TECHNICAL REPORT 2087
August 2015

Polar Coding with CRC-Aided List Decoding

David Wasserman

SSC Pacific
San Diego, CA 92152-5001

TECHNICAL REPORT 2087
August 2015

Polar Coding with CRC-Aided List Decoding

David Wasserman

SSC Pacific
San Diego, CA 92152-5001

SSC Pacific
San Diego, California 92152-5001

K. J. Rothenhaus, CAPT, USN
Commanding Officer

C. A. Keeney
Executive Director

ADMINISTRATIVE INFORMATION

The work described in this report was performed by the Space Systems Branch (Code 56270) of the ISR Division (Code 56200), Space and Naval Warfare Systems Center Pacific (SSC Pacific), San Diego, CA. The Naval Innovative Science and Engineering (NISE) Program at SSC Pacific funded this Applied Research project.

.

Released by
A. M. Mroczek, Head
Space Systems Branch

Under authority of
C. A. Wilgenbusch, Head
ISR Division

ACKNOWLEDGEMENT

The author would like to thank James Wagner, who did some of the work described in this report.

SB

EXECUTIVE SUMMARY

This report describes some results of the project "More reliable wireless communications through polar codes," funded in fiscal year 2015 by the Naval Innovative Science and Engineering (NISE) program at SSC Pacific.

OBJECTIVE

The purpose of the project is to determine if polar codes can outperform the forward error correction (FEC) currently used in Navy wireless communication systems. The project team has implemented an advanced decoding method called cyclic redundancy check (CRC)-aided list decoding.

RESULTS

Our simulation results show that polar coding can produce results very similar to the FEC used in the Digital Video Broadcasting - Satellite - Second Generation (DVB-S2) standard, and can provide up to 15 percent higher throughput by using code rates not provided in the DVB-S2 standard.

RECOMMENDATIONS

In any application for which the DVB-S2 FEC is considered, polar coding with CRC-aided list decoding with $N = 65536$ should also be considered.

CONTENTS

Figures

Tables

1. INTRODUCTION

Polar codes are a new type of forward error correction (FEC) codes, introduced by Arikan in [1], in which he proved that they can achieve the capacity of any binary memoryless symmetric channel with efficient encoding and decoding.

In fiscal years (FY) 2014 and 2015, SSC Pacific's Naval Innovative Science and Engineering (NISE) program funded the project "More Reliable Wireless Communications Through Polar Codes" to study polar codes and determine if they can outperform the forward error correction (FEC) currently used in Navy wireless communication systems. The project's FY 2014 results are described in [2]. In FY 2015 the project team has implemented an advanced decoding method called cyclic redundancy check (CRC)-aided list decoding, and simulated its performance in a wide variety of cases. This report documents the results of those simulations. The project's other FY 2015 work will be published separately.

Section 2 describes the basics of polar coding. CRC-aided list decoding is described in Section 3. Section 4 describes the scope of our study, and Section 5 presents the results. Section 6 gives details of how we implemented the decoder. Finally, Section 7 concludes the report.

2. POLAR CODING

Several versions of polar coding have been published. This section is intended to indicate which version is used in this work. For background and motivation of this material, see our previous technical report [2].

For any $n > 0$, we can specify a polar code of length $N = 2^n$ by choosing a subset $\mathcal{A} \subset \{1, \ldots, N\}$. If $\mathcal{A}$ has K elements, we get an (N, K) block code. $\mathcal{A}$ must be chosen well for good error-correction performance. We used the method of [3].

The polar encoder uses a row vector $\mathbf{u}$ of length N. Let $\mathbf{u}_\mathcal{A}$ be the subvector containing elements whose indices are in $\mathcal{A}$, and let $\mathbf{u}_{\mathcal{A}^C}$ be the subvector containing the remaining $N - K$ elements of $\mathbf{u}$. The encoder constructs $\mathbf{u}$ by filling $\mathbf{u}_\mathcal{A}$ with K information bits, and setting $\mathbf{u}_{\mathcal{A}^C}$ to predetermined values, usually all 0's. These predetermined values are called *frozen* bits. The encoder outputs $\mathbf{x} = \mathbf{u}\mathbf{G}_N$ where $\mathbf{G}_N$ is the N by N matrix $\mathbf{F}^{\otimes n}\mathbf{\Pi}_N$, where $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, $F^{\otimes n}$ is its nth tensor power, and $\mathbf{\Pi}_N$ is a permutation matrix called the *bit-reversal operator*, defined in Section VII of [1]. This encoder can be implemented with complexity $O(N \log N)$.

Arikan showed in [1] that polar codes can achieve capacity using a *successive cancellation* (SC) decoder. This decoder computes estimates $\hat{u}_1, \ldots, \hat{u}_N$, one at a time, in order, with complexity $O(N \log N)$.

2.1 SYSTEMATIC POLAR CODING

Systematic polar coding was introduced in [4]. The systematic polar encoder also computes $\mathbf{x} = \mathbf{u}\mathbf{F}^{\otimes n}\mathbf{\Pi}_N$, but the information bits are found in $\mathbf{x}$ rather than in $\mathbf{u}$. Specifically, [4] showed that for any vector of K information bits, there is a unique $\mathbf{u}$ such that $\mathbf{u}_{\mathcal{A}^C}$ is all 0's and $\mathbf{w}_\mathcal{A}$ is the specified information bits, where $\mathbf{w} = \mathbf{u}\mathbf{F}^{\otimes n}$.

The systematic SC decoder computes the estimate $\hat{\mathbf{u}}$ in the same way as the non-systematic decoder, and also computes $\hat{\mathbf{w}} = \hat{\mathbf{u}}\mathbf{F}^{\otimes n}$. Then $\hat{\mathbf{w}}_\mathcal{A}$ is the desired estimate of the information bits.

Arikan proved in [4] that systematic polar coding has the same frame error rate (FER)[1] as non-systematic polar coding. Arikan also provided simulation results showing that systematic polar coding has a lower bit error rate (BER) than non-systematic. This result has been replicated, but to our knowledge it has never been proven.

[1] Also known as block error rate.

3. LIST DECODING

Recall from Section 2 that the SC decoder estimates $\hat{u}_1, \ldots, \hat{u}_N$, one at a time, in order. For convenience, we introduce some non-standard notation. For any $k \leq N$ and any sequence of bits $b_1, b_2, \ldots, b_k$, let $\hat{P}(b_1, b_2, \ldots, b_k)$ be the decoder's estimate of the probability that $\mathbf{u}$ begins with $(b_1, b_2, \ldots, b_k)$. $\hat{P}(b_1, b_2, \ldots, b_k)$ depends on the decoder's input, but to make the notation simpler we do not include the decoder's input in the expression.[2]

The decoder's decisions can be viewed as choosing a path down a binary tree of height N, with 2^N leaves representing all possible decodings of a block. Figure 1 shows an example of the first four choices.

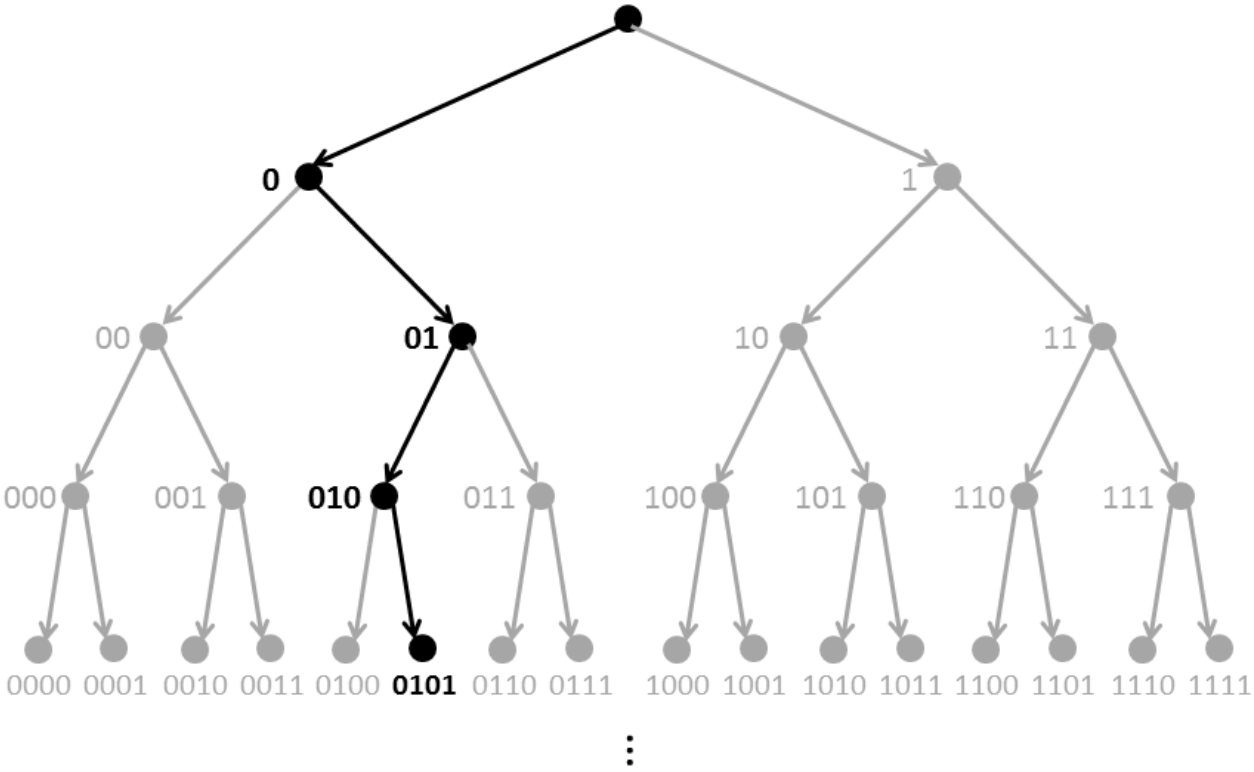

Figure 1. SC decoding tree.

