Preprint of a regular paper to appear in The International Conference on Dependable Systems and Networks (DSN) 2002.

32-Bit Cyclic Redundancy Codes for Internet Applications

Philip Koopman
ECE Department & ICES
Carnegie Mellon University
Pittsburgh, PA, USA
koopman@cmu.edu

Abstract

Standardized 32-bit Cyclic Redundancy Codes provide fewer bits of guaranteed error detection than they could, achieving a Hamming Distance (HD) of only 4 for maximum-length Ethernet messages, whereas HD=6 is possible. Although research has revealed improved codes, exploring the entire design space has previously been computationally intractable, even for special-purpose hardware. Moreover, no CRC polynomial has yet been found that satisfies an emerging need to attain both HD=6 for 12K bit messages and HD=4 for message lengths beyond 64K bits. This paper presents results from the first exhaustive search of the 32-bit CRC design space. Results from previous research are validated and extended to include identifying all polynomials achieving a better HD than the IEEE 802.3 CRC-32 polynomial. A new class of polynomials is identified that provides HD=6 up to nearly 16K bit and HD=4 up to 114K bit message lengths, providing the best achievable design point that maximizes error detection for both legacy and new applications, including potentially iSCSI and application-implemented error checks.

1. Introduction

Cyclic Redundancy Codes (CRCs) are used in a wide variety of computer networks and data storage devices to provide inexpensive and effective error detection capabilities. As data transfer rates and the amount of data stored increase, the need for simple, cheap, and robust error detection codes increases as well. Thus it is important to be sure that the CRCs in use are as effective as possible.

Unfortunately, standardized CRC polynomials such as the CRC-32 polynomial used in the IEEE 802.3 (Ethernet) network standard [IEEE85] are known to be grossly suboptimal for important applications. For example, the 802.3 CRC can detect up to three independent bit errors (Hamming Distance HD=4) in an Ethernet Maximum Transmission Unit (MTU) having a 1500 byte payload. But, the theoretical maximum is detection of five independent bit errors (HD=6) using identical error detection techniques with a better CRC polynomial.

New standards and applications are continually emerging that require a high degree of data integrity. While it is no small matter to refit a widely deployed standard such as Ethernet to a new error detection scheme, designers of emerging technology such as iSCSI (a protocol for Internet-based storage systems [IETF01]) are searching for improved CRC capabilities. However, no CRC polynomials have been previously identified that satisfy both a desire for high error detection performance at Ethernet MTU-length messages as well as good error detection performance for relatively long messages.

The challenge to finding ideal CRCs is that the effectiveness of any particular code is computationally expensive to determine, and finding the best code for any particular message length among all possible codes has in the past proven to be computationally intractable. This is particularly true of message lengths beyond 8K bits, which are commonly found on general-purpose computer networks and data storage devices.

In this paper we present the results of the first exhaustive exploration of the design space for 32-bit CRCs. The entire set of 1,073,774,592 distinct polynomials has been evaluated for effectiveness for data word sizes of 12112 bits.

A result of completing an exhaustive search is that a definitive list of classes of polynomials that can and cannot achieve HD better than the 802.3 CRC for MTU-sized messages has been created. The creation of this list led to the discovery of a previously unexplored class of polynomial that combines excellent performance for MTU-sized messages with good performance for longer messages. This class of polynomial provides a significantly improved alternative to the CRC currently being considered for iSCSI, yielding 5 bit error detection (HD=6) for MTU-size payloads and 3 bit error detection (HD=4) to 114K bits. The class of polynomial previously considered for iSCSI applications (which was only partially explored by previous

work) has now been proven to have no polynomials with HD>4 for MTU-sized messages. Additionally, two new classes of polynomials have been characterized that are comparable in effectiveness to previously known results, but have member polynomials with few feedback taps, potentially simplifying high-speed hardware implementations.

2. Background

Cyclic redundancy codes (also known sometimes as cyclic redundancy checks) have a long history of use for error detection in computing. [Peterson72] and [Lin83] are among the commonly cited standard reference works for CRCs. A treatment more accessible to non-specialists can be found in [Wells99].

A CRC can be thought of as a (non-secure) digest function for a data word that can be used to detect data corruption. Mathematically, a CRC can be described as treating a binary data word as a polynomial over GF(2) (*i.e.*, with each polynomial coefficient being zero or one) and performing polynomial division by a *generator polynomial* G(x). The generator polynomial will be called a CRC polynomial for short. (CRC polynomials are also known as feedback polynomials, in reference to the feedback taps of hardware-based shift register implementations.) The remainder of that division operation provides an error detection value that is sent as a Frame Check Sequence (FCS) within a network message or stored as a data integrity check. Whether implemented in hardware or software, the CRC computation takes the form of a bitwise convolution of a data word against a binary version of the CRC polynomial.

Error detection is performed by comparing an FCS computed on a piece of retrieved or received data against the FCS value originally computed and either sent or stored with the original data. An error is declared to have occurred if the stored FCS and computed FCS values are not equal. However, as with all digital signature schemes, there is a small, but finite, probability that a data corruption that inverts a sufficient number of bits in just the right pattern will occur and lead to an undetectable error. The minimum number of bit inversions required to achieve such undetected errors (*i.e.*, the HD value) is a central issue in the design of CRC polynomials.

The essence of implementing a good CRC-based error detection scheme is picking the right polynomial. The prime factorization of the generator polynomial brings with it certain potential characteristics, and in particular gives a tradeoff between maximum number of possible detected errors *vs.* data word length for which the polynomial is effective. Many polynomials are good for short words but poor at long words, and the converse. There are relatively few polynomials that are excellent for medium-length data words while still being good for relatively long data words.

