbf{z}$ is received, the decoder will not be able to determine which codeword was transmitted.

We will assume that both errors are insertions and that $\mathbf{z} = (z_1, \ldots, z_{n+1})$; the proof in the case where both are deletions is similar and will be omitted. Suppose the error that converts $\mathbf{c}_1$ into $\mathbf{z}$ is an insertion of a symbol after position p. More precisely, $c_{1,i} = c_{2,i} = z_i$ for $1 \leq i \leq p$, and due to the insertion of a symbol, $z_{p+1} = c_{1,p}$ also. But $c_{2,p+1}$ must coincide with z_{p+1} (otherwise there is already another insertion error occurring). Since $c_{2,p} = c_{2,p+1}$, these symbols must belong to different "Manchester pairs", and $c_{2,p+2} = 1 - c_{2,p+1}$. But $z_{p+2} = c_{2,p+2}$ while $c_{1,p+1} = z_{p+2} = 1 - c_{2,p+1}$. Thus, by induction, for $p < j < q$ where q is the position of the other single insertion, we have that $c_{1,j} = 1 - c_{2,j}$.

For this to happen, we need the subsequences of $\mathbf{c}_1$ and $\mathbf{c}_2$ between positions p and q to be alternating sequences, while $\mathbf{c}_1$ and $\mathbf{c}_2$ coincide before position p and after position q. Thus, the CRC codeword $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$ is on the form $(0 \ldots 01 \ldots 10 \ldots 0)$.

Now, the codeword $\mathbf{v}$ of the CRC code corresponds to the polynomial $v(x) = x^i \sum_{j=0}^{k} x^j$, for some i and k related to p and q. $v(x)$ is a codeword in the CRC-CCITT code if and only if $g(x)$ is a factor of $v(x)$. However, $g(x) = (x+1)p(x)$, where $p(x)$ is a primitive polynomial. Thus, $p(x)$ (and $g(x)$) divide $x^{32767}+1 = (x+1)\sum_{i=0}^{32766} x_i$, but $p(x)$ (and $g(x)$) do not divide $x^N + 1$ for $N < 32767$. Also, $g(x)$ does not divide $\sum_{i=0}^{32766} x_i$, since the latter polynomial has an odd number of terms. Hence, there are no codewords in the CRC-CCITT code on the form $(0 \ldots 01 \ldots 10 \ldots 0)$. ■

A. Decoding Performance

Figure 3 shows the simulated performance of the coded system under investigation. The three upper curves correspond to a stand-alone Manchester code, while the three lower curves correspond to the concatenated system with the stack decoder with a maximum selected distance of 4 (i.e., if more than four errors occur the decoder will always make a decoding error).

As Theorem 2 shows, $P(\text{Frame error}|\text{single error}) = 0$. Furthermore, empirically, the simulation indicates that $P(\text{Frame error}|\text{two errors occur in frame})$ is on the order of 10^{-4}.

In Figure 4 we show how the parameter D affects performance.

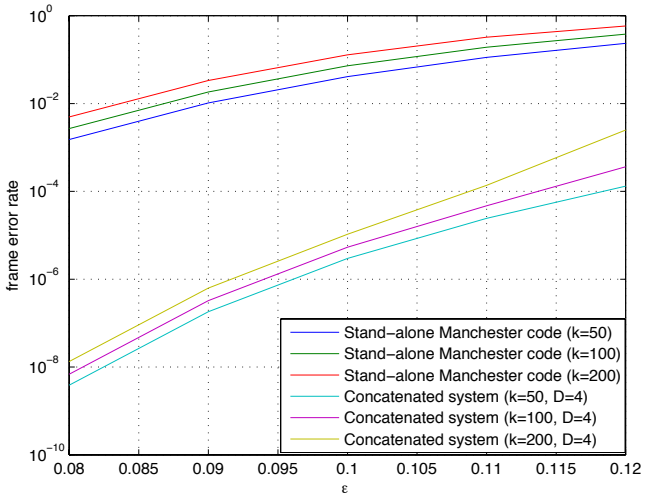

Fig. 3. Decoding results for different short frame lengths k, using either only the Manchester code, or also the CRC code for error correction.