This diagram shows that the decoder chose $\hat{u}_1 = 0$, $\hat{u}_2 = 1$, $\hat{u}_3 = 0$, and $\hat{u}_4 = 1$.

The SC decoder has a simple rule for choosing each bit. When choosing $\hat{u}_i$, if the ith bit is frozen, it chooses the known value of u_i. Otherwise, it computes the ratio

$$\frac{\hat{P}(\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_{i-1}, 0)}{\hat{P}(\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_{i-1}, 1)}.$$

It chooses $\hat{u}_i = 0$ if this ratio is greater than 1, and $\hat{u}_i = 1$ if this ratio is less than 1.

One major disadvantage of this method is that the decoder cannot correct an error made earlier in the process. In [5], Tal and Vardy introduced *list decoding*, also called *successive cancellation list decoding*

[2] $\hat{P}(b_1, b_2, \ldots, b_k)$ is computed using an efficient recursive formula, but the true probability cannot be computed efficiently. The estimate differs from the true probability because the recursive formula does not take into account the knowledge of frozen indices $> k$.

(SCL), to mitigate this problem. They found that list decoder's error correction performance was significantly better than SC, at the cost of higher computational complexity.

The list decoder follows multiple paths down the decoding tree. Figure 2 shows an example of a list decoder estimating u_1 through u_4.

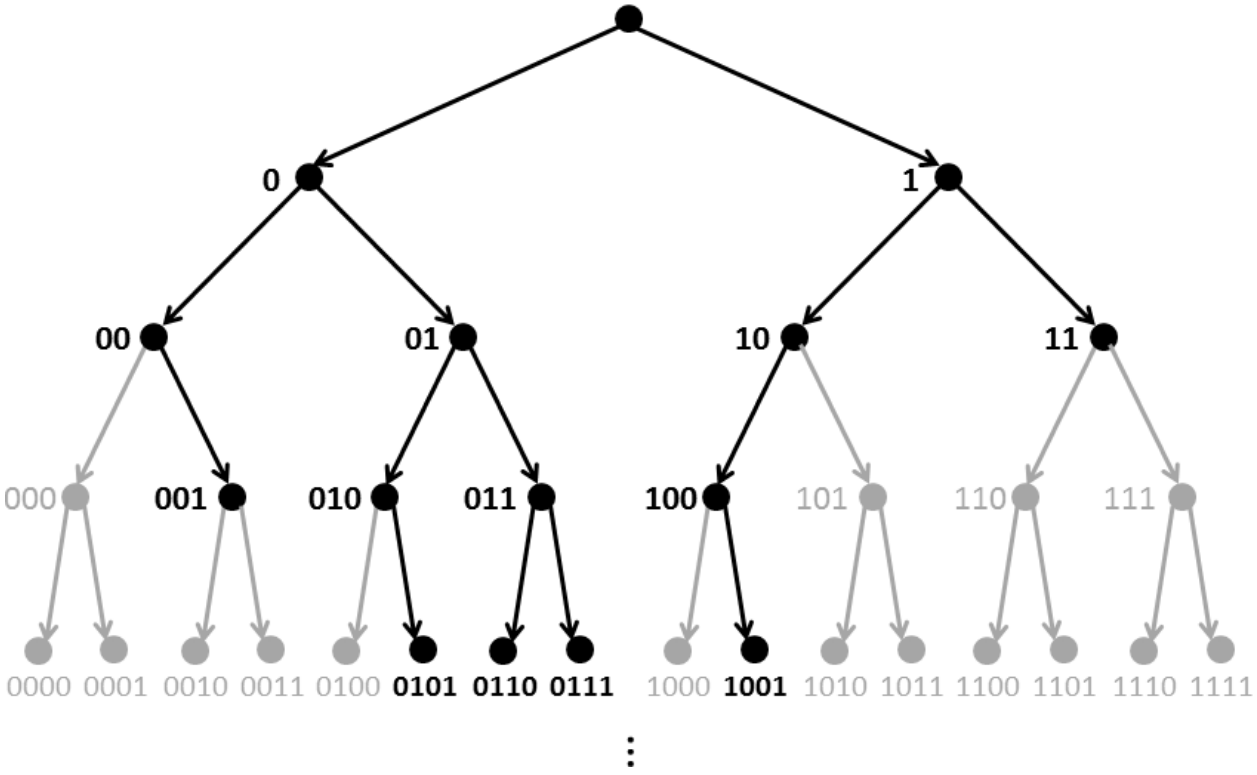

Figure 2. List decoding tree with $L = 4$.

For convenience, assume that none of the first four indices are frozen.[3] Instead of choosing if $\hat{u}_1$ is 0 or 1, the list decoder chooses both, and likewise for $\hat{u}_2$. In this way, it generates a *list* of candidate decodings. After two steps, the list contains four candidates: (0, 0), (0, 1), (1, 0), and (1, 1). After each non-frozen bit, the number of candidates doubles until it exceeds a predetermined maximum. This maximum is called the *list size*, and is represented by the symbol L.

Figure 2 shows an example with $L = 4$. In the third step, the decoder computes probability estimates for all eight possibilities: $\hat{P}(0,0,0)$, $\hat{P}(0,0,1)$, $\hat{P}(0,1,0)$, $\hat{P}(0,1,1)$, $\hat{P}(1,0,0)$, $\hat{P}(1,0,1)$, $\hat{P}(1,1,0)$, and $\hat{P}(1,1,1)$. It retains in the list the L candidates with the largest probability estimates. In the example, these are $(0,0,1)$, $(0,1,0)$, $(0,1,1)$, and $(1,0,0)$.

After the list has been pruned, the decoder does not compute probability estimates for all possibilities, but only the $2L$ possibilities that are extensions of the candidates in the list. In the example shown in 2, it computes $\hat{P}(0,0,1,0)$, $\hat{P}(0,0,1,1)$, $\hat{P}(0,1,0,0)$, $\hat{P}(0,1,0,1)$, $\hat{P}(0,1,1,0)$, $\hat{P}(0,1,1,1)$, $\hat{P}(1,0,0,0)$, and $\hat{P}(1,0,0,1)$. Again, the L candidates with the largest probability estimates are maintained in the list. The computational complexity of the list decoder is roughly L times that of an SC decoder.

After N steps, the decoder has a list L candidates that each include N bits, and it has a probability for each one. It chooses $\hat{\mathbf{u}}$ to be the most probable candidate.

[3] In most practical polar codes, the first four indices are all frozen.

Tal and Vardy experimented with list decoding a (2048, 1024) polar code and found that list decoding with small to moderate list sizes could match the FER of a maximum likelihood decoder.[4] The lower the E_b/N_0, the larger list size is needed: with $L = 2$ the list decoder matches the maximum likelihood decoder for $E_b/N_0 \geq 2.75$ decibels (dB), and with $L = 32$ it matches for $E_b/N_0 \geq 1.75$ dB. However, even with the improvement caused by list decoding, the error-correction performance of polar codes is poor compared to turbo and low-density parity check (LDPC) codes.

3.1 CRC-AIDED LIST DECODING

Tal and Vardy [5] found that a slight change could yield even better results, outperforming the LDPC code used in the WiMax standard. They ran simulations of list decoding and found that when the decoder produces the wrong answer, often the correct answer is in the list. Therefore, instead of always choosing the most probable candidate, it would help to be able to pick the correct answer out of this list. The use of a CRC helps to achieve this goal.

A CRC is an error detection code. It is specified by a binary polynomial of degree d, where d can be any positive integer. The CRC takes as input an information bit vector of any length, and outputs a binary vector of length d, called the *check bits* or the *parity bits*. Both the information bits and the check bits are sent to a receiver. The receiver applies the same CRC to the received information bits, and checks if the result matches the received check bits. If they do not match, an error has occurred. For a randomly chosen bit vector, the probability of passing the CRC is $\frac{1}{2^d}$. More details about CRC can be found in [7].

In polar coding with CRC-aided list decoding, we use an (N, K) polar code and a d-bit CRC. We start with $K - d$ information bits, and use the CRC to compute d check bits. We append the check bits to the information bits, and input all K bits to the polar encoder. The encoder outputs N bits, which are sent to the receiver. At the receiver, the list decoder works as described above until the last step. Instead of simply choosing the most probable candidate, the decoder uses the CRC to test the candidates for errors. If one or more candidates pass, the decoder outputs the most probable among those that pass. If none of the candidates pass, the decoder outputs the most probable candidate, and also outputs a flag indicating that the result is incorrect.

The price we pay for using CRC is that we have $K - d$ information bits instead of K. We can compensate by increasing K. This means we have fewer frozen bits, so the error-correction capability of the polar code is decreased, but in return we get additional error-correction capability from the CRC. If d is chosen well, we receive a net gain.

For the rest of this report we will use the symbol K_i for the number of information bits per block. K will always be the number of non-frozen bits in a polar code.

[4] A naive implementation of a maximum likelihood decoder would have to compute the probability of all 2^K possibilities. In [6] it was shown that this complexity can be reduced to roughly cubic, but this is still too high to be practical when N = 2048. Tal and Vardy did not implement a maximum likelihood decoder, but they did compute a lower bound of its FER and found that the list decoder came very close to this bound, close enough that the difference is not visible in a graph. The FER of the maximum likelihood decoder is less that the FER of the list decoder, but greater than the lower bound, so all three of these values are very close.

