Error detection by CRC

Stefan Höst

Error Control

On layer 2-4 the data is often considered as packets or frames consisting of bytes. When crossing the border from layer 2 to layer 1, it is instead viewed as a sequence of bits. Most often the individual bits are treated as independent and the position and meaning in the higher layer packet is not considered. To transmit the binary sequence from one place to another they must first be transformed to continuous signals, a process is called modulation. During the transmission the signals are exposed to various disturbances and distortion. It can for example be noise from the electrical circuits, interference from other users or impulse noise. At the receiver side the challenge is then to interprete the received signals and estimate the most likely transmitted binary sequence, which is called demodulation. Mostly this is done correctly, but sometimes the disturbances are too strong and there are errors in the estimated sequence. Depending on the characteristics of the disturbances the errors can be distributed differently. Often they are considered to be bitwise errors, independent of each other, but there can also be dependencies between the errors, they can be located in bursts. In this text we will assume that errors are independent.

The idea of error control coding is to add extra (redundant) data to the transmitted data, with the aim for the receiver side to be able to detect or even correct errors that occurred during transmission. There are many different flavours of codes, with different complexity. The simples imaginable error detection coding scheme is the parity bit. There an extra bit is added to a binary vector such that there is always an even number of ones. If the receiver sees a vector with an odd number of ones it can directly say that there has been an error during the transmission. This scheme, however, does only give the receiver means to detect an error in a vector, and not in which bit the error lies. To see how a bit error can actually be corrected, a repetition code can be used. At the transmitter each bit is transmitted three times. Then when the receiver sees three ones it assumes that a one is transmitted. But if it sees two ones and a zero, it sees that there has been an error. It can either be the single zero that is errored or both the ones. The most likely event is then that that it was a one transmitted and that there is one bit error in the transmission, so the receiver assumes it is a one.

In the above examples there is one error detection scheme and one error correction scheme. Both of them are quite primitive and there are more efficient ways to do it. For error corrections things are getting more complicated as the coding becomes more efficient. Today, there are codes that work very close to what is theoretically possible. However, for error detection the most widely used scheme is Cyclic Redundancy Check (CRC) coding. It can be seen as a generalisation of the described parity check coding. Instead of summing up all the bits in the binary vector to form one extra bit, there can be

different ways to sum to form several redundancy bits. The redundant bits can be placed anywhere in the transmitted vector, but in this text we will view then as appended at the end of the data

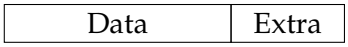

Figure 1: Redundant data is appended to the transmitted data.

In many cases when binary data is involved in derivations, modulo 2 arithmetic is used. Then the following addition and multiplication tables are used. It can be seen from this that the operations plus and minus are identical, hence, there is no meaning in using minus.

$\oplus$	0	1
0	0	1
1	1	0

$\otimes$	0	1
0	0	0
1	0	1

Table 1: Addition and multiplication for modulo 2 arithmetic.

Cyclic Redundancy Check (CRC)

CRC codes are often used for error detection over frames or vectors of a certain length. To get a convenient mathematical notation of the positions in the frame it can be expressed as a polynomial in x, where the exponent of x is the place marker of the coefficient.[1] The vector $\boldsymbol{a} = a_{L-1}a_{L-2}\ldots a_1a_0$ length L is represented by the degree $L-1$ polynomial

$$a(x) = \sum_{i=0}^{L-1} a_i x^i = a_{L-1}x^{L-1} + a_{L-2}x^{L-2} + \cdots + a_1 x + a_0$$

Example 1 *For example, the vector* $\boldsymbol{a} = 1000011$ *is transformed as*

$$\begin{aligned} a(x) &= 1x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0 \\ &= x^6 + x + 1 \end{aligned}$$

□

The idea of CRC coding can be seen as a generalization of the parity check bit described earlier. Parity bits work well for short vectors where they can be used to detect one bit error. However, if there are errors in two positions the resulting vector will still have av even number of ones, and the error will not be detected. To achieve a better error protection several parity bits can be added, derived over a different set of data bits. Here the procedure will first be described followed by an analysis of the error detection capabilities.

