Decoding and Encoding using CRC

RUCHITHA VUCHA (20161139)
SATHYA SRAVYA V (20161121)

Cyclic Redundancy Check (CRC)

- This is one of the most popular Error Detecting Codes being used today.
- Basic approach used for error detection is the use of redundancy, where additional bits are added to facilitate detection and correction of errors.
- CRC is based on binary division. In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- CRC is designed for and used to protect against common errors in communication channels but not against intentional alteration of data.

Characteristics

- This is the most powerful type of checksum which is able to detect accidental errors in data that is transmitted/stored between computers.
- According to the remainder of a polynomial division of the contents of the blocks, a short check value is attached to them.
- On the retrieval side, the calculation is repeated and, if the check values do not match, error correction has to be done in order to retrieve data. CRCs can be used for error correction.
- CRC uses Generator Polynomial which is represented by a key. Key is known to both sender and receiver side. Eg: Generator polynomial : $x^3 + x + 1$, key 1011
- Cyclic Redundancy Codes are among the best checksums available to detect and/or correct errors in communications transmissions.

- **Encoding:** Encoding is the process of applying a specific code, such as letters, symbols and numbers, to data for conversion into an equivalent cipher.
- **Decoding:** Decoding is the process of converting code into plain text or any format that is useful for subsequent processes. Decoding is the reverse of encoding.
- CRCs are most efficiently calculated in dedicated hardware. CRC algorithms are long division algorithms in disguise. They are readily implemented using shift-registers.
- But the modulo-2 binary division doesn't map particularly well to the instruction sets off-the-shelf processors. The reasons are:
  (i) Generally no registers are available to hold the very long bit sequence(numerator)
  (ii) Modulo-2 binary division is not the same as ordinary division. So even a processor with a division instruction can't help.

Modulo 2 Division

- The process of modulo 2 division is the same as the familiar division process used for decimal numbers except that there is a small change here.
- Instead of subtraction as in normal division here we use xor instead in modulo 2 division.
- In each step, a copy of the divisor(or data) is XORed with the k bits of the dividend.
- The result of the XOR operation (remainder) is (n-1) bits, which is used for the next step after 1 extra bit is pulled down to make it n bits long.
- When there are no bits left to pull down, we have a result. The (n-1)-bit remainder which is appended at the sender side.

At the Sender side

- Let k : Number of bits in data to be sent from sender side and

  n-k : Number of bits in the key obtained from generator polynomial.
- Then we'll create bit sequence of degree 'n' by multiplying polynomial representation of k bits, i(x) with x^{n-k}
- The **codeword b(x)** is finally produced by adding additional checkbits, i.e sequence r(x) which is the remainder of division of i(x)(dividend) by g(x)(divisor).

Note: Modulo 2 Division is used while dividing i(x) by g(x)

At the Sender side (contd..)

Hence, $$x^{n-k}.\, i(x) = q(x)g(x) + r(x)$$

$$b(x) = x^{n-k}.\, i(x) + r(x)$$

$$b(x) = q(x)g(x) + r(x) + r(x)$$

- The remainder can have a maximum degree of n-k-1
- The most significant ‘k’ bits in the codeword correspond to information sequence being sent, and least significant n-k bits correspond to remainder respectively(crc check bits)

Generator polynomial: $g(x) = x^3 + x + 1$

Information: $(1,1,0,0) \rightarrow i(x) = x^3 + x^2$

Encoding: $x^3 i(x) = x^6 + x^5$

$$
\begin{array}{r}
x^3 + x^2 + x \\
x^3 + x + 1 \overline{\big)\, x^6 + x^5} \\
\underline{x^6 + \quad x^4 + x^3} \\
x^5 + x^4 + x^3 \\
\underline{x^5 + \quad x^3 + x^2} \\
x^4 + \quad x^2 \\
\underline{x^4 + \quad x^2 + x} \\
x
\end{array}
\qquad
\begin{array}{r}
1110 \\
1011 \overline{\big)\, 1100000} \\
\underline{1011} \\
1110 \\
\underline{1011} \\
1010 \\
\underline{1011} \\
010
\end{array}
$$

Transmitted codeword:

$b(x) = x^6 + x^5 + x$

$\rightarrow \underline{b} = (1,1,0,0,0,1,0)$

Here the generator key is **g :**1011

Information to be passed is **i :** 1100

Multiply the information by $\mathbf{x^3}$ then we get $\mathbf{x^{n-k}.i}$ = 1100000.