4. PARAMETERS OF THE SIMULATIONS

The BER of CRC-aided list decoding depends on several parameters involving the channel, the polar code, the CRC, and the decoder. The following subsections describe these parameters.

4.1 THE CHANNEL

In this report, the "channel" encompasses everything between the encoder output and the decoder input. All channels used in this report are binary-input additive white Gaussian noise (AWGN) channels. Such a channel is determined by a parameter σ. The channel input b can be 0 or 1, and the output is $(-1)^b + n$, where n is a sample from a normal distribution with mean 0 and standard deviation σ. Each time another bit is sent, a new n is chosen, and all the n's are independent.

An AWGN channel can also be described by the ratio of *symbol energy* (E_s) to *noise spectral density* (N_0). We have normalized the energy so that $E_s = 1$ and $N_0 = 2\sigma^2$. We also frequently use the ratio E_b/N_0 where E_b is the *bit energy*, or more precisely, the *energy per information bit*. Thus, $E_b/N_0 = (E_s/N_0)/r$, where r is the code rate K_i/N. Both E_s/N_0 and E_b/N_0 are usually expressed in decibels.

Most FEC literature uses AWGN channels, so we did also, making it easier for us to compare our results to previous results on polar codes and other FEC.

4.2 THE POLAR CODE

For any N that is a power of 2 and any $K < N$, an (N, K) polar code can be specified by choosing a K-element subset of $\{1, \ldots, N\}$.[5] This subset is normally chosen to optimize the performance of the code for a particular channel. We constructed polar codes using the method of Tal and Vardy [3]. All the codes used in this report were optimized for AWGN channels of various E_s/N_0 values. The Tal/Vardy method requires specifying a fidelity parameter μ; we used $\mu = 512$.

The Tal/Vardy method, like most polar code construction methods, is designed to minimize the FER of an SC decoder. In the process, it computes an upper bound for this FER. In our previous work, we found that if $\mu \geq 256$ and FER $\leq 10^{-3}$, then this upper bound is very close to the true FER. In contrast, there is no method known to predict the performance of a list decoder except running a large number of trials. Thus, a code that is optimal for SC decoding in a given AWGN channel may not be optimal for list decoding, and a code designed for a different AWGN channel might perform better.

4.3 THE CRC

The CRC polynomial can be any polynomial with coefficients in $\{0, 1\}$ such that the constant term is 1. The number d of check bits equals the degree of the polynomial. Although d is determined by the polynomial, we treated them as two separate parameters: we can compare CRC lengths, or compare different polynomials at the same length. It is impossible to compare different lengths at the same polynomial.

[5] It is also possible to change the code by specifying the $N - K$ frozen bits. We follow the usual convention that all frozen bits are 0.

4.4 THE DECODER

After specifying the above parameters, there are two things left to vary, the list size and the decoder's knowledge of the channel. When the decoder receives a channel output y, it must compute the likelihoods: if a 0 is transmitted, what is the probability of receiving y? If a 1 is transmitted, what is the probability of receiving y? If the decoder knows that the channel is an AWGN channel, and knows the E_s/N_0 of the channel, it can compute the correct probabilities. If the decoder uses a channel model that differs from the true channel, it will compute different probabilities, which could degrade its performance. In our simulations, the decoder always used a channel model that exactly matched the channel.

4.5 SUMMARY OF PARAMETERS VARIED

We varied seven different parameters:

1. The E_s/N_0 of the AWGN channel ranged from -5.5 to 3.8 dB.
2. The block size N ranged from 512 to 65536.
3. The code rate $r = K_i/N$ ranged from 0.25 to 0.9.
4. The E_s/N_0 that the code was designed for ranged from 3 dB below to 3 dB above the E_s/N_0 of the channel.
5. The CRC length d ranged from 4 to 24.
6. The CRC polynomial: we used four different polynomials with d = 16, and one polynomial for each other length tested. All of these polynomials were taken from [8] or [9].
7. The list size L ranged from 1 to 64. Note that the list size can be any positive integer, but all the list sizes we have seen in the literature were powers of two. We also have tested only powers of 2.

We specify an FEC system by choosing values for parameters 2 through 7, and we test the system by computing the BER as a function of E_b/N_0.

Coding theorists typically compare FEC systems by specifying a target BER or FER, and finding the E_b/N_0 at which each system achieves the target. The difference between two such E_b/N_0's is called *coding gain*. For most of our tests, we used the target BER $= 10^{-5}$, because this is the target BER specified in some Department of Defense specifications, such as in Table XVI of [10]. An expert at SSC Pacific told us that for TCP/IP transmissions, the target BER is 10^{-8}. However, it takes a large number of trials to measure a BER this low, and we could only do this for a few cases.

In practice, N, r, and L must be chosen for the requirements of a particular system because they have a large effect on throughput, delay, and complexity. In contrast, the CRC length, CRC polynomial, and code design E_s/N_0 have little effect on throughput, delay, and complexity. Therefore, for any given combination of N, r, L, and target BER, there is a best combination of CRC length, CRC polynomial, and code design E_s/N_0: the combination that achieves the target BER at the lowest E_b/N_0.

5. RESULTS

The starting point for our investigation of CRC-aided list decoding was the example shown at the beginning of [5]: N = 2048, code rate = $1/2$, code design E_s/N_0 = -1.01 dB, CRC length = 16, and list size = 32. The authors of [5] did not specify what CRC polynomial they used. We started with the polynomial $x^{16} + x^{15} + x^2 + 1$, called "CRC-16-IBM" in [8].

Preliminary tests suggested that code design E_s/N_0 could be varied over a wide range without much effect on BER performance. As a result, we were not always careful in choosing the code design E_s/N_0. [6]

5.1 EFFECT OF VARYING LIST SIZE

Figure 3 shows the effect of changing the list size. All of these tests were performed using N = 2048, code rate = $1/2$, code design E_s/N_0 = -1.01 dB, and the CRC polynomial $x^{16} + x^{15} + x^2 + 1$, except that the test with L = 1 used no CRC. (With L = 1, a CRC cannot help correct errors.)

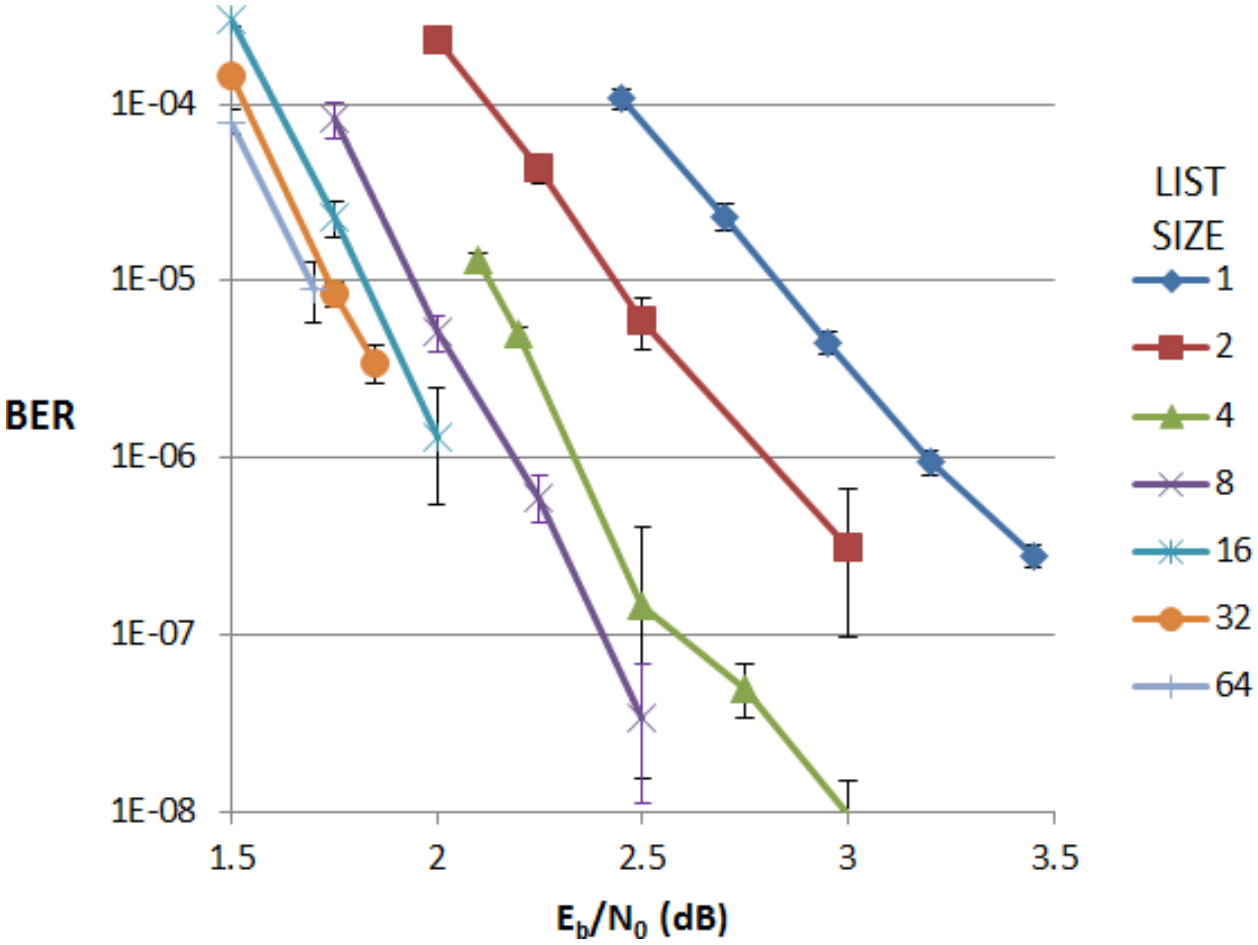

Figure 3. BER vs. E_b/N_0 for N = 2048, rate = $1/2$, and varying list sizes.

