International Journal of Computer Applications (0975 – 8887)
Volume 35– No.2, December 2011

An approach for a Standard Polynomial for Cyclic Redundancy Check

Hasan Md. Mohtarim
MBA
Institute of Business Administration
University of Dhaka

Ayesha Zaman
Lecturer
Dept. of Applied Physics,Electronics & Communication Engineering
University of Dhaka

Nowrin Hoque
MBA
University of Dhaka

ABSTRACT

In this paper Cyclic Redundancy Check codes are implemented to detect the effectiveness of various types of errors that might occur during the transmission of datastream carrying message signal through the internet. MATLAB simulation software is used to propose a standard polynomial to justify such errors. The justification has been done in such a way that bit position, number of bits of dataword and codeword of the generator polynomial are considered random. For getting accuracy, simulation at every step of error detection was done many a times in numerous ways. This helped to procure satisfactory result with the proposed polynomial.

General Terms

Transmission Technology, Data Communication, Data Security, Latency.

Keywords

Generic Polynomial, Cyclic Redundancy Check, Dataword, Codeword, Syndrome.

1. INTRODUCTION

When a message signal is sent from one network to another various distortion or change within the shape of that signal occurs. This is due to attenuation, presence of noise, high latency all these factors. Broadcast links and transmission links are the two main types of data transmission technology[1]. Many error-detecting and error-correcting codes are known, but both ends of the connection must agree on which one is being used. In addition, the receiver must have some way of telling the sender which messages have been correctly received and which has not. There are various error detection methods. Method of coding can be mainly classified as block coding and convolution coding. There are also other ways like evaluation of minimum hamming distance, internet checksum, cyclic coding. Among these, cyclic codes provide better performance in detecting different types of errors. These can easily be implemented in hardware and software . Also they are especially fast in action. Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword. Cyclic Redundancy Check (CRCs) codes are so called because the check (data verification) code is a redundancy (it adds zero information) and the algorithm is based on cyclic codes[2]. The central concept in detecting or correcting errors is redundancy. To be able to detect or correct errors, extra bits are needed to be sent with the data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct the corrupted bits.

2. BASIC TYPES OF ERRORS AND A PROPOSED POLYNOMIAL

Whenever bits flow from one point to another, they are subject to unpredictable changes because of interference. This interference can change the shape of the signal. In a single-bit error, a 0 is changed to a 1 or a 1 to a 0. In a burst error, multiple bits are changed. A burst error is more likely to occur than a single-bit error. The duration of noise is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise. Now a better way to understand cyclic codes and how they can be analyzed is to represent them as polynomials. A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit. Here ($x^4 + x^3 + x + 1$) this 4th order polynomial is chosen as a standard one ,whose binary pattern is represented as 11011.

3. ALGORITHM FOR CRC ERROR DETECTION

```
CRC_ENCODER (data_word, divisor)
k ← length of the data_word
n ← length of the code_word
c ← n – k
Augment c 0's to the right hand side of the data_word to create the augmented_data_word
remainder = MOD_2_DIV (augmented_data_word, divisor)
Append the remainder to the original data_word to create the code_word
CRC_DECODER (code_word, divisor)
remainder = MOD_2_DIV (code_word, divisor)
        If the remainder contains all 0's then
                the data_word is accepted
        Else

                the data_word is discarded
MOD_2_DIV (augmented_data_word, divisor)
Call the uppermost c bits of the message the remainder
Beginning with the most significant bit in the original message and for each bit position that follows, look at the c bit remainder:
        If the most significant bit of the remainder is a one then
```

```
            Set the appropriate bit in the quotient to a one, and
            XOR the remainder with the divisor and store the result back into the remainder
        Else
            Set the appropriate bit in the quotient to a zero, and
            XOR the remainder with zero (no effect)
        Left-shift the remainder, shifting in the next bit of the message.
Return the remainder
```

4. DESIGN OF CRC ENCODER-DECODER

A code is called cyclic if $[x_nx_0x_1...x_{n-1}]$ is a codeword whenever $[x_0x_1...x_{n-1}x_n]$ is also a codeword. CRCs are also classified as block coding. In block coding, the original message is divided into blocks, each of k bits, called datawords, r redundant bits are added to each block to make the length n = k + r. The resulting n-bit blocks are called codewords. Now the extra r bits are chosen or calculated. CRC is used in networks as LANs or WANs. Figure (1) depicts the principle of operation of CRC Encoder and Decoder.

