UNIT IV

Error Detection and Correction.

/ /

what is an error?

- Corruption of data in transission is error.
- Error may be caused by Noise, human errors, natural disasters, etc.

Types

- Single bit error — a single bit is changed during transmission
- Multi-bit error — lots of bits are incorrectly received.
- Burst Error — 2 or more bits in data unit have changed. in 8 bits burst.

Redundant bits

To detect and correct data sender will add some extra vendundant bits to the data.

Redundancy achieved through coding (block coding).

receiver can detect error given that
1. receiver has a list of valid codewords.
2. Original codeword has changed to an invalid one.

what is burst error?

/ /

Datawords : • k-bits long blocks of the original message.

• we add r redundant bits.

codeword → n length '$n = k + r$'

e.g

Datawords	Codewords
00	000
01	011
10	101
11	110

Hamming distance :

• Hamming distance between two words (of same size) is number of difference between the bits.

e.g. (000, 011)

0 0 0
0 1 1
→ × ✓ ✓ 2✓ → 2 difference, 2 hamming distance

If s-error occurred during transmission, the Hamming distance is s.

• Min hamming distance should be 1 more than the size of codeword.
if we don't do this +1, it might be mistaken for data bit.

/ /

Parity bit - a bit added to make parity (1's occurance even).

- appended with dataword → forms codeword.

Parity Check Decoder

DW → 1011

CW → 10111

r_0 — Parity bit

Cases :

1. No error . 10111 received → 1+0+1+1+1 % 2 → 0 (Syndrome bit.)

   1011 extracted .

2. 1 bit changes at a_1 10011 → 1+0+0+1+1 % 2 → 1 ← Syndrome bit

3. 1 bit changes at r_0 , 10110 → 1+0+1+1+0 % 2 → 1 ← Syndrome bit

   2,3 → dataword is discared.

4. Even number of errors cannot be detected, as syndrome will be 0 .

5. Three bits are changed, syndrome will be 1 , any odd number of errors can be found here. (works for odd errors).

Cycling Redundancy Check (CRC)

DW → k bits

CW → n bits

a divisor of size 'n-k+1' is used

while in generator

DW → 1001 000

```
          1010
1011 ) 1011000
      -1011
       010000
      -0000
        10000
       -1010
        00100
```

? Check this out again!

Checksum:

• Message divided into m-bits

e.g.

received checksum, → take 1's compliment → add 1's compliment with data words → if checksum is more than 'm' bits, divide into m bits and add them together → take 1's compli again → Syndrome = 0000 not corrupted.

Algo:

1. Split data in 'm' bits.
2. Convert bits (of size m) to decimals and add them. (Converting is optional).
3. Take compliment of addition (on sender side itself.)
4. (On receiver) add m bit data words with checksum value.
5. Take 1's compliment of result.
   a. if result = 0 (Syndrome) no corruptions.
   b. else data corrupted.

Data = 01110011, $\underline{m = 4}$ ↳ block of 4 bits

$\Rightarrow \underbrace{0+1+1+1}_{7} + \underbrace{0+0+1+1}_{3}$

7 + 3

$\Rightarrow (10)_{10} \Rightarrow (1010)_2$

1's compliment $\Rightarrow (0101)_2$

⇓ receiver

$\underbrace{0+1+1+1}_{7} + \underbrace{0+0+1+1}_{3} + \underbrace{0+1+0+1}_{5}$

7 + 3 + 5

$(15)_{10} \rightarrow (1111)_2$

1's compliment $\Rightarrow (0000)_2$

∴ Syndrome = 0000. ∴ No errors/data corruptions.

Internet Checksum

- Traditionally 16-bit checksum.

8-bit fletcher:

- data octets (bytes)
-

Adler Checksum

→32-bit checksum.

→3 differences

Forward Error Correction

- Hamming Distance
- XORing
- Chunk interleaving
- Combining Hamming distance & interleaving
- Compounding High and Low resolution packets.