Unfortunately, prime factorization of a polynomial is not sufficient to determine the achieved HD value for any particular message length. A polynomial with a promising factorization might be vulnerable to some combination of bit errors, even for short message lengths. Thus, factorization characteristics suggest potential capabilities, but specific evaluation is required of any polynomial before it is suitable for use in a CRC function. While many previous results for CRC effectiveness have been published, no previous work has attempted to achieve complete screening of all possible 32-bit polynomials.

3. Previously known 32-bit CRC polynomials

At a general level, the effectiveness of a CRC can be expressed as the minimum Hamming Distance ("HD") of the codewords created by appending computed CRC values to network messages or other data words of interest. If all resulting codewords have an inter-codeword Hamming Distance of at least m bits, then the CRC is guaranteed to detect all possible errors involving (*m*-1) or fewer bit inversions. Typically, a high percentage of bit errors numbering *m* or more are detectible. For CRC polynomials divisible by (x+1), all odd numbers of bit inversions are detected, but all even numbers of bit inversions suffer an undetected error rate approximately twice as high as for other polynomials. Finally, all burst errors of size less than or equal to the number of bits in the CRC are detected (that property is not the primary consideration of this work and remains intact for all the codes we consider).

A critical measurement of CRC effectiveness for general purpose computing is the HD at an Ethernet MTU message size of a 12112 bit data word. Thus the search for CRC polynomials can be concentrated on maximizing achieved HD for MTU-sized data words.

The IEEE 802.3 standard adopts the CRC polynomial:
$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
(this is irreducible, but not primitive). We represent this polynomial as a 32-bit hexadecimal number 0x82608EDB. The leading "8" of this number corresponds to the top four (x^{32} through x^{29}) polynomial coefficients, with lower order bits corresponding to lower order coefficients, down to the trailing "B," which refers to the terms (x^4+x^2+x). The "+1" term is implicit in this representation, permitting representing a polynomial of degree 32 using a 32-bit integer as is common practice in software CRC implementations.

A polynomial's effectiveness is evaluated by computing weights for that polynomial. A *weight* W_i is the number of occurrences of a combination of *i* error bits, including bit errors perturbing the CRC value, that would be undetected by a given polynomial for a given data word length. For ex-

ample, the 802.3 CRC has a weight at message length=12112 bits of $\{W_2=0; W_3=0; W_4=223059; ...\}$. This means this particular polynomial, when used with a 12112 bit data word, will detect all 2-bit errors, and detect all 3-bit errors, but fail to detect the 223,059 four-bit possible errors within $\binom{12144}{4} = \frac{12144!}{4!\ 12140!} = 906 \cdot 10^{12}$ different possible combinations of 4-bit errors that could occur across a 12144-bit codeword (slightly more than 1 out of every 2^{32} possible errors would be undetectable). While this is a small proportion of errors to go undetected, the increasing amount of data being transmitted and stored worldwide suggests that higher detection rates are desirable, or at least that lower detection rates should not be accepted without question as longer data words are being sent and stored.

Weights beyond the first non-zero weight are largely unimportant when evaluating a polynomial for general purpose network applications. That is because, assuming independent and moderate bit error rates (BERs), each successive number of bit errors is less likely to occur by a factor approximately equal to the BER value. So on a network with a 10^{-6} BER, a five-bit error is approximately 10^6 times less likely than a four-bit error. Additionally, because a CRC can always detect 1-bit errors, weights for positions less than 2 are always zero and thus not reported. (While situations with high BERs do occur, in most cases other error detection mechanisms such as message format errors, bit encoding phase violations, and high level protocol handshake failures also take place. Thus, CRC effectiveness at moderate BER values at which only a fraction of messages are corrupted is often the most important networking case from a practical point of view.)

Figure 1 shows a comparison of the HD values of various polynomials identified during the survey discussed in this paper. All HD values are exactly calculated for data word lengths up to 128K bits (131072 bits). Similarly, Table 1 gives the data word bit lengths stating which HD values apply to each polynomial. For example, the 802.3 polynomial has a HD greater than or equal to 8 up to a data word length of 91 bits, HD=7 to 171 bits, HD=6 to 268 bits, HD=5 to 2974 bits, HD=4 to 91607 bits, and HD=3 to at least 128K bits.

