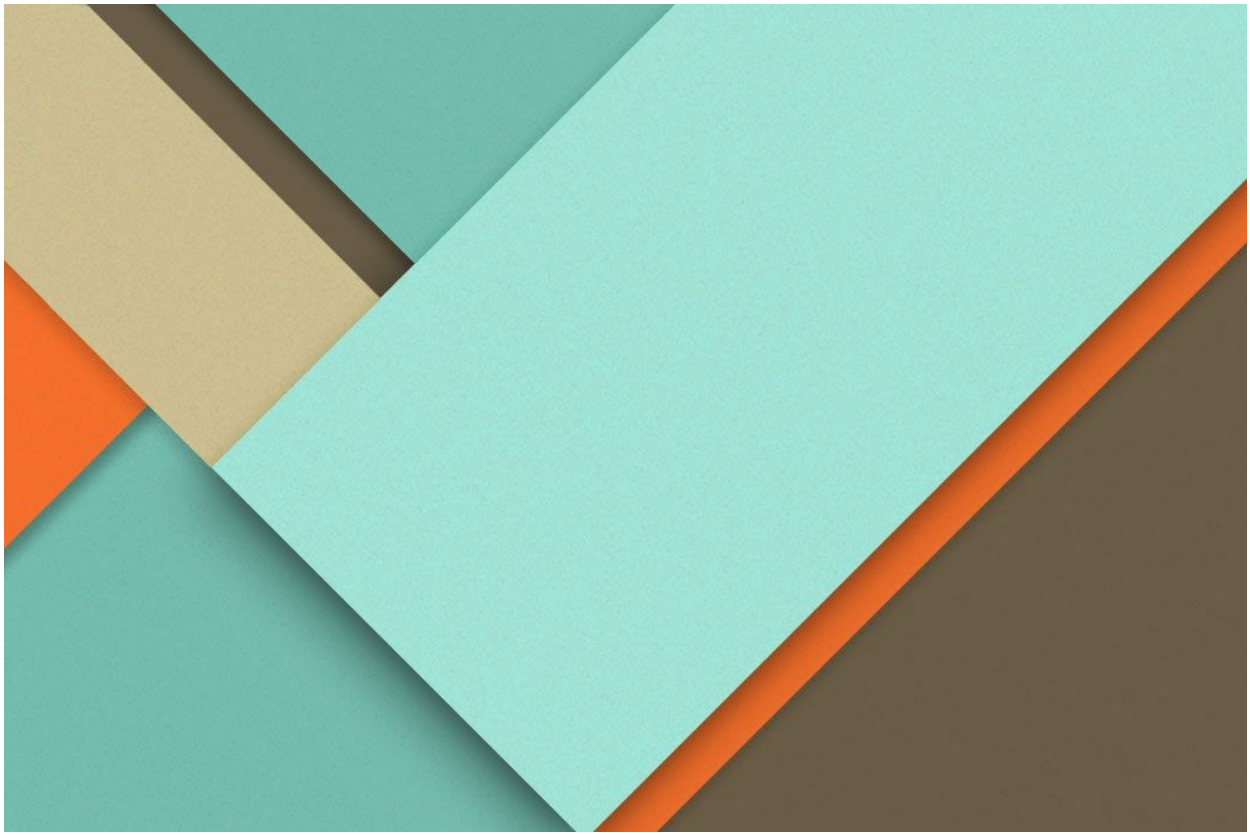


Assignment 2

17.01.2021

Name

Yousef Mohamed Fathy

ID

66

Overview

The requirement of this assignment is to build a reliable data transfer over UDP protocol

Goals

1. Allow client and server communication with a reliable data transmission

Specifications

A client sent a request with a file name to the local host server and the server. When the server receives the request, it reads the file and splits it into packets and sends them.

The packet needs to be acknowledged to be received. Packet loss is simulated by a ratio which represents the probability of a packet loss.

