M.S. Project Report

Performance of Cyclic Redundancy Codes for Embedded Networks

Tridib Chakravarty

Department of Electrical and Computer Engineering

Carnegie Mellon University

Prof. Phil Koopman, advisor

December 2001

Abstract

The Cyclic Redundancy Code (CRC) in a network message forms a primary defense against system failures. Because embedded systems often operate in noisy environments, it is vital that CRC polynomials be selected to be optimal with respect to detecting errors in expected traffic workloads. Unfortunately, most standard CRCs do not perform well for the short messages that are commonly sent in embedded system applications, and in general were apparently not designed with short messages in mind. This paper describes a methodology to determine an optimal CRC polynomial for applications using short messages. Additionally, standard 16-bit CRC polynomials are shown to be grossly suboptimal in error detection performance for short messages. Our methodology has identified an optimal 12-bit CRC that yields better error detection than the widely used CCITT 16-bit CRC for embedded network workloads having typical message lengths of 64 bits. Adoption of optimal CRCs would provide improved cost/detection performance tradeoffs for new dependable embedded system designs.

1 Introduction

Embedded networks are becoming prevalent in many types of embedded systems. Distributing functionality provides the benefits of modularity, reduced wiring costs, improved diagnosability, and an ability to create more flexible product family architectures. It is increasingly common for critical systems such as automobiles, aircraft, and industrial automation to use embedded networks. In fact, the work reported in this paper was motivated by a study of error detection performance on a new network for use in rail applications called the Train Communication Network (TCN) [IEC98].

Unlike desktop computer networks, embedded networks often have to provide dependable service in harsh environments. It is common to see elevated Bit Error Rates (BERs) in embedded systems, making it critical that error-detection codes perform well to prevent erroneous messages from causing system failures as they propagate throughout the system. Unfortunately, most embedded networks must operate at relatively low speeds, typically at 1 Mbit/sec or below, in order to be robust and provide global prioritization in support of real time scheduling requirements. Therefore, there is a significant tension between providing large error detection fields in messages to reduce the probability of undetected errors and providing small error detection fields to conserve network bandwidth.

Given that error-detection performance is important for embedded systems, one would naturally assume that particular error-detection codes in common use are optimal given their constraints. Unfortunately, they are not. In fact, error-detection performance can be improved by up to an order of magnitude by selecting an optimal code, with no change in general algorithm, performance or cost.

Embedded networks typically use a Cyclic Redundacy Code (CRC) as an error-dectection code to achieve a reasonable tradeoff. CRC implementations use a shift/XOR process to compute a message digest value that is appended to a network message, where this process is an implementation of a polynomial division over a Galois field [Blahut84]. If a receiving network node finds a mismatch between the received CRC field and the CRC value computed on the received message, it can be certain that the message has been corrupted in transmission. However, if a receiving node does not detect a received message/CRC mismatch, it has only a probabilistic assurance that the message is uncorrupted. Since there are many possible polynomials to use when implementing a CRC, the art is in picking a polynomial with optimal error-detecting properties.

Not all CRC polynomials are equally effective at performing error-detection and, as one would imagine, there are a number of standardized CRC polynomials in widespread use [Costello98]. However, the polynomials selected by these standards were chosen at a time when there was not enough computing power to exhaustively search all possibilities. Furthermore, these standards were primarily chosen for performance on long messages (64 bits to 1024 bits or longer) typical of desktop computer networks rather than for short messages (16 to 128 bits) seen in embedded system applications.Finally, because exhaustive search was impractical, mathematical techniques were used to limit search spaces, generally to polynomials that were a product of (x+1) and a primitive polynomial. While these approaches resulted in the selection of seemingly good polynomial values, they also led to previous researchers missing optimal polynomials that are dramatically more effective for short messages, and also measurably more effective for moderately long messages.

This thesis presents the results of the first complete, analytically exact search of CRC polynomial values for optimal performance. Furthermore, we focus on performance for short messages typical of embedded networks. We identify optimal CRC polynomials and show that they give performance of up to several orders of magnitude better than standard CRC polynomials on short messages subject to independent bit errors without compromising performance on burst error detection. The optimal polynomials can be directly substituted for standard polynomials with no change in algorithm, computation speed, or network bandwidth requirements to attain better error detection performance (assuming, of course, that both transmitting and receiving nodes use the same polynomial for their CRC calculations).

2. A Case Study: The TCN Multi-function Vehicle Bus

The Train Communication Network is being adopted as an international standard for use in critical transportation applications on trains [IEC98]. TCN includes a network specification to be used within nodes on a single vehicle (a single car in a train) called the Multifunction Vehicle Bus (MVB). The MVB operates at only 1.5Mbits per second due to constraints in achieving robust operation in the noisy environment of an electric train. Thus, messages must be kept short and message overhead must be minimized. The MVB makes use of 16, 32, and 64-bit payloads for each message, with each payload protected by an 8-bit error detection code. Longer messages, when used, are broken up into a sequence of 64-bit payloads that are each individually protected by an 8-bit error detection code.