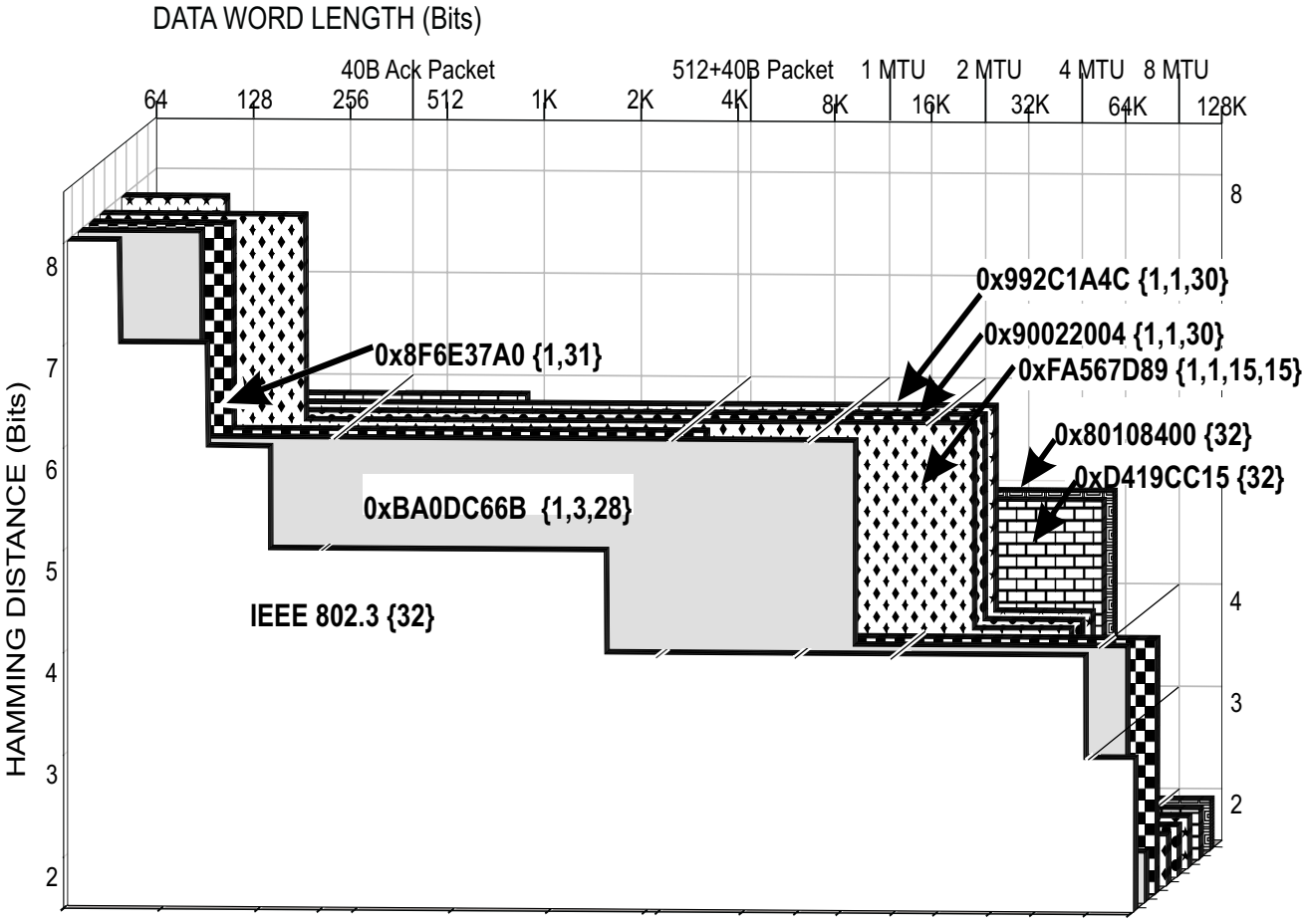

Figure 1. Error detection capabilities of selected 32-bit CRC polynomials.

Table 1. Message lengths in bits (exclusive of CRC field) for which the specified HD is achieved. (Computed to data word length of 131072.)

HD	IEEE 802.3 0x82608EDB {32}	Castagnoli (iSCSI) 0x8F6E37A0 {1,31}	Koopman 0xBA0DC66B {1,3,28}	Castagnoli 0xFA567D89 {1,1,15,15}	Koopman 0x992C1A4C {1,1,30}	Koopman 0x90022004 {1,1,30}	Castagnoli 0xD419CC15 {32}	Koopman 0x80108400 {32}
15	8-10							
14	—	8						
13	—	—						
12	11-12	9-20	8-16	8-11	8-16		8-17	
11	13-21	—	—	—	—		18-21	
10	22-34	21-47	17-18	12-24	17-26		22-27	
9	35-57	—	—	—	—		—	
8	58-91	48-177	19-152	25-274	27-134		28-58	
7	92-171	—	—	—	—		59-81	
6	172-268	178-5243	**153-16360**	**275-32736**	**135-32738**	**8-32738**	82-1060	
5	269-2974	—	—	—	—	—	**1061-65505**	**8-65505**
4	**2975-91607**	**5244-131072...**	16361-114663	32737-65502	32739-65506	32739-65506	—	—
3	91608-131072...	...	—	—	—	—	—	—
2	...	...	114664+	65503+	65507+	65507+	65506+	65506+

Several significant message lengths are marked on Figure 1. The two most frequently encountered message lengths on Internet traffic are 40-byte acknowledgment packets (400 bit data word including 80 bits of protocol overhead) and acknowledgment packets additionally containing 512 bytes of data (4496 bit data word). The Ethernet size for an MTU message is a 12112 bit data word (this forms a 12144 bit codeword including the 32-bit CRC value). Larger potential message sizes of interest are multiples of the 1500-byte MTU payload size.

While the general limit to HD that is attainable for an MTU-size message is 6, actually finding a polynomial that achieves that performance is computationally very expensive. The starting point for determining a HD value is based on the exploitation of the linearity of CRCs. Consider the fact that a data corruption is undetectable if and only if it transforms one codeword (some payload with its valid FCS value) into a different valid codeword. But because CRCs are linear, this means that the faulty bits that have been flipped from the original codeword have to themselves form a valid codeword. (In other words, the bits flipped in the message payload have to be compensated for by bits flipped in the FCS field, and the only way this can happen is if the entire set of bits flipped is itself a valid codeword.) This means that the actual data in a message payload is irrelevant in computing error detection abilities, which simplifies things greatly. Because each undetectable error pattern is itself a codeword, this also means that determining the minimum HD for a polynomial is equivalent to determining the lowest non-zero weight for that polynomial. Furthermore, the weights of a polynomial give the number of undetectable errors for corresponding numbers of error bits.

