Tittle: To demonstrate error detection and correction using Hamming Codes or CRC

Objectives : To implement error detection and correction techniques

Problem Statement: Write a program for error detection and correction for 7/8 bits ASCII codes using Hamming Codes or CRC. Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.

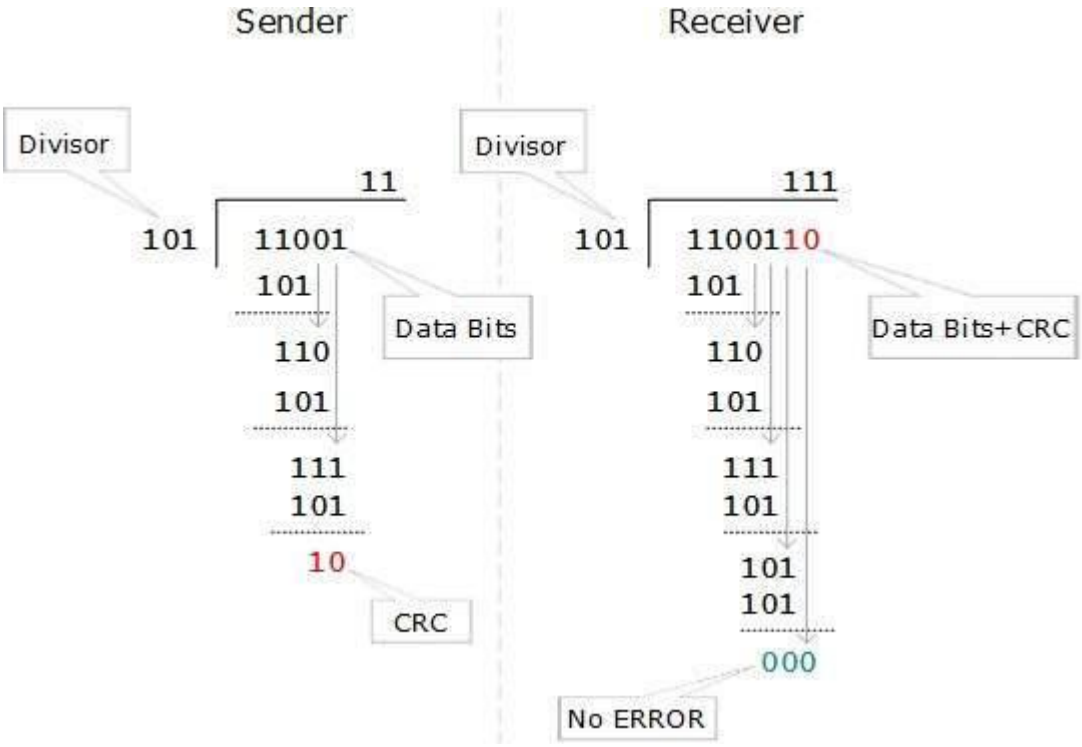
Outcome: Demonstrate Hamming Codes or CRC with example.

Software Requirements: C and wireshark

Operating System: Open source (Ubuntu)

THEORY:
Cyclic Redundancy Check: CRC

CRC is a different approach to detect if the received frame contains valid data. This technique involves binary division of the data bits being sent. The divisor is generated using polynomials. The sender performs a division operation on the bits being sent and calculates the remainder. Before sending the actual bits, the sender adds the remainder at the end of the actual bits. Actual data bits plus the remainder is called a codeword. The sender transmits data bits as codewords.



At the other end, the receiver performs division operation on codewords using the same CRC divisor. If the remainder contains all zeros the data bits are accepted, otherwise it is considered as there some data corruption occurred in transit.

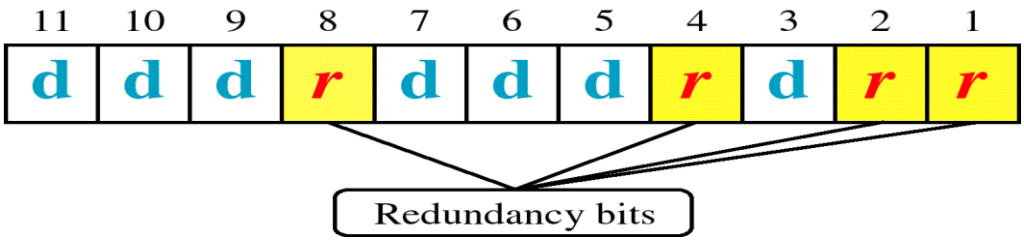
Hamming code

- **Hamming codes are a family of linear error-correcting codes that generalize the Hamming(7,4)-code**
- **Invented by Richard Hamming in 1950**

Hamming codes can detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors.