The MVB is operated in a master/slave polling fashion. That means that every message transmission is accomplished via having a master node send a polling message to a slave node, and the slave node respond with a message to be transmitted on the network. Master frames used for polling have a 16-bit data payload, meaning that at least 50% of the messages on the MVB have only 16 bit payloads. In many applications a significant number of slave messages will also have 16 bit payloads, further increasing the proportion of short messages being transmitted.

The MVB uses the following 7 bit polynomial to compute a 7-bit CRC value as part of the error detection code:

G(x) = x^7 + x^6 + x^5 + x^2 + 1

which in a software implementation would be represented by the hexadecimal number 0x53. This represents the FT2-class CRC polynomial from IEC 60870-5. The 7-bit CRC value computed using this polynomial applied across the payload is then extended with a parity bit to form an 8-bit error detection field. However, for the purposes of this thesis we ignore that parity extension of the 7-bit CRC and concentrate only on the CRC itself.

While in the process of reviewing the dependability of the MVB protocol, we assessed the effectiveness of its error detection coding scheme using a Monte Carlo simulation. In that simulation we randomly created message payloads, computed error detection codes, injected random bit flips, and evaluated error detection performance and in particular, $P_{undetected}$ as a function of BER. In order to provide a basis for comparison, we performed a simulation run over a long weekend that compared MVB performance against all possible 7-bit CRC polynomials and 8-bit CRC polynomials.

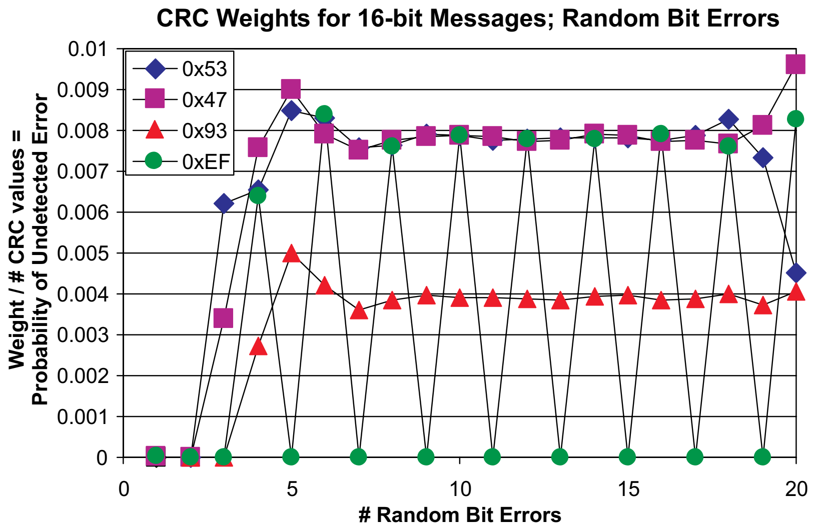

Figure 1. MVB CRC alternative performance with random bit flips

The simulations produced an unexpected result - the MVB polynomial of 0x53 would be outperformed by a factor of 7.4 times better in error detection, by polynomial 0x47 for 16-bit messages (Figure 1). This result was especially surprising given that in the case of this particular CRC polynomial, exhaustive computer search had been performed prior to adoption. Additionally, it was found that an 8-bit polynomial could significantly outperform the 7-bit polynomial + parity bit. The fact than an 8-bit polynomial is superior was not a huge surprise, but the fact that we found one that was not divisible by (x+1) was unexpected, because divisibility by (x+1) is widely recommended for CRC polynomials as it can detect all odd numbers of bit errors (in effect, incorporating a parity check into the CRC calculation).

As Figure 1 shows, both 0x53 and 0x47 detect all single and double bit errors (*i.e.*, they have a *weight* of zero for N=# errors of 1 and 2). However, 0x53 has a weight that is 1.83 times higher for triple-bit transmission errors on 16-bit messages. (Perhaps contrary to the intuition of of desktop system developers, triple bit errors are to be expected as a matter of course in noisy embedded control networks.) The fact that 0x47 performs slightly better for quadruple-bit errors is of no consequence since quadruple-bit errors are significantly less likely than triple-bit errors (as a gross approximation, each increasing number of bit errors is

BER less likely, with typical values for BER being 10^{-4} or 10^{-5} in moderately noisy systems, compared to perhaps 10^{-6} to 10^{-8} in non-embedded wired networks).

Figure 1 also shows the performance of two 8-bit CRC polynomials, 0x93 and 0xEF. (These were explored as a more effective alternative to using an independent parity bit; note that the data in Figure 1 does not include the parity bit for 7-bit CRCs.) 0xEF is the best-performing polynomial that is divisible by (x+1). However, 0x93, which is not divisible by (x+1) gives clearly superior performance. Both 8-bit CRC polynomials provide detection of all triple-bit errors, but 0x93 provides 2.3 times better protection against quadruple-bit errors (and given that quadruple-bit errors are much more likely than quintuple-bit errors for a given BER, this gives significantly better overall error detection performance).

