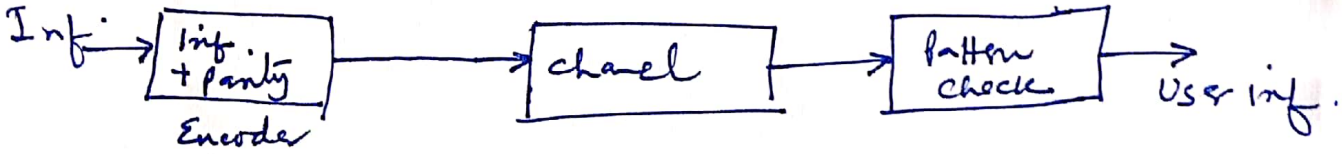
Detection and Correction of Errors.

Detection: To see if an error has occured. The central concept of error detection and subsequent correction is ~~redundancy~~ based on addition of redundant bits.

Simplest of error detection involves addition of Parity bit (as redundant bit)

~~Suppose~~ As an example, suppose we have to send k-bits of information. A single check bit (Parity bit) is added to form code word. The parity check determines on the basis of pre-decided even or odd parity; the presence or absence of error.



Parity ~~code~~ bit is calculated as modulo Sum of information bits

e.g. $b_{k+1} = b_1 + b_2 + \cdots + b_k$ modulo 2

$b_1, b_2 \cdots b_k$ are information bits.

This is an example of linear code

This rule assigns parity bit a value that will produce a code word that contains [e.g. 0 for even no, 1 for odd no.] for even no. of 1's [Case of even parity]

If during transmission code word undergoes a single error then corresponding code word contains [1 for even no]

This method detects Single bit error

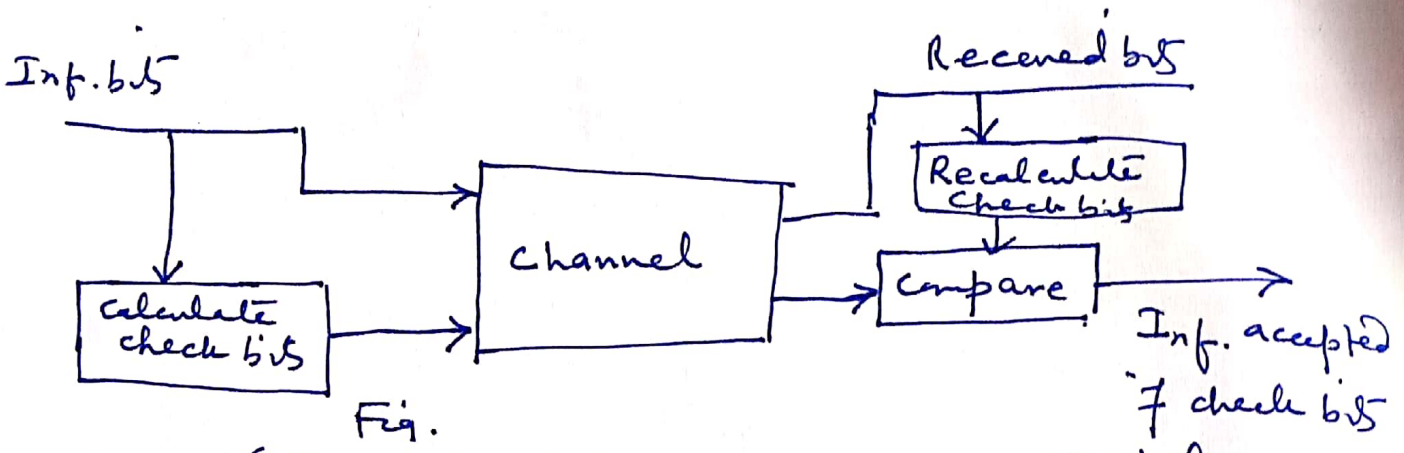


Fig.
(Error detection using check bits)

Observation 1: Error detection requires redundant bit(s) i.e. Inf. xmitted is over & above the reqd. minimum

eg. For Parity: If for (k+1) bits no. of redundant bits = 1
for k bits " " $\frac{k}{k+1}$

Observation 2: An error detection technique will fail to detect tr. errors that convert one valid code word into another valid code word.

eg. In Single Parity check code word (using even parity) even no. of transmission errors will always convert one valid code word to other valid code.

