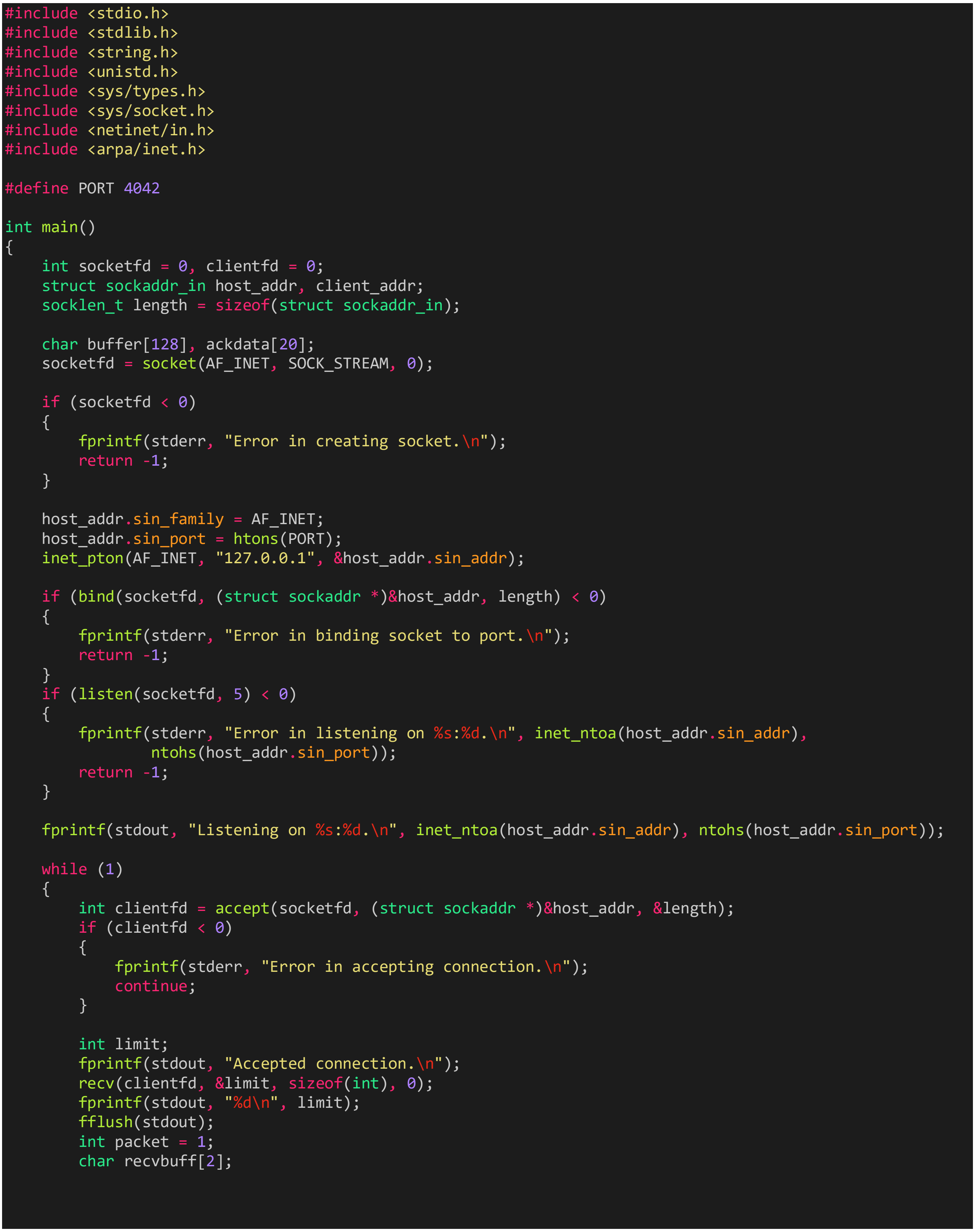
IMPLEMENTING THE NAGGLE’S ALGORITHM

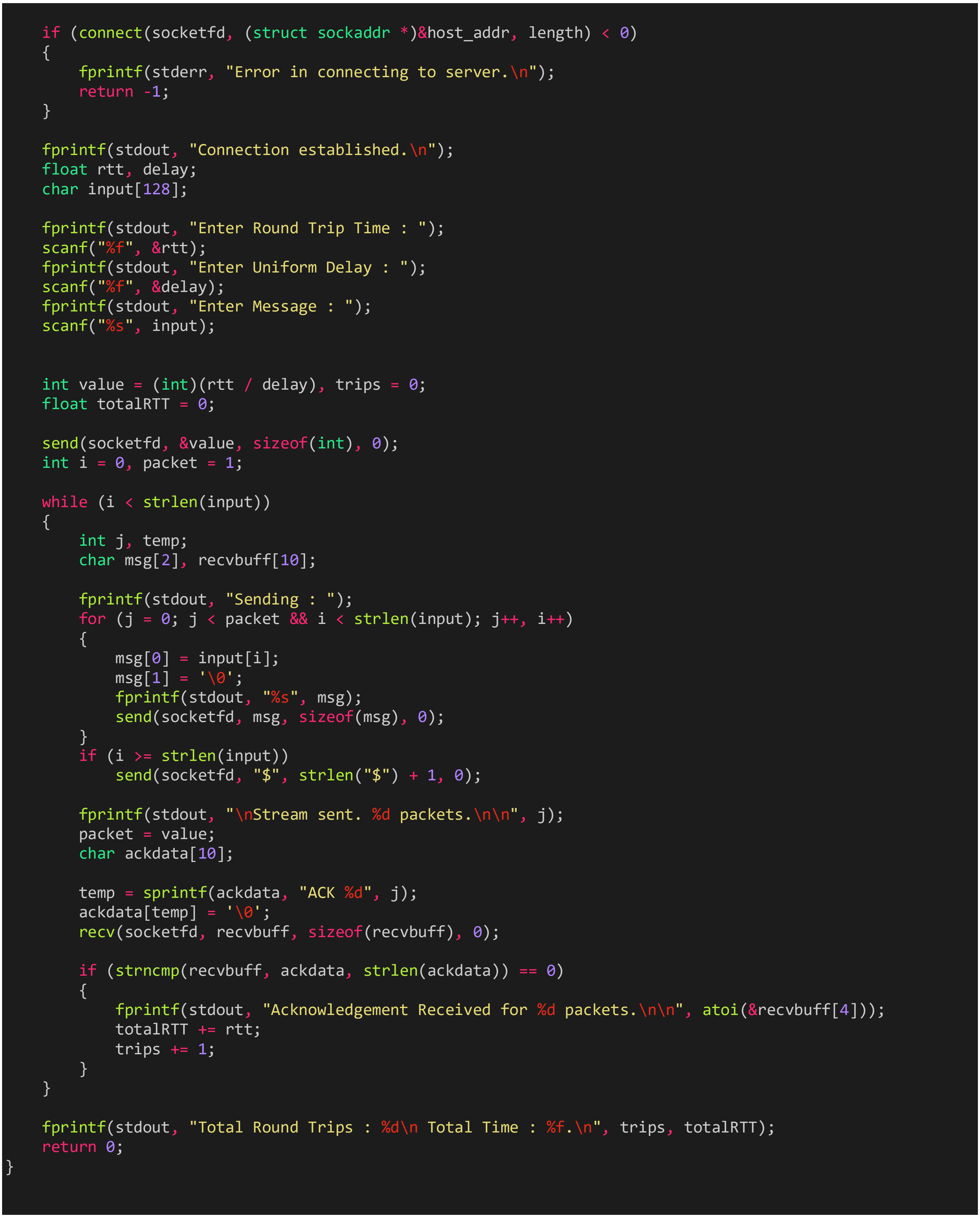
SERVER



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 4042

int main()
{
    int socketfd = 0, clientfd = 0;
    struct sockaddr_in host_addr, client_addr;
    socklen_t length = sizeof(struct sockaddr_in);

    char buffer[128], ackdata[20];
    socketfd = socket(AF_INET, SOCK_STREAM, 0);

    if (socketfd < 0)
    {
        fprintf(stderr, "Error in creating socket.\n");
        return -1;
    }

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

    if (bind(socketfd, (struct sockaddr *)&host_addr, length) < 0)
    {
        fprintf(stderr, "Error in binding socket to port.\n");
        return -1;
    }
    if (listen(socketfd, 5) < 0)
    {
        fprintf(stderr, "Error in listening on %s:%d.\n", inet_ntoa(host_addr.sin_addr),
                ntohs(host_addr.sin_port));
        return -1;
    }

    fprintf(stdout, "Listening on %s:%d.\n", inet_ntoa(host_addr.sin_addr), ntohs(host_addr.sin_port));

    while (1)
    {
        int clientfd = accept(socketfd, (struct sockaddr *)&host_addr, &length);
        if (clientfd < 0)
        {
            fprintf(stderr, "Error in accepting connection.\n");
            continue;
        }

