Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology, Pune-37

(An autonomous Institute of Savitribai Phule Pune University)

Department of Computer Engineering

Division	**TY-CS-D**
Batch	**2**
GR-no	**12420072**
Roll no	**60**
Name	**Anurag Shankar Nimkande**

Assignment No. 3

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

Steps for Error Detection & Correction (Hamming + CRC) with Wireshark

1. **Choose ASCII Data**
    - Take 7/8-bit ASCII characters as input for transmission.
2. **Apply Hamming Code (Error Correction)**
    - Encode each ASCII character into Hamming(12,8).
    - Add parity bits for error detection and single-bit error correction.
3. **Apply CRC (Error Detection)**
    - Append CRC bits (using a generator polynomial) to detect burst errors.
4. **Transmit Data Peer-to-Peer**
    - Use socket programming (TCP/UDP) to send encoded data between two systems (or two terminals on same machine).
5. **Capture Packets in Wireshark**
    - Start Wireshark capture.
    - Use a display filter like ip.addr==<peer_IP> or tcp.port==<port>.
    - Observe the transmitted ASCII + Hamming + CRC encoded packets.
6. **Introduce Transmission Errors**
    - Simulate bit flips in transmitted packets.
    - Observe Hamming correction (1-bit) and CRC detection (multi-bit).
7. **Verify at Receiver**
    - Receiver decodes Hamming, corrects errors.
    - CRC check ensures integrity.
    - Recovered ASCII text should match original.

Program Code (CRC):

```
import java.util.Scanner;

public class CRC {

	public static String crcEncode(String data, String divisor)

	{

		StringBuffer codeword = new StringBuffer(data).append("000");

		String parity = Integer.toBinaryString(Integer.parseInt(codeword.toString(),2) % Integer.parseInt(divisor,2));

		System.out.println("Bits to Append: "+String.format("%3s", parity).replace(' ', '0'));

		codeword.replace(codeword.length()-3, codeword.length(), String.format("%3s", parity).replace(' ', '0'));

		return codeword.toString();

	}


	public static String xorOp(String str1, String str2)

	{

		return (String.format("%4s", Integer.toBinaryString(Integer.parseInt(str1,2) ^ Integer.parseInt(str2,2))).replace(' ', '0'));

	}


	public static String crcDecode(String data, String divisor)

	{

		StringBuffer check = new StringBuffer(data);

		for(int i=0;i<4;i++)

		{

			if(check.charAt(i) == '1')

			{

				check.replace(i, i+4, xorOp(check.substring(i, i+4),divisor));

			}
```

```
                else
                {
                        check.replace(i, i+4, xorOp(check.substring(i, i+4),"0000"));
                }
        }

        return check.toString().contains("1") ? "Error Occured : Data Mismatch..." : "Data Properly Transmitted...";
    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("ENTER DATA:");
        String data = sc.nextLine();
        System.out.println("ENTER DIVISOR:");
        String div = sc.nextLine();

        String codeword = crcEncode(data,div);
        System.out.println("Codeword = "+codeword);

        System.out.println("ENTER THE MISMATCH OR CORRECT STRING:");
        String status = crcDecode(sc.nextLine(),div);
        System.out.println("Receiver Status = "+status);
        sc.close();
    }
}
```

Output:

ENTER DATA:

1010

ENTER DIVISOR:

1011

Bits to Append: 011

Codeword = 1010011

ENTER THE MISMATCH OR CORRECT STRING:

1010111

Receiver Status = Error Occured : Data Mismatch...

Program Code (Hamming Code):

```
import java.util.*;

public class Hamming {

	public static int[] encode(int data[])
	{
		int code[] = new int[2*data.length-1];

		code[2] = data[0];
	code[4] = data[1];
	code[5] = data[2];
	code[6] = data[3];

	code[0] = code[2] ^ code[4] ^ code[6];
	code[1] = code[2] ^ code[5] ^ code[6];
	code[3] = code[4] ^ code[4] ^ code[6];

		return code;
	}

	public static void decode(int data[])
	{
		int p1 = data[0] ^ data[2] ^ data[4] ^ data[6];
	int p2 = data[1] ^ data[2] ^ data[5] ^ data[6];
	int p4 = data[3] ^ data[4] ^ data[5] ^ data[6];

	int syn = p4*4 + p2*2 + p1*1;
```

```
        if(syn == 0)
            System.out.println("No Error Detected....");
        else
        {
            System.out.println("Wrong Word : "+Arrays.toString(data));
            System.out.println("Error Was Detected at pos:"+syn);
            data[syn-1] = data[syn-1] == 0 ? 1:0;
            System.out.println("Corrected Word : "+Arrays.toString(data));
        }
            }

            public static void main(String[] args) {

                    int data[] = {1,0,1,1};

                    System.out.println("Data Word = "+Arrays.toString(data));
                    int codeword[] = encode(data);

                    System.out.println("Code Word = "+Arrays.toString(codeword));
                    codeword[5] = codeword[5] == 0 ? 1:0;
                    decode(codeword);



            }

}
```

Output:

Data Word = [1, 0, 1, 1]

Code Word = [0, 1, 1, 1, 0, 1, 1]

Wrong Word : [0, 1, 1, 1, 0, 0, 1]

Error Was Detected at pos:2

Corrected Word : [0, 0, 1, 1, 0, 0, 1]

Conclusion:

In this experiment, error detection and correction for 7/8-bit ASCII codes was implemented using Hamming Codes and CRC. Hamming Codes enabled single-bit error correction while CRC provided strong detection of burst errors. The encoded packets were transmitted in peer-to-peer mode and analyzed with Wireshark, where packet traces confirmed the reliability and integrity of the communication. This demonstrates that combining Hamming and CRC ensures robust and error-resilient data transfer.