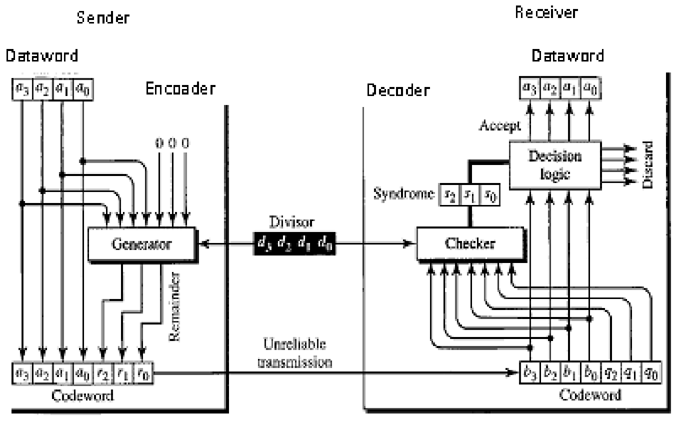

Fig 1: CRC encoder and decoder

In the encoder, the dataword has k bits (8 here); the codeword has n bits (12 here). The size of the dataword is augmented by adding n - k (4 here) 0's to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size n - k + 1 (5 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (r_3r_2 r_1r_0) is appended to the dataword to create the codeword. The decoder receives the possibly corrupted codeword. A copy of all n bits is fed to the checker which is a replica of the generator. The remainder produced by the checker is a syndrome of n - k (4 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all as, the 8 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 8 bits are discarded (error).

5. CYCLIC CODE ANALYSIS

A cyclic code can be analyzed to find its capabilities by using polynomials. The following is defined where f(x) is a polynomial with binary coefficients.

Dataword: d(x)   Codeword: c(x)   Generator: g(x)
Syndrome: s(x)   Error: e(x)

If s(x) is not zero, then one or more bits is corrupted. However, if s(x) is zero, either no bit is corrupted or the decoder failed to detect any errors. In a cyclic code,

I. If $s(x) \neq 0$, one or more bits is corrupted.
2. If s(x) = 0, either
a. No bit is corrupted. or
b. Some bits are corrupted, but the decoder failed to detect them.

In this analysis the intention is to find the criteria that must be imposed on the generator, g(x) to detect the type of error . Now the relationship among the sent codeword, error, received codeword, and the generator is also found as shown below:
Here, received codeword =c(x) + e(x)
In other words, the received codeword is the sum of the sent codeword and the error. The receiver divides the received codeword by g(x) to get the syndrome. This can be written as-

$$\frac{received\ codeword}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

The first term at the right-hand side of the equality does not have a remainder (according to the definition of codeword). So the syndrome is actually the remainder of the second term on the right-hand side. If this term does not have a remainder (syndrome =0), either e(x) is 0 or e(x) is divisible by g(x). The first case is simple if there is no error; the second case is very important. Those errors that are divisible by g(x) are not caught. In a cyclic code, those e(x) errors that are divisible by g(x) are not caught.

6. SIMULATION & RESULTS

One main program CRC.m and one sub-program DIV.m is used to check the effectiveness of the proposed polynomial and finally to implement Cyclic Redundancy Check as an error detection method using MATLAB.

6.1 Single bit error

A single-bit error is e(x) =xi, where i is the position of the bit. If a single-bit error is caught, then xi is not divisible by g(x). If g(x) has at least two terms (which is normally the case) and the coefficient of x0 is not zero (the rightmost bit is 1), then e(x) cannot be divided by g(x).The proposed polynomial is(x4 + x3 + x + 1). So it can detect all single-bit errors. No xi can be divisible by x + 1. In other words, xi/(x + 1) always has a remainder. So the syndrome is nonzero. Any single-bit error can be caught.

6.2 Two isolated single bit errors

For the proposed polynomial shown with the function g(x) two isolated single bit error is expressed as e(x) =: xj + xi. The values of i and j define the positions of the errors, and the difference (j-i) defines the distance between the two errors. Hence e(x)/g(x) always has a remainder for any i, j<n. So every two isolated single bit error is caught.