When this is divided by g(x) the remainder r = 010

After adding r(x) we get b = 1100010

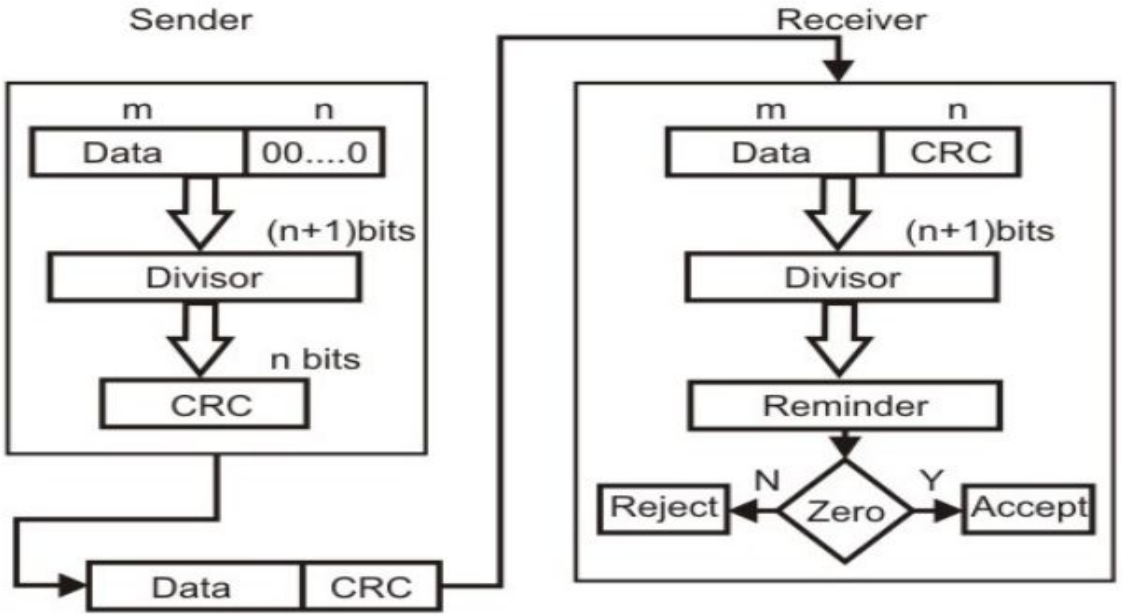
At the Receiver

From the calculations at the sender it's clear that the code word sent is a multiple of generator g(x) as b(x) where we have used the fact that in modulo 2 arithmetic r(x) + r(x) = 0.

$$\mathbf{b(x) = x^{n-k}.\, i(x) + r(x) \;= q(x)g(x) + r(x) + r(x) = q(x)g(x)}$$ **i.e a multiple of g(x)**

- The codeword received is divided by g(x),
  - If the remainder is non zero
    - Then an error has been detected,Receiver rejects the data and asks the sender for retransmission.
  - If the remainder is zero
    - Receiver assumes that no error occurred in the data during the transmission.Receiver accepts the data.

Block diagram



If a k bit message is to be transmitted, the transmitter generates an r-bit sequence, known as Frame Check Sequence (FCS) so that the (k+r) bits are actually being transmitted. Now this r-bit FCS is generated by dividing the original number, appended by r zeros, by a predetermined number. This number, which is (r+1) bit in length, can also be considered as the coefficients of a polynomial, called Generator Polynomial. The remainder of this division process generates the r-bit FCS. On receiving the packet, the receiver divides the (k+r) bit frame by the same predetermined number and if it produces no remainder, it can be assumed that no error has occurred during the transmission.

