Name : Gourav Kumar Shaw

Enrollment Id. : 2020CSB010

Section: Gx

Subject : Computer Network Lab (CS 3272)

Assignment 4 extension: UDP Socket Application

Code:

server.cpp

```
// 2020CSB010 GOURAV KUMAR SHAW

#include <iostream>
#include <cstring>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.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 << "Usage: " << 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 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 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 receiving packet.\n";
            continue;
        }

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

        // Send 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;
}
```

client.cpp

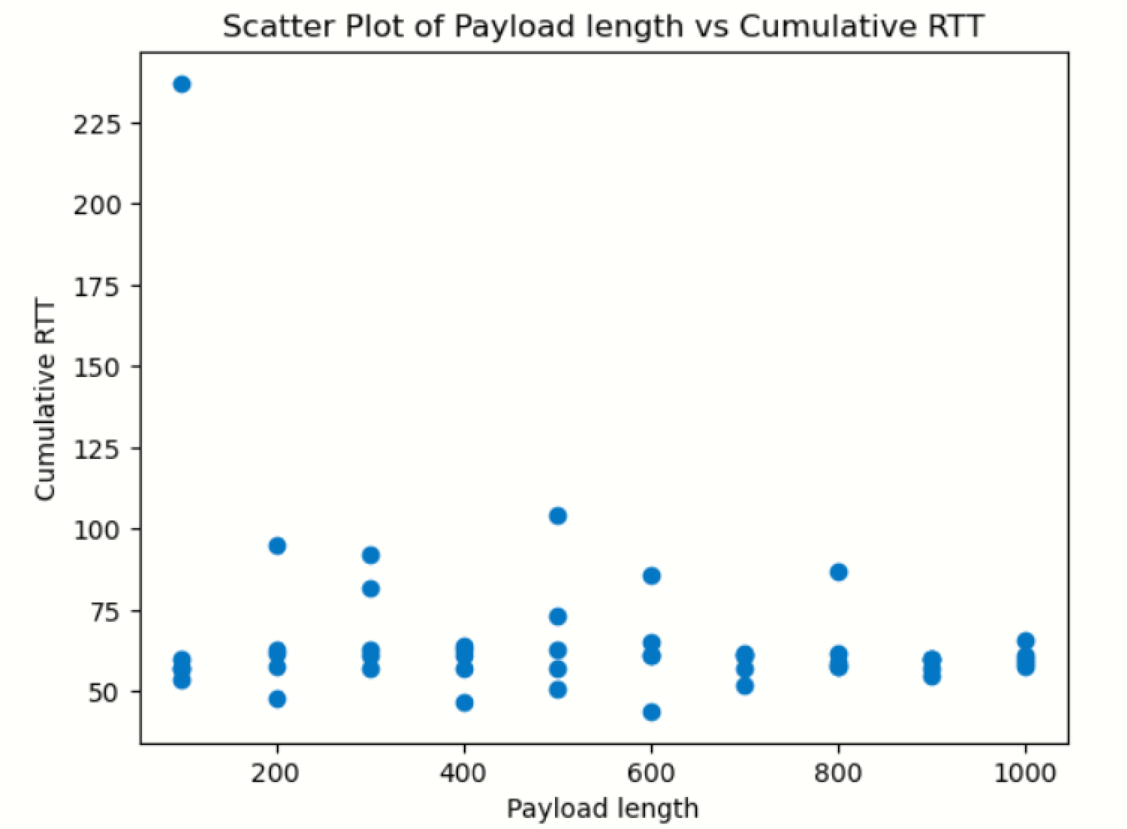
```
// 2020CSB010 GOURAV KUMAR SHAW

#include <bits/stdc++.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/time.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

#define MAXSIZE 1600
using namespace std;
// Driver code
int main(int argc, char *argv[])
{
    // defining the payload

    unordered_map<int, int> m;
    FILE *fp1;
    int k = 100;

    vector<int> vec1;
    vector<int> vec2;
    for (int i = 0; i < 10; i++)
    {
        m[i] = k;
        k += 100;
    }
    if (argc != 6)
    {
        std::cerr << "Use the following arguments: " << argv[0] << "
[ip_address] [port_number] [TTL] [NumPackets] [Filename.csv]\n";
        return 1;
    }
    srand(time(NULL));
    const int PORT = atoi(argv[2]);
    int sockfd;
    char filename[10];
    memcpy(filename, argv[5], sizeof(argv[5]));
    printf("Filename is %s\n", filename);

    struct sockaddr_in servaddr;
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
```

```
    {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

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

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);


    socklen_t len;
    struct timeval timestamp;
    struct timeval timestamp1;

    int numPacket = atoi(argv[4]);

    fp1 = fopen(filename, "w");
    fprintf(fp1, "%s,", "sl.no");
    fprintf(fp1, "%s,", "payload_length");
    fprintf(fp1, "%s \n", "Cumulative_RTT");
    for (int i = 0; i < numPacket; i++)
    {
        char buffer[MAXSIZE];
        memset(buffer, '\0', MAXSIZE);
        int payload_length = m[i % 10];
        printf("Payload length made as:%d\n", payload_length);
        int ttl = atoi(argv[3]);
        if (ttl % 2 != 0)
            ttl = ttl - 1;

        cout <<"ttl as: "<< ttl << endl;
        // making payload with random alphabets
        char payload[payload_length + 1];
        for (int j = 0; j < payload_length; j++)
        {
            payload[j] = 'a' + rand() % 26;
        }
        payload[payload_length] = '\0';

        gettimeofday(&timestamp, NULL);
        long int microseconds = timestamp.tv_usec;
        if (i <= 9)
        {
            buffer[0] = '0';
```

