NAME

MOHAMMAD SHAAD

REGISTRATION NUMBER

21BCE1542

CLASS

COMPUTER NETWORKS

FACULTY

PUNITHA K MA’AM

LAB

EXERCISE 7

AIM

TO IMPLEMENT THE SOCKET PROGRAMMING USING **UDP**

PROCEDURE

1. Open two terminals (one for server and another one for client)
2. Execute the server socket program
3. Execute the client program

CODES

server.c

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

#define MAX_BUFFER_SIZE 1024
#define CHECKSUM_BITS 5

// Function to calculate the checksum value and return as
binary string
char* calculateChecksum(const char* binaryCode)
{
    unsigned int sum = 0;
    unsigned int i;

    for (i = 0; i < strlen(binaryCode); i++)
    {
        sum += binaryCode[i] - '0';  // Convert character to
integer
        sum %= (1 << CHECKSUM_BITS); // Keep the sum within
the bit range
    }

    // Convert checksum to binary string
    char* checksumBinary = malloc(CHECKSUM_BITS + 1);
    for (i = 0; i < CHECKSUM_BITS; i++)
    {
        checksumBinary[CHECKSUM_BITS - i - 1] = (sum & (1 <<
i)) ? '1' : '0';
    }
    checksumBinary[CHECKSUM_BITS] = '\0';

    return checksumBinary;
}

int main()
{
    int sockfd;
    struct sockaddr_in serverAddr, clientAddr;
    char buffer[MAX_BUFFER_SIZE];

    // Create UDP socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0)
    {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }
    memset(&serverAddr, 0, sizeof(serverAddr));
    memset(&clientAddr, 0, sizeof(clientAddr));

    // Configure server address
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    serverAddr.sin_port = htons(12345); // Choose a suitable
port number

    // Bind the socket to the server address
    if (bind(sockfd, (const struct sockaddr*)&serverAddr,
sizeof(serverAddr)) < 0)
    {
        perror("Binding failed");
        exit(EXIT_FAILURE);
    }

    while (1)
    {
        char* checksumBinary;
        unsigned int clientAddrLen = sizeof(clientAddr);

        memset(buffer, 0, MAX_BUFFER_SIZE);

        // Receive binary code from client
        int len = recvfrom(sockfd, buffer, MAX_BUFFER_SIZE,
MSG_WAITALL,
                           (struct sockaddr*)&clientAddr,
&clientAddrLen);
```

```
        if (len < 0)
        {
            perror("Error in receiving message");
            exit(EXIT_FAILURE);
        }

        // Calculate checksum and get as binary string
        checksumBinary = calculateChecksum(buffer);

        printf("Received Binary Code: %s\n", buffer);
        printf("Calculated Checksum: %s\n", checksumBinary);

        // Send the checksum back to the client
        if (sendto(sockfd, checksumBinary,
strlen(checksumBinary), 0,
                   (struct sockaddr*)&clientAddr,
clientAddrLen) < 0)
        {
            perror("Error in sending message");
            exit(EXIT_FAILURE);
        }

        free(checksumBinary);
    }

    close(sockfd);