We found these results suspicious since they contradicted previous exhaustive search results used in selecting 0x53. However, we are sure of their accuracy based on recalculating them using exact analytic techniques reported in the next section. What we believe has happened is that previous searches were performed for 64-bit message payloads, in which polynomial 0x53 has slightly better performance than 0x47 (a factor of 1.02). However, the majority of messages for the MVB will have 16-bit payloads, making 16 bit payload error detection of prime importance. We have been told that the MVB uses a 7-bit CRC for non-technical reasons even though it was known than an 8-bit CRC would provide superior error detection.

The fact that an international standard CRC polynomial for use in critical systems could have selected a non-optimal CRC despite an exhaustive computer-based search raises a broader question. How many other standard CRC values are suboptimal? And what can we do to find provably optimal CRC polynomials for use in embedded systems? The following sections review previous work in this area and report the results of an exhaustive search for the best possible CRC polynomials.

3. **Previous Work**

Techniques to determine the probability of undetected transmission errors ($P_{undetected}$) using CRCs has been studied by numerous mathematicians for many years. Equations, efficient algorithms and even custom built hardware have been used to determine $P_{undetected}$ for various polynomials and suggest improvements to existing CRC standard approaches. A few representative pieces of work are discussed in the following paragraphs, although work is too extensive in this field to discuss at length.

Mathematical techniques to find an optimal CRC polynomial have been limited in that they either deal with statistical averages over long messages or require significant computational power to evaluate. In all previous published research, the authors have specifically limited their search for optimal polynomials to polynomials divisible by (x+1). This is in part because CRCs that are the product of a primitive polynomial and (x+1) have clean theoretical properties, in part because divisibility by (x+1) guarantees detection of all odd numbers of bit errors, and in part simply because of cultural inertia.

Wheal & Miller have suggested that the performance of CRC polynomials changes for shorter message sizes [Wheal83]. However there is no recommendation for dealing with that effect.

[Wagner86] has published the mathematics behind an efficient approach for evaluating the effectiveness of a particular CRC polynomial, which has been adopted in our work and is described in detail in a later section.

Wolf & Blakeney [Wolf88] used an analytic technique for exactly evaluating the probability of undetected errors (this approach is noted in their paper to be similar to previous work by Fujiwara *et al.*). Based on that result a CRC called CRC-16Q was proposed having better properties than previous standard CRCs. More recently, Chun & Wolf have created special-purpose hardware for searching the CRC polynomial space [Chun94]. That hardware was limited to searching for certain types of polynomials, and in particular only polynomials divisibility by (x+1). CRC-16Q was also the best 16-bit polynomial identified via hardware searching.

4. Methodology

The goal we have set is to analytically determine the optimal CRC polynomials for embedded networks. This means computing the optimal polynomial for all commonly used CRC sizes with short messages in the presence of independent bit errors at moderately high BERs. This was done by computing $P_{undetected}$ for every possible CRC polynomial of every CRC size of interest for a variety of message lengths. For example, there are 2^6 possible 7-bit CRC polynomials (the highest bit in the polynomial value must be set, resulting in 2^6 possible settings for lower order bits), and all 2^6 were evaluated. Only CRC polynomials themselves were evaluated — the issue of adding a parity bit as done in the MVB was eliminated from consideration since it is an orthogonal concern.

The first approach to evaluation was a Monte Carlo approach involving setting the data payload value, injecting bit flip faults in both message and CRC fields, and then tallying the proportion of undetected errors. Experiments were done involving random placement of a predetermined number of bit flips per message (*i.e.*, based on their weight distributions) rather than based solely on BER to speed up simulation times, with results then scaled by BER after the fact. Complete simulations of all 7- and 8-bit CRC polynomials were completed and used to determine that further investigation was warranted. However, these simulations were too slow to be useful for larger CRC polynomials, and had margins of error large enough to preclude picking a single polynomial as optimal, even on runs lasting several days on a server-class workstation.

In order to speed up execution time and produce exact results, we embarked upon a series of improved approaches that exploit the linearity of the CRC calculation. We present an intuitive explanation below. [Wagner86] describes this same general approach with more mathematical rigor, although without some of the significant implementation optimizations described below since that description is for obtaining the exact performance of a single CRC instead of comparatively ranking multiple CRCs.

4.1. Fast CRC Weight Evaluation

CRC and XOR computations are linear over a Galois Field, and can be though of as being analogous to division and addition for integers. This means in particular that the CRC of an XOR of two numbers is equal to the XOR of the CRC of those two numbers, a property often exploited in computing the CRC of an intentionally modified value:

(1) CRC(A xor B) = CRC(A) xor CRC(B)

If we model a message as a payload part P having N bits and an error detection field E having J bits, we can also model the bits flipped by injected noise as a corruption vector having two parts: a payload corruption vector V and an error detection field corruption vector W. A corrupted message having corrupted payload X and corrupted error detection field Y could then be thought of as the XOR of the original message P||E with the corruption vector V||W given delivered message X||Y. (Note that some of the bits in V or W might be zero depending on where corrupting bits actually occur across the length of the concatenated message P||E.)

