**Name : Shrinivas Hatyalikar**

**Roll No. : 26**

**Div : CS-B**

**Assignment 6**

**Develop a client server using UDP Berkeley socket primitives for chat application in  
peer to peer and client server mode. Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.**

**Code**

**Server**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <winsock2.h>

#define PORT 12345

#define MAX\_BUFFER\_SIZE 1024

void error(const char \*msg) {

    perror(msg);

    exit(1);

}

int main() {

    WSADATA wsa;

    SOCKET server\_socket;

    struct sockaddr\_in server\_addr, client\_addr;

    int addr\_len = sizeof(client\_addr);

    char buffer[MAX\_BUFFER\_SIZE];

    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {

        error("Failed to initialize Winsock");

    }

    // Create socket

    server\_socket = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (server\_socket == INVALID\_SOCKET) {

        error("Error creating socket");

    }

    // Initialize server address structure

    memset(&server\_addr, 0, sizeof(server\_addr));

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_port = htons(PORT);

    server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

    // Bind socket to the server address

    if (bind(server\_socket, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == SOCKET\_ERROR) {

        error("Error on binding");

    }

    printf("Chat server is running on port %d...\n", PORT);

    while (1) {

        int bytes\_received = recvfrom(server\_socket, buffer, MAX\_BUFFER\_SIZE, 0, (struct sockaddr \*)&client\_addr, &addr\_len);

        if (bytes\_received < 0) {

            error("Error receiving data");

        }

        buffer[bytes\_received] = '\0';

        printf("Received from client: %s\n", buffer);

        // Reply to the client

        printf("Enter message to send: ");

        fgets(buffer, MAX\_BUFFER\_SIZE, stdin);

        if (sendto(server\_socket, buffer, strlen(buffer), 0, (struct sockaddr \*)&client\_addr, addr\_len) < 0) {

            error("Error sending data");

        }

    }

    closesocket(server\_socket);

    WSACleanup();

    return 0;

}

**Client**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <winsock2.h>

#define PORT 12345

#define MAX\_BUFFER\_SIZE 1024

void error(const char \*msg) {

    perror(msg);

    exit(1);

}

int main() {

    WSADATA wsa;

    SOCKET client\_socket;

    struct sockaddr\_in server\_addr;

    char buffer[MAX\_BUFFER\_SIZE];

    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {

        error("Failed to initialize Winsock");

    }

    // Create socket

    client\_socket = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (client\_socket == INVALID\_SOCKET) {

        error("Error creating socket");

    }

    // Initialize server address structure

    memset(&server\_addr, 0, sizeof(server\_addr));

    server\_addr.sin\_family = AF\_INET;

    server\_addr.sin\_port = htons(PORT);

    server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

    while (1) {

        printf("Enter message to send: ");

        fgets(buffer, MAX\_BUFFER\_SIZE, stdin);

        if (sendto(client\_socket, buffer, strlen(buffer), 0, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {

            error("Error sending data");

        }

        int bytes\_received = recvfrom(client\_socket, buffer, MAX\_BUFFER\_SIZE, 0, NULL, NULL);

        if (bytes\_received < 0) {

            error("Error receiving data");

        }

        buffer[bytes\_received] = '\0';

        printf("Received from server: %s\n", buffer);

    }

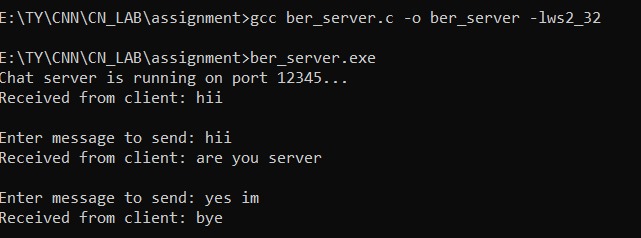
    closesocket(client\_socket);

    WSACleanup();