[6] The results in this section are not listed in the order they were obtained. For some of these tests, the choice of code design E_s/N_0 was influenced by the results shown in Section 5.6.

The figure shows significant improvement as list size is increased from 1 to 4, and rapidly diminishing improvements as list size is increased further. We chose to use $L = 4$ for most of our subsequent experiments.

5.2 EFFECT OF VARYING BLOCK SIZE

Figure 4 shows the effect of changing the block size. All of these tests were performed using L = 4, code rate = $1/2$, code design E_s/N_0 = -1.01 dB, and the CRC polynomial $x^{16} + x^{15} + x^2 + 1$.

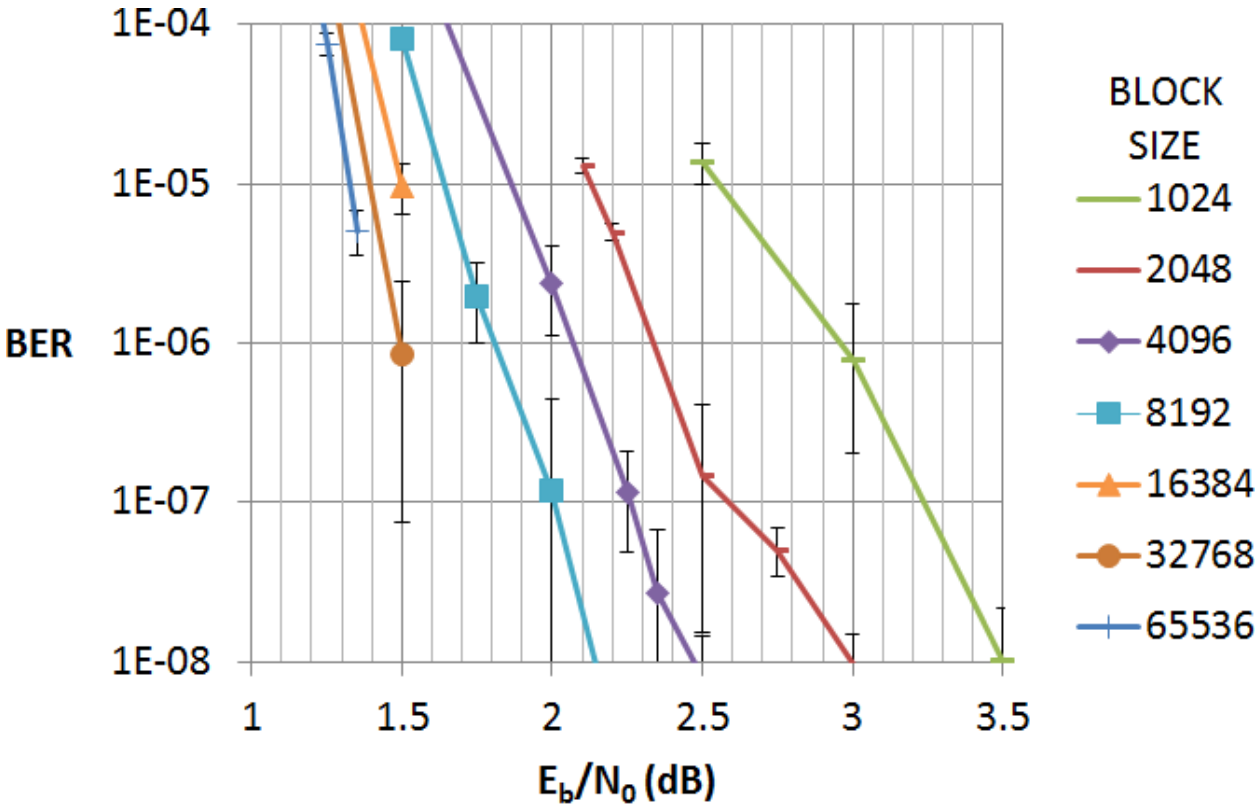

Figure 4. BER vs. E_b/N_0 for L = 4, rate = $1/2$, and varying block sizes.

Again we see diminishing improvement with each doubling of the block size.

5.3 EFFECT OF VARYING CODE RATE

For this set of tests, we selected various combinations of block size and code rate, and determined the E_b/N_0 at which the BER is 10^{-5}. All of these tests were done with L = 4 and the CRC polynomial $x^{16} + x^{15}+x^2+1$. The code design E_s/N_0 was chosen to be close to the E_s/N_0 at which we expected to achieve BER = 10^{-5}.

The previous figures showed BER over a range of E_b/N_0 values, but Figure 5 shows only the E_b/N_0 needed to achieve BER = 10^{-5}. Although E_b/N_0 is still on the horizontal axis, it is now the dependent variable, with code rate as the independent variable. The same curve tells us, for a given E_b/N_0, what is the highest code rate we can use while keeping the BER below 10^{-5}.

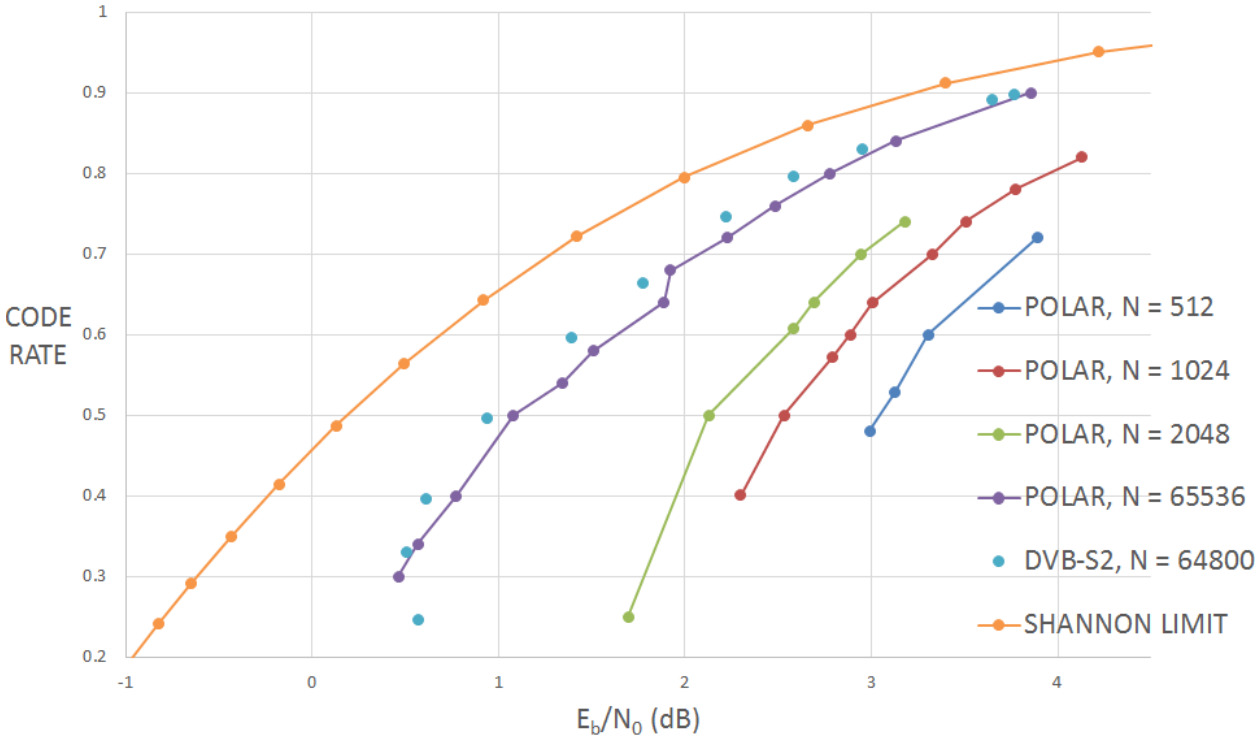

Figure 5. E_b/N_0 needed to achieve BER = 10^{-5} for various block sizes and code rates.

As suggested by an expert at SSC Pacific, we compared the results to the performance of the FEC code used in the DVB-S2 standard. In Figure 5, the DVB-S2 points are not connected because these points correspond to all 11 code rates in the standard. In contrast, the code rate of the polar codes can be adjusted in increments of $1/N$.

Ideally, we should see a smooth curve for each polar block size. We believe the jaggedness is mostly caused by inconsistency in how the code design E_s/N_0's were chosen.

We did not compute error bars for this graph. For each polar code data point, we estimated the required E_b/N_0 by interpolating the BER between two E_b/N_0 values that we tested. The interpolation was linear when viewed on a log-log scale. This produces some error because the BER curve is not perfectly linear, and the points we interpolated between were affected by sampling error. To limit the error, we simulated at least 10^5 blocks for each point, and the E_b/N_0's we interpolated between were at most 0.25 dB apart.

BER performance of the DVB-S2 FEC is hard to find, because within the standard the relevant metric is packet error rate (PER), not BER. We estimated the DVB-S2 points using Figures 4 and 22 of [11]. We assumed that a BER of 10^{-5} corresponds approximately to a PER of 10^{-3}. This should not make much difference because the PER curves shown in Figure 4 of [11] are very steep. We believe these points are correct within 0.05 dB.

Figure 6 shows the same data as Figure 5, but with E_s/N_0 on the horizontal axis instead of E_b/N_0. This is useful because, from a code design perspective, the E_s/N_0 is given,[7] but the E_b/N_0 can be changed by changing r. Recall these are related by $E_b/N_0 = (E_s/N_0)/r$.