Algorithm

Server:

- Our aim here is to pass string from sender side to receiver side and detect if any accidental errors occurred.
- The string i(x) is converted into binary string data.
- Augment n-k-1 zeros at the end of the binary data (i.e multiply by $\mathbf{x^{n-k}}$).
- Divide it by generator polynomial g(x) using modulo 2 division.
- Add the remainder r(x) to the binary data to get the codeword.
- Now the data (encoded using the CRC code using the key) is sent to the receiver.

Algorithm

Receiver:

- The receiver receives the codeword from the sender.
- Now the receiver has to decode in order to find any accidental errors that occured in the transmission.
- Divide the received codeword again by the key of generator polynomial using modulo 2 division to check if it's a multiple of generator polynomial.
- **Zero remainder:** This implies that no errors have occurred in the data during transmission. ACK is sent to the Sender.
- **Non-zero remainder:** This implies that error has occurred in the data during transmission. NACK is sent to the Sender so that it can retransmit the data again.

Standardized Polynomial Codes

The generator polynomials which are highly endorsed,

CRC type	Polynomial	Used in
CRC-8	$x^8 + x^2 + x + 1$	ATM header error check
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x + 1$	ATM AAL CRC
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$ $= (x + 1)(x^{11} + x^2 + 1)$	Bisync
CRC-16	$x^{16} + x^{15} + x^2 + 1$ $= (x + 1)(x^{15} + x + 1)$	Bisync
CCITT-16	$x^{16} + x^{12} + x^5 + 1$	HDLC, XMODEM, V.41
CCITT-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	IEEE 802, DoD, V.42, AAL5

Error detecting capability

Cases where error detection fails

1. Single errors: $e(x) = x^i \qquad 0 \le i \le n-1$

If $g(x)$ has more than one term, it cannot divide $e(x)$.

2. Double errors: $e(x) = x^i + x^j \quad 0 \le i \le j \le n-1$

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

If $g(x)$ is primitive, it will not divide $(1 + x^{j-1})$ for $j - i \le 2^{n-k} - 1$.

3. Odd number of errors: $e(1) = 1$ if number of errors is odd.

If $g(x)$ has $(x + 1)$ as a factor, then $g(1) = 0$ and all codewords have an even number of 1s.

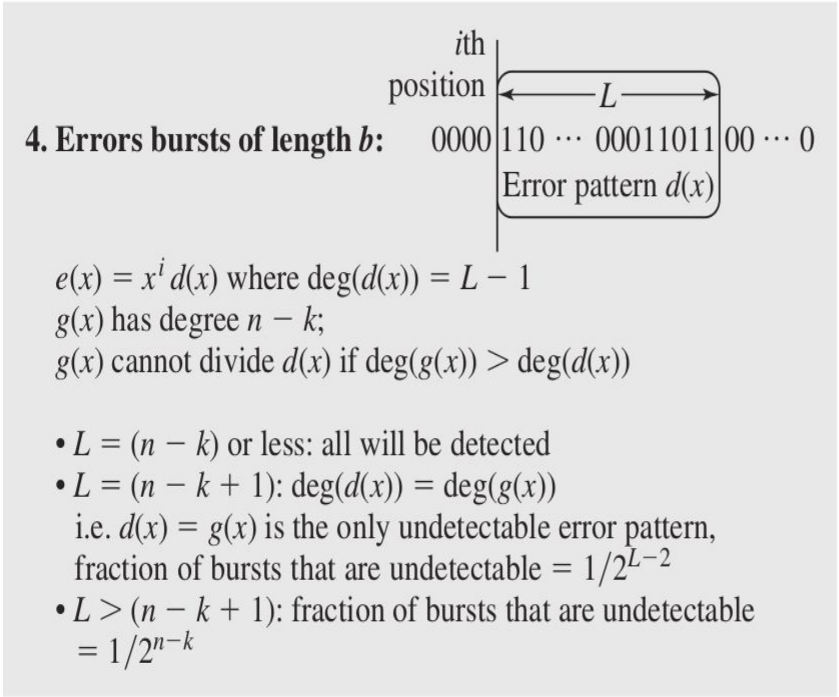
$X^n .M(X)/P(X) = Q(X) + R(X) / P(X)$

Single errors: if $e(x) = x^i$, g(x) has at least has 2 coefficients . Then e(x) is not divisible by g(x).So all single errors can be detected.

