Analysis Of The Effectiveness Of Error Detection In Data Transmission Using Polynomial Code Method

Daniel N. Owunwanne, Howard University, USA

ABSTRACT

Data transmitted from one location to the other has to be transferred reliably. Usually, error control coding algorithm provides the means to protect data from errors. Unfortunately, in many cases the physical link can not guarantee that all bits will be transferred without errors. It is then the responsibility of the error control algorithm to detect those errors and in some cases correct them so that upper layers will receive error free data. The polynomial code, also known as Cyclic Redundancy Code (CRC) is a very powerful and easily implemented technique to obtain data reliability. As data transfer rates and the amount of data stored increase, the need for simple and robust error detection codes should increase 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 Ethernet network standard are known to be grossly suboptimal for important applications, (Koopman, 2002). This research investigates the effectiveness of error detection methods in data transmission used several years ago when we had to do with small amount of data transfer and data storages compared with the huge amount of data we deal with nowadays. A demonstration of erroneous bits in data frames that may not be detected by the CRC method will be shown. A corrective method to detect errors when dealing with humongous data transmission will also be given.

Keywords: Data Transmission, Error Detection, Polynomial Code, Cyclic Redundancy Code, CRC Method, Checksum

1. INTRODUCTION

The Cyclic Redundancy Code, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission. In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probability if an error occurred or not in the transmission. If an error occurred, the receiver sends a "negative acknowledgement" (NAK) back to the sender, requesting that the message be retransmitted.

The technique is also sometimes applied to data storage devices, such as in a disk drive. In this situation each block on the disk would have check bits, and the hardware might automatically initiate a reread of the block when an error is detected, or it might report the error to software. The material that follows speaks in terms of a "sender" and a "receiver" of a "message," but it should be understood that it applies to storage writing and reading as well.

The aim of an error detection technique is to enable the receiver of a message transmitted through a noisy (error-introducing) channel to determine whether the message has been corrupted. To do this, the transmitter constructs a checksum which is a function of the message, and appends it to the message. The receiver can then use the same function to calculate the checksum of the received message and compare it with the appended checksum to see if the message was correctly received. For example, if we chose a checksum function which was simply the sum

of the bytes (numbers) in the message modulo 256 i.e., 2^8 (8-bit register), then it might go as follows (all numbers are in decimal):

Message: 6 23 4
Message with checksum: 6 23 4 33
Message after transmission: 6 27 4 33

In the above, the second byte of the transmitted message was corrupted from 23 to 27 by the communications channel. However, the receiver can detect this by comparing the transmitted checksum (33) with the computer checksum of 37 (6 + 27 + 4). If the checksum itself is corrupted, a correctly transmitted message might be incorrectly identified as a corrupted one. However, this is a safe-side failure. A dangerous-side failure occurs where the message and/or checksum is corrupted in a manner that results in a transmission that is internally consistent. Unfortunately, this possibility is completely unavoidable and the best that can be done is to minimize its probability by increasing the amount of information in the checksum, that is, by widening the checksum from one byte to two bytes, (Williams, 1993).

There are several techniques for generating check bits that can be added to a message. Perhaps the simplest is to append a single bit, called the "parity bit," which makes the total number of 1 bits in the *code vector* (message with parity bit appended) even or odd. If a single bit gets altered in transmission, this will change the parity from even to odd (or vice versa). The sender generates the parity bit by simply summing the message bits in modulo 2 arithmetic, that is, by using *XOR method*. It then appends the parity bit (or its complement) to the message. The receiver can check the message by summing all the message bits in modulo 2 and checking that the sum agrees with the parity bit. Equivalently, the receiver can sum all the bits (message and parity) and check that the result is 0 (if even parity is being used). This simple parity technique is often said to detect 1 bit error. Actually it detects errors in any odd number of bits, including the parity bit.

The CRC algorithms were originally developed for detection of line transmission errors. They were designed to be fast and easy to implement in hardware. The selection of generator polynomial is the most important part of implementing the CRC algorithm. The polynomial is chosen to maximize the error detecting capabilities (minimizing collision probability). The most important attribute of the polynomial is its length (the number of the highest nonzero coefficient), because of its direct influence of the length of the computed checksum.

There are also communication standards approved by IEEE or ITU organizations, which define CRC polynomials used in various communication protocols. When creating a new polynomial, the general advice is to use an irreducible polynomial, which means that the polynomial cannot be divided by any polynomial, except itself with zero reminder, (Petr Hlávka et al, 2005).

