An Analysis of Potential CRC Polynomials Using Simulated Bit Errors in Transmitted Packets

Mark J. Boyd
Department of Computer Engineering
University of California, Santa Cruz
Santa Cruz, California 95064 USA

Abstract

The ability to detect errors in a transmitted packet is currently available in the data-link layer by use of a Cyclic Redundancy Check (CRC).

We analyze the selection of CRC polynomials in the context of error detection when transmitting packets of bits with rare, memoryless errors. We simulate errors in transmitted packets and show why standard CRC polynomials should not be used.

Based on a realistic simulation, we show that equivalent-size primitive polynomials provide over one trillion times better protection against undetected errors than the current international standards.

1 Introduction

As the speed and volume of packet transmissions continues to grow, the ability to reliably detect errors becomes increasingly important. The standard procedure for error detection for decades has been Cyclic Redundancy Check (CRC).

As part of the network data link layer, CRC-32 is implemented in most protocol standards (Ethernet-ANSI/IEEE Standard 802.3, Token Ring - ANSI/IEEE Standard 802.4, 100VG-AnyLAN - Proposed 100Mbit ANSI/IEEE Standard 802.12, 100 BaseT - Proposed 100Mbit ANSI/IEEE Standard 802.3, MAN - Proposed ANSI/IEEE Standard 802.6, ADCCP - ANSI X3.66 -1979 / FED-STD-1003A, Fibre Channel - ANSI X3T11 Standard).

CRC is a *polynomial code*, meaning that bit strings are treated as representations of polynomials with coefficients of 0 and 1 only. Any k-bit string is considered a polynomial with k terms, ranging from x^{k-1} to x^0.

CRC is implemented by dividing some m-bit frame by some universally known standard g-bit *generator polynomial.* The remainder of this polnomial division is the **checksum** of the frame.

Since this **checksum** is fairly distinct (depending upon the choice of the *generator polynomial*), we can calculate this **checksum** before transmitting the frame, and then send the value with the frame. The receiver then performs the same calculation on the received frame. If the calculated **checksum** matches the received **checksum**, it is very likely the received frame has no errors.

Since a receiver assumes a matching **checksum** indicates a correctly transmitted frame, it is critically important that the chance of an undetected error is very small. One technique for achieving this is to increase the size of the *generator polynomial.* Much like a hash function, the larger the polynomial, the greater the number of unique buckets into which we can put packets. CRC-32 provides 33 bits, approximately 4 billion unique **checksums** with which to validate transmitted frames.

If the difference between a transmitted frame T and its erroneous received counterpart frame E were entirely random, CRC-32 would allow an erroneous packet to be accepted as valid with a probability $P(E_{accept})$ which is less than 2^{-32}.

This estimate has proven to be quite naive. Recent inspection of real data has shown that under some circumstances, undetected errors are much more common than previously thought. Two assumptions cause the estimate to be inaccurate. The first is that **checksummed** frames do not contain random bits, the second is due to the inaccuracy of assuming an even distribution of hamming distances of frames which hash into the same **checksum** bucket.

The first assumption has been adequately examined elsewhere. Here we instead focus on the hamming distances of frames which generate the same **checksum**, and show how the selection of a correct *generator polynomial* improves the error detecting capability of CRC by several orders of magnitude over the current international standards.

2 Background

Several international standard CRC *generator polynomials* are widely in use. They vary in degree from 12 to 32 (and are represented by bitstrings from 13 to 33 bits long).

CRC-32's generating polynomial is $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x^{1}+1$ which is equivalent to 4,374,732,215.

CRC-16's generating polynomial is $x^{16}+x^{15}+x^{2}+1$ which is equivalent to 98309.

CRC-CCITT's generating polynomial is $x^{16}+x^{12}+x^{5}+1$ which is equivalent to 69665.

CRC-12's generating polynomial is $x^{12}+x^{11}+x^{3}+x^{2}+x^{1}+1$ which is equivalent to 6159.

Interestingly, none of these polynomials is primitive. Furthermore, only CRC-32 has an approximately even ratio of zeros to ones in its representation.

3 Model

In order to generate an adequate simulation, our model assumed rare, random bit errors could occur during the transmission of a frame. To simplify the simulation, the distribution of errors was entirely random. We also assumed the chance of a frame having *any* erroneous bits was less than 50 percent.