Double errors: $e(x) = x^i (1 + x^{j-i})$ can be detected if g(x) (of degree n-k) is primitive as it can't divide$(1 + x^{j-i})$ unless $j-i > 2^{n-k}$.

Odd no. of errors: e(1)=1 , if (x+1) is a factor of g(x) then g(1) = 0 and hence can't divide e(x).

Cases where error detection fails



$X^n .M(X)/P(X) = Q(X) + R(X) / P(X)$

1.Error bursts of length b: $e(x) = x^i d(x)$ If the error burst involves L consecutive bits, then the degree of d(x) is L − 1. e(x) is divisible by g(x) only if d(x) is divisible by g(x).

2.If the burst error has length L = n − k + 1, that is, degree of d(x) = degree of g(x), then d(x) is divisible by g(x) only if d(x) = g(x). probability that error goes undetected is $1/2^{n-k-1}$.

3. In the case of L > n − k + 1 the fraction of bursts that is undetectable is $1/2^{n-k}$.

Improve Efficiency :

- The most common way to improve efficiency of the CRC calculation is to throw memory at the problem.
- For a given input remainder and generator polynomial, the output remainder always be the same i.e for a given dividend and divisor the remainder is always the same.
- So we can precompute the output remainder for each of the possible byte-wide input remainders and store the results in a lookup table.
- The lookup table can be used to speed up the CRC calculations for a given message. The speedup is realized because the message can now be processed byte by byte, rather than bit by bit.

CRC 3: Key: 1010 does not detect most of the errors whereas for the same strings key 1011 detects errors in data

CRC 3: Key: 1001 does not detect most of the errors whereas for the same strings key 1011 detects errors in data

CRC 3: Key: 1100 does not detect most of the errors whereas for the same strings key 1011 detects errors in data

Standard crc, 1011 detects errors while 1000 and 0100 don't

```
sathya@sathya-Inspiron-5559:~/cn2$ python2 c.py
Enter data you want to send->iaygdcisiucjsdcviydsgifdhvdcvusydgcishiufhviguygdsytfyfdyfsyudgvctsdfcyfcutvcgtdfycuagi8sftgvatsffcxasvxhgsvfcytcf
udsgcysdcduysgcyusdtsucdsvycusdvutfdyfcysvdgjdscfsdytcdsidhgyugwefcygeisfcusifhcudgfuygvduysdgyics
1101001110000111110011100111111001001100011110100111100111101001111010111000111101010111001111001001100011111011010100111110011100100111001111
0111110100111001101100100110100011101101100100110001111101101110101111001111110011100100110011111000111101001111001111010001101001111010111001
0110100011101101101001110011111101011111001110011111100100111001111110011110100110011011100111001101100100111100111001101110011111100111101011
0010011001111111011010001111101001110011110010011001101100011111100111001101100011111010111101001110110110000111100111111010011001001100110111
0111000111110101110000111001111101001111000111001111001101110100110011111101101100001111010011100111100110110011011000111111000110000111100111
1011011110001101000110011111100111110111011001101100011111100111101001100011110011011101011100100111001111001111100011111001111001110010011000
1111001001110101111100111100111100111111000111111001111010111001111001001110100111001111101011100011110010011100111111011011100111000111110101
1100111100100111011011101011110100110011011001001111001110011011000111111001111001111101101100100110011111101010110010011100111100011110011011
0111100100111100111101001100011110010011100111101001110010011010001100111111100111101011100111111011111001011100110110001111111001110011111100101
1101001111001111001101100011111010111001111010011100110110100011000111110101110010011001111100110111010111110011100111111011011001001110101111
1001111001111001001100111111100111010011100011111 0011
length of data
1483
Error in data
sathya@sathya-Inspiron-5559:~/cn2$
```