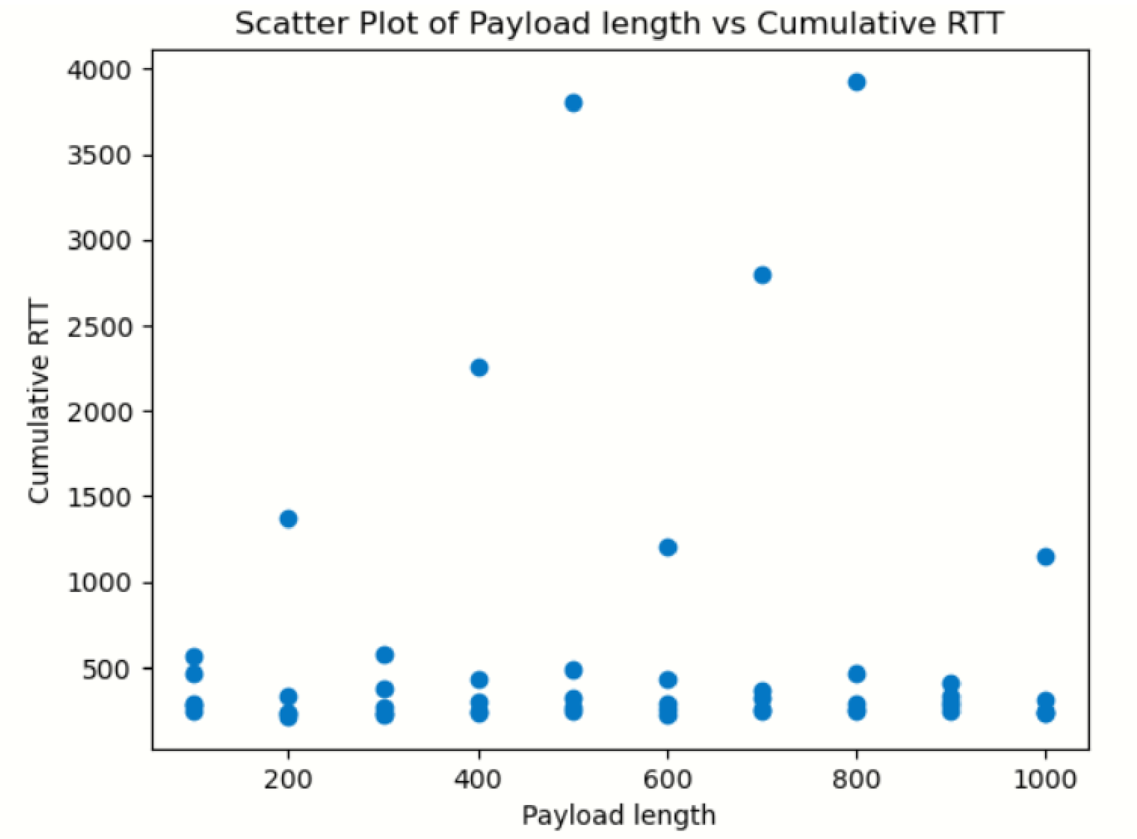
```
            snprintf(buffer + 1, MAXSIZE, "%d%ld", i, microseconds); // i and
microseconds variables, formatting them into a string using the specified
format string, and then writing that string to the buffer starting at the
second position of the buffer
        }
        else
            snprintf(buffer, MAXSIZE, "%d%ld", i, microseconds);

        buffer[8] = ttl;
        buffer[9] = payload_length;


        strcat(buffer, payload);
        strcat(buffer, filename);
        int n;
        while (ttl)
        {
            sendto(sockfd, (const char *)buffer, strlen(buffer),
                   MSG_CONFIRM, (const struct sockaddr *)&servaddr,
                   sizeof(servaddr));
            std::cout << "Client:  "
                      << "Sequence Number:" << buffer[0] << buffer[1] <<
std::endl
                      << "time:" << microseconds << std::endl
                      << "ttl:" << ttl << std::endl;
            std::cout << "above message sent\n";
            n = recvfrom(sockfd, (char *)buffer, MAXSIZE,
                         MSG_WAITALL, (struct sockaddr *)&servaddr,
                         &len);
            ttl = buffer[8];
            ttl--;
            buffer[8] = ttl;
        }

        gettimeofday(&timestamp1, NULL);
        long int microsecond = timestamp1.tv_usec;// gives the time in
microseconds
        ttl = buffer[8];
        buffer[n] = '\0';


        int Cumulative_RTT = microsecond - microseconds;
        cout << "Starting time is:" << microseconds <<" microseconds"<< endl;
        cout << "ending time is:" << microsecond <<" microseconds"<< endl;
        cout << "This is the Cumulative_RTT-->" << Cumulative_RTT <<"
microseconds"<< endl;

        vec1.push_back(payload_length);
```

```
        vec2.push_back(Cumulative_RTT);

        char temp1[5];
        char temp2[5];

        fprintf(fp1, "%d,", i);
        fprintf(fp1, "%d,", payload_length);
        fprintf(fp1, "%d \n", Cumulative_RTT);
    }
    for (auto it : vec1)
    {
        cout << it << " ";
    }
    cout << endl;
    for (auto it1 : vec2)
    {
        cout << it1 << " ";
    }

    int err = close(sockfd);
    return 0;
}
```