Given:

P || E is original message concatenated with error detection field

V || W is the corruption vector

X || Y is the received corrupted message

then:

(2) E = CRC(P); by definition

(3) X = P xor V; Y = E xor W; due to linearity of the CRC function

An error is undetected if and only if the value of the corrupted CRC Y exactly matches the value of the CRC computed on the corrupted received message X. If there is a mismatch, then that indicates successful detection of bit errors. Thus,:

(4) CRC(X) = Y; for any undetected error

but because of linearity, this means that the contents of the original message don't really matter — only the corruption vector:

(5) CRC(V xor P) = E xor W

(6) CRC(V) xor CRC(P) = CRC(P) xor W

(7) CRC(V) = W; for any undetected error.

What this means is that an error is undetected if and only if the CRC of the corruption vector of the payload part V is equal to the corruption vector of the error detection field W.

Based on this result, it is possible to compute an analytic exact result *weight distribution* for undetected errors (*i.e.*, the number of undetected errors having each number of bits set in the corrupting bit pattern). This is done by considering all 2^N possibilities of different patterns of bit flips in the payload without regard to the CRC size. This computation can be performed efficiently by first computing the CRC of every single bit position out of the N bits in the payload (computing N CRC values) and storing them in a lookup table, then XORing together various combinations of these CRC values to form the CRC of the payload corruption vector V for any particular bit flip pattern. It is important to note that for every possible combination of bit errors injected into a payload, there is *exactly one* error detection field corruption vector W that will produce an undetected error. But, this is equal to CRC(V), and so is available based on XORing table values as just described. What is really desired is a bit count (to know how many bits are required to force each particular undetected error scenario), which is bit_count(V) + bit_count(CRC(V)).

Thus, the algorithm for computing the weight distribution for a particular polynomial is to examine all 2^N possible combinations of corruption vectors V and tally the bit count of V and CRC(V) added together for each corruption vector. The result, when weighted by BER probabilities for each particular number of bit errors, gives $P_{undetected}$ for one particular CRC polynomial. This entire computation is repeated for every possible polynomial, resulting in a list from which the optimal polynomial value can be selected. Executing this algorithm for all 7-bit CRC polynomials with 16-bit payloads takes 4.5 seconds on a 500 MHz Alphastation programmed in C and gives an analytically exact result.

4.2. **Optimizing for Comparisons**

With payloads of 32 bits and longer, execution times are still too long even with this improved algorithm. A further optimization not published previously is based on the realization that we are interested in ranking polynomial choices rather than computing an exact weight distribution for every polynomial. Thus, we usually only need the weights of the first one or two error bits for which there are any undetected errors rather than all possible error vectors. ([Wheal83] makes the observation that only the first one or two weights matter, although they do not propose this as a mechanism to speed up polynomial searches.) While computing this is slightly complicated because the number of error bits in the inner loop varies depending on the bits in CRC(V), it is guaranteed that if all cases of corruption vector V with at least B bits have been considered, then since the number of bits in CRC(V) is obviously non-negative, all possible message corruptions with weight B or higher have been considered. Because every succeeding number of B possible corrupted bits is significantly less likely with reasonable BERs, then only the first few non-zero weights need be considered to evaluate $P_{undetected}$ to a known error bound. (Further details of this process are straightforward and are omitted due to space constraints.)

There is one additional optimization involving computing the dual code of the CRC polynomial that would further reduce the search space by a factor related to the width of the CRC [Wagner86]. However, the speedup gained by examining only the first few non-zero weights is a more powerful optimization, and gives large enough speedups that it is not worth performing this additional optimization for the short message lengths we are concerned with. While exact weightings are not required for the search phase, for completeness we compute the exact weightings for standard CRC and the optimal CRC polynomial once it has been found.

The execution speed of the algorithm is determined by the number of corruption vectors considered times the number of CRC polynomials examined plus negligible overheads for setting up CRC lookup tables and the like. The number of corruption vectors considered is bounded by $\left(\frac{N}{B}\right)$ where N is the number of bits in the payload and B is the number of errors in the first, second, or occasionally a third non-zero code weighting as determined by the error bounding algorithm. The number of CRC polynomials is 2^{J-1}, where J is the number of bits in the CRC. Computation times become significant on current computing platforms for CRCs with more than about 24 bits and payloads with more than about 64 bits, but neither of these limitations is of much significance for embedded networks.

4.3. **Optimal Polynomial Selection**

The methodology for selecting an optimal polynomial for an embedded workload consists of the following steps:

- Select the number of bits desired in the CRC, using the results in this paper as a guide to expected error detection capabilities.
- Find the weights of all polynomials of that CRC size for each length of message in the workload. This can be time consuming for large messages, but need be done only once and the results saved for use on future workloads.
- Find the CRC with the optimal error detection properties using a weighted average of $P_{undetected}$ for each message length. This weighting is required because each polynomial has a different ratio of performance tradeoff on long vs. short messages.