1000 doesn't detect error

```
Enter data you want to send->iaygdcisiucjsdcviydsgifdhvdcvusydgcishiufhviguygdsytfyfdyfsyudgvctsdfcyfcutvcgtdfycuagi8sftgvatsffcxasvxhgsvfcytcf
udsgcysdcduysgcyusdtsucdsvycusdvutfdyfcysvdgjdscfsdytcdsidhgyugwefcygeisfcusifhcudgfuygvduysdgyics
1101001110000111110011100111110010011000111101001111001111010011110101110001111010101110011110010011000111110110110100111110011100100111001111
0111110100111001101100100110100011101101100100110001111101101110101111001111110011100100110011111000111101001111001111010001101001111010111001
0110100011101101101001110011111101011111001110011111100100111001111110011110100110011011110011100110110010011110011100110111001111110011110101
0010011001111111011011000111110100111001111001001100110110001111110011100110110001111101011110100111011011000111100111111010011001001100110111
0111000111110101110000111001111101001111000111001111001101110100110011111101101100001111010011100111100110110011011000111111000110000111100111
1011011110001101000110011111100111110110110011011000111111001111010011000111100110111010111001001110011110011111000111111001111001111001001100
1111001001110101111100111100111100111110001111110011110101111001111001001110100111001111101011100011111001001110011111101101111001110001111101011
1100111100100111011011101011110100110011011001001111001110011011000111111001111001111101101100100110011111101010110010011100111100011110011011100
0111100100111100111101001100011110010011100111101001110010011010001100111111100111101011100111111011111001011100110110001111110011100111110010100
1101001111001111001101100011111010111100111101001110011011010001100011111010111001001100111110011011101011111001110011111110110110010011101011111
1001111001111001001100111111100111010011100011111001111000111101011100100110011111001101110101111100111001111110110110010011101011111001111001
0111000111110101110011101111000110100011001111110011110110110011011000111111001111010011000111100110111010111001001110011110011111000111111001111
1001111001001100011110010011101011111001111001111001111100011111100111101011110011100100111010011100111111010111000111100100111001111101101111
0111000111110101110011101111000110100011001111110011110110110011011000111111001111010011000111100110111010111001001110011110011111000111111001111
1001111001001100011110010011101011111001111001111001111100011111100111101011110011100100111010011100111111010111000111100100111001111101101111
0111000111110101110011
sathya@sathya-Inspiron-5559:~/cn2$
```

0100 doesn't detect error

```
sathya@sathya-Inspiron-5559:~/cn2$ python2 c.py
Enter data you want to send->iaygdcisiucjsdcviydsgifdhvdcvusydgcishiufhviguygdsytfyfdyfsyudgvctsdfcyfcutvcgtdfycuagi8sftgvatsffcxasvxhgsvfcytcf
udsgcysdcduysgcyusdtsucdsvycusdvutfdyfcysvdgjdscfsdytcdsidhgyugwefcygeisfcusifhcudgfuygvduysdgyics
1101001110000111110011100111110010011000111101001111001111010011110101110001111010101110011110010011000111110110110100111110011100100111001111
0111110100111001101100100110100011101101100100110001111101101110101111001111110011100100110011111000111101001111001111010001101001111010111001
0110100011101101101001110011111101011111001110011111001001110011111100111101001100110111100111001101100100111100111001101110011111100111101011
0010011001111111011010001111101001110011110010011001101100011111100111001101100011111010111010011101101100011110011111101001100100110011011110
0111000111110101110000111001111101001111000111001111001101110100110011111101101100001111010011100111100110110011011000111111000110000111100111
1011011110001101000110011111100111110111010011011000111111001111010011000111100110111010111001001110011111001111110001111110011110011100100110000
1111001001110101111100111100111100111110001111110011110101111001111001001110100111001111101011100011110010011100111111011011100111000111110101
1100111100100111011011101011110100110011011001001111001110011011000111111001111001111101101100100110011111101010110010011100111110001111001101110
0111100100111100111101001100011110010011100111101001110010011010001100111111100111101011100111111011111001011100110110001111110011100111110010100
1101001111001111001101100011111010111001111010011100110110100011000111110101110010011001111100110111010111110011100111111011011001001110101111
1001111001111001001100111111110011101001110001111110011
length of data
1483
THANK you Data ->1101001110000111110011100111110010011000111101001111001111010011110101110001111010101110011110010011000111110110110100111110011
1100100111001111001111101001110011011001001110011100110110000111111001111001111101101100100110011111101010110010011100111110001111001101110
0111100100111100111101001100011110010011100111101001110010011010001100111111100111101011100111111011111001011100110110001111110011100111110010100
1101001111001111001101100011111010111001111010011100110110100011000111110101110010011001111100110111010111110011100111111011011001001110101111
1001111001111001001100111111110011101001110001111110011
0111000111110101110011
sathya@sathya-Inspiron-5559:~/cn2$
```