6.3 Odd number of errors

A generator with a factor of x+1 can catch all odd number of errors. So the proposed polynomial g(x) = x4 + x3 + x + 1 can detect all odd number of errors.

6.4 Burst errors

Here burst error function is defined as $e(x) = x^j + ... + x^i$ or $e(x)= x^i(x^{j-i} + ... + 1)$. In this type remainder of $(x^{j-i} + ... + 1)/(x^r + ... + 1)$ must not be zero; where r is the degree of polynomial. For the detection of burst error three cases are taken into consideration:

Case 1: $L \leq r$, all burst errors with length smaller than or equal to the number of check bits will be detected.

Case 2: $L = r+1$, $s(x) = 0$ and the probability of undetected error of length r+1 is $(1/2)^{r-1}$.

Case 3: $L > r+1$, $s(x) = 0$ and the probability of undetected error of length greater than r+1 is $(1/2)^r$; L implies error length.

7. GRAPHICAL REPRESENTATION

To understand the effectiveness of the proposed polynomial for the detection of the above mentioned three types of errors; a simulation based comparison was done with the proposed polynomial to that of other standard polynomials like (x3 + x +1), (x4 + 1), (x7 + x6 +1), (x6 +1). It was found that the proposed polynomial was able to detect with 100% accuracy the single bit error, two isolated single bit error and burst error upto 11 bit. The simulation results are graphically shown below.

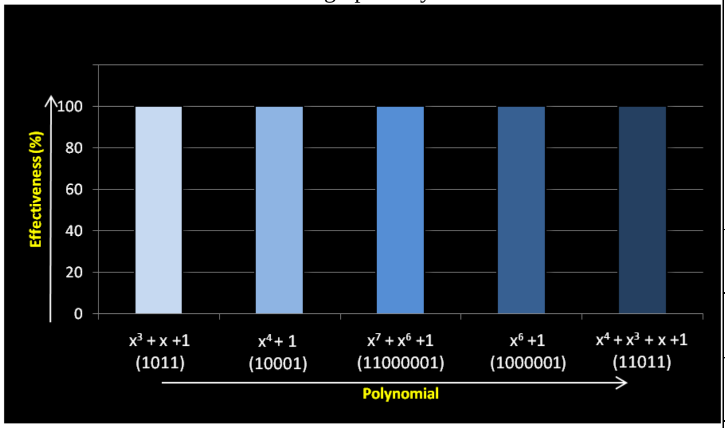

Fig 2: Graphical representation of the effectiveness of different polynomials for Single-bit Error

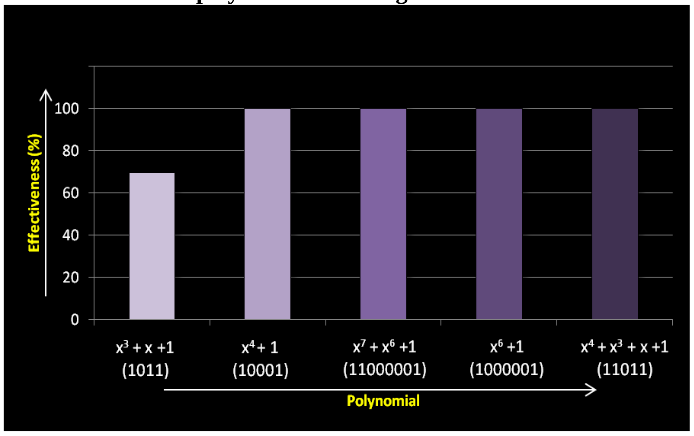

Fig 3: Graphical representation of the effectiveness of different polynomials for Two Isolated Single-bit Errors

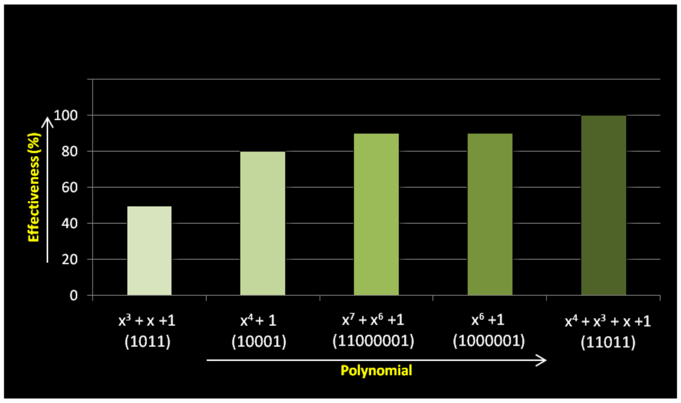

Fig 4: Graphical representation of the effectiveness of different polynomials for Burst Error (up to 11 bit)