The result will be an optimal polynomial tailored to the expected execution environment. If the workload is unknown, we recommend identifying the top several polynomials at the longest message lengths of interest (usually there are several that are about the same in performance) and then selecting the polynomial among them that has excellent performance for the expected average message length. While not optimal, this should produce better results than adopting most standard CRC polynomials.

5. **Analytic Results**

The exhaustive search techniques described in the previous section were used to identify the optimal CRC polynomial for 16-bit payloads that have reasonable performance on 64-bit payloads. These optimal codes are comapred to current standards. The results immediately highlighted the fact that different polynomial codes perform optimally for different message legths. We ran all 16 bit CRC codes over message lengths 10 bits - 128 bits and determined optimal polynomial codes within that range.

5.1. Overall Results

Table 1 shows the performance differences between standard CRC polynomials and optimal CRC polynomials for a weighted combination of 16-bit and 64-bit payloads. The evaluations were performed at a moderately high BER of 10^{-5}, although the approximate improvements would be similar for other reasonable BERs. It is interesting to note that most of the optimal CRC values are not divisible by (x+1) even though that is one of the usual criterion by which CRC polynomials are selected.

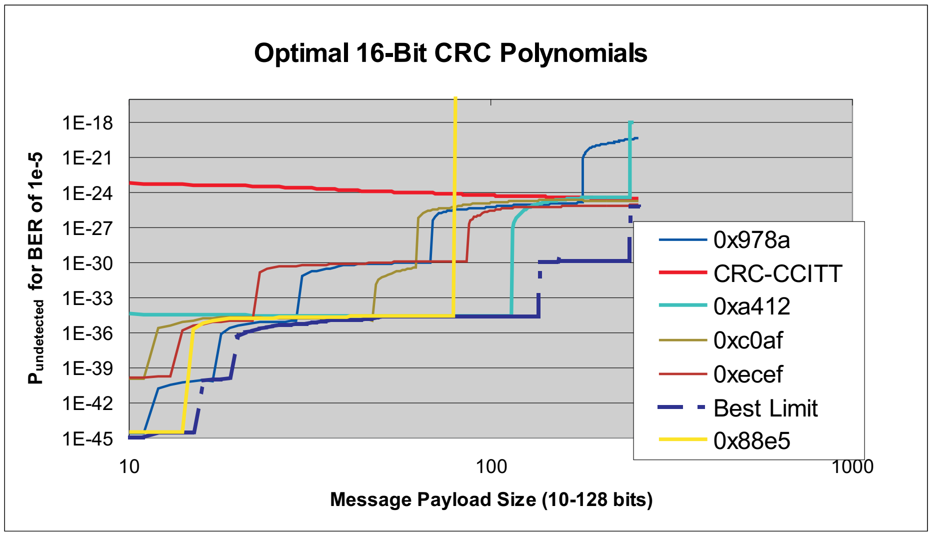

Figure 2. Optimal 16 bit CRC codes for message lengths 10 bits - 128 bits

Figure 2. highlights the result that certain codes perform optimally for certain message length ranges. We ran simultations for all 2^{16} 16 bit polynomials for message lengths of 10 bits - 128 bits. The above graph shows Probability $_{\text{undetected}}$ vs Message Length for a number of CRCs that perform optimally in this range. Additionally, we also plot the performance of CRC-CCITT (16 bit CRC standard) as a comparison. At every message length, there is a polynomial code that gives the lowest $\text{Probability}_{\text{undetected}}$ for message corruptions. For every message length, the lowest $\text{Probability}_{\text{undetected}}$ is plotted to give the "Best-Limit" curve in the graph above. That curve is the best (lowest $\text{Probability}_{\text{undetected}}$) any polynomial can achieve at a given length. Thus, optimal codes will straddle the "Best Limit" line. A number of polynomials perform optimally for a specific message legth but the truly optimal codes are those that perform optimally for a message length range.

Consider the polynomial code 0x88e5 performs optimally for the range 10 bits - 14 bits and then stradles the "Best Limit" curve again for the range 30 bits - 79 bits. Polynomial 0xa412 is not optimal for smaller messages but performs optimally from 40 bit messages up-until 122 bit messages. Thus, if we were to expect some messages in the 10 bits - 14 bits range and majority of messages in the 30 bits - 79 bits range, we would be guaranteed that 0x88e5 is the optimal code for this range. The polynomial standard CRC-CCITT is sub-optimal for this range and has a $\text{Probability}_{\text{undetected}}$ high above the "Best-Limit" curve for this range, proving that it is not effective to protect messages in this range.

5.2. Burst Error Results

CRCs are also useful in detecting errors other than independent bit errors. Burst errors are important as they are prevalent in high noise environments in which a noise disturbance can corrupt numerous simultaneous bits in a network. We conducted experiments to ensure that the recommended polynomials offer the same level of burst error protection as the standards while providing superior independent bit-error detection.