∴ The techniques operate on following principle:
For a given frame of bits, additional bits that constitute an error detecting code are added by transmitter. The code is calculated as a function of xmitted bits.

The receiver performs same function and compares the results. A detected error occurs only if there is a mismatch.

Error Correction:

Once the error has been detected, correction is necessary to get the exact message.

In error correction, we need to know exact no. of bits that are corrupted and their location in message. If we need one single error in eight bit data, we need 8 possible error locations; if we need to correct 2-errors in data unit of same size, we need to consider 28 possibilities. For more data with more errors, difficulty in receiver is complex.

To overcome this difficulty/complexity; there are two approaches for error correction.

Forward Error Correction (FEC)

FEC is detection of errors, followed by further processing of received information that attempts to correct the errors

FEC is used in Satellite communication & Deep Space Communication. Here provision of return channel is expensive, therefore, not available. Accordingly, no retransmission request acceptable.

FEC may be used for audio CD recording where FEC provides robustness to errors to facilitate reproduction of clear sound in presence of smudges & scratches of disk surface.

It may be pointed out that FEC is possible when no. of errors is small.

4-DC.

Retransmission or (ARQ)

Correction by retransmission is a technique in which receiver detects occurance of an error and asks (Req45) sender to resend the message. Resending is repeated untill a message arrives that the receiver believes is error free. This technique is also called Automatic Repeat Request (ARQ).

Forward Error detection & correction

Hamming code:

Hamming code allows detection of upto 2 errors. Single bit can be corrected. Although there are some Hamming codes that can correct more than one error, here we focus on single bit error correcting code.

Hamming Distance: One of the central concepts in coding for error control (Error Control: Detect & Then Correct) is idea of Hamming distance. The Hamming distance between two words (of same size) is no. of differences between the corresponding bits.

e.g. Hamming distance between:

000, 011 represented as d(000, 011) is 2

Similarly d(10101, 11110) is 3. Thus we can find this by XOR operation between two words also, e.g.,

XOR(000, 011) = 2 1's.

Minimum Hamming distance:

Smallest distance between all possible pairs

eg. for two words

d(00000, 01011) = 3 d(00000, 11110) = 4

∴ Min Hamming distance dmin = 3.

It may be mentioned, any coding scheme needs to have at least three parameters: Code word size = n data word size = k, and Min. Hamming distance dmin.

It is stated

To guarantee the detection upto s errors, in all cases, the Hamming distance in block code must be $d_{min} \geq s+1$; ∴ [For 2 error detection $d_{min} = 3$.]

used to all forward

Hamming codes were orginally designed for $d_{min} = 3$ That means they can detect upto 2 errors. Some Hamming codes can detect more than 2-errors. However, we limit the present Studis upto detection of 2-errors.

In Hamming code scheme, the bit positions for Data bits (D) and Hamming bits/parity bits (P) are as under:

1	2	3	4	5	6	7	8	9	
P_1	P_2	D	P_4	D	D	D	P_8	D	.....

Each data bit (D) is checked by no. of parity bits/Hamg bits

~~Data bit is checked by no.~~

eg. A data bit in position 6 is checked by P_4 & P_2 ($6 = 2^2 + 2^1$)... so on is given in Table

Even-odd parity can be used

~~If even parity is used~~

~~Illustration:~~

Data bit Position	Parity bit			
	P_1	P_2	P_4	P_8
3	x	x		
5	x			x
6			x	
7	x		x	
9	x			x
10		x		x
11	x			
12			x	x

Table:

7-Dec

Illustration:

Q1 Generate code word for 'k' = 1001011; Assume even Parity

1	2	3	4	5	6	7	8	9	10	11
P_1	P_2	1	P_4	0	0	1	P_8	0	1	1

First Parity bit P_1: ✓ ⇒ From table 3, 5, 7, 9, 11 — 1 0 ✗ 1 0 1 — ∵ $P_1 = 1$ ∵ No. of 1's should even but they are odd.

2nd Parity bit P_2: From Table 3, 6, 7, 10, 11 — 1 0 1 1 1 — ∵ No 7 1's even. $P_2 = 0$

3rd Parity bit P_4: 0 0 1 1 1 — $P_4 = 1$

P_8: 0 1 1 0 — $P_8 = 0$

∴ Hamming code : 10110010011. (Hamming code with even Parity.