Thus, there is a relatively simple way to determine the number of k-bit undetected errors for an r-bit CRC polynomial used to provide error detection for an n-bit payload.

All possible combinations of bit patterns with k bits set in an r+n wide bit field can be tested to see if they form a codeword. If no such bit patterns are codewords, then the CRC has W_k=0 and provides perfect detection of k-bit errors. To determine the first k_{max} weights of a polynomial, this enumeration can be iterated by increasing k over the range of $\{2,..,k_{max}\}$. The complexity of each iteration for each polynomial considered is the combinatorial complexity of considering (n+r) thinks k at a time, which is proportional to: $\binom{n+r}{k} = \frac{(n+r)!}{k!\ (n+r-k)!}$. This has a computational complexity of approximately $O((n+r)^k)$ for each value of k for small values of k, with the $k_{max}th$ iteration dominating the computational complexity.

This computation must be repeated for every possible r-bit polynomial. Each polynomial has its top bit set. Additionally, only one of each pair of reciprocal polynomials need be checked [Peterson72] (reciprocal polynomials are readily identified by the fact that their coefficients are bit-reversed from each other). These facts reduce the potential search space from 2^r to 2^{r-2} polynomials, yielding approximately 2^{30} distinct 32-bit CRC polynomials to be evaluated. (There are a few more than 2^{30} polynomials because polynomials that are palindromes are self-reciprocal and thus do not permit elimination of a companion reciprocal polynomial from consideration.) This makes the algorithmic complexity for a complete search $O(2^{r-2}(n+r)^k)$

A pure brute force approach would be dominated by the computation time required to evaluate all combinations of 12144 bits taken 6 at a time ($4.45 \cdot 10^{21}$) for each of approximately 2^{30} candidate polynomials. This requires examination of more than $4.78 \cdot 10^{30}$ bit combination/polynomial pairs. Even at one billion such pairs per second evaluated by each of one million parallel processors, this computation would take 151 million years to complete, and thus is intractable.

Mathematicians have spent many years creating less computationally intensive approaches. The culmination of that work for 32-bit CRCs can be found in [Castagnoli93]. Castagnoli *et al.* evolved Fujiwara's techniques [Fujiwara85] based on constructing dual codes. Additionally, they built special purpose hardware that was able to evaluate the weights of polynomials that had been carefully selected based on prime factorization characteristics. The complexity of this technique (when implemented with special-purpose hardware capable of many concurrent operations) for evaluating a polynomial at a given codeword length is 2^{32} operations. While searching all 2^{30} candidate polynomials was still intractable, searching selected polynomials from promising classes was feasible, with execution time reported to be 107 to 215 seconds per polynomial using a 40 MHz clock. (At this rate, evaluation of all polynomials would have taken in excess of 3600 years on the single copy of special-purpose hardware available, so only partial exploration was performed. Performing a massively parallel distributed computation using otherwise idle computers would be impossible because special-purpose hardware was required to attain this level of computational speed.)

In describing previous work and results we use the following shorthand notation to represent factorization of a polynomial: $\{d_1, .., d_k\}$, where each "d" represents the degree of a factor. Thus "{1,3,28}" represents the set of all polynomials whose irreducible factorization is: "$(x+1)(x^3+..+1)(x^{28}+..+1)$" (*i.e.*, has irreducible factors of degrees 1, 3, and 28).

Using their special-purpose hardware, Castagnoli *et al.* found optimal (lowest-weight at the lowest HD) polynomials for several factorization classes of 32-bit polynomials. Three classes of polynomials of those examined are potentially of interest when improving upon the 802.3 CRC. (It should be noted that Castagnoli's work was not specifically intended to address this particular problem. However, it is the most relevant existing data source available for that purpose.) Those three classes are {1,1,15,15} polynomials, {32} polynomials, and {1,31} polynomials.

First, [Castagnoli93] reports that the optimal {1,1,15,15} polynomial is 0xFA567D89=($0x1 \cdot 0x1 \cdot 0x4008 \cdot 0x642f$), which gives HD=6 up to almost 32K bits. (No polynomial gives HD=6 at exactly 32K bit data word length.) This polynomial would be suitable for MTU-sized data words, but does not work well above 32K bit. (The published polynomial in Table XI of [Castagnoli93] has an error; it is incorrectly given as 1F6ACFB13, but should have been 1F4ACFB13, a one-bit difference. The factorization given in that table yields the correct polynomial that matches one of the polynomials found in our results. Thus it can be assumed this is merely a minor data transcription error. The incorrectly published polynomial has HD=6 up to a length of only 382 bits and so should not be used.)

Second, [Castagnoli93] reports that a {32} polynomial, 0xD419CC15 (irreducible, although not primitive), gives HD=5 up to almost 64K bits. (No polynomial gives HD=5 at exactly 64K bit data word length.) This polynomial would improve upon the 802.3 CRC by one bit of HD, and extend coverage at HD=5 out to almost 64Kb, making it an attractive alternative for messages longer than an MTU. However, it drops to HD=2 above 65505 bits. This polynomial was selected from a restricted class of irreducible polynomials, but nonetheless achieves the best possible HD values at 32Kb to 64Kb.