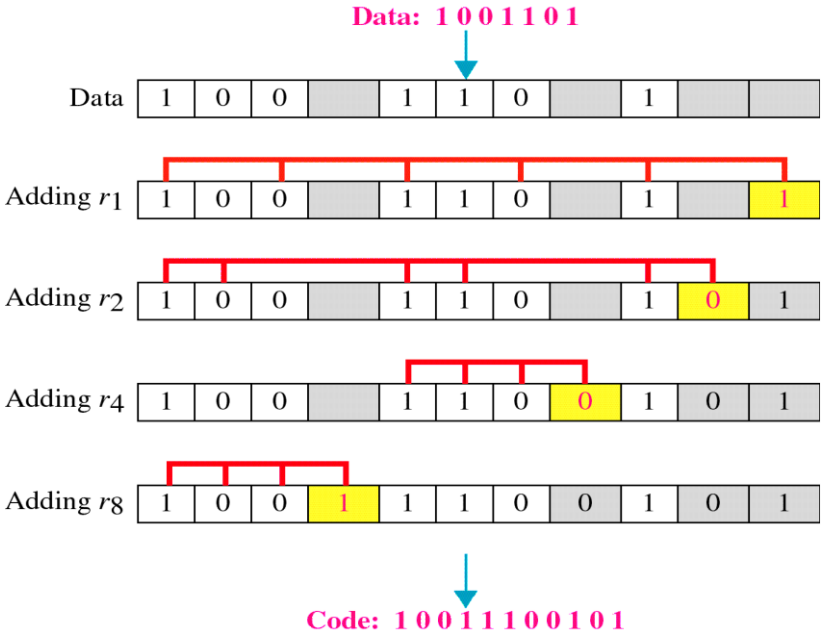
General algorithm

- **The following general algorithm generates a single-error correcting (SEC) code for any number of bits.**
- **Number the bits starting from 1: bit 1, 2, 3, 4, 5, etc.**
- **Write the bit numbers in binary: 1, 10, 11, 100, 101, etc.**
- **All bit positions that are powers of two (have only one 1 bit in the binary form of their position) are parity bits: 1, 2, 4, 8, etc. (1, 10, 100, 1000)**
- **All other bit positions, with two or more 1 bits in the binary form of their position, are data bits.**
- **Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.**
- **Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.**
    - **Parity bit 1 covers all bit positions which have the least significant bit set: bit 1 (the parity bit itself), 3, 5, 7, 9, etc.**
    - **Parity bit 2 covers all bit positions which have the second least significant bit set: bit 2 (the parity bit itself), 3, 6, 7, 10, 11, etc.**
    - **Parity bit 4 covers all bit positions which have the third least significant bit set: bits 4–7, 12–15, 20–23, etc.**
    - **Parity bit 8 covers all bit positions which have the fourth least significant bit set: bits 8–15, 24–31, 40–47, etc.**
    - **In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.**



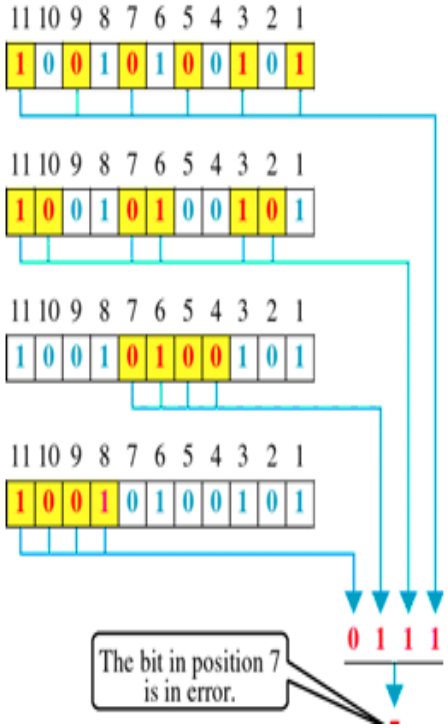
Example

Error detection



Error correction

ERROR DETECTION



Conclusion: Hence we have implemented CRC and Hamming code.