Server side

The server starts at a local IP address with 8port number 8080. On receiving a request, It creates a child process to handle the request with a new file descriptor.

```
int main()
{
    int port_number;
    int server_fd, client_fd;
    struct sockaddr_in server_addr {};
    struct sockaddr_in client_addr {};
    socklen_t addr_len = sizeof(struct sockaddr);
    int broadcast = 1;

    port_number = 8080;

    memset(&server_addr, 0, sizeof(server_addr));
    memset(&client_addr, 0, sizeof(client_addr));

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(port_number);
    server_addr.sin_addr.s_addr = INADDR_ANY;
    memset(&(server_addr.sin_zero), '\0', 8);


    if ((server_fd = socket(PF_INET, SOCK_DGRAM, 0)) == -1)
    {
        cerr << "Can't create a socket! Quitting" << endl;
        return -6;
    }

    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &broadcast, sizeof(int)))
    {
        perror("setsockopt server_fd:");
        return 1;
    }

    if (bind(server_fd, (struct sockaddr *)&server_addr, sizeof(struct sockaddr)) == -1)
    {
        cerr << "Can't bind the socket! Quitting" << endl;
        return -2;
    }

    cout << "Server is start listening on port 8080"<< endl;
```

```
    char* buf = new char[600];

    while(true)
    {
        memset(buf,0, 600);

        int bytesReceived = recvfrom(server_fd, buf, 600, 0, (struct sockaddr*)&client_addr,  &addr_len);
        if (bytesReceived == -1)
        {
            perror("Error in recv(). Quitting\n");
            return -5;
        }
        if (bytesReceived == 0)
        {
            perror("Client disconnected !!! \n");
            return-5;
        }

        auto* data_packet = (struct packet*) buf;

        //delegate request to child process
        pid_t pid = fork();
        if (pid == 0)
        {
            //child process
            if ((client_fd = socket(PF_INET, SOCK_DGRAM, 0)) == -1)
            {
                cerr << "error during creating a child process" << endl;
                return -6;
            }
            if (setsockopt(client_fd, SOL_SOCKET, SO_REUSEADDR, &broadcast, sizeof(int)))
            {
                perror("setsockopt server_fd:");
                return 1;
            }
            handle_request(client_fd, client_addr, addr_len, data_packet);
        }
    }
}
```