[1] It works very much like discrete transforms like z-transform or DFT, but here x does not have a mathematical meaning.

Let the data to be transmit consist of a length k binary vector, and represent it by the degree $k-1$ polynomial

$$d(x) = d_{k-1}x^{k-1} + d_{k-2}x^{k-2} + \cdots + d_1 x + d_0$$

Then, to add redundant bits so the total length of the codeword is n, we should add $n-k$ bits. These redundant bits, the CRC bits, can be represented by the degree $n-k-1$ polynomial

$$r(x) = r_{n-k-1}x^{n-k-1} + \cdots + r_1 x + r_0$$

and the codeword polynomial can be written as

$$\begin{aligned} c(x) &= d(x)x^{n-k} + r(x) \\ &= d_{k-1}x^{n-1} + \cdots + d_1 x^{n-k+1} + d_0 x^{n-k} + r_{n-k-1}x^{n-k-1} + \cdots + r_1 x + r_0 \end{aligned}$$

To derive the CRC polynomial a degree $n-k$ *generator polynomial* is used

$$g(x) = x^{n-k} + g_{n-k-1}x^{n-k-1} + \cdots + g_1 x + 1$$

It is a binary polynomial with degree $n-k$ where the highest and lowest coefficients are non-zero, i.e. $g_{n-k} = 1$ and $g_0 = 1$. Then the CRC polynomial is derived as[2]

$$r(x) = R_{g(x)}\big(d(x)x^{n-k}\big)$$

The polynomial division is performed in the same manner as normal polynomial division, except that all coefficients are binary and modulo 2 arithmetic is used. The procedure is shown in the following example.

Example 2 *Assume we have the data word* $\boldsymbol{d} = 1001$ *and want to add three CRC bits. In this case we get* $k = 4$ *and* $n = 7$*. The data word is represented as a polynomial as* $d(x) = x^3 + 1$*. Then find a degree three generator polynomial, say* $g(x) = x^3 + x + 1$*. Later in the text it will be considered how too chose these polynomials. Performing the division* $d(x)x^3/g(x)$*,*

$$\begin{array}{r|l} & x^3+x \\ \hline x^3+x+1 & x^6+\quad\ \ x^3 \\ & \underline{x^6+\ x^4+x^3} \\ & \quad\ x^4 \\ & \quad\ \underline{x^4+\quad x^2+x} \\ & \qquad\ x^2+x \end{array}$$

gives that

$$\frac{d(x)x^3}{g(x)} = \frac{x^6+x^3}{x^3+x+1} = x^3 + x + \frac{x^2+x}{x^3+x+1}$$

Hence, the CRC polynomial is

$$r(x) = R_{x^3+x+1}\big(x^6+x^3\big) = x^2 + x$$

and the codeword polynomial

$$c(x) = x^6 + x^3 + x^2 + x$$

The codeword rewritten as a binary vector becomes $\boldsymbol{c} = 1001110$*.* □

[2]The notation $R_a(b)$ means the reminder from the division b/a. That is, if a and b are polynomials we can (uniquely) find d and r in $b = a \cdot d + r$ where $\deg(r) < \deg(a)$. In some texts it is denoted with the modulo operator as $\mod(b, a)$

To see how the receiver side can use this codeword to detect errors we first need derive some properties of it. Let $z(x)$ denote the quotient in the division $d(x)x^{n-k}/g(x)$. In the previous example $z(x) = x^3 + x$. Then, the data polynomial can be written as

$$d(x)x^{n-k} = g(x)z(x) + r(x)$$

Equivalently, since addition and subtraction are identical in modulo 2 arithmetic, the codeword polynomial becomes

$$c(x) = d(x)x^{n-k} + r(x) = g(x)z(x)$$

That is, all polynomials are divisible by the generator polynomial $g(x)$ and all polynomials (with degree less than n) that are divisible by $g(x)$ are polynomials for codewords. In other words we have the following theorem.[3]

Theorem 1 *A polynomial $c(x)$ with $\deg(c(x)) < n$ is a codeword if and only if $g(x)|c(x)$.*

If $c(x)$ is transmitted over a channel and there occur errors, they can be represented by an addition of the polynomial $e(x)$, and the received polynomial is $y(x) = c(x) + e(x)$.