2. BACKGROUND

Cyclic Redundancy Codes (also known as Cyclic Redundancy Checks) have a long history of use for error detection in computing. W. Peterson (Peterson72) and Shu Lin (Lin83) as well as Andrew Tanenbaum (Tanenbaum2003) are among the commonly cited standard reference works in CRCs. Actually, CRCs can be used to detect medium induced errors in messages transferred over communication channels. They are also commonly used to protect the integrity of data stored in memory. In practice CRCs have been found to be an adequate protection mechanism for deployment in everyday communication systems for medium induced errors. But, this experience is insufficient to assure ultra-dependable operation (Paulitsch et al, 2005).

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), that is, with each polynomial coefficient being zero or one and performing polynomial division by a *generator polynomial* G(x). CRC polynomials are also known as feedback polynomials which is 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, (Koopman, 2002). Error detection is performed by comparing a 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 - the Hamming Distance (HD) value is a central issue in the design of CRC polynomials. HD is discussed in the next section.

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 *versus* data word length for which the polynomial is effective. Many polynomials are good for short words but poor at long words. 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. 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. HAMMING DISTANCE

The Hamming Distance (HD) between two strings or codewords of equal length is the number of positions for which the corresponding symbols are different. That is, it measures the minimum number of substitutions required to change one into the other, or the number of *errors* that transformed one string into the other. Another way to define HD is that, it is the number of bit positions in which two codewords differ. See the following examples:

	Codeword1		**Codeword2**		**HD**
i.	**1011101**	and	**1001001**	is	2.
ii.	**2173896**	and	**2233796**	is	3.
iii.	**toned**	and	**roses**	is	3.

This is a cube HD:

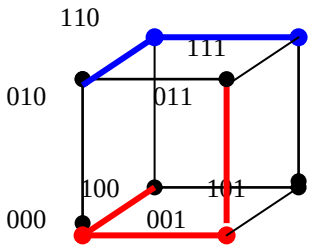


Figure 1: Determining Hamming Distance

As seen in Figure 1 above, following the red path, from 100 to 011, it has distance 3. This is quite obvious if you count the total number of the paths. Also, following the blue path, from 010 to 111, the distance is 2.

The following C++ code computes the Hamming Distance (HD) between two strings of equal distance.

```
const  int  m, n;
 int Hd;
 char  S1[m];
 char  S2[n];
   ---
      if (m == n)
        {
          for (int i = 0;  i < m;  i ++)
              {
                if  (S1[i]  != S2[i])
                     Hd  += 1;
                     sum = Hd;
                }
                return  sum;
        }
```

The significance of HD is that if two codewords are a Hamming distance ***d*** apart, it will require ***d*** single-bit errors to convert one into the other. The error detecting properties of a code depend on its Hamming distance because to detect d errors, you need a distance of d+1 code.

4. POLYNOMIAL CODES IN DETECTING ERRORS

In data transmission, a simplex channel is not preferred if an error is detected, but most often, error detection followed by retransmission is preferred because it is more efficient to handle. A polynomial code is a widespread means of detecting errors in data transmission. Polynomial coding treats each message as a polynomial equation with each bit in the message representing a coefficient in the equation. Polynomial codes are based on treating bit strings as representations of polynomial with coefficients of 0 and 1 only. For instance, a k-bit frame is regarded as the coefficient list for a polynomial with k terms, ranging from x^{k-1} to x^0. Such a polynomial is said to be of degree k – 1. The higher order (leftmost) bit is the coefficient of x^{k-1}; the next bit is the coefficient of x^{k-2}, and so on. For example, 110001 has 6 bits and thus represents a six term polynomial with coefficients 1, 1, 0, 0, 0, and 1 as $x^5 + x^4 + x^0$.