Third, [Castagnoli93] reports that the best evaluated polynomial of the form {1,31}, where the larger factor is primitive, is 0x8F6E37A0=($0x1 \cdot 0x7ADA129F$). This

code has the promising property of keeping HD=4 out to very long data words, but only has HD=6 up to less than half an Ethernet MTU. However, the authors state that the search was limited by available compute time on the special-purpose hardware, and that there was only time to investigate 47,000 such codes out of $6.93 \cdot 10^7$ possibilities (there are more {1,31} polynomials than that, but the authors of that study only considered 31-bit polynomials that were primitive).

The fact that the exploration of {1,31} polynomials was incomplete leaves open the intriguing possibility that there might be other, previously unknown, polynomials that achieve HD=6 for an Ethernet MTU without sacrificing achieving HD=4 at data words sizes in excess of 64Kb. Additionally, there might be other forms of polynomials not explored that provide other similarly useful message length *vs.* error detection capability tradeoff points.

4. An exhaustive search for 32-bit CRCs

While it is easy to argue that retrofitting an existing standard such as Ethernet is impractical, there always seem to be new standards being created that might well adopt a superior CRC polynomial. For example, the team creating the draft iSCSI standard is in the process of designing a protocol that will involve messages of MTU size or larger and that will use 32-bit CRCs to assure data integrity of transmitted messages [Satran01].

A study of CRC effectiveness was completed by Sheinwald *et al.*[Sheinwald00] as part of the iSCSI definition effort. That report recommends adoption of Castagnoli's {1,31} polynomial 0x8F6E37A0 to achieve good performance on short data words while not sacrificing HD=4 performance on longer data words. A good way to improve upon this selection would be to find a polynomial with HD=5 (or even HD=6) at 12112 bits that still has HD=4 to somewhat beyond 64Kb.

4.1. The search technique

Rather than embellish upon existing mathematically-based approaches, we opted for a brute-force enumerative approach that could be implemented with high efficiency on standard computing platforms. Beyond allowing us to build upon mature software that had already proven itself in use for embedded network error detection evaluation, using software on conventional computing platforms permitted achieving the following goals:

- Examine all possible polynomials without being limited to those which have certain theoretical properties.
- Reproduce previous results via an independent methodology both to validate our approach and demonstrate reproducibility for previous work. Indeed, our work discovered an data reporting error in [Castagnoli93], which might have caused a problem if someone had simply used the published polynomial given without investigation of its properties. Additionally, this validation step provided an independent check against any transient errors that might possibly have affected those earlier computations on special-purpose hardware.
- Attain scalability via using idle computing cycles and riding the technology curve of standard platforms, both of which are difficult to achieve with the use of custom hardware that is required to attain high speed operation with other approaches.

The software used for the search was very carefully optimized and tuned C++ code running on a Digital Unix Alphastation platform (Sparcstation and Windows 2000/PC platforms were also used with identical source code). The software examines all possible combinations of k bit errors across an n-bit data word plus r-bit FCS field, with r=32. While previously argued herein (and by previous publications) that this approach was intractable, success was achieved by spending extreme care building the code for speed and using the following algorithmic complexity-reduction techniques in combination:

- **Filtering out polynomials** rather than computing exact weights. Since it was desirable to improve upon the HD=4 802.3 CRC performance, all polynomials were first evaluated for non-zero weights for 2-, 3-, and 4-bit errors. If any of these weights were non-zero, there was no need to compute weights for 5 and 6 bit errors.
- **Early bailout of weight evaluation.** Furthermore, there is no need to compute exact weights for 2, 3, and 4 bits for most polynomials. This is because the first non-zero contribution to a weight dooms a polynomial to failure, so there is no point in continuing the weight computation. Thus, only one undetected pattern of errors need be found for any polynomial at a given length, with a result of terminating evaluation of a polynomial and filtering it out of consideration. Because only a tiny fraction of polynomials has HD>4 at 12112 bit data word length, this permitted short-circuiting the computation of almost all polynomials quite quickly compared to complete computation.
- **Exploiting common behavior** of error detection failures. After the first few thousand polynomials were checked, it was determined that the majority of polynomials had at least one undetected error that involved bits in the FCS field. Therefore errors with one or two FCS bits inverted were tried first, speeding the average time to encountering an undetected error. (It should be emphasized that all reported polynomials with HD>4 were the result of exact weight computations.

This approach merely maximizes the filtering speed by looking for likely undetected error cases first.)

- **Filtering with increasing lengths.** Because the cost of filtering each candidate is in the worst case $O((n+r)^4)$ to filter for 4-bit error vulnerabilities, polynomials can be first filtered at a shorter length n. For example, evaluating polynomials for HD>4 at length 1024 is almost 17,500 times faster than at length 12112 bits, and successfully filters the overwhelming majority of polynomials evaluated. Because the HD of a polynomial can only stay equal or be reduced with increasing data word length n, any polynomial filtered at a short length can be removed from consideration before filtering the remaining polynomials for that same HD at longer lengths.
- **Inverse filtering with decreasing lengths.** Once candidate polynomials are identified via filtering, inverse filtering can be applied to determine a maximum length at which any of a set of polynomials achieves a particular HD value. Iterative evaluations decrease lengths to establish successively shorter upper length bounds. Runs at long lengths reject all polynomials quickly using the early-out filtering approach, providing a firm upper length bound by proving no polynomials examined achieve the desired HD value. Reducing that bound until run time increases dramatically gives a good detection tool to find the maximum length for which the HD being filtered for can be achieved. (An example is provided below.)