A burst error of length N is conventionally defined as an error that affects N continuous bits. We however describe a burst of length N as a burst vector with bit T and bit T+N corrupted and then consider all possible corruptions within this window of length N. We adopt this definition because it characterizes a real life burst more closely, in which bits are randomly flipped in an error window.

We characterized burst errors into the following: (1) bursts that affect only the message part (2) bursts that corrupt the message are well as the CRC, by straddling over the message boundary into the CRC and (3) bursts that only corrupt the CRC. We need not simulate the third case, as by definition these bursts will be detected. We however simulate the first and second case.

In the first case, if the burst error vector resides completely in the CRC, then the only way that this burst will go undetected is if its CRC = 0. Thus, when the burst resides completely in the message, its CRC is computed and if it is 0, then the burst is marked as undetected.

If the burst straddles the message and CRC, then we look at the corruption in the CRC area. If the corruption in the CRC area is the same as the CRC of the corruption in the message area, then the burst is marked an undetected corruption.

This was repeated for all polynomials for message lengths of interest. We found that all polynomials of the same length, generated the exact same weights for burst errors when applied to a particular message length. Thus, all polynomials of the same length offer the same error detection capabilities for burst errors..

Like the independent bit errors, burst errors were also validated using two independent implementations by separate individuals.

5.2.CRC-CCITT vs. Optimal 16-bit CRC

CRC-CCITT is in use in a number of embedded systems, including all systems employing the Echelon LonWorks network. However, as an inspection of Table 1 reveals, $P_{undetected}$ for both 16- and 64-bit payloads is only about as good as that provided by the optimal 12-bit CRC, and are a factor of more than 10^7 worse than an optimal 16-bit CRC. The improvements for the optimal 16-bit CRC compared to standard 16-bit CRCs are so large that a further discussion is warranted to explain these results.

The CRC-CCITT polynomial is: $x^{16} + x^{12} + x^5 + 1$, which corresponds to hexadecimal value 0x8810 for our purposes (the conversion to a hex value is accomplished by dropping the "+1" term and representing all the non-zero coefficients of powers of x by a 1 bit in the value). By comparison, the optimal 16-bit CRC found in these experiments was 0x978A, corresponding to a polynomial value of $x^{16} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^4 + x^2 + 1$, which is not divisible by (x+1).

Table 2. Weights of CRC-CCITT compared to optimal 16-bit CRC weights for 16-bit messages.

	CRC-CCITT 0x8810		Optimal 16-bit CRC 0x978A	
N (bits)	WEIGHT	FRACTION UNDETECTE D	WEIGHT	FRACTION UNDETECTE D
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	16	0.000445	0	0
5	0	0	0	0
6	50	0.000055	0	0
7	0	0	30	0.000009
8	308	0.000029	227	0.000022
9	0	0	595	0.000021
10	1750	0.000027	761	0.000012
11	0	0	1575	0.000012
12	6908	0.000031	3869	0.000017

Table 2 shows the weights of both CRC-CCITT and the optimal polynomial for 16-bit payloads. CRC-CCITT permits 16 distinct quadruple-bit errors to go undetected (*i.e.*, it has a weight of 16 for N_1=4 total corrupted message and CRC bits) out of a possible 2^{16}-1 possible 16-bit message corruption vectors. In contrast, the first non-zero weight on the optimal polynomial is for N_2=7 (N_2=5 for 64-bit messages reported in Table 1). Using a simple approximation, this would mean that one would expect the optimial CRC polynomial to be $BER^{(N_2-N_1)}$ times better. For a BER of 10^{-5}, this is approximately $(10^{-5})^3 = 10^{-15}$, which is in the ballpark of the exact value of $9.00 \times 10^{-41} / 4.45 \times 10^{-24} = 2.02 \times 10^{-17}$ based on results in Table 1.

As a further check on this result, one of the error vectors that the analysis shows goes undetected is:

0x00018810

This error vector can be split into payload and CRC portions, with the payload value of:

0x0001

And a CRC value of:

0x8810

Which is correct by inspection, since the 0x0001 represents a corruption of the very last bit shifted into the CRC, resulting in a CRC value equal to the feedback polynomial of 0x8810. (Note that this value is not the same as would be computed by a real CCITT CRC function, since the CCITT calculation also includes some bit complementing and other factors that do not bear on the math of the CRC itself CCITT.)

This example reveals a basic property of CRCs: the first non-zero weight of a CRC is at a value of N no larger than the number of non-zero terms in the CRC polynomial (including the +1 term). Thus, since

CRC-CCITT has only 4 non-zero terms, it is guaranteed to be vulnerable to 4-bit errors as found in the experiments. Also, since the optimal 16-bit CRC has 9 bits set, it is reasonable that it has a non-zero weight at 7 bits (which is less than the bounding value of 9 non-zero coefficients in its polynomial). To our knowledge this is a novel observation, although somewhat obvious in retrospect. [Wolf88] has previously noted that a small number of non-zero coefficients is undesirable, but did not propose a mechanism to explain the effect.