CRC 4: Key: 10000 does not detect any errors whereas for the same strings key 10011 detects errors in data

```
ruchitha@G5-5587:~/crc$ python2 s.py
String to be sent:hasvghdshvcdjscbjhvjnkjfnjkfgbsdcjsuihfiudhvjdfjvbhdfvbhjdbhfvbvhjfbvhjbfjbhjjbhhbjhfhjbvhbdjfhfdbcjhsbjhfbvbhfvbfskjckhjfsbhjfbhvfbhvjhfbhjfvbjfdvbjhfdvbjhfbvhbvfhjbffhjvyufbgjhbfvhhjbv
fvhbfdfjhvbdfhbdfvbhjfbvjdfhbhdbvhjbhdfhvfbdfmkvndfnvdjfnvkjdfnfd
('The length of the string is', 1757)
('The key is', '10000')
No error found1101000110000111100111110110110011111010001100100111001111010001110110110001111001001101010111001111000111100010110101011010001110110110101011011101101011110101011001101101110110101011010111
1001101100111110001011100111100100110001111010101110011111010111010011101000110011011010011110101110010011010001110110110101011001001100110110101011101101100010110100011001001100110111011011000101101000111
0101011001001100010110100011001101110110110001011101101101000110101011001101100010111011011010001101010110001011001101101010110001011010001101010110101011000101101000110100011000101101010110100011001101101
1000110101011000101110110110100011000101100100110101011001101101000110011011001001100010110001111010101101000111001111000101101010110100011001101100010111011011000101101000110011011101101100010110011011101
0111101011110101011000111101011110100011010101100110111001111000101101000110101011001101100010110100011101101100110110001011010001110110110101011010001100110110001011010001101010110011011101101100010110101
1011
ruchitha@G5-5587:~/crc$ python2 s.py
String to be sent:sytdfcyudsbvihidfuhvuodhfbhdsuygcsvdubfiduby7grvbfyuhbfsvbuhsbvusdcbsracsfvxhgavchbdchjbdsiycbuydshbcjhscksncikbyudbuhsbchfhjfbjsbhfdbcycbsdjbcnuydsbuyhybfhussdvgvchgsdvghdschgcsdjbchjbj
dshgvgdshvhsvcdsvhsvghcsdyctsvdcjbjbhscjsvdyhybcddhskfggshcrgsjhguferhgbvkjrdnvkjdsvbjubdvjkdnvkjnfdvhjdfvbjnfdjknvkjdfbvferihvuiehnveiufnvkjefdnkjnvjhbdxbedhbcwhcbjhbsdhcjdbsdbcjksdnkjfdbjdksjhbjhbhj
('The length of the string is', 2701)
('The key is', '10000')
No error found1110011111100111101001100100110011011000111111001111010111001001110011110001011101101101001110100011010011100100110011011101011101000111011011101011101111110010011010001100110110001011010001
1001001110011111010111110011100111110001111100111110110110010011101011100010110011011010011100100111010111000101111001110111111001111110010111011011000101100110111110011110101110100011000101100110110011111
0110110001011101011101000111001111000101110110111010111100111100100110001111000101110011111001011000011100011111001111001101110110111100011010001100111110000111101101100011110100011000101100100110001111101
0001101010110001011001001110011110100111110011100011110001011101011111001110010011100111101000110001011000111101010110100011100111100011110101111100111101110110001111010011101011110001011110011110101111001
0011000101110101110100011100111100010110001111010001100110110100011010101100110110001011010101110011110001011010001100110110010011000101100011111100111000111100010111001111001001101010110001011000111110111
0111
ruchitha@G5-5587:~/crc$ python2 s.py
String to be sent:hasvghdshvcdjscbjhvjnkjfnjkfgbsdcjsuihfiudhvjdfjvbhdfvbhjdbhfvbvhjfbvhjbfjbhjjbhhbjhfhjbvhbdjfhfdbcjhsbjhfbvbhfvbfskjckhjfsbhjfbhvfbhvjhfbhjfvbjfdvbjhfdvbjhfbvhbvfhjbffhjvyufbgjhbfvhhjbv
fvhbfdfjhvbdfhbdfvbhjfbvjdfhbhdbvhjbhdfhvfbdfmkvndfnvdjfnvkjdfnfd
('The length of the string is', 1757)
('The key is', '10011')
Error in data
ruchitha@G5-5587:~/crc$ python2 s.py
String to be sent:sytdfcyudsbvihidfuhvuodhfbhdsuygcsvdubfiduby7grvbfyuhbfsvbuhsbvusdcbsracsfvxhgavchbdchjbdsiycbuydshbcjhscksncikbyudbuhsbchfhjfbjsbhfdbcycbsdjbcnuydsbuyhybfhussdvgvchgsdvghdschgcsdjbchjbj
dshgvgdshvhsvcdsvhsvghcsdyctsvdcjbjbhscjsvdyhybcddhskfggshcrgsjhguferhgbvkjrdnvkjdsvbjubdvjkdnvkjnfdvhjdfvbjnfdjknvkjdfbvferihvuiehnveiufnvkjefdnkjnvjhbdxbedhbcwhcbjhbsdhcjdbsdbcjksdnkjfdbjdksjhbjhbhj
('The length of the string is', 2701)
('The key is', '10011')
Error in data
ruchitha@G5-5587:~/crc$
```