[7] In a broader RF design perspective, the E_s/N_0 can be changed in several ways, such as changing the transmit power or the symbol duration.

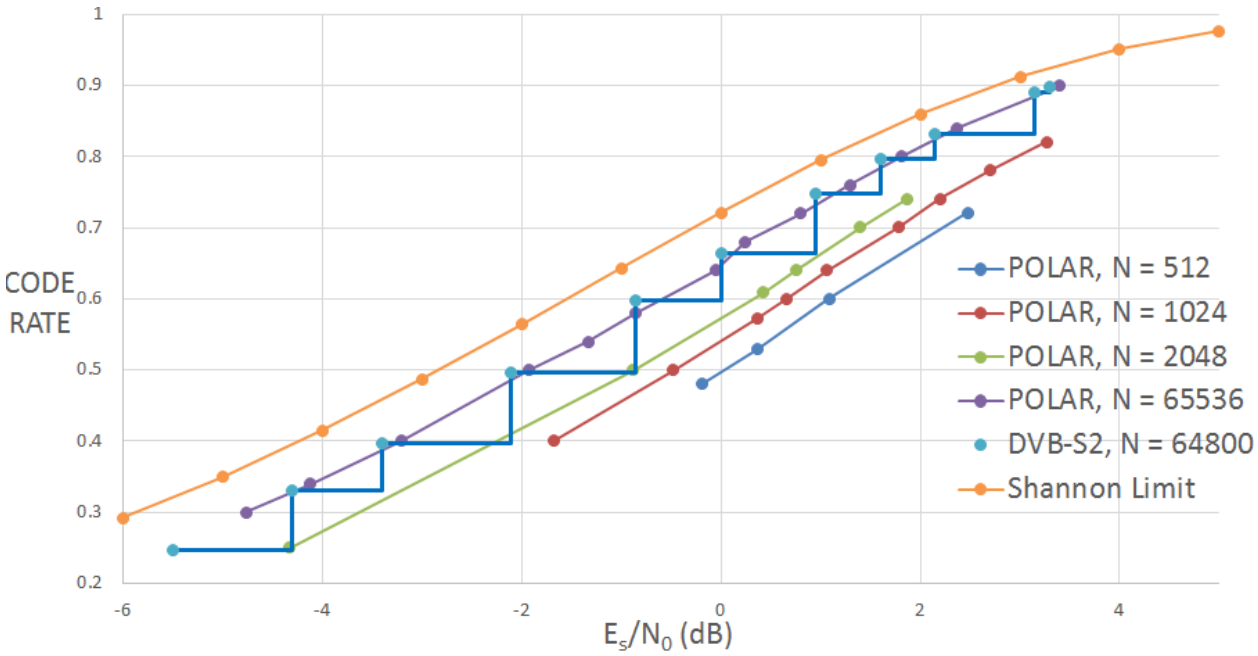

Figure 6. E_s/N_0 needed to achieve BER = 10^{-5} for various block sizes and code rates

The DVB-S2 results form a stairstep curve due to the limited choice of code rates. For example, suppose the E_s/N_0 is -2.1 dB and we are required to maintain BER $\leq 10^{-5}$. Using DVB-S2, we can achieve a code rate of 0.5. Now suppose the E_s/N_0 increases to -1.3 dB. This is not high enough to use a higher-rate DVB-S2 code, so a DVB-S2 user would continue to use rate 0.5. In contrast, using a polar code of approximately the same length, at -1.3 dB we could use rate 0.54. In the range -2 dB $< E_s/N_0 <$ -0.9 dB, polar codes of length 65536 can provide up to 15% more throughput than DVB-S2 codes.

5.4 EFFECT OF VARYING CRC POLYNOMIAL

We compared four different polynomials of length 16:

1. The previously mentioned CRC-16-IBM.
2. $x^{16} + x^{14} + x^{13} + x^{12} + x^{10} + x^8 + x^6 + x^4 + x^3 + x^1 + 1$, from Table 3 of [9]. We refer to this polynomial as "Koopman".
3. $x^{16} + x^{12} + x^5 + 1$, called "CRC-16-CCITT" in [8].
4. $x^{16} + x^{15} + x^{14} + x^{11} + x^6 + x^5 + x^2 + x^1 + 1$, called "CRC-16-CDMA2000" in [8].

Figure 7 shows the results of using these four polynomials with N = 2048, $r = 1/2$, list size 4, and design E_s/N_0 -1.01 dB. Figure 8 shows the results of using these four polynomials with N = 2048, $r = 1/2$, list size 32, and design E_s/N_0 -1.25 dB. In both cases, the design E_s/N_0 was chosen to be approximately the E_s/N_0 at which the BER is 10^{-5}.

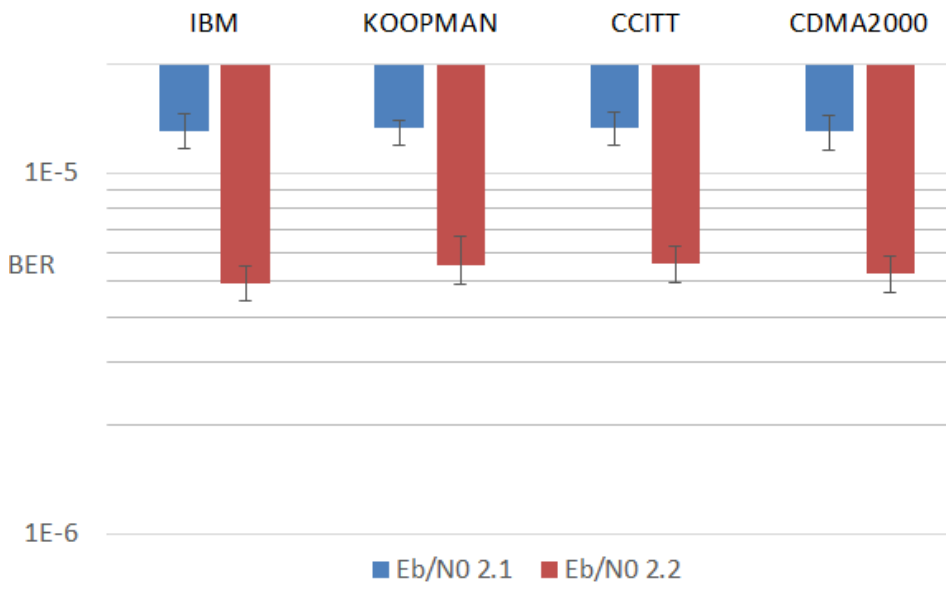

Figure 7. Comparison of 16-bit CRCs with list size 4.

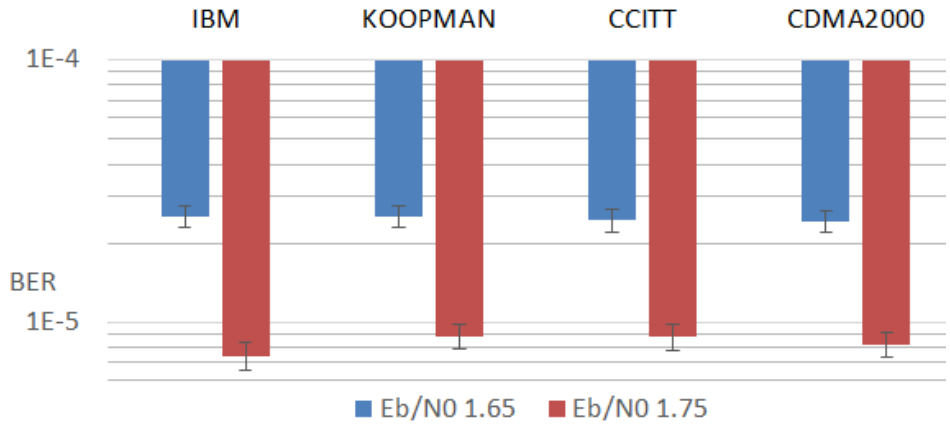

Figure 8. Comparison of 16-bit CRCs with list size 32.

The effect of changing the polynomial is smaller than the measurement error, and too small to matter. For each of the eight systems tested, we interpolated to find the E_b/N_0 at which the BER is 10^{-5}. For both $L = 4$ and $L = 32$, CRC-16-IBM was best, but the difference between best and worst was 0.005 dB and 0.013 dB, respectively. We decided it was not necessary to do further comparisons of polynomials with the same length.

5.5 EFFECT OF VARYING CRC LENGTH

We tested CRC-aided list decoding using the CRC lengths and polynomials shown in Table 1.

As mentioned in Section 4.5, the best CRC length may depend on N, the code rate, the list size, and the target BER. We ran tests to determine the best length in a few different cases.

Table 1. CRC polynomials tested.

CRC length	Polynomial	Source
4	$x^4 + x + 1$	CRC-4-ITU from [8]
6	$x^6 + x + 1$	CRC-6-ITU from [8]
8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$	CRC-8 from [8]
10	$x^{10} + x^9 + x^5 + x^4 + x + 1$	CRC-10 from [8]
12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$	CRC-12 from [8]
14	$x^{14} + x^9 + x^8 + x^7 + x^6 + x^4 + 1$	from Table 3 of [9]
16	$x^{16} + x^{15} + x^2 + 1$	CRC-16-IBM from [8]
24	$x^{24} + x^{22} + x^{20} + x^{19} + x^{18} + x^{16} + x^{14} + x^{13}$ $+x^{11} + x^{10} + x^8 + x^7 + x^6 + x^3 + x + 1$	CRC-24 from [8]

5.5.1 Preliminary Test

In this test we compared CRC lengths d = 8, 12, 16, and 24. For all four lengths, we used L = 4 and the same polar code, with $N = 2048$ and $K = 1040$, designed for E_s/N_0 -1.01 dB. We tested them at E_b/N_0 2.1 and 2.2 dB. The results are shown in Figure 9.

