Name : Gourav Kumar Shaw
Enrollment Id. : 2020CSB010
Section: Gx
Subject : Computer Network Lab (CS 3272)

Assignment 4: UDP Socket Application

Code:

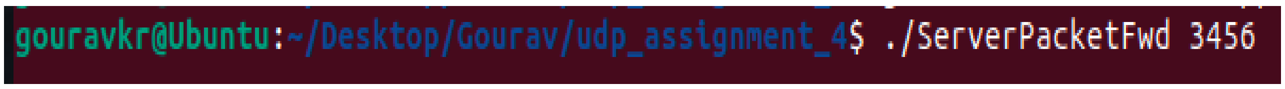
ServerPacketFwd.cpp

```cpp
// 2020CSB010 GOURAV KUMAR SHAW

#include <iostream>
#include <cstring>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>

#define BUFFER_SIZE 1024

const int MAX_PAYLOAD_SIZE = 1000;

int main(int argc, char * argv[]) {

    if (argc < 2) {
        std::cerr << "Use: " << argv[0] << " <port>\n";
        return 1;
    }

    int port = atoi(argv[1]);

    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        std::cerr << "Error in creating socket.\n";
        return 1;
    }

    struct sockaddr_in servaddr, cliaddr;
    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(port);

    if (bind(sockfd, (const struct sockaddr *) &servaddr, sizeof(servaddr)) <
0) {
        std::cerr << "Error in binding socket.\n";
        close(sockfd);
        return 1;
    }

    char buffer[BUFFER_SIZE];
    socklen_t len;

    len = sizeof(cliaddr);

    while (true) {
        struct Packet {
            uint16_t sequence_number;
            uint32_t timestamp;
            uint8_t ttl;
            uint8_t payload[MAX_PAYLOAD_SIZE];
        } recv_packet;

        int n = recvfrom(sockfd, &recv_packet, BUFFER_SIZE, MSG_WAITALL,
(struct sockaddr *) &cliaddr, &len);
        if (n < 0) {
            std::cerr << "Error in receiving packet.\n";
            continue;
        }

        // Decrementing TTL value by one
        recv_packet.ttl--;

        // Sending the same packet back to the client
        if (sendto(sockfd, &recv_packet, sizeof(buffer), MSG_CONFIRM, (const
struct sockaddr *) &cliaddr, len) < 0) {
            perror("sendto");
            continue;
        }
    }

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

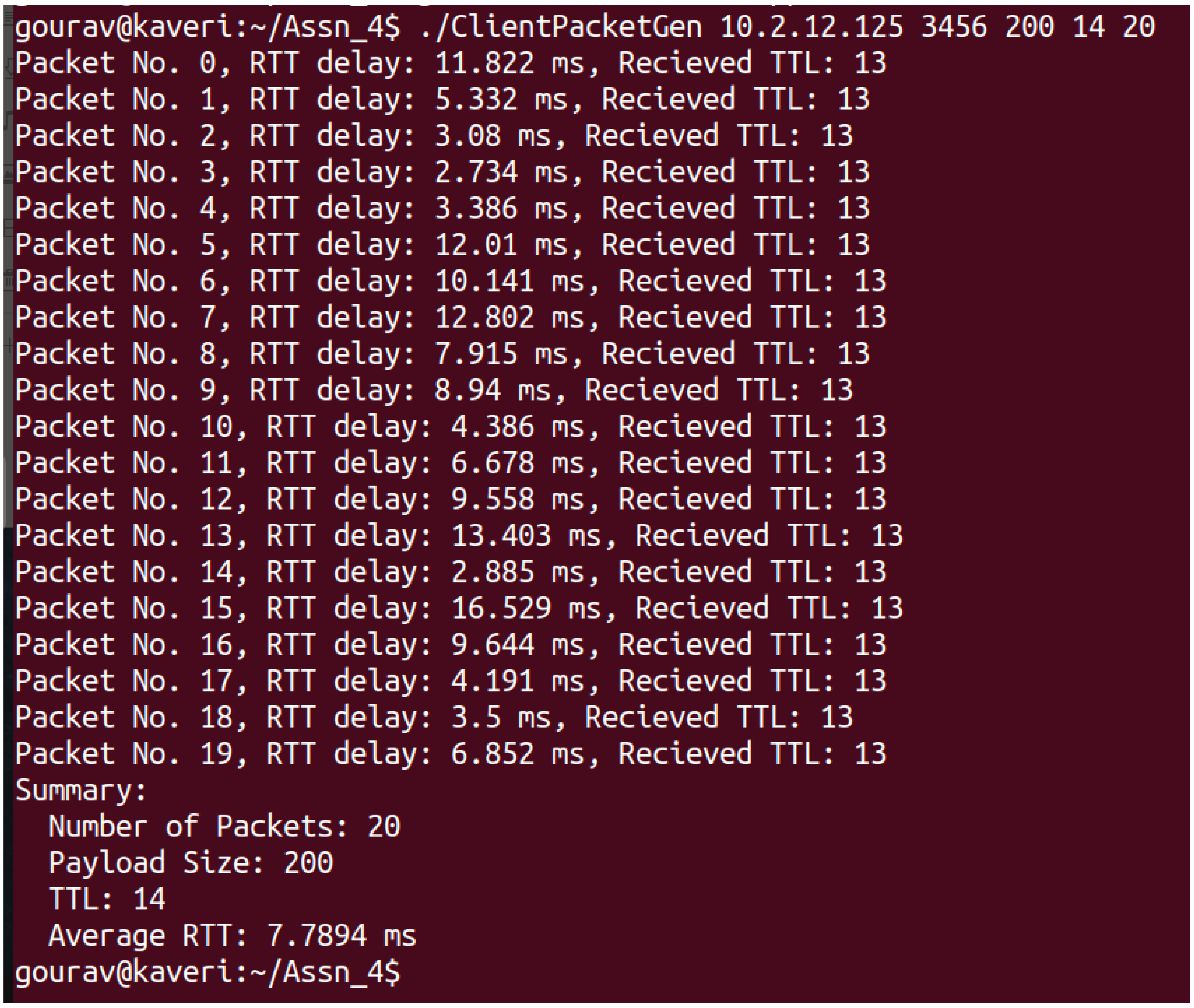
ClientPacketGen.cpp