        int limit;
        fprintf(stdout, "Accepted connection.\n");
        recv(clientfd, &limit, sizeof(int), 0);
        fprintf(stdout, "%d\n", limit);
        fflush(stdout);
        int packet = 1;
        char recvbuff[2];
        while (1)
        {
            int j, temp;
            fprintf(stdout, "Receiving : ");

            for (j = 0; j < packet; j++)
            {
                recv(clientfd, recvbuff, sizeof(recvbuff), 0);
                if (strncmp(recvbuff, "$", sizeof("$")) == 0)
                {
                    break;
                }
                fprintf(stdout, "%s", recvbuff);
            }

            fprintf(stdout, "\nEnd of stream. %d packets.\n\n", j);
            char ackdata[10];

            temp = sprintf(ackdata, "ACK %d", j);
            ackdata[temp] = '\0';
            fprintf(stdout, "Sending acknowledgment for %d packets.\n", j);
            send(clientfd, ackdata, strlen(ackdata) + 1, 0);

            if (j < packet)
            {
                close(clientfd);
                break;
            }
            packet = limit;
        }
    }

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

CLIENT

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 4042

int main()
{
    int socketfd = 0;
    struct sockaddr_in host_addr;
    socklen_t length = sizeof(struct sockaddr_in);
    socketfd = socket(AF_INET, SOCK_STREAM, 0);

    if (socketfd < 0)
    {
        fprintf(stderr, "Error in creating socket.\n");
        return -1;
    }

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



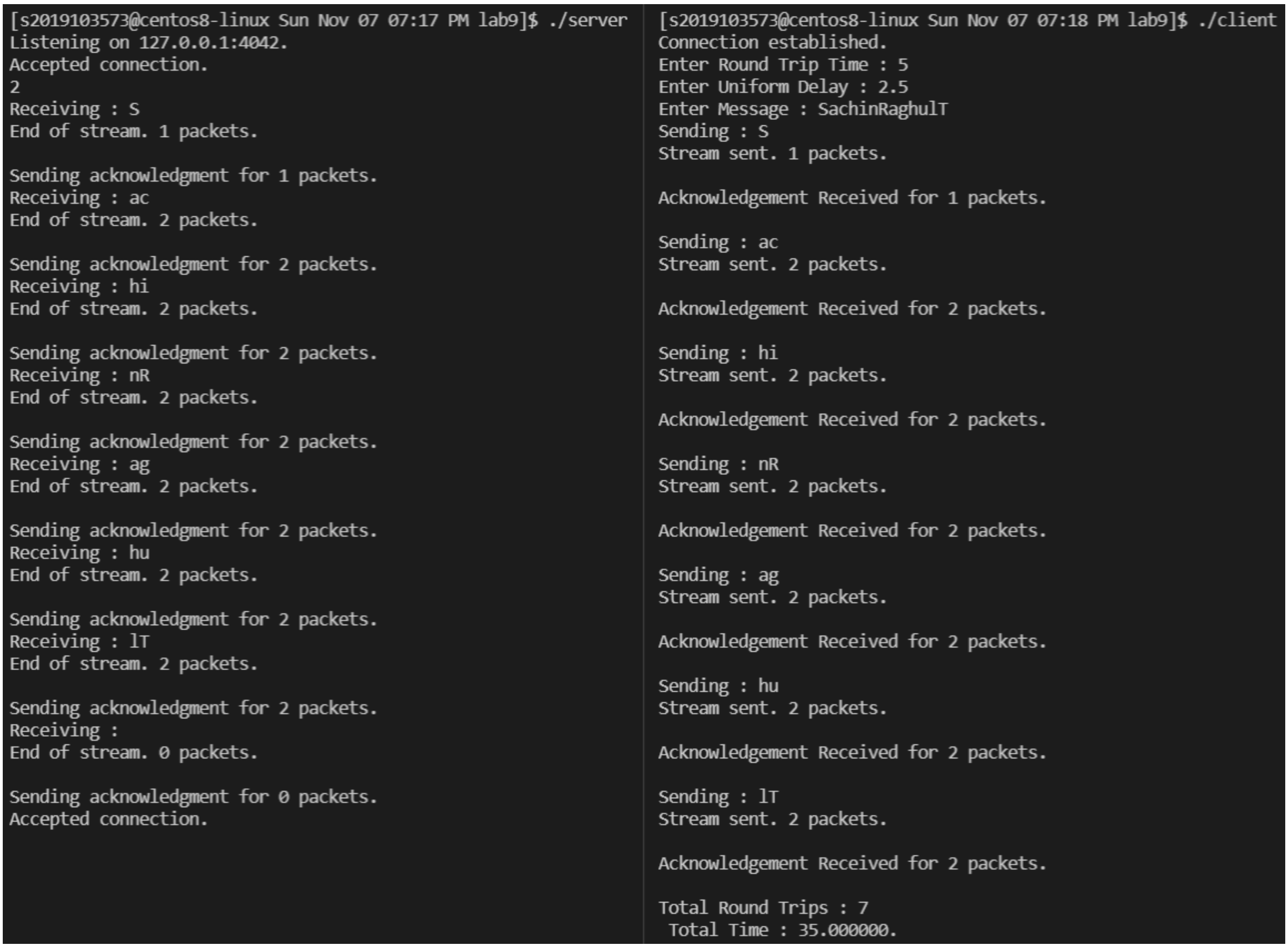
```c
    if (connect(socketfd, (struct sockaddr *)&host_addr, length) < 0)
    {
        fprintf(stderr, "Error in connecting to server.\n");
        return -1;
    }

    fprintf(stdout, "Connection established.\n");
    float rtt, delay;
    char input[128];

    fprintf(stdout, "Enter Round Trip Time : ");
    scanf("%f", &rtt);
    fprintf(stdout, "Enter Uniform Delay : ");
    scanf("%f", &delay);
    fprintf(stdout, "Enter Message : ");
    scanf("%s", input);


    int value = (int)(rtt / delay), trips = 0;
    float totalRTT = 0;

    send(socketfd, &value, sizeof(int), 0);
    int i = 0, packet = 1;

    while (i < strlen(input))
    {
        int j, temp;
        char msg[2], recvbuff[10];

        fprintf(stdout, "Sending : ");
        for (j = 0; j < packet && i < strlen(input); j++, i++)
        {
            msg[0] = input[i];
            msg[1] = '\0';
            fprintf(stdout, "%s", msg);
            send(socketfd, msg, sizeof(msg), 0);
        }
        if (i >= strlen(input))
            send(socketfd, "$", strlen("$") + 1, 0);

        fprintf(stdout, "\nStream sent. %d packets.\n\n", j);
        packet = value;
        char ackdata[10];

        temp = sprintf(ackdata, "ACK %d", j);
        ackdata[temp] = '\0';
        recv(socketfd, recvbuff, sizeof(recvbuff), 0);

        if (strncmp(recvbuff, ackdata, strlen(ackdata)) == 0)
        {
            fprintf(stdout, "Acknowledgement Received for %d packets.\n\n", atoi(&recvbuff[4]));
            totalRTT += rtt;
            trips += 1;
        }
    }

    fprintf(stdout, "Total Round Trips : %d\n Total Time : %f.\n", trips, totalRTT);
    return 0;
}
```