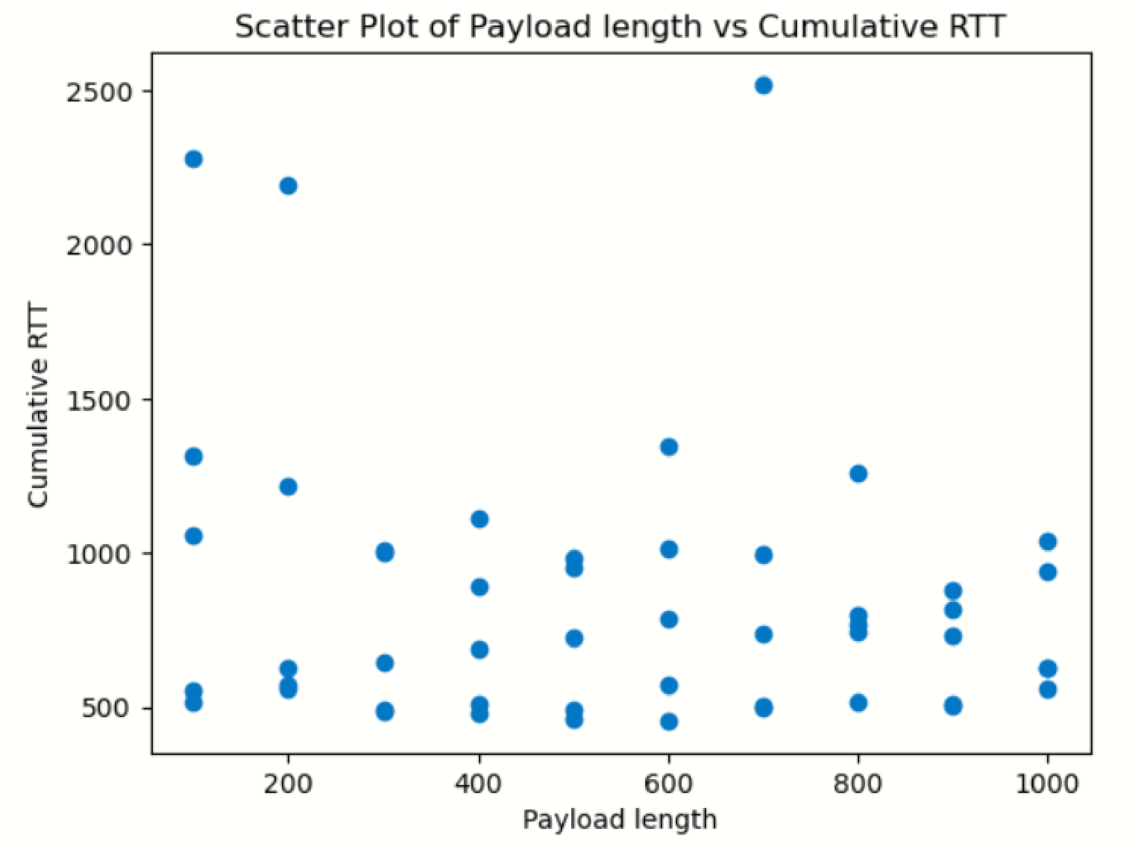
Scatter plot for TTL=2



Scatter Plot for TTL=8



Scatter Plot for TTL= 16



Qn 1. What do you observe? What information does the slope of the graph contain?

Ans:

- The plot between cumulative RTT (Round-Trip Time) and payload length of packets provide insights into the behavior of the network in this scenario. The slope of this plot can indicate the relationship between the RTT and the payload length of the packets.
- If the slope of the plot is positive, it indicates that as the payload length of the packets increases, the RTT also increases. This could be due to the fact that larger packets take longer to transmit and process than smaller packets, which increases the overall RTT. In this case, a steeper slope may indicate that the increase in RTT with respect to the increase in payload length is more significant.
- Conversely, if the slope of the plot is negative, it indicates that as the payload length of the packets increases, the RTT decreases. This could be due to the fact that larger packets can be processed more efficiently than smaller packets, leading to a shorter RTT overall. In this case, a steeper slope may indicate that the decrease in RTT with respect to the increase in payload length is more significant.
- It is worth noting that the slope of the plot alone may not provide a complete picture of the network performance, as other factors such as network congestion, packet loss, and network topology can also impact the RTT and payload length of the packets.

<u>Qn 2.</u> Repeat the process when T = 8 and 16 and mention your understanding.

<u>Ans:</u>