Q2 Detect & Correct the Single error in following Hamming code 10110010111 (Assume even Parity.

1	2	3	4	5	6	7	8	9	10	11	
1	0	1	1	0	0	1	0	1	1	1	
First check for 3,5,7,9,11	1		1		0		1		1		odd no. of 1's ∴ Fail.
Second check 3,6,7,10,11		0	1			0	1			1	Even no. of 1's ∴ Pass.
3rd check.				1	0	0	1				Pass.
4th check								0	1	1	1 Fail

P_1 & P_8 indicate error ∴ 9th bit position in error

∴ Correct code word : 10110010011

Internet checksum

8-Dec

Several Protocols (TCP, IP, UDP) use check bits to detect Errors. With IP checksum is calculated for the contents of Header and included in ~~Specific~~ Special field. The checksum must be recalculated at every router. The checksum has been selected because of its ease of implementation in S.W. rather than H.W.

The algorithm assumes that header consists of certain number L of 16 bit word:

$b_0, b_1, b_2, \ldots b_{L-1}$

16 bit, 16 bit, ... 16 bit

The 16-bit checksum bk corresponds to parity bit and is calculated as by the following algorithm:

1. Each 16 bit word is treated as integer and L words are added modulo $2^{16}-1$
2. checksum then consists of negative inverse of x:
   $b_L = -x$
3. The checksum b_L is then treated as dedicated field of header

The contents of all headers including checksums field must then Satisfy following pattern:

$$0 = b_0 + b_1 + b_3 + \ldots + b_{L-1} + b_L \text{ modulo } 2^{16}-1$$

Each router can then check for the errors in the header by calculating the preceding equation for each header received.

9-Dec

Example:

Suppose we use 4-bit words (instead 2 16 bit)

b_0 b_1 b_2 b_3

4 bit 4 bit 4 bit 4 bit

These are added modulo $2^4 - 1 = 15$

Likewise:

If we have 2 - 4 bit words

say 1100 & 1010

We capite their decimal which is 12 & 10

Addition fields 12 + 10 = 22

capite 22 modulo $2^4 - 1$

22 modulo 15 = 7

The no which added to 7 and then capuring modulo 15 should equate to zero.

(7 + (8)) modulo 15 = 0

↓

The no if added to 7 ~~does~~ with modulo is 0

∴ (8) is ~~complete inverse of~~ checksum

convert 8 to 4 bit = $\boxed{1000}$ checksum.

And note: (22 + 8) modulo 15 = 0 [Also].

Cyclic Redundancy Check (CRC):

One of the most common & most powerful, error detection codes is CRC which can be described as:

Given a k bit block of bits
Transmitter generate n bit sequence (for error detection) as frame check sequence (FCS), so that resulting frame, consisting of (k+n) bits is divisible by some pre-determined number. (What number ?)
Receiver then divides the incomming frame by the same number & if there ~~are~~ is no remainder, assumes that was no error.

~~Illustrate~~ The procedure can be presented in three ways:

(a) Modulo 2 Arithmetic (uses binary addition with no carries i.e Exclusive OR operation)
(b) Polynomials
(c) Digital logic.

Modulo 2 Arithmetic:

$$\begin{array}{r} 1111 \\ +\,1010 \\ \hline 0101 \end{array} \qquad \begin{array}{r} 1111 \\ -\,0101 \\ \hline 1010 \end{array} \qquad \begin{array}{r} 11001 \\ \times\ 11 \\ \hline 11001 \\ 110010 \\ \hline 101011 \end{array}$$

$$2^2(1011)$$

$$1\times2^3+0\times2^2+1\times2^1+1\times2^0$$

$$1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+0\times2^0$$

Now define:

✓ T = (k+n) – bit frame to be transmitted
↑
M = k bit message, first k bits of T
F = ~~last~~ n-bit FCS, last n bits of T ← Appended fm L to R
P = Pattern of (n+1) bits; predetermined divisor

$$T = 2^n M + F$$ ✓ ✓

(By multiplying M by 2^n, we have an effect of shifting M by n bits and padded out result with zeros)

Adding F yields the concatination of M & F which is T

Now $\frac{2^n M}{P} = Q + \frac{R}{P}$

∵ Division is modulo 2, the remainder (R) is shorter than the divisor.