8. COMPARISON OF DIFFERENT POLYNOMIALS

Generator Polynomial	No. of Execution of the Program	Detection			Effectiveness			
		Single-bit Error	Two Isolated Single-bit Error	Burst Error (up to 11 bit)	Single-bit Error	Two Isolated Single-bit Error	Burst Error (up to 11 bit)	Overall
$x^3 + x + 1$ (1011)	10	10	7	5	100 %	70%	50%	73.33 %
$x^4 + 1$ (10001)	10	10	10	8	100 %	100%	80%	93.33 %
$x^7 + x^6 + 1$ (11000001)	10	10	10	9	100 %	100%	90%	96.67 %
$x^6 + 1$ (1000001)	10	10	10	9	100 %	100%	90%	96.67 %
$x^4 + x^3 + x + 1$ (11011)	10	10	10	10	100 %	100%	100 %	100 %

Table: Data Analysis for Different Polynomials

From the above comparison table it is very conspicuous that the accuracy of error detection capability of the proposed polynomial (x4 + x3 + x + 1) is the maximum than the other standard polynomials. After 11 bit of burst error this polynomial got no effectiveness.

9. OVERALL PERFORMANCE ANALYSIS

Most of the cases the generator polynomial $(x^3 + x +1)$can detect two isolated single bit errors but sometimes it cannot detect two isolated errors. This is not a good polynomial for detecting burst errors. In case of the polynomial $(x^4 + 1)$ cannot detect two errors which are four positions apart. The location of these two errors can be anywhere, but if their distance is 4, this generator is unable to detect them. $x^7 + x^6 +1$ is good for detecting single-bit errors and two isolated single-bit errors. Another (x^6 +1) can detect which has a length less than or equal to 6 bits. 3 out of 100 burst errors with a length of 7 will slip by. 16 out of 1000 burst errors with a length of 8 or more will slip by. The overall performance of all of these polynomials with the proposed generator polynomial is shown graphically in figure 5.

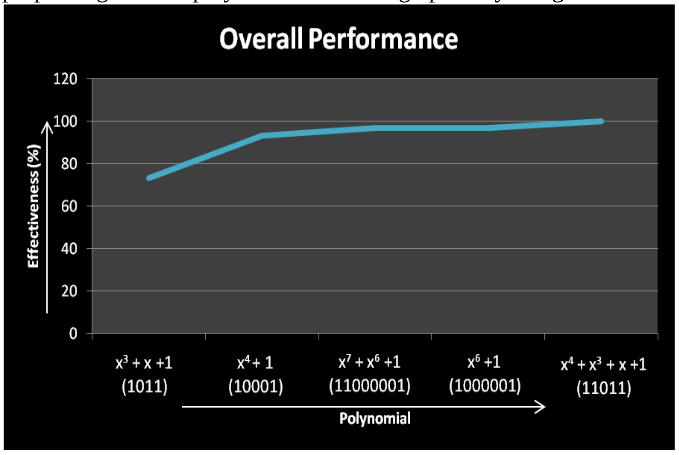

Fig 5: Graphical representation of the overall performance of different polynomials for all error types.

10. CONCLUSION

The mechanism of CRC system is a little difficult to implement as the division process is to be done using Modulo-2 arithmetic. Selection of the generator polynomial is the most sophisticated part of this work. A polynomial has been proposed considering some standard features.The polynomial results quite successfully up to 11 bit error. The proposed polynomial can detect single bit error, two isolated single bit error and burst error with cent percent efficiency.

11. ACKNOWLEDGEMENTS

The authors thank to all the experts who have worked earlier in this field of computer networking and security system .

12. REFERANCES

[1] Andrew S. Tanenbaum, Computer Networks; 4th edition.

[2] Behrouz A. Forouzan, Data Communication and Networking; 4th edition.

[3] MathWorks, available at www.mathworks.com

[4] Mathtools.net, available at www.mathtools.net