OUTPUT :-

Message : SachinRaghulT

Round Trip Time : 5

Uniform Delay : 2.5



```
[s2019103573@centos8-linux Sun Nov 07 07:17 PM lab9]$ ./server
Listening on 127.0.0.1:4042.
Accepted connection.
2
Receiving : S
End of stream. 1 packets.

Sending acknowledgment for 1 packets.
Receiving : ac
End of stream. 2 packets.

Sending acknowledgment for 2 packets.
Receiving : hi
End of stream. 2 packets.

Sending acknowledgment for 2 packets.
Receiving : nR
End of stream. 2 packets.

Sending acknowledgment for 2 packets.
Receiving : ag
End of stream. 2 packets.

Sending acknowledgment for 2 packets.
Receiving : hu
End of stream. 2 packets.

Sending acknowledgment for 2 packets.
Receiving : lT
End of stream. 2 packets.

Sending acknowledgment for 2 packets.
Receiving :
End of stream. 0 packets.

Sending acknowledgment for 0 packets.
Accepted connection.
```

```
[s2019103573@centos8-linux Sun Nov 07 07:18 PM lab9]$ ./client
Connection established.
Enter Round Trip Time : 5
Enter Uniform Delay : 2.5
Enter Message : SachinRaghulT
Sending : S
Stream sent. 1 packets.

Acknowledgement Received for 1 packets.

Sending : ac
Stream sent. 2 packets.

Acknowledgement Received for 2 packets.

Sending : hi
Stream sent. 2 packets.

Acknowledgement Received for 2 packets.

Sending : nR
Stream sent. 2 packets.

Acknowledgement Received for 2 packets.

Sending : ag
Stream sent. 2 packets.

Acknowledgement Received for 2 packets.

Sending : hu
Stream sent. 2 packets.

Acknowledgement Received for 2 packets.

Sending : lT
Stream sent. 2 packets.

Acknowledgement Received for 2 packets.

Total Round Trips : 7
 Total Time : 35.000000.
```

Total Round Trips : 7

Total Time : 35.0000 ms