CRC 4: Key: 10110 does not detect most of the errors whereas for the same strings key 10011 detects errors in data

CRC4, when key is 10011 error is detected, but when key is 10100 no error was found

```
sathya@sathya-Inspiron-5559:~/cn2$ python2 client_encoder.py 10011 1118
String to be sent:q3efhdbwoufgywoeufygruvcasbdhcbdukfvalhdvbcWUDHIQUWEDCBWUDHVBDSJKBCaudsfvaeufbvceijvcbauhvfbaljvbhljvhbahdfkvagbdlfjvbha sdlj
vbhlashvbdasldhvjbarhrgrthy
('The length of the string is', 1062)
('The key is', '10011')
11100011100111100101110011011010001100100110001011101111101111111010111001101100111111100111101111101111110010111101011100110111100111001111110
01011101011110110110001111000011110011110001011001001101000110001111000101100100111010111010111100110111011011000011101100110100011001001110110
11000101100011101011110101011000100100100010010011010001101010110101111000101100010010000111000010101011110101011000100100100010101101000010100
01001010011100101010010111000010100001111000011110101110010011100111001101110110110000011100101111010111001101100010111011011000111100101110100
11101010111011011000111100010110000111101011101000111011011001101100010110000111011001101010111011011000101101000110110011010101110110100011
00010110000111010001100100110011011010111110110110000111001111100010110010011011001100110110101011101101100010110100011000011000001110011110010
01101100110101011101101100010110100011011001100001111001111010001110110110001011001001100001111001111011001100100110100011101101101010110001011
00001111001011010001110010110011111100101110100110100011110010000
Error found, Retransmit data and remainder is 0011
sathya@sathya-Inspiron-5559:~/cn2$ python2 client_encoder.py 10100 1119
String to be sent:q3efhdbwoufgywoeufygruvcasbdhcbdukfvalhdvbcWUDHIQUWEDCBWUDHVBDSJKBCaudsfvaeufbvceijvcbauhvfbaljvbhljvhbahdfkvagbdlfjvbha sdlj
vbhlashvbdasldhvjbarhrgrthy
('The length of the string is', 1062)
('The key is', '10100')
11100011100111100101110011011010001100100110001011101111101111111010111001101100111111100111101111101111110010111101011100110111100111001111110
01011101011110110110001111000011110011110001011001001101000110001111000101100100111010111010111100110111011011000011101100110100011001001110110
11000101100011101011110101011000100100100010010011010001101010110101111000101100010010000111000010101011110101011000100100100010101101000010100
01001010011100101010010111000010100001111000011110101110010011100111001101110110110000011100101111010111001101100010111011011000111100101110100
11101010111011011000111100010110000111101011101000111011011001101100010110000111011001101010111011011000101101000110110011010101110110100011
00010110000111010001100100110011011010111110110110000111001111100010110010011011001100110110101011101101100010110100011000011000001110011110010
01101100110101011101101100010110100011011001100001111001111010001110110110001011001001100001111001111011001100100110100011101101101010110001011
00001111001011010001110010110011111100101110100110100011110010000
Thankyou, DATA RECEIVED,remainder is 0000
sathya@sathya-Inspiron-5559:~/cn2$
```