The child process handle the request ass follow:

```
void handle_request(int client_fd, sockaddr_in client_addr, socklen_t addr_len, packet *packet)
{
    vector<vector<char>> data;
    int data_length = packet->len - sizeof(packet->seqno) - sizeof(packet->len);
    string file_name = string(packet->data, 0, data_length);
    cout << "file name: "<<string(packet->data, 0, data_length)<<endl;
    data = get_data(file_name);
    send_data(client_fd, client_addr, addr_len, data);
}

vector<vector<char>> get_data(string file_name)
{
    vector<vector<char>> file_data;
    vector<char> row;
    char c;
    ifstream fin;
    fin.open(file_name);
    if(fin)
    {
        int counter = 0;
        while(fin.get(c))
        {
            if(counter < 499)
                row.push_back(c);
            else
            {
                file_data.push_back(row);
                row.clear();
                row.push_back(c);
                counter = 0;
                continue;
            }
            counter++;
        }
        if(counter > 0) file_data.push_back(row);
    }
    else
    {
        perror("An error occurred or no file exist with that name");
        exit(1);
    }
```

```
        perror("An error occurred or no file exist with that name");
        exit(1);
    }
    fin.close();
    return file_data;
}

void send_data(int client_fd, struct sockaddr_in client_addr, socklen_t addr_len,  vector<vector<char>> data)
{
    int seqno ;
    for (int i = 0; i < data.size(); ++i)
    {
        seqno = 500*i;
        struct packet p = create_packet(data.at(i), seqno, data.at(i).size());
        cout <<"data size = " << data.at(i).size()<<endl;

        char* buf = new char[600];
        memset(buf, 0, 600);
        memcpy(buf, &p, sizeof(p));
        double prob = rand() % 100  ;
        cout << prob << "---" << PLP * 100<< endl;
        if( prob > PLP * 100)
        {
            int bytesSent = sendto(client_fd, buf, 600, 0,(struct sockaddr *)&client_addr, sizeof(struct sockaddr));
            if (bytesSent == -1)
            {
                perror("couldn't send the ack");
                exit(1);
            }
        }
        int sret;
        fd_set readfds;
        struct timeval timeout {};

        FD_ZERO(&readfds);
        FD_SET(client_fd, &readfds);

        timeout.tv_sec = TIMEOUT;
        timeout.tv_usec = 0;

        sret = select(client_fd+1, &readfds, nullptr, nullptr, &timeout);
```


```
        sret = select(client_fd+1, &readfds, nullptr, nullptr, &timeout);
```


```
        if(sret == 0)
        {
            cout << "No acks received for packet "<< i << ". Resending" << endl;
            i--;
            continue;

        }
        else if(sret == -1)
        {
            perror("error in select");
            exit(1);
        }
        else
        {
            memset(buf,0, 600);
            int bytesReceived = recvfrom(client_fd, buf, 600, 0, (struct sockaddr*)&client_addr,  &addr_len);
            if (bytesReceived == -1)
            {
                perror("Error in recv(). Quitting");
                exit(1);
            }
            cout <<"ack received for packet "<< i <<endl;
        }
    }
}

packet create_packet(vector<char> data, uint32_t seqno, int data_len)
{
    struct packet p;
    p.len = data_len + sizeof(p.len) + sizeof(p.seqno);
    p.seqno = seqno;
    memset(p.data, 0, 500);
    strcpy(p.data, data.data());
    return p;
}
```

Client side

The client start with a request contains the file name and enter a loop receiving the packets. For each packet received, An acknowledge is sent. When there are no packets received, the client stores the file packets in one file.

```
int main()
{
    memset(&serv_addr, 0, sizeof(serv_addr));
    // reading client.in
    ifstream fin;
    ofstream fout;
    string file_name;
    string line;
    port_number = 8080;
    file_name = "input.txt";
    // server info
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(port_number);
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    memset(&(serv_addr.sin_zero), '\0', 8);

    // create client socket
    if ((sockfd = socket(PF_INET, SOCK_DGRAM, 0)) == -1)
    {
        cerr << "Can't create a socket! Quitting" << endl;
        exit(EXIT_FAILURE);
    }

    // send a packet with the filename
    struct packet p = create_packet(file_name);

    char* buf = new char[600];
    memset(buf,0, 600);
    memcpy(buf, &p, sizeof(p));

    send_request(buf);
```

```
    while(true)
    {
        int sret;
        fd_set readfds;
        struct timeval timeout {};
        FD_ZERO(&readfds);
        FD_SET(sockfd, &readfds);
        timeout.tv_sec = TIMEOUT;
        timeout.tv_usec = 0;
        sret = select(sockfd+1, &readfds, nullptr, nullptr, &timeout);

        if(sret == 0)
        {
            cout << "No response from the server!!. XxXxXxTerminatingxXxXxX" << endl;
            break;
        }
        else if(sret == -1)
        {
            perror("error in select");
            exit(1);
        }

        memset(buf,0, 600);
        int bytesReceived = recvfrom(sockfd, buf, 600, 0, (struct sockaddr*)&serv_addr,  &addr_len);
        if (bytesReceived == -1)
        {
            perror("Error while receivng packet");
            exit(EXIT_FAILURE);
        }

        cout <<"The Packet number "<<i<<" is received " << packets.size() << endl;

        auto* data_packet = (struct packet*) buf;
        extract_data(data_packet);
        send_ack(data_packet->seqno);
        i++;
    }
```

```
    fout.open(file_name);
    cout <<"file size: "<< packets.size() << " packets" << endl;
    map<uint32_t, vector<char>> :: iterator it;
    for (it=packets.begin() ; it!=packets.end() ; it++){
        for(char c : (*it).second){
            fout << c;
        }
    }
    fout.close();
}
void send_ack(uint32_t seqno)
{
    struct ack_packet ack;
    ack.cksum = 0;
    ack.len = sizeof(ack);
    ack.ackno = seqno;

    char* buf = new char[600];
    memset(buf,0, 600);
    memcpy(buf, &ack, sizeof(ack));

    int bytesSent = sendto(sockfd, buf, 600, 0,(struct sockaddr *)&serv_addr, sizeof(struct sockaddr));
    if (bytesSent == -1)
    {
        perror("couldn't send the ack");
        exit(1);
    }
}
```

