Pradeep KM
3122225001092

Exercise 8 : Error Control

Server algorithm

1. Create a TCP socket.
2. Bind to INADDR_ANY on port 8080.
3. Listen for incoming connections.
4. Accept an incoming connection.
5. Read the received Hamming code.
6. Check for errors in the Hamming code.
7. If an error is detected, correct it.
8. Close the socket.

Client algorithm

1. User enters a 4-bit data string.
2. Calculate the 7-bit Hamming code.
3. Set the second bit of the Hamming code to '0'.
4. Create a TCP socket.
5. Connect to the server at 127.0.0.1 on port 8080.
6. Send the Hamming code to the server.
7. Close the socket.

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>

#define PORT 8080

void checkAndCorrectHammingCode(char *receivedCode) {
    int hammingBits[7];
    for (int i = 0; i < 7; i++) {
        hammingBits[i] = receivedCode[i] - '0';
    }

    int p1 = hammingBits[0] ^ hammingBits[2] ^ hammingBits[4] ^ hammingBits[6];
    int p2 = hammingBits[1] ^ hammingBits[2] ^ hammingBits[5] ^ hammingBits[6];
    int p4 = hammingBits[3] ^ hammingBits[4] ^ hammingBits[5] ^ hammingBits[6];

    int errorPosition = p4 * 4 + p2 * 2 + p1 * 1;

    if (errorPosition == 0) {
        printf("No error detected in received data.\n");
    } else {
        printf("Error detected at position: %d\n", errorPosition);
```

```
        hammingBits[errorPosition - 1] ^= 1;
        printf("Corrected code: ");
        for (int i = 0; i < 7; i++) {
            printf("%d", hammingBits[i]);
        }
        printf("\n");
    }
}

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[8] = {0};

    server_fd = socket(AF_INET, SOCK_STREAM, 0);

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    bind(server_fd, (struct sockaddr *)&address, sizeof(address));
    listen(server_fd, 3);

    new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen);

    read(new_socket, buffer, 7);
    printf("Received code: %s\n", buffer);

    checkAndCorrectHammingCode(buffer);

    close(new_socket);
    close(server_fd);
    return 0;
}
```

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>

#define PORT 8080

void calculateHammingCode(char *data, char *hammingCode) {
    int dataBits[4];
    int hammingBits[7];
```

```
    for (int i = 0; i < 4; i++) {
        dataBits[i] = data[i] - '0';
    }

    hammingBits[2] = dataBits[0];
    hammingBits[4] = dataBits[1];
    hammingBits[5] = dataBits[2];
    hammingBits[6] = dataBits[3];

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

    for (int i = 0; i < 7; i++) {
        hammingCode[i] = hammingBits[i] + '0';
    }
    hammingCode[7] = '\0';
}

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char data[5], hammingCode[8];

    printf("Enter 4-bit data: ");
    scanf("%4s", data);

    calculateHammingCode(data, hammingCode);
    printf("Hamming code to send: %s\n", hammingCode);

    hammingCode[1] = '0';

    sock = socket(AF_INET, SOCK_STREAM, 0);

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr);

    connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));

    send(sock, hammingCode, strlen(hammingCode), 0);
    printf("Hamming code sent\n");

    close(sock);
    return 0;
}
```

Pradeep KM
3122225001092

Output