1. **Increasing the TTL value from 2 to 8 will allow packets to traverse a greater number of routers or hops before being discarded.** This means that packets can travel over a longer distance and may take a longer time to reach their destination.
2. As for the effect on Round-Trip Time (RTT), it may increase or decrease depending on the network conditions. If the network is congested or has a high latency, increasing the TTL value may result in a longer RTT as packets take longer to reach their destination. On the other hand, if the network is not congested and has low latency, increasing the TTL value may have little effect on the RTT.
3. **Increasing the TTL value from 8 to 16 will allow packets to traverse a greater number of routers or hops before being discarded.** <u>This means that packets can travel over a longer distance and may take a longer time to reach their destination.</u>
4. As for the effect on Round-Trip Time (RTT), it may increase or decrease depending on the network conditions. If the network is congested or has a high latency, increasing the TTL value may result in a longer RTT as packets take longer to reach their destination. On the other hand, if the network is not congested and has low latency, increasing the TTL value may have little effect on the RTT.
5. In general, the effect of changing the TTL value on RTT will depend on various factors, including the network topology, congestion levels, and the distance between the source and destination hosts. **<u>However, increasing the TTL value from 8 to 16 is likely to have a smaller impact on RTT than increasing it from 2 to 8 because the number of additional hops that packets can travel is relatively smaller</u>**.