Example 3 *According to the previous example the codeword transmitted over the channel is $\boldsymbol{c} = 1001110$. Assuming the channel introduces an error in the third bit, the received vector is $\boldsymbol{y} = 1011110$. The error vector can be seen as $\boldsymbol{e} = 0010000$ and the*

$$\boldsymbol{y} = \boldsymbol{c} \oplus \boldsymbol{e} = 1001110 + 0010000 = 1011110$$

where the addition is performed positon-wise. Since modulo 2 arithmetic is used the function $a+1$ will always invert the bit a. Expressed in polynomial form the error polynomial is $e(x) = x^4$, and the received polynomial

$$y(x) = c(x) + e(x) = x^6 + x^4 + x^3 + x^2 + x$$

□

The receiver can use the fact that $g(x)$ is a factor of each transmitted codeword. Similar to the parity check example, where the receiver detects an error if the received vector has an odd number of ones, in this case the receiver will detect an error if $g(x)$ is not a factor. To check this the reminder of the division $c(x)/g(x)$ is derived as

$$\begin{aligned} s(x) &= R_{g(x)}\big(y(x)\big) = R_{g(x)}\big(c(x) + e(x)\big) \\ &= R_{g(x)}\Big(R_{g(x)}\big(c(x)\big) + R_{g(x)}\big(e(x)\big)\Big) = R_{g(x)}\big(e(x)\big) \end{aligned}$$

This quantity plays a central role in coding theory and is normally called the *syndrome* of the received vector. Notice that the syndrom is directly a function of the error since $R_{g(x)}\big(c(x)\big) = 0$.

[3]The notation $a|b$ means the division b/a has a zero reminder, i.e. that b is a multiple of a. For integers, for example $3|12$ but $3 \not| 10$.

Example 4 *In the previous example the received vector* $\boldsymbol{y} = 1011110$ *was considered. The corresponding polynomial representation is* $y(x) = x^6 + x^4 + x^3 + x^2 + x$. *With the generator polynomial* $g(x) = x^3 + x + 1$ *the following division*

$$\frac{y(x)}{g(x)} = \frac{x^6 + x^4 + x^3 + x^2 + x}{x^3 + x + 1} = x^3 + \frac{x^2 + x}{x^3 + x + 1}$$

gives the syndrom $s(x) = R_{g(x)}(y(x)) = x^2 + x$. *Since this is non-zero an error has been detected.* □

If there are no errors in the transmission then the error polynomial is $e(x) = 0$ and the syndrome $s(x) = 0$. So the criteria for assuming error free transmission at the receiver side is that $s(x) = 0$. In the case when there are errors, and $e(x) \neq 0$, these will be detected if $s(x) \neq 0$. But in some cases, when $e(x) = g(x)p(x)$ for some nonzero polynomial $p(x)$, the syndrom will also be zero and the error will not be detected. That is, an error event will not be detected if it is a codeword. In the following we will consider what types of errors that can detect with the method.

First, assume that we have a single error in position i, i.e. $e(x) = x^i$. Since $g(x)$ has at least two nonzero coefficients so will $g(x)p(x)$, and it cannot be x^i. Hence, all single errors will be detected by the scheme. In fact, by using the generator polynomial $g(x) = 1 + x$ we are back to adding a parity check bit, which is used to detect one error.