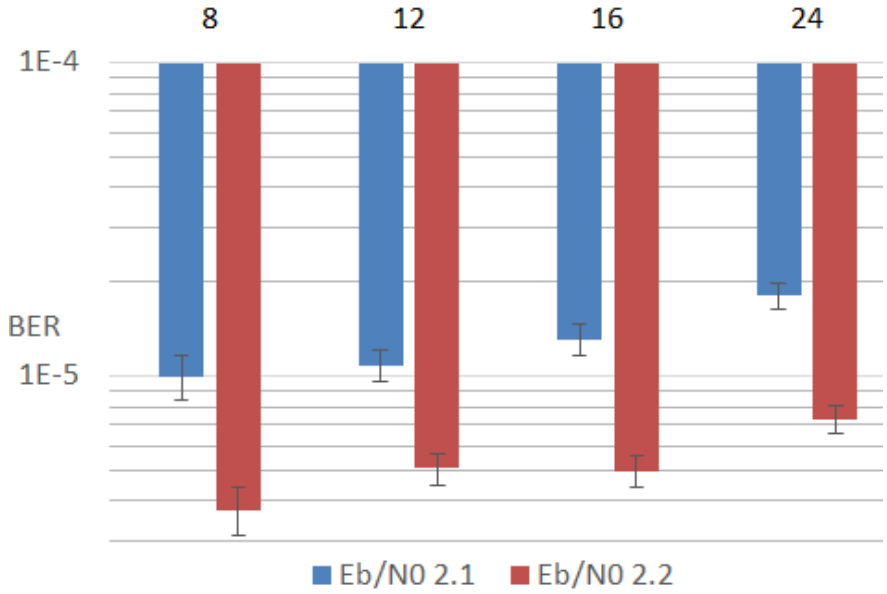

Figure 9. Comparison of CRC lengths with N = 2048, K = 1040, and list size 4.

We can see that length 8 is better than the others. This result may be confusing. In all four cases, the frozen bits were the same, providing the same error correction, and the longer CRCs are better at filtering out incorrect answers on the list, so how can the shorter CRC win? The reason is that we tested them at the same E_b/N_0, not at the same E_s/N_0. The code rate was $r = \frac{1040-d}{2048}$, so a larger d meant a smaller r and lower E_s/N_0, i.e., more noise. The effect of the longer CRC was not enough to compensate for more noise.

This was not exactly a fair test. In this test, the 8-bit CRC was used with rate $1032/2048$, and the longer CRCs were used with lower rates. As shown in Section 5.3, using a lower code rate normally produces a coding gain. Therefore, if we repeated the test with all four lengths at rate $1032/2048$, the longer lengths would have performed even worse.

In subsequent tests, we always compared codes with the same length and rate.

5.5.2 Baseline Test

Since the lengths $d \geq 12$ performed poorly in the preliminary test, in the baseline test we compared CRC lengths 4, 6, 8, and 10. In all four cases, we used $N = 2048$, $r = 1/2$, design E_s/N_0 = -0.91 dB, and list size 4. We tested them at E_b/N_0 2.05, 2.15, and 2.25 dB. The results are shown in Figure 10.

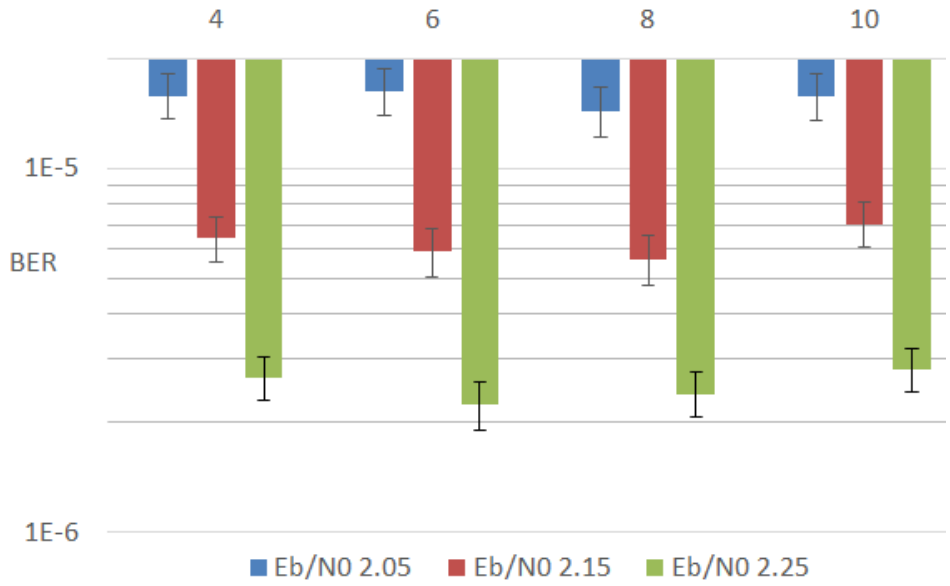

Figure 10. Comparison of CRC lengths with N = 2048, $r = 1/2$, and list size 4.

The 8-bit CRC appears to be best of the four, with the lowest BER at both 2.05 and 2.15 dB. The 6-bit CRC performed slightly better at 2.25 dB, but the difference is well within measurement error.

5.5.3 Test With a Different List Size

For this test, we used list size 32, $N = 2048$, $r = 1/2$, and design E_s/N_0 = -1.36 dB. With a larger size, there is a higher probability that the list will include an incorrect codeword that passes the CRC. Intuitively, it makes sense to mitigate this by using a longer CRC. That is why we chose to test a longer CRC in this test than in the baseline test. We compared CRC lengths 4, 8, 10, and 12. We tested them at E_b/N_0 1.7, 1.8, and 1.9 dB. The results are shown in Figure 11.

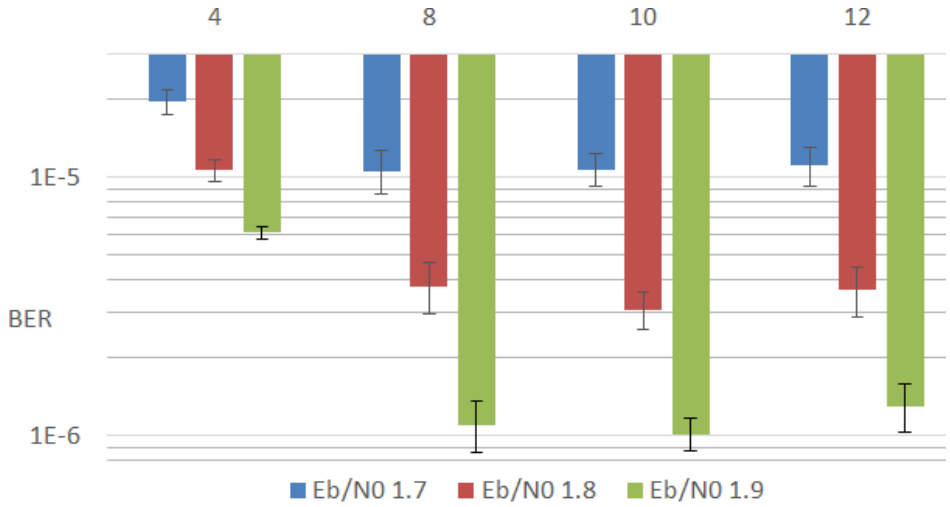

Figure 11. Comparison of CRC lengths with N = 2048, $r = 1/2$, and list size 32.

The 10-bit CRC appears to be best of the four, with the lowest BER at both 1.8 and 1.9 dB. The 8-bit CRC performed slightly better at 1.7 dB, but the difference is well within measurement error.

5.5.4 Test With a Different Code Rate

For this test, we used $r = 0.7$, $N = 2048$, list size 4, and design E_s/N_0 = 0.9 dB. We compared CRC lengths 6, 8, and 10. We tested them at E_b/N_0 2.9, 3.0, and 3.1 dB. The results are shown in Figure 12.

The 8-bit CRC appears to be best, with the lowest BER at both 3.0 and 3.1 dB. The 6-bit CRC performed slightly better at 2.9 dB, but the difference is well within measurement error.

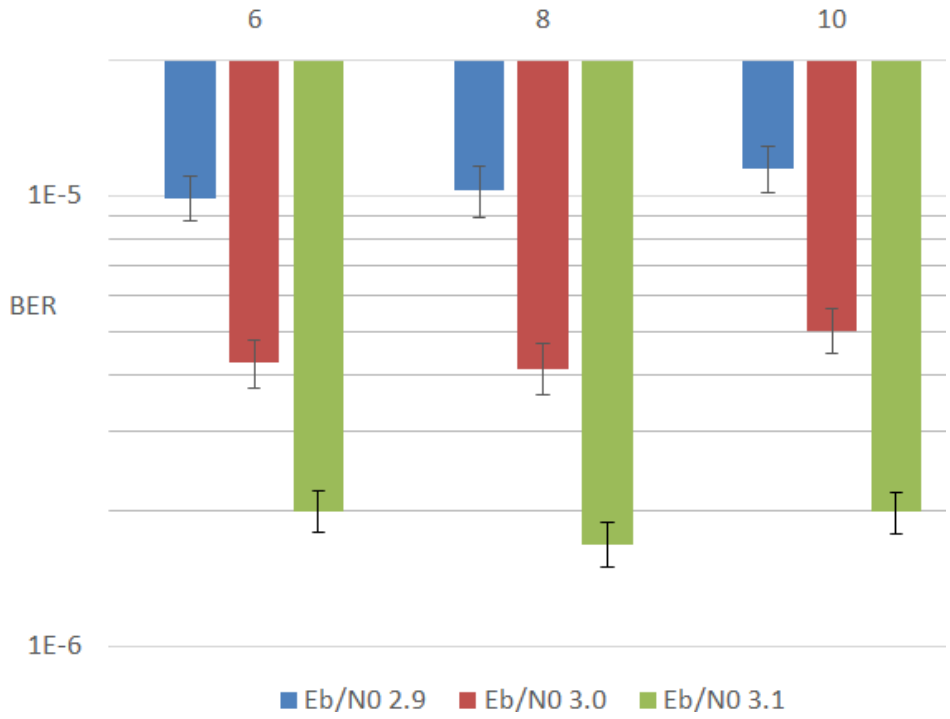

Figure 12. Comparison of CRC lengths with N = 2048, $r = 0.7$, and list size 4.