6. Discussion of Results

The analysis results demonstrate that significant gains in error detection capability can be obtained by using CRC polynomials other than the standard polynomials that are in use worldwide. While there have been a few previous publications that indicated the standard CRCs were not optimal, this is the first work that has discovered such dramatic opportunities for improvement. Finding such a large opportunity for improvement in widely used standard approaches is a somewhat startling result, and so merits some discussion as to why it is so and speculation as to why it apparently has not been found, or at least published, before.

6.1. Examination of short payloads

Previous work has generally concentrated on long messages, and thus has simply never considered shorter messages in which the performance of different CRC polynomials becomes more pronounced. To be fair, this is appropriate for desktop computing networks in which long messages are sent most of the time. In particular, error rates are generally quite low in such networks, making multi-bit errors likely only in the longer messages. However, in embedded networks such as the MVB discussed earlier, shorter messages can be the ones of concern and bit error rates are often quite high due to operation in harsh environments, strong electromagnetic interference, and the like. Generally standards bodies creating embedded networks have simply adopted existing CRC polynomials without examining the assumptions behind them, or in the case of the MVB apparently evaluated polynomial performance at the maximum payload length instead of the expected payload length.

The physical mechanism for the dramatic differences in performance for short payloads has to do with what happens if an error is injected in the last few bits of a payload to be fed into the CRC computation. Errors in most of the bits of a payload are fairly thoroughly mixed by the CRC computation, with any arbitrary number of bit flips causing approximately half the bits in the CRC to be modified (this is because a CRC function is in effect a pseudo-random number generator, meaning that any change in input pseudo-randomly flips half the bits of the CRC output on average).

However, consider a corruption of just the last bit shifted into a CRC. This last bit corruption will modify the CRC value by XORing it with the value of the CRC polynomial. Thus, it is possible in general terms to get an undetected error by flipping the last bit of the message and flipping bits corresponding to the polynomial value. If the CRC polynomial has only a few bits set (*i.e.*, a few non-zero coefficients in the polynomial), then this creates a vulnerability to an undetected error of only a few bits in magnitude. With long messages the effects of these last few bits are diluted, and to a degree can become lost in the noise of average performance over a long message. Thus, analysis of short message performance is required to clearly identify polynomials that are superior for shorter messages.

6.2. Examination of polynomials not divisible by (x+1)

Mathematical analysis and even previous custom polynomial searching hardware ([Chun94] was built to search exclusively for polynomials divisible by (x+1), because these polynomials find all errors having an odd number of corrupted bits. However, there is a potential problem with this approach in that pushing $P_{undetected}$ to zero for odd numbers of bit errors approximately doubles it for even numbers of bit errors (compare the graphs for 0x93 and 0xBC on Figure 1 — 0xEF is divisible by (x+1)). But, because it is only the first few non-zero terms that matter in the weight distributions for independent bit error detection, using a polynomial divisible by (x+1) creates a false sense of efficiency. (It is interesting to note that we have not seen a graph like Figure 1 in any previous publication. Perhaps because previous graphs were limited to BER graphs there was a loss of opportunity to build intuition.)

What we have found via exhaustive search of all polynomials is that it is usually the case that optimal polynomials are not divisible by (x+1). This effect is most dramatic on 16-bit CRCs where polynomials divisible by (x+1) have non-zero weights for 4-bit errors while the optimal polynomial has a zero weight for 4-bit errors. While the optimal polynomial has a non-zero weight for 5-bit errors on 64-bit messages, that is of little consequence since it does not have a 4-bit error vulnerability.

6.3. Number of Non-Zero Polynomial Coefficients

It is worth noting that one of the usual criteria for selecting polynomials several decades ago was minimizing non-zero polynomial coefficients to conserve hardware gates. This consideration is largely irrelevant in current hardware and completely irrelevant in software implementations, but apparently survives in the form of standard CRC values retaining a small number of non-zero coefficients. And, as discussed previously, polynomials with few non-zero coefficients are suseptible to undetectable errors by a bit flip pattern identical to the polynomial coefficients plus the last bit shifted into the CRC computation.

6.4. Increases in computational power

A final reason that it may be time for these findings is that the last time these issues were widely considered by the research community was in the mid-1980s, when a mainframe CPU had less than 1% of the computing power available on a current workstation. Even custom-built hardware discussed more recently in [Chun94] had a clock rate less than one tenth the speed of the workstations we have used for this work. Thus, part of the reason this new result has emerged after all these years may be simply that until now there was not enough readily available computing power to perform an exhaustive search of possibilities.

6.5. Validation

It is important to validate the results of any computer-assisted analysis. The software used to perform the computations were verified and the results in this paper were validated by the following methods:

- Using a Monte Carlo simulation to explore the search space before applying optimized search techniques.
- Comparing refined search techniques against the original Monte Carlo simulation.
- Comparing the refined search approach against a mathematical description of a similar search technique found in the literature.
- Performing a code inspection of the refined search software (and comparing outputs of optimized and unoptimized versions of the code for producing identical results).
- Comparing the refined search approach results with results of a Monte Carlo simulator written independently (and with independent supervision by someone who did not know the mathematical techniques being used).
- Finding that results for 7- and 8-bit CRC values for the MVB CRC matched results performed independently a number of years ago with respect to optimal polynomial selection for 64-bit messages.