When the polynomial code method is used, the sender and the receiver must agree upon a ***generator polynomial***, **G(*x*)** in advance, (Tanenbaum, 1989). Both the high order and low order bits of the generator must be 1. To compute the **checksum** for some frames with ***m*** bits, corresponding to the polynomial **M(*x*),** the frame must be longer than the generator polynomial, **G(*x*).**

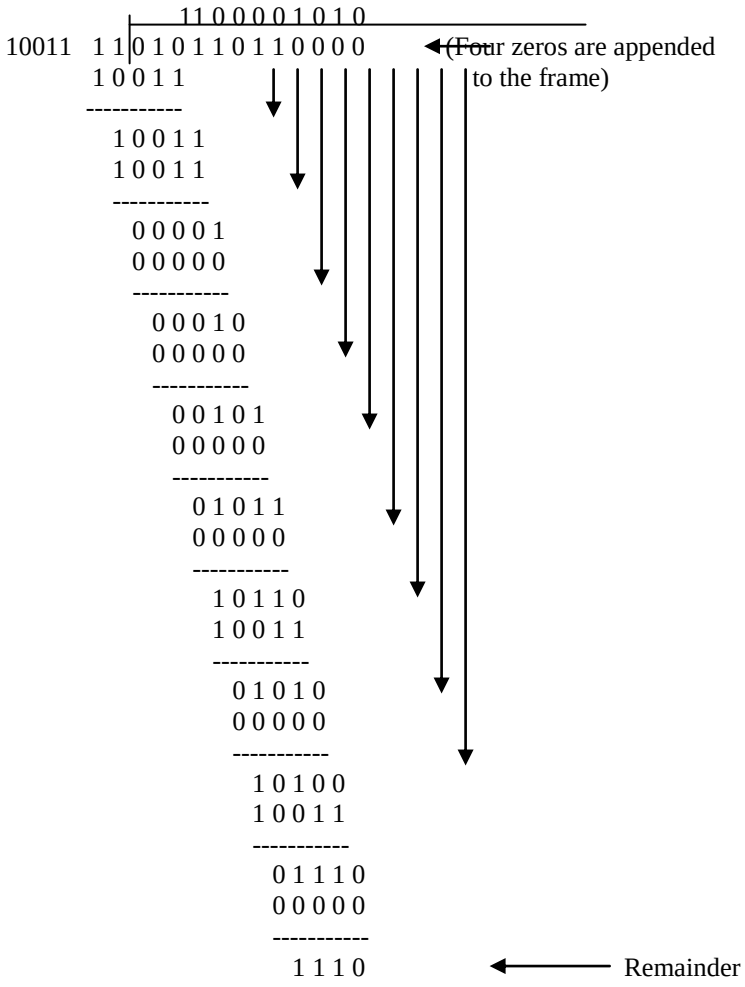
The basic idea is to append a checksum to the end of the frame in such a way that the polynomial represented by the checksummed frame is divisible by **G(*x*).** If there is a remainder, there has been a transmission error, otherwise, there is no error if there is no remainder. The following algorithm was used in Andrew Tanenbaum's book (Tanenbaum, 2003) in computing the checksum:

a. Let **r** be the degree of G(x). Append r zero bits to the low order end of the frame so that it now contains m + r bits, which corresponds to the polynomial $x^rM(x)$.

b. Divide the bit string corresponding to G(x) into the bit string corresponding to $x^rM(x)$ using modulo 2 division arithmetic.

c. Subtract the remainder (which is always *r* or fewer bits) from the bit string corresponding to $x^rM(x)$ using modulo 2 subtraction arithmetic. The result is the checksummed frame to be transmitted, called ***T(x).***

For example, calculate the checksum of this frame: 1101011011 with $G(x) = x^4 + x + 1$.



Transmitted Frame (Tx): 1101011011 1110

Figure 2: Calculation of Checksum

Generally, in TCP, the algorithm used to calculate the checksum as seen in Figure 2 above by the sending device is the same algorithm employed by the receiver to check the data received and to ensure that there were no errors, (Kozierok, 2005).

4.1 Analysis of the Method

Upon transmitting the checksummed frame (see figure 2 above), the receiver divides the *T(x)* by *G(x)*, the result is always zero if there was no error. But, if a transmission error occurred such that, instead of the polynomial for *T(x)* arrived, *T(x) + E(x)* was received. The receiver divides the checksummed frame (T(x) + E(x)) by G(x). The result of the computation is simply E(x)/G(x). Each 1 bit in *E(x)* corresponds to a single bit that has been inverted. That is, if there are ***k*** bits in *E(x)*, ***k*** single-bit errors have occurred. It is very important to know that a polynomial code with ***r*** check bits will detect all burst errors of length ≤ ***r***. Also, it should be noted that when an error burst longer than ***r + 1*** bits occurs, or several shorter bursts occur, the probability of a bad frame getting through unnoticed is $1/2^r$ assuming that all bit patterns are equally likely.