5.5.5 Test With a Different Block Size

For this test, we used $N = 32768$, $r = 1/2$, list size 8, and design E_s/N_0 = -2.25 dB. We compared CRC lengths 6, 8, and 10. We tested them at E_b/N_0 1.05 and 1.15 dB. The results are shown in Figure 13.

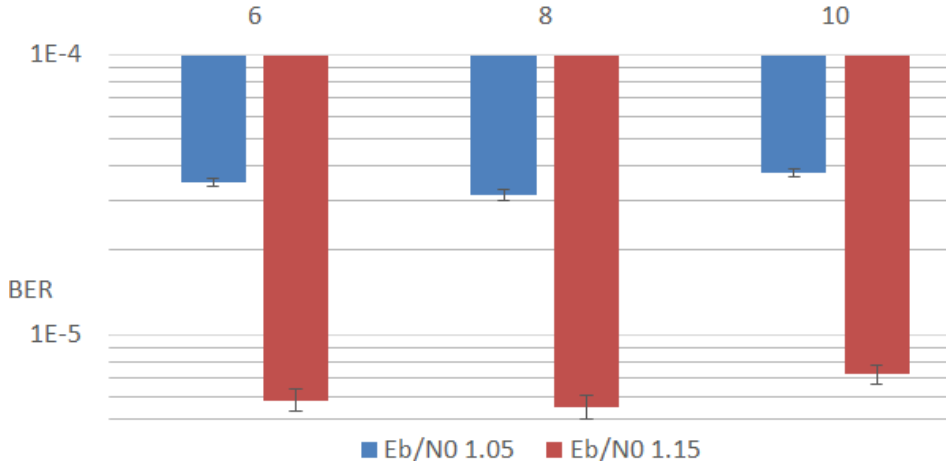

Figure 13. Comparison of CRC lengths with N = 32768, $r = 1/2$, and list size 8.

The 8-bit CRC appears to be best, with the lowest BER at both 1.05 and 1.15 dB.

5.5.6 Test With a Different Target BER

For this test, we used $N = 2048$, $r = 1/2$, design E_s/N_0 = -0.91 dB, and list size 4, which are the same parameters used in the baseline test. We tested CRC lengths 6, 8, 10, and 12 at E_b/N_0 2.6 dB, and lengths 6, 8, 10, 12, 14, and 16 at E_b/N_0 2.7 dB. The results are shown in Figure 14.

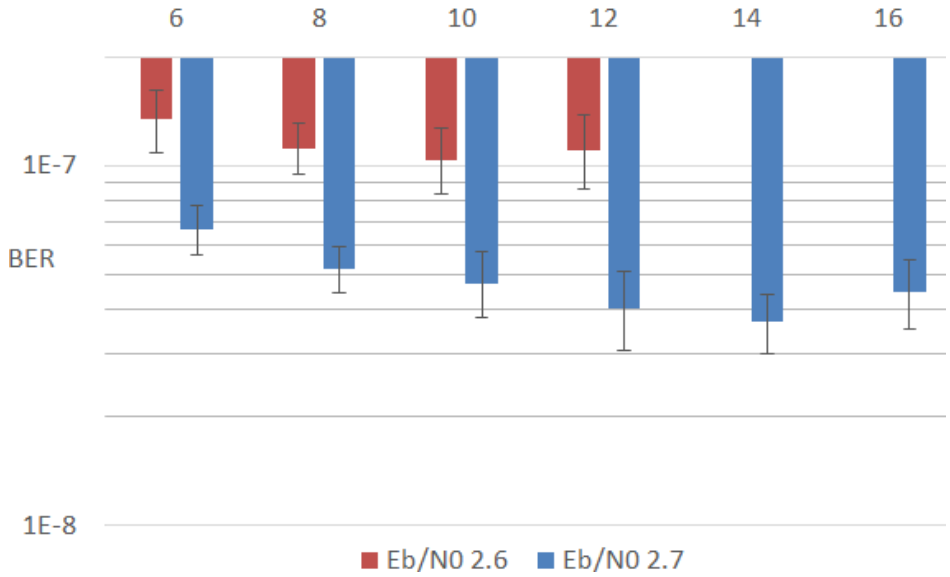

Figure 14. Comparison of CRC lengths with N = 2048, $r = 1/2$, and list size 4.

Recall that in the baseline test, length 8 was best for a target BER of 10^{-5}. It appears that length 10 is best for a target BER of 10^{-7}, and length 14 is best for a target BER of $4 \cdot 10^{-8}$. It appears that, in general, a lower target BER means a larger optimal CRC length. One previously published result supports this hypothesis: Figure 3 of [12] shows a comparison between CRC lengths 8 and 32, with $N = 1024$, $r = 0.84$, and list size 128 [8], with E_b/N_0 ranging from 2 to 5 dB. Both achieve BER approximately 10^{-5} at E_b/N_0 4 dB. For lower E_b/N_0 (equivalently, for higher target BER), length 8 outperforms length 32, while for $E_b/N_0 > 4$ dB (target BER $< 10^{-5}$), length 32 outperforms length 8, with a coding gain of about 0.5 dB at BER $= 10^{-7}$. We do not recall seeing any other published results comparing two different CRC lengths.

5.5.7 CRC Length Summary

For a list size of 4 and a target BER of 10^{-5}, use an 8-bit CRC. The ideal CRC length increases if the list size is increased or the target BER is decreased.

5.6 EFFECT OF VARYING CODE DESIGN E_S/N_0

We would like to have a rule that when the channel E_s/N_0 is x dB, we should use a code designed for E_s/N_0 $x + \delta$ dB. The ideal δ may depend on N, r, L, and E_s/N_0, but we hope to find a δ that will be near-optimal in a wide range of cases.

For this test, we used $N = 2048$, $r = 1/2$, a 16-bit CRC, and list size 4. The δ values we tested were -3, -2, -1, -0.5, 0, 0.5, 1, 2, and 3, and we tested all of these at E_b/N_0 2.1 and 2.2 dB (equivalently, E_s/N_0

[8] No polar code construction method is specified in [12]; in particular, no design E_s/N_0 is specified. It is possible that the results in Figure 3 of [12] were obtained using multiple design E_s/N_0's to match the E_s/N_0's tested.

-0.91 and -0.81 dB). Thus, we tested a total of 18 different codes. The results are shown in Figure 15. We obtained the best results with $\delta = -0.5$.

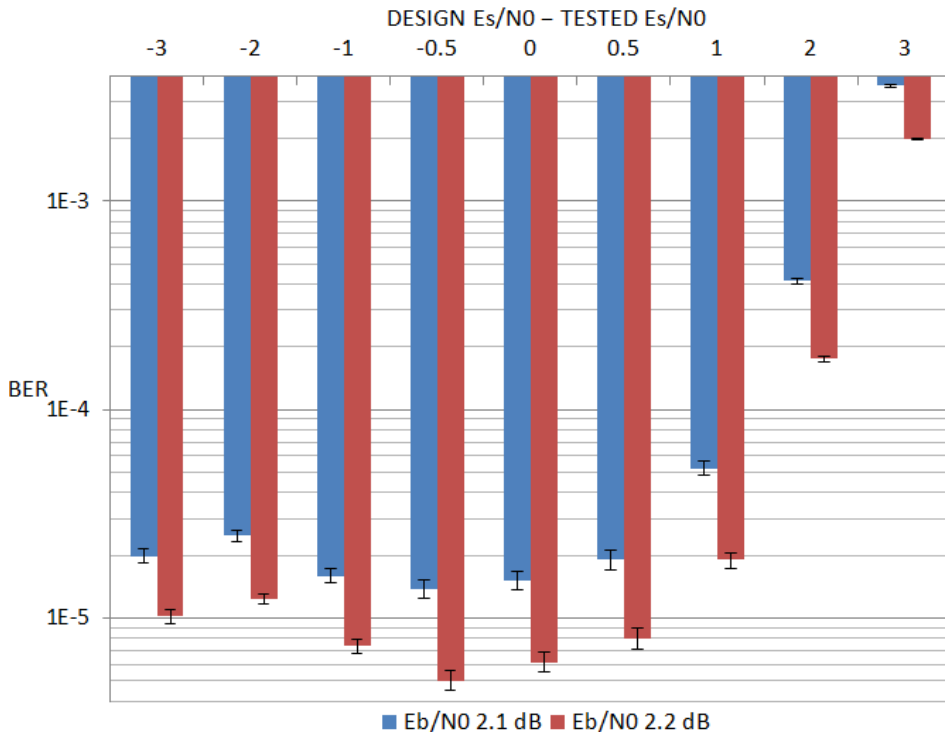

Figure 15. Comparison of design E_s/N_0's with N = 2048, $r = 1/2$, 16 bit CRC, and list size 4.