```
// 2020CSB010 GOURAV KUMAR SHAW

#include <iostream>
#include <cstdlib>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <cstring>
#include <chrono>
#include <ctime>
#include <unistd.h>

using namespace std;
using namespace chrono;

const int MAXIMUM_PAYLOAD_SIZE = 1000;

// Making the Packet structure
struct Packet
{
    uint16_t sequence_number;
    uint32_t timestamp;
    uint8_t ttl;
    uint8_t payload[MAXIMUM_PAYLOAD_SIZE];
};

int main(int argc, char *argv[])
{
    if (argc != 6)
    {
        cerr << "Use: " << argv[0] << " <ServerIP> <ServerPort> <P> <TTL>
<NumPackets>" << endl;
        return EXIT_FAILURE;
    }

    const char *server_ip = argv[1];
    const int server_port = atoi(argv[2]);
    const int payload_size = atoi(argv[3]);
    const int ttl = atoi(argv[4]);
    const int num_packets = atoi(argv[5]);

    if (ttl % 2 != 0)
    {
        cerr << "Error: TTL value must be even" << endl;
        return EXIT_FAILURE;
    }
    if (payload_size > MAXIMUM_PAYLOAD_SIZE)
    {
        cerr << "Error: Payload size too large" << endl;
        return EXIT_FAILURE;
    }

    // Now Creating a datagram socket
    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0)
    {
        cerr << "Error creating socket" << endl;
        return EXIT_FAILURE;
    }

    // Set server address and port
    struct sockaddr_in servaddr;
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(server_ip);
    servaddr.sin_port = htons(server_port);
    double total_rtt = 0;

    // Generate and send packets
    for (int i = 0; i < num_packets; i++)// iterating no. of packets time
    {
        // Generate packet
        Packet packet;
        packet.sequence_number = htons(i);
        packet.timestamp =
htonl(duration_cast<microseconds>(system_clock::now().time_since_epoch()).coun
t());// gives value in microseconds
        packet.ttl = ttl;
        memset(packet.payload, 'a', payload_size);