Using this remainder as FCS (∵ FCS $n < k$)

Then $T = 2^n M + R$ ← use as frame check sequence

$\frac{T}{P} = \left(Q + \frac{R}{P}\right) + \frac{R}{P}$

$\frac{T}{P} = Q + \frac{R}{P} + \frac{R}{P}$

$\frac{T}{P} = Q + \frac{R+R}{P} = Q + \frac{0}{P}$ ∵ $R+R=0$ (Binary add. without carry)

or $\frac{T}{P} = Q$. ✓

∵ There is no remainder, therefore, FCS is easily generated by using R as ~~remainder~~, FCS.

∴ To find remainder: Divide $2^n M/P$ (This is FCS) (R).

(At xmitter)

At receiver: Divide T/P

For finding $T = 2^n M +$ Remainder (R)

The pattern P is chosen to be 1 bit longer than the desired FCS, the exact bit pattern depends on type of errors expected. At minimum both higher & lower order bits must be 1.

12-Dec

Message M = 1010001101 (10 bits)

Pattern P = 110101 (6 bits)

FCS R = To be calculated (5 bits)

$2^n M = 2^5 (1010001101) = 101000110100000$

```
                1101011011  ← Q.
        ┌──────────────────
P → 110101 ) 101000110100000
             110101
             ------
              111011
              110101
              ------
                11100
                11010
                -----
                 11110
                 11010
                 -----
                  10100
                  11010
                  -----
                   11100
                   11010
                   -----
                    11110  ← R
```

To find T

$T = 2^n M + \text{Remainder}$

$2^n M = 101000110100000$

R. 1110

T = 101000110101110 → This T is Transmitted

13-Dec

```
                1101010110
P → 110101 | 101000 11010 1110
             110101
             -------
              11101 1
              11010 1
              -------
               1110 10
               1101 01
               -------
                111 110
                110 101
                -------
                 101 111
                 110 101
                 -------
                  110 101
                  110 101
                  -------
                  000 000  ← R
```

If the receiver will fail to detect an error if Tr is divisible by P.
E is divisible by P.
This is an unlikely occurrence

∵ There is no R, ∴ There are no errors (assumption)

* There is a concise method for specifying one or more errors. An error results in reversal of a bit. This is equivalent to taking XOR of bit and 1 (modulo 2 addition of 1 to bit). Thus the errors in an (n+k) bit frame can be represented by an (n+k) bit field with 1's in each error position

∴ $T_r = T \oplus E$

T_r is received frame
E = Error pattern with 1's in position of errors
Tr = Received frame

Polynomials:

In polynomial viewing of CRC we express all value in terms of a polynomial of dummy variable

eg. $M = 110011$

$M(x) = x^5 + x^4 + x + 1$

$P = 11001$

$P(x) = x^4 + x^3 + 1$

The CRC process can now be described as

1. $$\frac{x^n M(x)}{P(x)} = Q(x) + \frac{R(x)}{P(x)}$$

1010 → 1011 (1)
1011
0001

2. $$T(x) = x^n M(x) + R(x)$$

An error $E(x)$ will be undetectable if divisible by $P(x)$.

Following can be shown that following errors are not divisible by $P(x)$ & hence detectable

- All single bit errors
- All double bit errors, as long as $P(x)$ has at least 3 1's.
- Any odd no. of errors as long as $P(x)$ contains a factor $(x+1)$
- Any burst error for which length of burst is less than length of divisor polynomial; that is less than or equal to FCS.
- Most large burst errors.

Four versions of $P(x)$ usually used

CRC-12 = $x^{12} + x^{11} + x^3 + x^2 + x + 1$

CRC-16 = $x^{16} + x^{15} + x^2 + 1$

CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$

$$CRC\text{-}32 = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

15-Dec

It can be shown that if error patterns are equally likely, then for a burst error of length $(r+1)$

The probability of undetected error $= \frac{E(x)}{P(x)} = \frac{1}{2^{r-1}}$ $(r+1)$

For longer bursts

$$\text{Probability} = \frac{1}{2^r} \quad (r = \text{length of FCS})$$

For eros

Ref: Ramabadran T Gaitonde, S "A Tutorial on CRC computations" IEEE Micro, August 88