Thus, significant speedup was achieved by using a variety of filtering techniques to avoid computing exact weights, and instead using an early-out approach to detect non-zero weights without completing full weight calculations. Nonetheless, filtering preserves the property that all results reported at the end of the process are exact. While this discussion is in the context of MTU-sized messages, the techniques are generally applicable to any CRC selection process.

An example of the somewhat subtle tradeoffs involved in using these filtering techniques in combination is the creation of the data shown in Figure 1 and Table 1. Consider the task of determining precisely where the 802.3 polynomial from Figure 1 transitions from HD=5 to HD=4. Given knowledge that this transition cannot happen at a payload size greater than 64K bits, a reasonable baseline method to do this might be to perform a classical binary subdivision search for the transition over a span up to 64K bits, looking for successively smaller intervals in which the lower of two message lengths has HD=5 and the upper of two message lengths tested has HD=4.

A straightforward approach would be to compute the first 5 weights of the polynomial for each length considered in the search and then evaluate them. If we assume that 64K bit payloads have HD=4, then the first point to evaluate in a binary subdivision search might be 32K bits. Evaluating the first 5 weights at 32K bits takes a long time (we estimate it would take more than 5 months, which is far longer than we were willing to wait for a test run to complete).

A speedup can be obtained by realizing that computing the value of the fifth weight is unnecessary. In fact, all that need be done is compute the first four weights (this is a use of the *filtering* technique previously described). If all four weights are zero for that length, then it is certain that HD≥5. If any of the first four weights are non-zero, then HD≤4. Thus the break point from HD=4 to HD=5 can be found simply by looking for the shortest length at which the fourth weight becomes non-zero. Computing only the first four weights at 32K bits takes approximately 7 minutes – a substantial speedup.

A further improvement can be made by implementing *early bailout*. With early bailout, the evaluation software is modified to check the computation of weights periodically to detect any of weights 2 through 4 being non-zero, with the computation bailing out as soon as that happens. The result is not a precise weight value, but rather a logical flag that is true if any of weights 4 or less is non-zero, and false if all of them are zero. Because this is actually the only information needed to perform the search for the break-point (namely, deciding whether HD is above 4 or not), this result suffices. The execution time now depends on the particular polynomial and order of evaluation. Adding the *exploitation of common behavior* optimization as well to provoke a bail-out as early as possible results in an execution time of less than 7 seconds at 32K bits, compared to 7 minutes with just filtering. Note that this optimization is probabilistic, and depends on the particular polynomial being tested as well as message lengths. But it works very well in practice, especially when there are a large number of undetectable errors that are spread throughout the evaluation space for any particular polynomial.

Given that evaluations of longer payloads can still take a significant amount of time in the context of examining a billion polynomials, a further improvement is to evaluate the polynomial at HD=4 for 256 bits, 512 bits, 1K bits, 2K bits, and so on until the HD=5 to HD=4 break point is straddled (*filtering with increasing lengths*). Exact evaluation at 4K bits for HD=4 takes less than 6 seconds, and evaluation at 2K bits takes less than 1.5 seconds, straddling the break point. This means that a search exploiting increasing lengths followed by a binary search to narrow results within the first interval spanning the break point succeeds in less than a minute of total CPU time. This approach also takes less time than a full binary subdivision search even though it generally requires a few more evaluations, because the

evaluations performed are concentrated on smaller sized payloads and thus run quickly.

The final optimization of *inverse filtering* relies upon a further exploitation of the early-out evaluation speedup to provide prediction of results via monitoring of execution time as well as fast computation of upper length bounds. Consider evaluating the 802.3 polynomial at the break point being discovered, i.e. at lengths of 2974 bits and 2975 bits with early-out evaluation. Evaluation of the first four weights at 2974 bits takes 2.7 seconds to determine that all four are zero. However, evaluation at 2975 bits takes only 1.9 seconds to determine that there is *at least* one undetected 4-bit error at that length. Full evaluation to find out there is in fact exactly one such undetected error takes the full 2.7 seconds at 2975 bits, but the early-out approach does not have to complete the full computation since it happens to find the undetected error 1.9 seconds into the computation. This illustrates that early-out location of an undetected error at a longer length can be faster than discovering that all errors are detected at a shorter length.

Thus, a search that biases its selections to increase the probability that it will look above a break point rather than below it will tend to run faster. In this particular case the speed differential is small enough that the results of employing this technique are not clear-cut. But at longer message lengths the results are significant. An application of particular importance is attempting to find the highest length at which no possible polynomial provides a particular HD as opposed to a search that evaluates many polynomials offering a particular HD.

As a further example of an opportunity for inverse filtering, computing that HD<6 for 0xBA0DC66B at 16361 bits takes 7.4 seconds via finding at least one non-zero weight among the first five weights. But confirming that at 16360 bits has HD=6 would take approximately 19 days. Thus when finding this breakpoint, the binary subdivision search strategy was modified to abort an evaluation after 30 seconds and to consider long execution time to be an implicit confirmation of HD=6 for any particular length, homing in on 16361 as the shortest length with HD<6. Then a single calculation at a length of 16360 can be permitted to run to completion in order to confirm the result.

Of course many combinations of these filtering techniques are possible. An important one for this work was first obtaining a list of HD=5 and HD=6 polynomials at Ethernet MTU data word lengths (as a filtering step based on increased lengths). Then this small list of polynomials was inverse filtered with lengths working downward from 128K bits to find the maximum data word lengths for HD=5 and HD=6 without having to actually compute complete HD values.