CRC4, when key is 10011 error is detected, but when key is 10010 no error was found

Commonly observed results

Performance

Given that g(x) is at least not divisible by X and is not divisible by (X+1), it's guaranteed all burst errors of a length equal to the degree of polynomial are detected and all burst errors affecting an odd number of bits are detected.

- CRC can detect all burst errors of less than the degree of the polynomial
- CRC detects most of the larger burst errors with a high probability.

Usual Performance of standard CRC16 vs other 16 bit polynomials

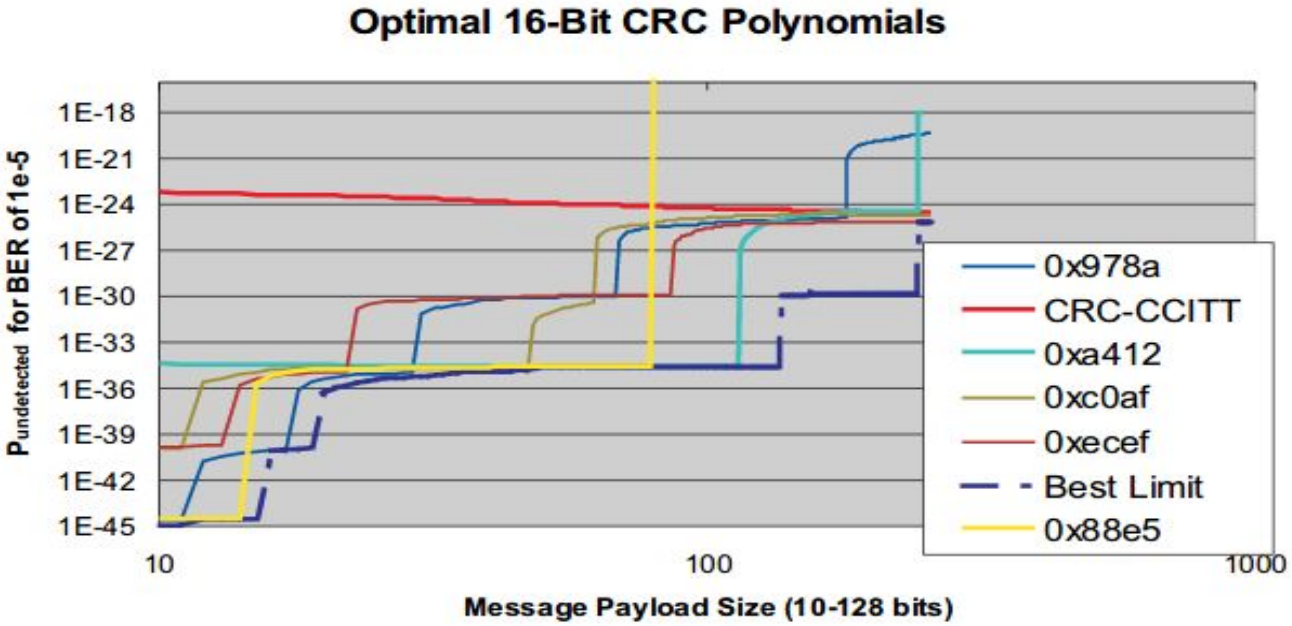


Figure 2. Optimal 16 bit CRC codes for message lengths 10 bits - 128 bits

As the length of bits in the message transmitted increase, probability of undetected errors increase for all 16 bit generator polynomials except for the standard CRC-CCIT generator polynomial.

This is because of unique properties of CRC-CCIT.

Thank you.