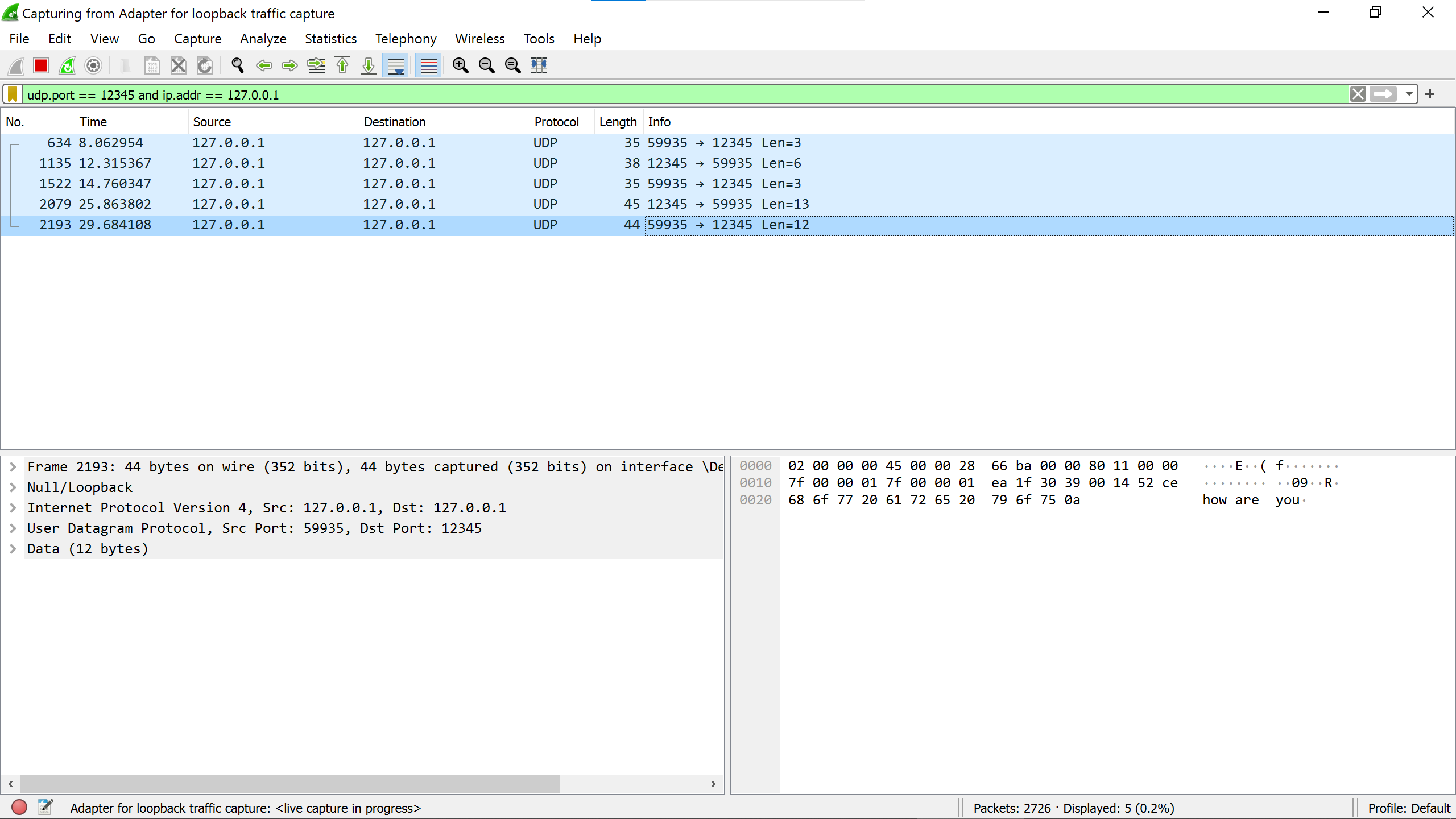
    return 0;

}

**Output**



**Wireshark packet tracing**

****