We have not had time to repeat this test with different parameters, but we did use this result with N = 65536 and several different code rates, to improve the results shown in Figure 5. Several points on the N = 65536 curve appeared to be too far to the right relative to the others. We recomputed these points using a δ near -0.5, and these points moved significantly to the left. As a result, other points now appear to be too far right.

6. IMPLEMENTATION DETAILS

Our decoder software is written in C++. It is based on the pseudocode in [5], which is designed to be used with a systematic polar encoder. Unlike most decoders described in the polar coding literature, this decoder does not use the logarithmic domain for probabilities. We used type `long double` for the probabilities.[9]

We made only four changes to the pseudocode from [5]. First, we corrected a mistake in algorithm 15: lines 1 and 2 are misplaced; they belong after "continue". The same correction recently appeared in a later version of the paper [13]. Second, we found that Algorithm 12 did not work as written. The indices in `pathIndexToArrayIndex` are used both in `arrayPointer_P`, to point to float arrays, and in `arrayPointer_C`, to point to bit arrays. We found that when a float array is copied, the corresponding bit array must also be copied, and vice versa, so that the indexing remains consistent.

The third change is necessary because the pseudocode in [5] is for a non-CRC-aided decoder. The CRC-checking part of the algorithm is described in the text, but no pseudocode is provided. We modified the `findMostProbablePath` algorithm to implement CRC-checking.

Before calling `findMostProbablePath`, the decoder computes L candidate codewords $\hat{\mathbf{x}}_0$ through $\hat{\mathbf{x}}_{L-1}$ and a probability p_l for each one. The function `findMostProbablePath` keeps track of four variables:

- `pPrime`, the highest probability found so far
- `lPrime`, the index where `pPrime` was found
- `pPrimeCrc`, the highest probability found so far among codewords that pass CRC
- `lPrimeCrc`, the index where `pPrimeCrc` was found

`lPrimeCrc` is initialized to -1, which is invalid, and the other three are initialized to 0.

`findMostProbablePath` has a loop where the index l ranges from 0 to $L-1$. If $p_l >$ `pPrime`, the function updates `pPrime` and `lPrime`. Then if $p_l >$ `pPrimeCrc`, the function checks to see if $\hat{\mathbf{x}}_l$ passes CRC. If it passes, the function updates `pPrimeCrc` and `lPrimeCrc`. At the end of the loop, it computes the flag `passedCrc` = (`lPrimeCrc` > -1). If the flag is true, it returns lPrimeCrc as the most probable path index, otherwise it returns lPrime. Note that it is usually not necessary to check CRC for every codeword in the list. `passedCrc` is also returned via pointer.

Checking the CRC is a three-step process. `findMostProbablePath` must first call a bit-reversal function to compute $\hat{\mathbf{w}}_l = \mathbf{\Pi}_N \hat{\mathbf{x}}_l$, then copy some of the entries to form $(\hat{\mathbf{w}}_l)_{\mathcal{A}}$, before it can compute the CRC.

The final change we made is that in [5], the decoder outputs the entire N-bit codeword $\hat{\mathbf{x}}$, but our decoder outputs its estimate of the K-bit input to the polar encoder. This output is via pointer, and the return value is `passedCrc`.

To test the decoder, we repeatedly generate K_i random bits, encode them, simulate transmission through an AWGN channel, decode the channel output, and compare the random bits to the first K_i bits output by the decoder. We did not count errors in the CRC bits.

[9] We used two different compilers. This type was 16 bytes in one compiler and 12 bytes in the other.

6.1 DECODING SPEED

Reference [13] shows that the list decoder requires $O(LN \log N)$ operations. With N = 65536, $r = 1/2$, and L = 4, our decoder runs in 0.3 seconds on one core of an Intel® Xeon® X7560 processor. This speed enables throughput of up to 100 kbps.

Our decoder is designed to be as simple as possible. Other CRC-aided list decoder designs have been proposed to run much faster, both in software and hardware. Almost all such results concern block sizes 512, 1024, or 2048. For example, [12] showed that a list decoder with N = 2048, r = 0.84, and L = 32 could run in 433 ms, with throughput 33 Mbps. We have not seen any results that combined high speed with the error-correction performance that we achieved with N = 65536. The closest is [14], which describes a decoder with N = 32768, r = 0.9, and L = 32, but the speed of this decoder is not given. It achieves BER 10^{-5} at E_b/N_0 4.2 dB, which is 0.3 dB worse than what we achieved at the same code rate with N = 65536 and L = 4. We have not seen any other examples of CRC-aided list decoding with $N > 2048$ in the existing literature.

If we assume that the decoder of [12] can be scaled up to N = 65536 with runtime proportional to $O(N \log N)$, it would run in 0.02 seconds, with throughput up to 23 Mbps, and would probably provide a small coding gain over our decoder.

7. CONCLUSION

In any application for which the DVB-S2 FEC is considered, polar coding with CRC-aided list decoding with N = 65536 should also be considered. With L = 4 in an AWGN channel it approximates the DVB-S2 results when compared at the same data rate, and can provide up to 15% higher throughput by using code rates not provided in the DVB-S2 standard. The current runtime of 0.3 s may be fast enough for some applications up to 100 kbps. Higher speed can be achieved using techniques in the existing literature. Further work is needed to accomplish the following:

- Incorporate these techniques into our software
- Test both polar coding and DVB-S2 coding with BER target 10^{-8}
- Test both polar coding and DVB-S2 coding in more complex channel models such as Nakagami fading

REFERENCES

1. Arıkan, E. 2009. "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073.

2. Wasserman, D. 2014. "Polar Codes," Technical Report 2054. Space and Naval Warfare Systems Center Pacific (SSC Pacific), San Diego, CA.

3. Tal, I. and Vardy, A. 2013. "How to Construct Polar Codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582.

4. Arıkan, E. 2011. "Systematic Polar Coding," *IEEE Communications Letters*, vol. 15, no. 8, pp. 860–862.

5. Tal, I. and Vardy, A. 2012. "List Decoding of Polar Codes," *arXiv preprint arXiv:1206.0050*.

6. Kahraman, S. and Celebi, M. 2012. "Code Based Efficient Maximum-likelihood Decoding of Short Polar Codes," *Proceedings of the 2012 IEEE International Symposium on Information Theory Proceedings (ISIT)* (pp. 1967–1971). July 1–6, Cambridge MA. IEEE.

7. Castagnoli, G., Ganz, J., and Graber, P. 1990. "Optimum Cycle Redundancy-check Codes With 16-bit Redundancy," *IEEE Transactions on Communications*, vol. 38, no. 1, pp. 111–114.

8. Anonymous. 2015. "Cyclic Redundancy Check - Wikipedia, the free encyclopedia," `https://en.wikipedia.org/wiki/Cyclic_redundancy_check`, accessed: 2015-07-10.

9. Koopman, P. and Chakravarty, T. 2004. "Cyclic Redundancy Code (CRC) Polynomial Selection for Embedded Networks," *Proceedings of the International Conference on Dependable Systems and Networks* (pp. 145–154). June 28–July 1, Florence, Italy. IEEE.

10. Department of Defense. 2011. "Department of Defense Interface Standard MIL-STD-188-110C: Interoperability and Performance Standards for Data Modems."

11. Morello, A. and Mignone, V. 2006. "DVB-S2: The Second Generation Standard for Satellite Broadband Services," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 210–227.

12. Sarkis, G., Giard, P., Vardy, A., Thibeault, C., and Gross, W. J. 2015. "Unrolled Polar Decoders, Part II: Fast List Decoders," *arXiv preprint arXiv:1505.01466*.

13. Tal, I. and Vardy, A. 2015. "List Decoding of Polar Codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226.

14. Sarkis, G., Giard, P., Vardy, A., Thibeault, C., and Gross, W. J. 2014. "Increasing the Speed of Polar List Decoders," *2014 IEEE Workshop on Signal Processing Systems (SiPS)* (pp. 1–6). October 2–22, Belfast, Ireland. IEEE.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
August 2015	Final	

4. TITLE AND SUBTITLE

Polar Coding with CRC-Aided List Decoding

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

David Wasserman

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

SSC Pacific, 53560 Hull Street, San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER

TR 2087

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Innovative Science and Engineering (NISE) Program (Applied Research)
SSC Pacific, 53560 Hull Street, San Diego, CA 92152–5001

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report describes some results of the project "More reliable wireless communications through polar codes," funded in Fiscal Year 2015 by the Naval Innovative Science and Engineering (NISE) program at SSC Pacific.

The purpose of the project is to determine if polar codes can outperform the forward error correction (FEC) currently used in Navy wireless communication systems. The project team has implemented an advanced decoding method called cyclic redundancy check (CRC)-aided list decoding.

Our simulation results show that polar coding can produce results very similar to the FEC used in the Digital Video Broadcasting - Satellite - Second Generation (DVB-S2) standard, and can provide up to 15 percent higher throughput by using code rates not provided in the DVB-S2 standard.

In any application for which the DVB-S2 FEC is considered, polar coding with CRC-aided list decoding with $N = 65536$ should also be considered.

15. SUBJECT TERMS

Mission Area: Communications

polar codes, polar encoder, polar code construction, cyclic redundancy check, polar coding algorithms, list decoding, polar decoder, Tal/Vardy Method, forward error correction

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			David Wasserman
U	U	U	U	31	19B. TELEPHONE NUMBER *(Include area code)* (619) 553-3003

INITIAL DISTRIBUTION

84300	Library	(2)
85300	Archive/Stock	(1)
56270	D. Wasserman	(1)

Defense Technical Information Center
Fort Belvoir, VA 22060–6218 (1)

Approved for public release.

SSC Pacific
San Diego, CA 92152-5001