The assumption of rare errors is key to our analysis of the undetected error rate, $P(E_{accept})$. Obviously, if the chance of a bit flip is .5, the problem is identical to using the **checksum** as a semi-distinct identifier of distinct frames. Although a common and efficient use of **checksums** is to identify if a file or compressed image matches the one which we wish to access, our analysis focuses on error detection, and transmissions with very high error rates are rarely found (either the protocol negotiates a new transmission rate or the connection is terminated).

Furthermore, our model seeks simply to compare the $P(E_{accept})$ ratio between a g-bit *generator polynomial G(x)* and an equivalent-degree *G'(x)* alternate candidate. Although this ratio is affected by the bit error rate e, the simulation portion of the experiment requires no information about e beyond the assumptions given above. In fact, e is only used in the last step of generating the analytical results.

4 Simulation

The interactive simulator for this project is available at *http://www.cse.ucsc.edu/ mjboyd/j1/CRC.html*

A simple CheckSum java applet can be found at *http://www.cse.ucsc.edu/ mjboyd/j2/CheckSum.html*

The simulator generates two frames which have the same **checksums** and then calculates the hamming distance between them. This distance indexes a histogram array cell, which is incremented.

Hamm Dist	CRC 16	CRC 16P	CRC 12	CRC 12P
4	7	0	1	0
5	0	0	0	0
6	91	11	12	9
7	0	24	0	30
8	668	122	283	134
9	0	489	0	497
10	4625	1374	3034	1517
11	0	4087	0	4236
12	21789	9911	20242	10090
13	0	21366	0	21862

Table 1: Results of 2 million Simulated Erroneous Packets. Frame size is 40. Numbers indicate occurrences of distances for two random packets with identical checksums.

	CRC 16	CRC P16	CRC 12	CRC P12
$P(E_{accept})$	10^{-23}	10^{-35}	10^{-24}	10^{-35}

Table 2: Probability of an undetected error in 2 million Simulated Erroneous Packets. Frame size is 40. Numbers are approximated within one order of magnitude.

5 Results

Simulations were run for four *generator polynomials.* Two were international standards, the other two were primitive (CRC-12P is 6199, CRC-16P is 98299).

For both of the international standards, there were occurrences of frames with the same **checksum** and hamming distances of 4. Given an error rate e of 10^{-6}, the primitive polynomials were over one trillion times less likely to allow an undetected error. These results reflected an average simulation run, and did not vary significantly with changes in the frame size.

6 Conclusion

Using a primitive polynomial as the *generator polynomial* for detecting errors in transmitted frames provides better error detection than the current non-primitive international CRC standards. If the bit error rate is low (on the order of 10^{-6}) these primitive polynomials provide a trillion times better error detection than their international standard counterparts.

7 Annotated Bibliography

Computer Networks, 3rd edition, by Andrew S. Tanenbaum, 1996, pp. 182-190. Demonstrates the calculation procedure and gives polynomials for CRC-12, CRC-16, and CRC-CCITT.

TCP/IP Illustrated, Volume 1: The Protocols, by W. Richard Stevens, 1994.

The Art of Computer Programming: Volume 2, by Don Knuth, 1998.

Numerical Recipes in C: The Art of Scientific Computing, by W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, 1988.

"Performance of Checksums and CRCs over Real Data," by C. Partridge, J. Hughes and J. Stone in Proceedings of the SIGCOMM Conference, ACM, 1995, pp. 68-76.

"Procedure for Computing CRC-32 Values," Microsoft Systems Journal, March 1995, pp. 107-108.

"Byte - wise CRC Calculations" by Aram Perez in IEEE Micro, June 1983, pp. 40-50. Shows how to create a lookup table which is the best way to implement CRC in software (versus the shifts that are done when implemented in hardware).

"A Tutorial on CRC Computations" by Tenkasi V. Ramabadran and Sunil S. Gaitonde in IEEE Micro, August 1988, pp. 62-75.

"Cyclic Redundancy Checks for Data Integrity or Identity" by William H. Press and Saul A. Teukolsky, Computers in Physics, Jul/Aug 1989, pp. 88-91.