        // Send packet to server
        if (sendto(sockfd, &packet, sizeof(packet), 0, (struct sockaddr
*)&servaddr, sizeof(servaddr)) < 0)
        {
            cerr << "Error sending packet" << endl;
            return EXIT_FAILURE;
        }

        // Receive packet from server
        Packet recv_packet;
        struct sockaddr_in recvaddr;
        socklen_t len = sizeof(recvaddr);
        if (recvfrom(sockfd, &recv_packet, sizeof(recv_packet), 0, (struct
sockaddr *)&recvaddr, &len) < 0)
        {
            cerr << "Error receiving packet" << endl;
            return EXIT_FAILURE;
        }

        // Calculate RTT delay
        uint16_t seq_num = ntohs(recv_packet.sequence_number);
        uint32_t sent_time = ntohl(recv_packet.timestamp);
        uint32_t recv_time =
duration_cast<microseconds>(system_clock::now().time_since_epoch()).count();
        double rtt_delay = static_cast<double>(recv_time - sent_time) /
1000.0;
        uint8_t recv_ttl = recv_packet.ttl;
        cout << "Packet No. " << seq_num << ", RTT delay: " << rtt_delay << "
ms"
             << ", Recieved TTL: " << static_cast<int>(recv_ttl) << endl;
        total_rtt = total_rtt + rtt_delay;
    }
    // Print summary statistics
    double average_rtt = total_rtt / (double)num_packets;
    cout << "Summary:" << endl;
    cout << "  Number of Packets: " << num_packets << endl;
    cout << "  Payload Size: " << payload_size << endl;
    cout << "  TTL: " << ttl << endl;
    cout << "  Average RTT: " << average_rtt << " ms" << endl;

    close(sockfd);
    return EXIT_SUCCESS;
}
```

Output:



```
gouravkr@Ubuntu:~/Desktop/Gourav/udp_assignment_4$ ./ServerPacketFwd 3456
```



```
gourav@kaveri:~/Assn_4$ ./ClientPacketGen 10.2.12.125 3456 200 14 20
Packet No. 0, RTT delay: 11.822 ms, Recieved TTL: 13
Packet No. 1, RTT delay: 5.332 ms, Recieved TTL: 13
Packet No. 2, RTT delay: 3.08 ms, Recieved TTL: 13
Packet No. 3, RTT delay: 2.734 ms, Recieved TTL: 13
Packet No. 4, RTT delay: 3.386 ms, Recieved TTL: 13
Packet No. 5, RTT delay: 12.01 ms, Recieved TTL: 13
Packet No. 6, RTT delay: 10.141 ms, Recieved TTL: 13
Packet No. 7, RTT delay: 12.802 ms, Recieved TTL: 13
Packet No. 8, RTT delay: 7.915 ms, Recieved TTL: 13
Packet No. 9, RTT delay: 8.94 ms, Recieved TTL: 13
Packet No. 10, RTT delay: 4.386 ms, Recieved TTL: 13
Packet No. 11, RTT delay: 6.678 ms, Recieved TTL: 13
Packet No. 12, RTT delay: 9.558 ms, Recieved TTL: 13
Packet No. 13, RTT delay: 13.403 ms, Recieved TTL: 13
Packet No. 14, RTT delay: 2.885 ms, Recieved TTL: 13
Packet No. 15, RTT delay: 16.529 ms, Recieved TTL: 13
Packet No. 16, RTT delay: 9.644 ms, Recieved TTL: 13
Packet No. 17, RTT delay: 4.191 ms, Recieved TTL: 13
Packet No. 18, RTT delay: 3.5 ms, Recieved TTL: 13
Packet No. 19, RTT delay: 6.852 ms, Recieved TTL: 13
Summary:
  Number of Packets: 20
  Payload Size: 200
  TTL: 14
  Average RTT: 7.7894 ms
gourav@kaveri:~/Assn_4$
```