UTM

UNIVERSITI TEKNOLOGI MALAYSIA

School of Computing, Faculty of Engineering

Universiti Teknologi Malaysia (UTM)

Error Correction

NETWORK COMMUNICATION

SEM1_2021

PROJECT

SUBJECT CODE: SECR 1213-02

Prepared by:

Lee Sze Yuan (A19EC0068)

Submitted to: Dr. Syed Hamid Hussain Madni

Introduction

After conducting some research on the topic “How to do Error Correction after detecting Error” given by my lecturer, Dr Syed Hamid Hussain Madni, I will share my sharing and understanding on the topic in this 3 pages report.

First of all, there is 4 methods of error detection (which we have learnt in our class)

1. Single Parity Check
2. 2D Parity Check
3. Checksum
4. Cyclic Redundancy Check (CRC)

All 4 methods mentioned above used different ways and logics to perform error detection. The errors that can be detected include single-bit error, multiple-bits error and burst error.

However, based on my study and research, I found that all these methods can only detect errors. The drawback of the above method is that either it can only tell whether the data have error, but cannot tell which bits are in error. (except for the 2D bit parity). Or will fail to detect error at certain conditions. Besides, most of them cannot perform error correction. Especially, CRC. CRC can only detect errors but not correcting errors

So, I will explain the 2 common error correction methods I learnt.

Error Correction

Backward Error Correction

The working principle of this method is very simple. When the receiver detecs an error in the data it received (through any of the 4 methods mentioned), It will request the sender to transmit the same data again. This process repeats until there is no error detected in the received data.

This method is simple, but it is only efficient when the cost of retransmission of data is low. If the cost is too high, the 2nd method will be applied.

Forward Error Correction

To correct the error in the data unit, the receiver must know which bit in the frame is in error. To locate the bit in error, redundant bits are used as parity bits for error detection.

To know how many redundant bits(r) to be used, we need the Algorithm of Hamming code. What is this algorithm about? I think I will just use an example here for the following explanation.

Let say, we have a data 1010

To calculate the number of redundant bit, we use this formula $2r >= d+r+1$

Total number of data bits 'd' = 4

Number of redundant bits r : $2r >= d+r+1$
$2r >= 4+r+1$

Therefore, the value of r is 3 that satisfies the above relation.

Total number of bits = d+r = 4+3 = 7;

The number of redundant bits is 3. The three bits are represented by r1, r2, r4. The position of the redundant bits is calculated which corresponds to the raised power of 2. Therefore, their corresponding positions are 1, 2, 4.

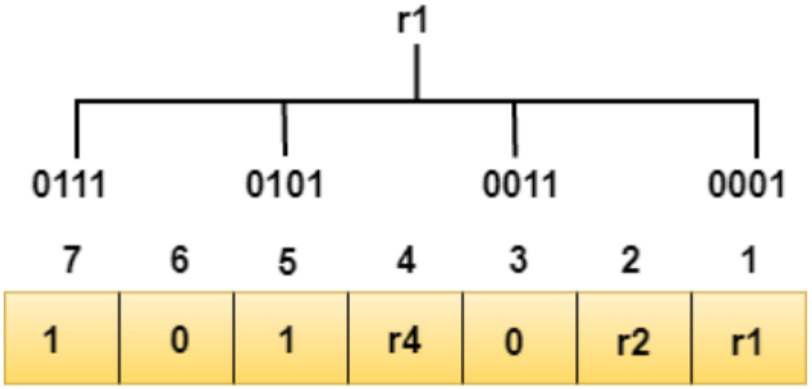
Sender

Sender will have determine the value of r1, r2 and r3 bits first

1	0	1	r4	0	r2	r1

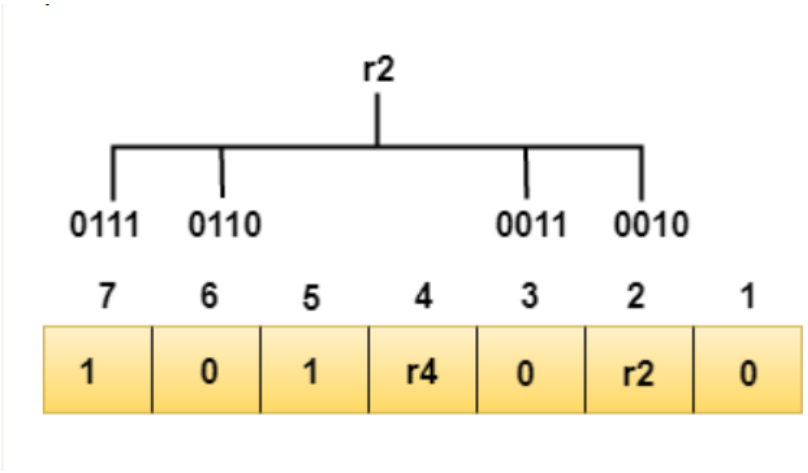
Determining the r1 bit

The r1 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the first position.