If there are two errors during the transmission, let say in position i and j, where $i < j$, the error polynomial is

$$e(x) = x^j + x^i = x^i\big(x^{j-i} + 1\big)$$

This error will not be detected in the case when $g(x)$ divides $x^{j-i} + 1$. From discrete math and the theory of finite fields it is known that if $deg(g(x)) = L$ the least K such that $g(x)|x^K + 1$ does not exceed $2^L - 1$. Furthermore, it is always possible to find a polynomial for which there is equality, i.e. $K = 2^L - 1$ is the least integer such that $g(x)|x^K + 1$. These polynomials are called *primitive polynomials*. So, by using a primitive polynomial $p(x)$ of degree $\deg(p(x)) = L$ all errors of the $x^{j-i} + 1$ will be detected as long as $j - i < 2^L - 1$. By choosing the codeword length n such that $n - 1 < 2^L - 1$ there is no combination of i and j such that $g(x)$ divides $x^{j-i} + 1$. Hence, all double errors will be detected.

To see how the system can cope with three errors, we notice that if a binary polynomial is multiplied with $x+1$ it will contain an even number of nonzero coefficients[4]. Thus, if the generator polynomial contains the factor $x+1$, all polynomials on the form $g(x)z(x)$ will have an even number of nonzero coefficients, and all occurrences of an odd number of errors will be detected. Therefore, it is common to chose the generator polynomial as a primitive polynomial of large enough degree and multiply it with $x + 1$.

So, to chose a generator polynomial, assume that the length of the data frame is k bits. Then to be able to detect double errors a primitive polynomial of degree L. To get the generator polynomial this should be multiplied with $x + 1$. That is, the degree of the generating polynomial is $L + 1$ and the length of the codeword is $n = k + L + 1$. The primitive polynomial should be chosen such that

$$k + L < 2^L - 1$$

[4]For binary polynomials it can be found that $(x + 1)(x^\alpha + x^{\alpha-1} + \cdots + x^{\alpha-\beta}) = x^{\alpha+1} + x^{\alpha-\beta}$ where $\alpha, \beta \in \mathbb{Z}^+$. Generalizing this leads to that $(x + 1)q(x)$ will have an even number of nonzero coefficients.

To summarize the error detection capabilities, if the generator polynomial is chosen correctly we can always detect single, double and triple errors. Apart from this we can also detect all odd number of errors.

Some well known CRC generator polynomials are

$$g(x) = x^8 + x^2 + x + 1 \quad \text{(CRC-8)}$$
$$g(x) = x^{10} + x^9 + x^5 + x^4 + x^2 + 1 \quad \text{(CRC-10)}$$
$$g(x) = x^{16} + x^{12} + x^9 + x^5 + x^4 + x^2 + 1 \quad \text{(CRC-16)}$$
$$g(x) = x^{16} + x^{15} + x^2 + 1 \quad \text{(CRC-16)}$$
$$g(x) = x^{16} + x^{12} + x^5 + 1 \quad \text{(CRC-CCITT)}$$
$$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad \text{(CRC-32)}$$

A list of some primitive polynomials

$p(x)$	$p(x)$
$x^2 + x + 1$	$x^{10} + x^3 + 1$
$x^3 + x + 1$	$x^{11} + x^2 + 1$
$x^4 + x + 1$	$x^{12} + x^6 + x^4 + x + 1$
$x^5 + x^2 + 1$	$x^{13} + x^4 + x^3 + x + 1$
$x^6 + x + 1$	$x^{14} + x^{10} + x^6 + x + 1$
$x^7 + x^3 + 1$	$x^{15} + x + 1$
$x^8 + x^4 + x^3 + x^2 + 1$	$x^{16} + x^{12} + x^9 + x^7 + 1$
$x^9 + x^4 + 1$	$x^{17} + x^3 + 1$

References

[1] Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall, 2nd edition, 1992. Available at web.mit.edu/dimitrib/www/datanets.html.

[2] Behrouz Forouzan. *Data Communications and Networking*. McGraw Hill, 5th edition, 2013.

[3] Shu Lin and Daniel J. Costello. *Error Control coding*. Prentice Hall, 2nd edition, 2004.

[4] Robert McEliece. *Finite Fields for Computer Scientists and Engineers*. Springer, 1986.