4.2. Experimental results

In the end, even with all the filtering and computational techniques that could be brought to bear, the enumeration of all billion possible 32-bit polynomials was a formidable task. The initial filtering lasted from late May to early September 2001 and made use of otherwise idle workstations. Approximately 50 Alphastations (an even mix of 400 MHz and 500 MHz processors) were kept running continuously for over three months, and 30 UltraSparc machines were used intermittently for two months. When the computations had been completed, all polynomials with HD>4 at a 12112 bit data word length had been discovered via filtering out polynomials failing to have zero 2-, 3-, and 4-bit weights.

The average computation rate was approximately two polynomials filtered per second per CPU. This filtering technique implemented on a general purpose workstation was an order of magnitude more efficient than exact evaluation using special purpose hardware as reported in [Castagnoli93]. Specifically, filtering was more than 200 times faster in absolute terms, but took advantage of newer technology with a 10-time faster clock speed for an overall speedup of more than 20 on a clock-for-clock basis. The time required to filter any particular polynomial was variable because it depended on how long it took to encounter the first non-zero weight, but the vast majority of polynomials benefitted from examining errors involving one or two bits in the FCS field first.

Further filtering at HD=5 of those polynomials left 21,292 polynomials with HD=6 at 12112 bit message lengths. Evaluating the precise weight of each HD=6 poly-

Table 2. Number of polynomials having HD=6 at MTU length for different irreducible factorizations.

# Factors	Size of Factors	# Distinct Polynomials
3	{1,1,30}	658
3	{1,3,28}	448
4	{1,1,15,15}	9887
4	{1,1,2,28}	895
4	{1,3,14,14}	4154
5	{1,1,1,1,28}	448
5	{1,1,2,14,14}	2639
6	{1,1,1,1,14,14}	2263

nomial is still impractical, but is expected to become practical in a few years with faster workstations.

Of the HD=5 polynomials, the {1,1,15,15} class investigated by Castagnoli *et al.* was verified to indeed provide its claimed properties. Further filtering analysis indicated that the class {1,1,30} had similar properties. A potentially useful polynomial reported in Table 1 is 0x90022004=(0x1·0x1·0x2FFF5FFE), which is the polynomial with the fewest non-zero coefficients that attains HD=6 up to almost 32Kb. (Having only five non-zero coefficients may help in creating high-speed combinational logic implementation of CRCs by reducing logic synthesis minterms.) 0x992C1A4C=(0x1·0x1· 0x2D095216) was also selected for characterization as a representative {1,1,30} polynomial and has error detection performance comparable to Castagnoli's {1,1,15,15} polynomial.

As it turns out, all polynomials with HD=6 were divisible by (x+1) as shown in Table 2. This gives them the property of incorporating an implicit parity bit, enabling them to detect all odd number of bit errors.

No polynomials were found that best Castagnoli's {32} primitive polynomial at or above 12112 in terms of HD. Due to computational resource limitations, filtering was not attempted on the very large number of primitive 32-bit polynomials to see if there were one with HD>4 at a length beyond 1060 bits. However, it is certain that none has HD>4 at 12112 bits because all {32} polynomials found at that length are irreducible but non-primitive. The {32} polynomial 0x80108400 was identified as a polynomial with the minimum possible number of non-zero coefficients that achieved HD=5 up to nearly 64Kb.

Inverse filtering was used to ensure that there were no possible polynomials of any class with HD=6 at or above 32739 bits and no polynomials with HD=5 at or above 65507 bits of data word length. The newly found polynomials reported in Table 1 extend one or two bits of data word length past the Castagnoli polynomials at HD=5 or HD=6, although this is only a negligible improvement for most applications.

4.3. A better iSCSI candidate polynomial

An example of an application for a new 32-bit CRC polynomial is iSCSI. iSCSI is a work in progress, but the point of discussing it is to demonstrate that new polynomials can and are being sought after for new standards as well as demonstrate a concrete opportunity for improvement over existing recommended 32-bit polynomials.

[Sheinwald00] concludes that [Castagnoli93]'s {1,31} polynomial 0x8F6E37A0 presents a good tradeoff between being no worse than the 802.3 CRC for MTU-size messages, and maintaining HD=4 up to large data word sizes. And in fact, this polynomial is in the draft versions of iSCSI documents (*e.g.*, [Satran01]).

Given that single Ethernet-sized packets are likely to be transported on an iSCSI network in addition to packed multi-MTU data storage packets protected by a single CRC, it would seem there is an advantage to having HD=6 error detection coverage for MTU-sized data words in addition to maintaining the HD=4 detection achieved by the iSCSI polynomial for long messages. Improved error detection performance can be achieved by using the {1,3,28} polynomial 0xBA0DC66B=(0x1·0x6·0x82CA9A0) described in Table 1. This polynomial achieves HD=6 up to almost 16Kb and HD=4 up to 114,663 bits, which is more than 9 times an Ethernet MTU data word size and sufficiently large for iSCSI purposes. Thus, the use of this newly evaluated polynomial class offers an opportunity for improved error detection for an emerging standard.

4.4. Other potential applications

Stone *et al.* [Stone00] discovered that corrupted network packets are far more prevalent than might be anticipated from bit error rates alone, with CRCs being relied upon to detect corrupted data once every few thousand packets. As a solution they strongly urge use of an application-level error checking code to supplement network error checking. The polynomials described in this paper offer a variety of length vs. error detection performance tradeoffs for such application usage.