The following three polynomials are commonly used and have become international standards (Tanenbaum, 1989):

CRC-12 = $x^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-16 = $x^{16} + x^{15} + x^2 + 1$
CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$

All the three polynomials above have x + 1 as a prime factor. CRC-12 is used when the character length is 6 bits. The other two are used for 8-bit characters. A 16-bit checksum such as CRC-16 or CRC-CCITT catches all single and double errors.

As can be seen, the above polynomials can only be used effectively in error detections when the character bits are 8 or less. But, when the character bits are 16, 32, or more as we deal with humongous data transmission these days, then, there is a need for a more robust error detection polynomial code that can handle the transmission of such frames effectively. Polynomial codes such as the following should be standardized by ***ISO*** to effectively handle error detection in a frame of higher capacity.

CRC-32 (IEEE) $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

CRC-64 $x^{64} + x^{62} + x^{57} + x^{55} + x^{54} + x^{53} + x^{52} + x^{47} + x^{46} + x^{45} + x^{40} + x^{39} + x^{38}$ (ECMA) $+ x^{37} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{27} + x^{24} + x^{23} + x^{22} + x^{21} + x^{19} + x^{17} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^4 + x + 1$

The above generator polynomial are more robust than the conventional ones and they are necessary for multimedia applications especially as the amount of IP-based traffic increases. The polynomial used in IEEE 802, that is CRC-32 has the property that detects all bursts of length 32 or less and all bursts affecting an odd number of bits, (Tanenbaum, 2003).

4.2 Undetected Erroneous Bits

There are certain times when some erroneous bits in data frames may be transmitted undetected by CRC as a result of the transmission assumptions used in that system. Paulitsch et al did an enormous work in this area. They stated that some communication protocols compute the CRC frame check sequence (FCS) over several fields but send only some fields as part of the message, (Paulitsch et al, 2005). For example: the figure below shows a calculated FCS frame at the Source and Destination as well as the transmitted message. If the data at the Source (data A) is the same as the Destination data (data A'), then the ability of CRCs to detect errors in the communication line is not influenced.

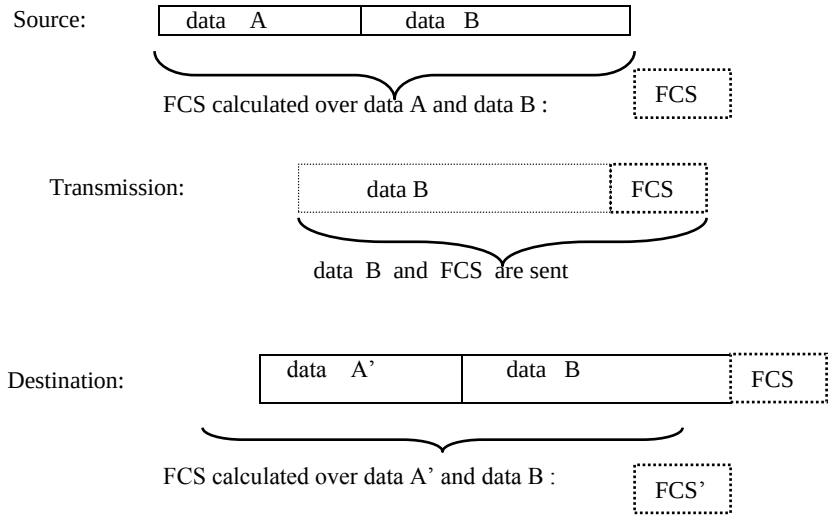


Figure 3: Calculation of an Extended CRC Check

The only consequence of including data A in the CRC calculation is the need to consider the overall length (length of data A and data B) for the achievable Hamming distance (HD). One of the reasons for including some information (data A) into the CRC calculation but not transmitting it might be that this data is already part of a different protocol layer (e.g. the address in TCP/IP), or that the agreement on a protocol version needs to be enforced without the expense of communication bandwidth. The CRC Frame Check Sequence (FCS) calculation with additional data that is not sent was called an *extended CRC check*.

If the data included in the CRC calculation but not sent - is different at the source and the destination: data A and data A' (in Figure 3), then the ability of the CRC to detect failure in transmitted data is decreased. The decrease is dependent on the number and positioning of bits that are different between data A and data A'. Even a single bit error in data transmission can lead to an undetectable error if more than HD - 1 bit differs in the unsent data.