    return 0;
}
```

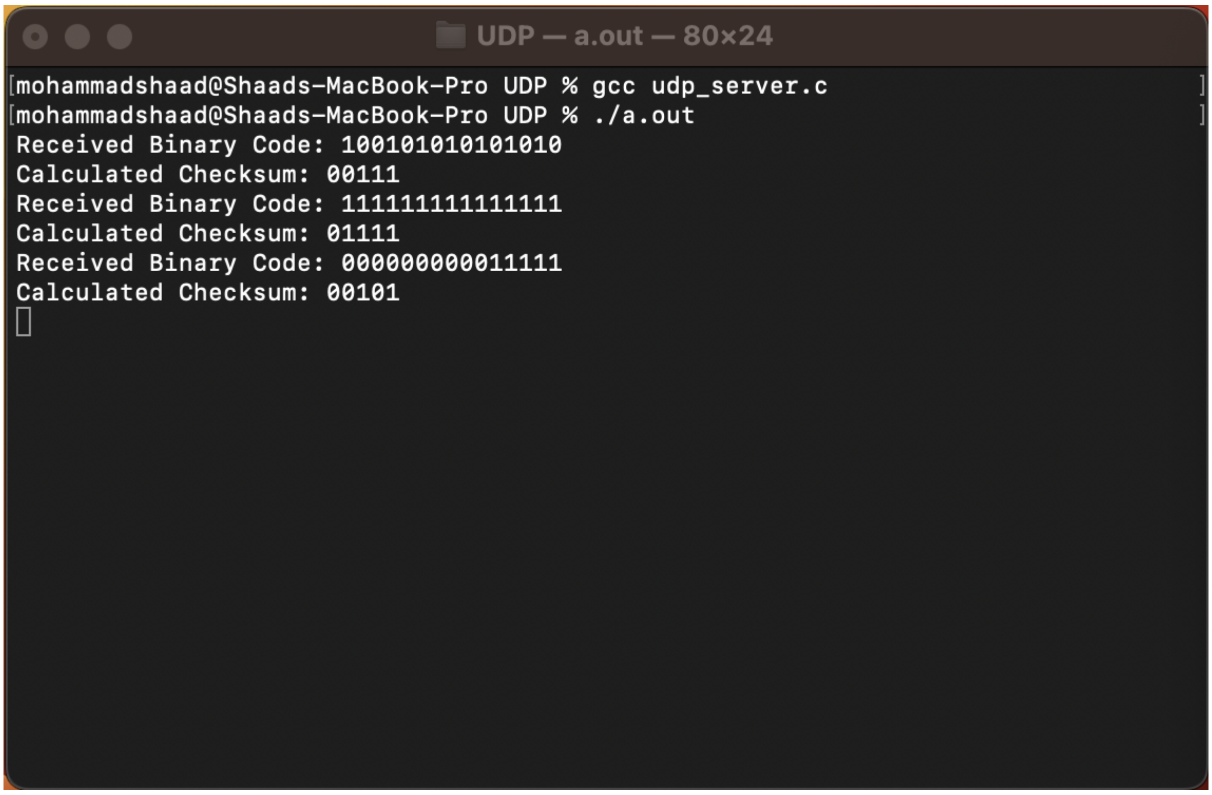
client.c

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

#define MAX_BUFFER_SIZE 1024

int main()
{
    int sockfd;
    struct sockaddr_in serverAddr;

    // Create UDP socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0)
    {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&serverAddr, 0, sizeof(serverAddr));

    // Configure server address
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(12345); // Server port number
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); //
Server IP address
```