Another potential application for a 32-bit CRC polynomial that has both HD=6 for Ethernet MTU length messages and HD=4 to longer lengths is for jumbo packets in Gigabit Ethernet. Currently available Gigabit Ethernet cards seem to support a de facto standard of 9000-byte jumbo packet payload sizes (data word size of 72112 bits), and such an approach is entering consideration for standardization. These jumbo packets use the existing IEEE 802.3 polynomial. It might possibly be argued that since new interface cards have to be designed to operate at high bit rates, these new cards could have both the legacy polynomial for slower speed backward compatible messages plus a new polynomial for high-speed messages. Unfortunately there is probably already enough hardware already built for Gigabit Ethernet that doing so is unrealistic. But the opportunity might well remain for the next generation of Ethernet cards beyond 1 Gigabit per second speeds.

4.5. Validation

Software validation was accomplished by a combination of reproducing known results for exhaustive searches of 8- and 16-bit polynomials, creating unit and system test programs, comparing answers obtained with "simple" code to

optimized code, and comparing results to existing publications for 32-bit polynomials.

Two key invariants unrelated to the software implementation were monitored. Polynomials divisible by (x+1) were checked to ensure that all odd-numbered weights computed were in fact zero, even though the software did not exploit this fact when performing evaluations. Additionally, weight values were ensured to be non-decreasing when computed over increasing payload lengths. (This weight check revealed a 32-bit counter overflow problem in an early version of the code. That problem would not have affected the results presented herein even if it had not been fixed; but finding it provided some reassurance that results were being monitored quite closely.)

5. Conclusions

An exhaustive search of all possible 32-bit CRC polynomials has revealed the existence of a class of polynomials that provides an excellent combination of error detection performance for long and short network messages. A representative of that polynomial class is: 0xBA0DC66B ($x^{32}+x^{30}+x^{29}+x^{28}+x^{26}+x^{20}+x^{19}+x^{17}+x^{16}+x^{15}+x^{11}+x^{10}+x^{7}+x^{6}+x^{4}+x^{2}+x+1$) = ($x+1$)($x^{3}+x^{2}+1$)($x^{28}+x^{22}+x^{20}+x^{19}+x^{16}+x^{14}+x^{12}+x^{9}+x^{8}+x^{6}+1$). This polynomial achieves HD=6 beyond one Ethernet MTU (to a 16,360 bit data word length) and HD=4 to 114,663 bits, which is more than 9 times the length of an Ethernet MTU. This gives two additional bits of error detection ability at MTU-sized data words compared to the Ethernet CRC standard polynomial while not sacrificing HD=4 capability for data word sizes up to and beyond 72K bits.

Beyond the discovery of new polynomials, this work reproduces results using direct evaluation of error detection capability that were previously only obtainable using mathematically based techniques. The results contained herein have been found to be consistent with a variety of previously reported results, including results previously created via special-purpose hardware. The availability of a more efficient search capability on standard hardware platforms opens up the possibility of identifying optimal polynomials that are customized to the particular message lengths of specific applications and special-purpose communication networks.

Finally, because complete coverage of all possible polynomials was obtained, certain classes of polynomials have been conclusively ruled out as viable for providing HD=6 coverage for MTU-size data words. This includes all 32-bit primitive polynomials and all polynomials that are not divisible by (x+1). One unexpected finding was a publication error in the only previously published polynomial that achieved HD=6 for MTU-size data words.

Application of these results is possible in newly emerging network and data storage standards such as Internet SCSI protocols as well as application-level CRCs that provide increased data integrity checks.

6. Acknowledgments

Significant use was made of equipment donated by Digital Equipment Corporation (now Compaq). Additional equipment support was provided by Intel.

7. References

[Castagnoli93] Castagnoli, G., Braeuer, S. & Herrman, M., "Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits", *IEEE Trans. on Communications*, Vol. 41, No. 6, June 1993.

[Fujiwara85] Fujiwara, T., Kasami, T., Kitai, A. & Lin, S., "On the undetected error probability for shortened hamming codes", *IEEE Trans. on Communications*, vol. 33, no. 6, 1985, pp. 570-573.

[IEEE85] *IEEE standards for local area networks: carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*, ANSI/IEEE Std 802.3-1985.

[IETF01] Internet Engineering Task Force, "IP Storage (ips) Charter," *http://www.ietf.org/html.charters/ips-charter.htm*, accessed Nov. 10, 2001.

[Lin83] Lin, Shu & d. Costello, *Error Control Coding*, Prentice-Hall, 1983.

[Peterson72] Peterson, W. & E. Weldon, *Error-Correcting Codes*, MIT Press, Second Edition, 1972.

[Satran01] Satran, J. *et al.*, "iSCSI", Internet-Draft work in progress, *http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-08.txt*, Sept. 30 2001, accessed Nov. 10, 2001.

[Sheinwald00] Sheinwald, D., *et al.*, "iSCSI CRC/Checksum Considerations", Internet-Draft work in progress, *http://search.ietf.org/internet-drafts/draft-sheinwald-iscsi-crc-00.txt*, May 7, 2001, accessed Nov. 10, 2001.

[Stone00] Stone, J. & Partridge, C., "When the CRC and TCP checksum disagree", *ACM SIGCOMM Computer Communication Review: Proc. of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Aug. 2000, pp. 309-319.

[Wells99] Wells, R., *Applied coding and information theory for engineers*, Prentice-Hall, 1999.

ERRATA

The following corrections have been made to this paper:

8/3/2014: Page 4, table 1
CRC 0x992C1A4C provides HD=6 up to length 32738 bits and HD=4 starting at 32739 bits. The original paper indicated HD=6 only up to 32737 bits. Thanks to Berthold Gick for reporting this issue.