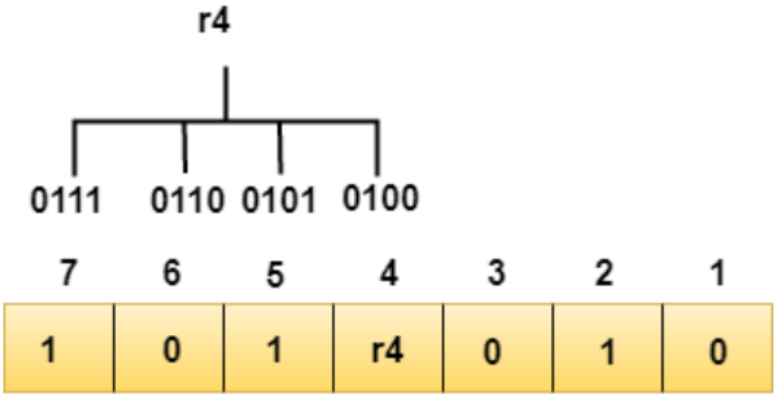


We observe from the above figure that the bit positions that include 1 in the first position(in binary number) are 1, 3, 5, 7. Now, we perform the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r1 is even, therefore, the value of the r1 bit is 0.

We repeat the above steps for r2 and r3 bits



bit positions that include 1 in the second position are 2, 3, 6, 7. After performing the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r2 is odd, therefore, the value of the r2 bit is 1.



bit positions that include 1 in the third position are 4, 5, 6, 7. With even-parity check at these bit positions, we know that the total number of 1 at these bit positions corresponding to r4 is even, therefore, the value of the r4 bit is 0.

So, the data will be transfer to the receiver is

1	0	1	0	0	1	0

Receiver

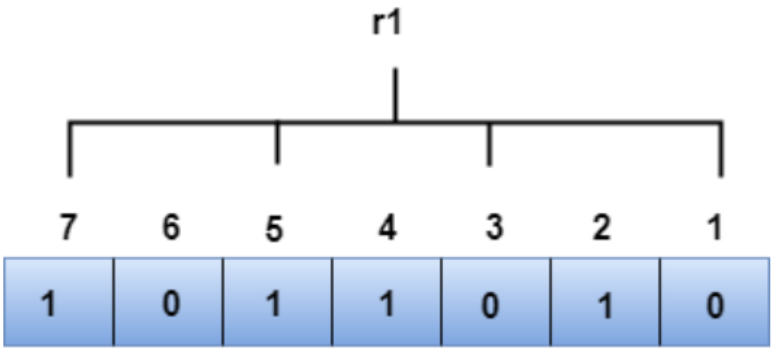
Let say it receive this inead of teh one sender sends (we purposely make it to have error)

1	0	1	1	0	1	0

Receiver now need to check even-parity on R1, R2 and R4 bits

R1 bit

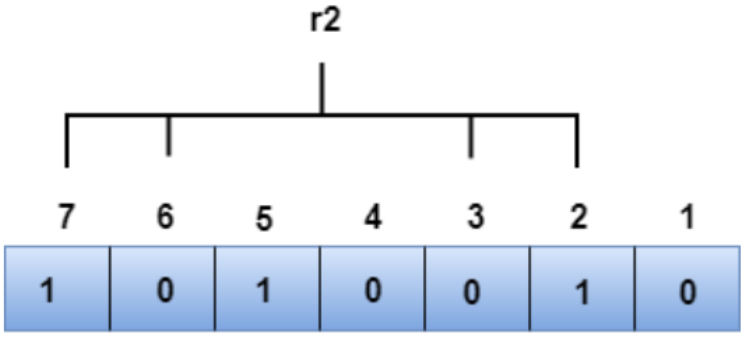
The bit positions of the r1 bit are 1,3,5,7



the binary representation of r1 is 1100. Now, we perform the even-parity check, the total number of 1s appearing in the r1 bit is an even number. Therefore, the value of r1 is 0.

R2 bit

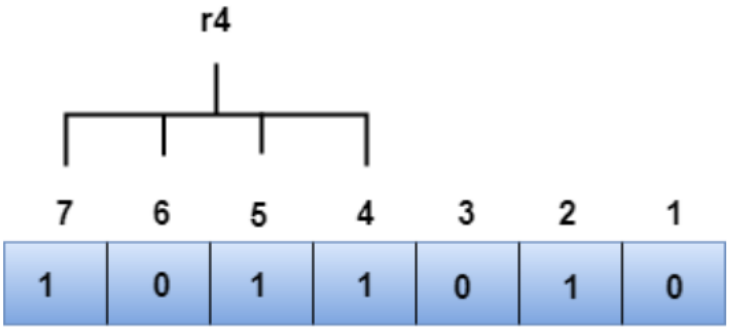
The bit positions of r2 bit are 2,3,6,7.



the total number of 1s appearing in the r2 bit is an even number. Therefore, the value of r2 is 0.

R4 bit

The bit positions of r4 bit are 4,5,6,7.



the total number of 1s appearing in the r4 bit is an odd number. Therefore, the value of r4 is 1.

So, from here we get the value of R1, R2 and R4

The binary representation of redundant bits, i.e., r4r2r1 is 100, and its corresponding decimal value is 4. Therefore, we know the error occurs in a 4th bit position. The bit value must be changed from 1 to 0 to correct the error.

Bibliography

java T point. (n.d.). *Error Correction*. java T point. Retrieved January 21, 2021, from https://www.javatpoint.com/computer-network-error-correction#