Another example is the Implicit Acknowledgement in Time Triggered Protocol/ Communication (TTP/C). The Hamming Distance (HD) of TTP/C is at least 6 (TTTech, 2004) for frames that use the CRC to protect 2024 bits or fewer. For certain types of frames, TTP/C calculates the CRC over the data and header (including the version number/schedule identifier and the membership vector), but the version number and the membership vector are not transmitted. Because of the implicit acknowledgment mechanism in TTP/C, when a sending node experiences a fault that prevents any receiver from receiving its frame, it is placed into a separate group from other transmitters, and the membership vector of nodes in these different groups can differ by up to two bits. Sending nodes then compute their transmitted FCS based on these different membership vectors. Because the Hamming Distance of the TTP/C CRC is at least 6, and 2 bits of Hamming Distance can be consumed by membership vector differences, this leaves only a Hamming Distance of 4 bit inversions to detect possible bit errors in transmitted data. This clearly increases the probability of an undetected error, (Paulitsch et al, 2005).

4.3 Analysis of CRC Method

As stated earlier, a polynomial code is a widespread means of detecting errors in data transmission. Since, the probability of erroneous frames getting through the transmission links undetected by the CRC is: $\boldsymbol{1/2^r}$. In this section, we calculate the probability of erroneous data (usually in frames) passing through the system undetected using various degrees of the generator polynomial, **G(x)**, which is the same as **r**.

i) When r = 8
The prob. = $(1/2)^8$
= 0.004

ii) When r = 16
The prob. = $(1/2)^{16}$
= 0.000015

iii) When r = 32
The prob. = $(1/2)^{32}$
= 2.33^{-10}

iv) When r = 64
The prob. = $(1/2)^{64}$
= 5.42^{-20}

We can see that the higher the degree of the generator polynomial G(x), the lower the error in the transmission system and the more impeccable the system becomes.

5. CONCLUSION

The selection of generator polynomial ***G(x)*** is the most important part in implementing the CRC algorithm. The polynomial must be chosen to maximize the error detecting capabilities while minimizing overall collision probabilities. As established in section 4 above, the probability of a bad frame getting through unnoticed is $1/2^r$ and r is the degree of G(x), therefore, CRC-16, CRC-32, and CRC-64 has a probability of failure equal to 0.000015, 2.33^{-10}, and 5.42^{-20} respectively. This means that the more robust the generator polynomials are, the more effective they are in detecting bad frames even when transmitting humongous data. The length (the number of the highest nonzero coefficient) of the polynomial is another important attribute in implementing the CRC algorithm because of its direct influence in the length of the computed checksum.

AUTHOR INFORMATION

Dr. Daniel Owunwanne is an Assistant Professor at Howard University, Washington DC, USA. His areas of research interest are Data Interoperability in Federated Databases, Software Fault Tolerance, and Data Transmission. Dr. Owunwanne has presented many papers in these areas of his research interest and as well as in an interdisciplinary areas in both national and international conferences. He has also published many papers in conference proceedings and refereed journal articles. He teaches Databases, C++, Software Design, Data Communications and Networks, and Systems Analysis and Design.

REFERENCES

1. Koopman, Philip (2002), 32-Bit Cyclic Redundancy Codes for Internet Applications, The International Conference on Dependable Systems and Networks (DSN) 2002.
2. Kozierok, C. M. (2005), *TCP/IP Checksum Calculation*. Retrieved. TCP/IP Guide: www.tcpipguide.com/free/t_TCPChecksumCalculationandtheTCPPseudoHeader.htm
3. Paulitsch, Michael et al (2005), Coverage and the Use of Cyclic Redundancy Codes in Ultra-Dependable Systems. Dependable Systems and Networks, 2005.
4. Petr Hlávka, et al (2205), CRC64 Algorithm and Verification. The CESNET Technical Report, December 15, 2005.
5. Tanenbum, Andrew S. (1989), *Computer Networks* (2nd Edition), Prentice Hall.
6. Tanenbaum, Andrew S. (2003), *Computer Networks* (4th Edition), Prentice Hall PTR, www.phptr.com
7. TTTech Computertchnick GmbH, Time-Triggered Protocol Communication, *TTP/C*, TTTech Comp. GmbH, Austria 2005.
8. Williams (1993), A Painless Guide to CRC Error Detection Algorithms, RockSoft™ Pty, Ltd, www.geocities.com/CapeCanaveral/Launchpad/3632/CRCguide.htm