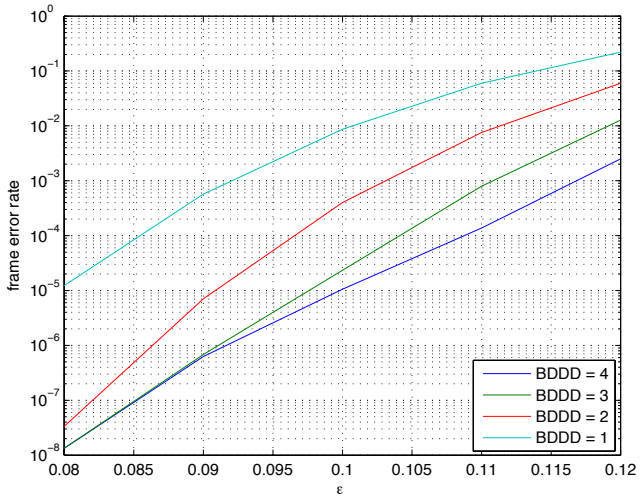

Fig. 4. Decoding results for information frame length $k = 200$, depending on Bounded Distance Decoding threshold D (BDDD).

B. Complexity

The stack decoder's decoding performance is limited by the maximum distance D. Because of ordering of the stacks, the number of states per symbol is determined by the number of

errors that actually occurs. Empirically, the average number $A(i,k)$ of states (created and put on the stack) per decoded bit with i errors occurring and with an information block of k bits increases very slightly with k for $i = 1$, approximately as $\sqrt{k}$ for $i = 2$, and at a rate slower than k for $i = 3$. For $k = 200$, we have $A(0, 200) = 1$, $A(1, 200) = 3.8$, $A(2, 200) \approx 66$, $A(3, 200) \approx 435$, and $A(4, 200) \approx 4000$, if the line 1.100 of Figure 2 is not implemented. Otherwise, $A(2, 200) \approx 90$.

C. Alternative Choices of Modulation Codes

At ITA 2011 [1], we discussed a similar channel model but applied for the reader-to-tag channel. In that setting, the receiver (the tag) has strictly limited computational power. Thus, it makes sense to protect the information by choosing a modulation code that limits the amount of errors, rather than allowing many errors that the receiver can decode (at a considerable computational effort).

Some of the modulation codes described in [1] might be candidates also for a tag-to-reader channel, as alternatives to the Manchester code mandated by most standards. However, the concatenated decoder structure might be messier than what is described in this paper.

D. Alternative Choices of Error Correcting Codes

There are recent code constructions targeted specifically at insertion/deletion channels (see, e.g., [9, 11]). However, such codes have much lower code rates than the CRC codes. Finding better codes in this case is an open problem.

E. Soft Decoding

A similar decoding can in principle be applied to a channel output quantized to more levels, at the expense of bigger decoding tables and an increase in decoder complexity. It is an open question whether or how much this could improve the decoding performance.

References

[1] A. I. Barbero, E. Rosnes, G. Yang, and Ø. Ytrehus, "Constrained codes for passive RFID communication," in *Proc. Inf. Theory Appl. Workshop (ITA)*, San Diego, CA, Feb. 2011, pp. 496–504.

[2] S. Lin and D. J. Costello, Jr., *Error Control Coding, Second Edition*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[3] T. Kløve, *Codes for Error Detection*. Singapore: World Scientific Publishing Co., 2007.

[4] K. Witzke and C. Leung, "A comparison of some error detecting CRC code standards," *IEEE Trans. Commun.*, vol. 33, no. 9, pp. 996–998, Sep. 1985.

[5] E. Rosnes, G. Yang, and Ø. Ytrehus, "Exploiting the CRC-CCITT code on the binary erasure channel," in *Proc. 6th Int. Symp. Turbo Codes and Iterative Information Processing*, Brest, France, Sep. 2010, pp. 344–348.

[6] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and substitutions of symbols," *Dokl. Akad. Nank. SSSR*, vol. 163, pp. 845–848, 1965.

[7] L. Cheng and H. C. Ferreira, "Rate-compatible pruned convolutional codes and Viterbi decoding with the Levenshtein distance metric applied to channels with insertion, deletion, and substitution errors," in *Proc. 7th IEEE AFRICON Conf.*, vol. 1, Gaborone, Botswana, Sep. 2004, pp. 137–143.

[8] L. Cheng, H. C. Ferreira, and T. G. Swart, "Bidirectional Viterbi decoding using the Levenshtein distance metric for deletion channels," in *Proc. Inf. Theory Workshop (ITW)*, Chengdu, China, Oct. 2006, pp. 254–258.

[9] K. A. S. Abdel-Ghaffar, H. C. Ferreira, and L. Cheng, "Correcting deletions using linear and cyclic codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5223–5234, Oct. 2010.

[10] T. Mori and H. Imai, "Viterbi decoding considering insertion/deletion errors," in *Proc. Int. Symp. Inf. Theory (ISIT)*, Whistler, BC, Canada, Sep. 1995, p. 145.

[11] Ø. Ytrehus, "On codes for error correction and block synchronization," in *Proc. 39th Annual Allerton Conf. Commun., Control, and Computing*, Monticello, IL, Oct. 1997, pp. 432–439.