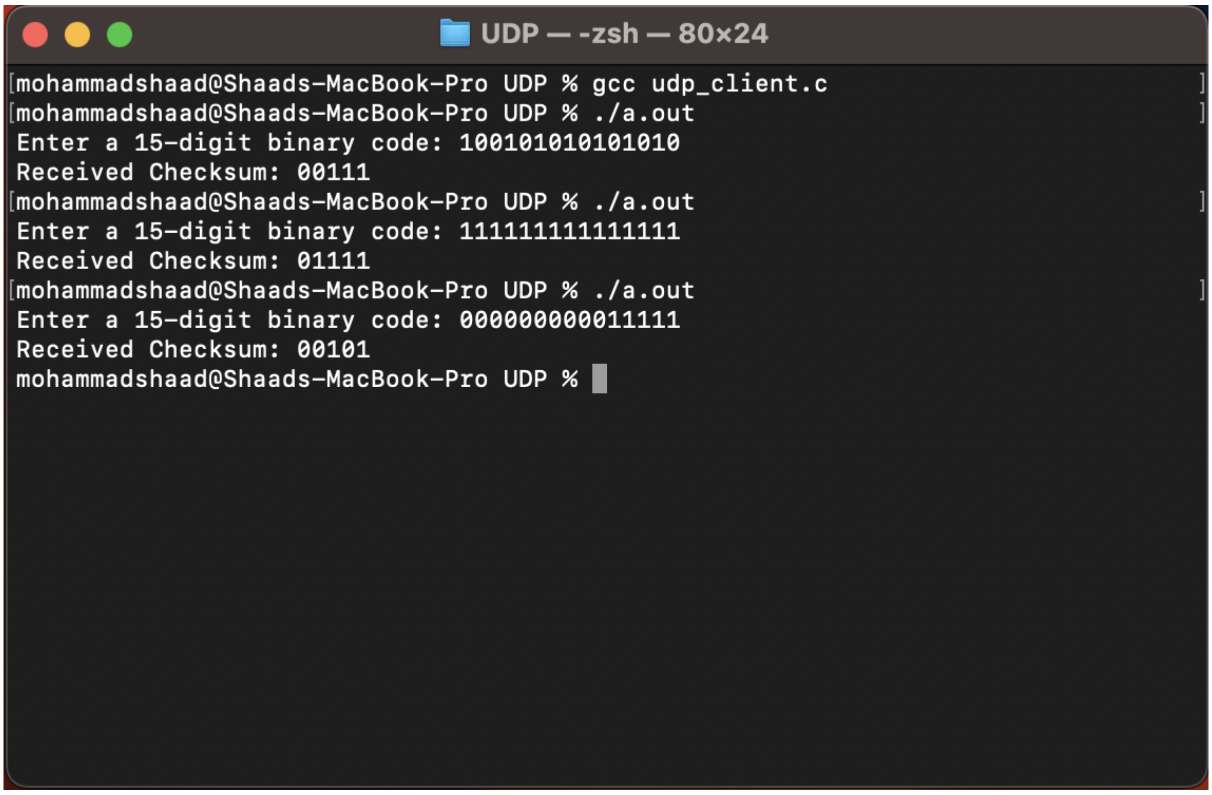
```
    char binaryCode[16];

    printf("Enter a 15-digit binary code: ");
    fgets(binaryCode, sizeof(binaryCode), stdin);

    // Send the binary code to the server
    if (sendto(sockfd, binaryCode, strlen(binaryCode), 0,
               (struct sockaddr*)&serverAddr,
sizeof(serverAddr)) < 0)
    {
        perror("Error in sending message");
        exit(EXIT_FAILURE);
    }

    char checksumBinary[6]; // 5 bits + null terminator
    memset(checksumBinary, 0, sizeof(checksumBinary));

    // Receive the checksum from the server
    if (recvfrom(sockfd, checksumBinary,
sizeof(checksumBinary), MSG_WAITALL,
                 NULL, NULL) < 0)
    {
        perror("Error in receiving message");
        exit(EXIT_FAILURE);
    }

    printf("Received Checksum: %s\n", checksumBinary);

    close(sockfd);

    return 0;
```

```
}
```

OUTPUT


UDP — a.out — 80×24

```
mohammadshaad@Shaads-MacBook-Pro UDP % gcc udp_server.c
mohammadshaad@Shaads-MacBook-Pro UDP % ./a.out
Received Binary Code: 100101010101010
Calculated Checksum: 00111
Received Binary Code: 111111111111111
Calculated Checksum: 01111
Received Binary Code: 000000000011111
Calculated Checksum: 00101
```



```
[mohammadshaad@Shaads-MacBook-Pro UDP % gcc udp_client.c                    ]
[mohammadshaad@Shaads-MacBook-Pro UDP % ./a.out                             ]
Enter a 15-digit binary code: 100101010101010
Received Checksum: 00111
[mohammadshaad@Shaads-MacBook-Pro UDP % ./a.out                             ]
Enter a 15-digit binary code: 111111111111111
Received Checksum: 01111
[mohammadshaad@Shaads-MacBook-Pro UDP % ./a.out                             ]
Enter a 15-digit binary code: 000000000011111
Received Checksum: 00101
mohammadshaad@Shaads-MacBook-Pro UDP %
```

RESULT

Therefore we learned how to implement the Server-Client application which returns the checksum value using UDP