6.6. Impact

While error detection performance for multi-bit errors beyond a certain point seems to be of secondary concern to people accustomed to desktop computing, this issue is of vital importance in the embedded systems world. This is becauseembedded networks tend to be noisy, the number of installed units tends to be high, the consequences of failures can be dramatic, and the cost of spending unnecessary bits on CRC values can be significant.

While the opportunity to improve the MVB error detection by a factor of 1.46 is enticing, it is probably not a dramatic enough gain to justify re-engineering a system already commited to a stanrdards document (at least until a major overhaul of the standard is performed in a decade or so).

However, the MVB is important as an illustration that new communication standards are being created continually, and that in general they tend to adopt "proven" CRC polynomials without necessarily investigating how appropriate they may be for embedded applications. It is hoped that this approach and the data in this paper illustrate that the effects of computing a CRC across short messages are significant, as are the opportunities for dramatically increased performance, significantly reduced CRC size, or a tradeoff of both.

7. Conclusions and Future Work

We have presented a methodology and example calculations of how to determine the optimal CRC polynomial for use on an embedded network with a mix of short to moderate-sized message traffic.

This is the first publication to present exhaustive CRC polynomial search results across all polynomials rather than limiting the search space to polynomials divisible by (x+1). As a result, optimal polynomials that outperform the divisible-by-(x+1) class were found that can substantially improve error detection performance. Additionally, a focus on short payloads discovered that some polynomials that have essentially identical performance for long messages can have dramatically different performance for short payloads. The results presented take both of these issues into account.

As a somewhat surprising result, we have found that standard 16-bit CRCs give poor performance for the number of bits spent within a message, under-performing an optimal CRC by a factor of 10^7 for a BER of 10^{-5} (the under-performance increases as BER gets smaller). In fact, the 16-bit CRC-CCITT can be out-performed by an optimal 12-bit CRC for independent bit errors (although a 12-bit CRC cannot match a 16-bit CRC for burst errors). We hope that future embedded network designers will adopt a CRC optimal for their planned network workload.

While it is perhaps unreasonable to expect existing protocols to change CRC polynomials, there are many new embedded network protocols being developed, and that trend shows no signs of abating. It is hoped that newer protocols will adopt these improved CRC polynomials rather than suboptimal existing polynomials. It is also possible that optimal polynomials will be phased into use as new versions of standardized protocols are introduced over time. Doing so will reduce the probability of incidents occuring due to corrupted data being accepted by distributed embedded system nodes due to unlucky combinations of bit corruption patterns.

Finally, it is important to note that there are considerations beyond the mathematical effectiveness of CRCs that are important in designing protocols. These include analog considerations in the network interfaces and tradeoffs made in bit encoding approaches. However, having a CRC that is tailored to an expected workload can provide significant benefits at a cost equal to or even less than currently standardized approaches.

8. Acknowledgements

This work was funded by ADtranz Corporation and PITA. PITA is the Pennsylvania Infrastructure Technology Alliance, a grant sponsored by the Pennsylvania Department of Community and Economic Development. Equipment support was provided by Intel, Compaq and Microsoft. Thanks go to Aditi

Bajoria for creating the Monte Carlo simulation software used for validation, and to Hubert Kirmann for stimulating and insightful discussions.

9. **References**

[Blahut84] Richard E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley, London, 1984.

[Costello98] Daniel J. Costello Jr., Joachim Hagenauer, Hideki Imai, and Stephen B. Wicker, "Applications of Error-Control Coding," *IEEE Transactions on Information Theory*, Vol. 44 number 6, October 1998.

[Chun94] Dexter Chun and Jack Keil Wolf, "Special Hardware for Computing the Probability of Undetected Error for Certain Binary CRC Codes and Test Results", *IEEE Transactions on Communications*, Vol. 42, No. 10, p. 2769, October 1994

[IEC98] International Electrotechnical Commission working group 22 of IEC Technical Committee 9: Electric railway equipment. *Train Communication Network*, working document 61375-1, 1998-06.

[Wagner86] M.Wagner, "On the Error Detecting Capability of CRC Polynomials", *Informationstechnik it*, 28. Jahrgang, Heft 4/1986, p. 236

[Wheal83] M.G. Wheal and M.J.Miller, "The probability of Undetected Error with Error Detection Codes", *IREEECON International Sydney 83, 19th International Electronics Convention and Exhibition*, Digest of Papers, p. xxvii+678, 464-6

[Wolf88] Jack Keil Wolf and Robert D. Blakeney, "An Exact Evaluation of the Probability of Undetected Error for Certain Shortened Binary CRC Codes", *MILCOM 88*, IEEE, 15.2.1-15.2.6.