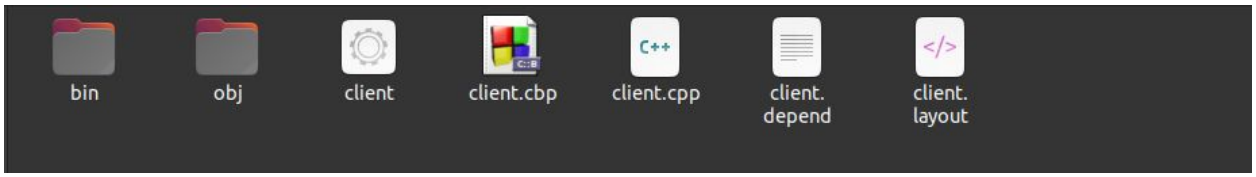
```
void extract_data(struct packet *data_packet)
{
    uint32_t seqno = data_packet->seqno;
    vector<char> data;

    int size = data_packet->len - sizeof(seqno) - sizeof(data_packet->len);
    cout <<"data size = " << size << endl;
    data.reserve(size);

    for (int i = 0; i < size; ++i)
    {
        data.push_back(data_packet->data[i]);
    }

    packets.insert(make_pair(seqno, data));
}

packet create_packet(const string& data)
{
    struct packet p {};
    p.seqno = 0;
    p.len = data.length() + sizeof(p.len) + sizeof(p.seqno);
    strcpy(p.data, data.c_str());
    return p;
}
```

```
void send_request(char* buf)
{
    int sret;
    fd_set readfds;
    struct timeval timeout {};

    FD_ZERO(&readfds);
    FD_SET(sockfd, &readfds);

    timeout.tv_sec = TIMEOUT;
    timeout.tv_usec = 0;

    int bytesSent = sendto(sockfd, buf, 600, 0,(struct sockaddr *)&serv_addr, sizeof(struct sockaddr));
    if (bytesSent == -1)
    {
        perror("couldn't send the packet");
    }

    sret = select(sockfd+1, &readfds, nullptr, nullptr, &timeout);

    if(sret == 0)
    {
        cout << "No response from the server! \n Resending packet..." << endl;
        send_request(buf);
    }
    else if(sret == -1)
    {
        perror("error in select");
        exit(1);
    }
}
```

Sample runs

The server folder has input.txt file

The client folder does not have the file



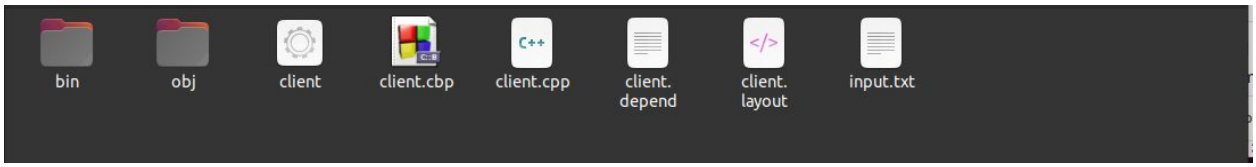
```
mrjoe@MrJoe: ~/MrJoe/C++Projects/Assignemt2/server
mrjoe@MrJoe:~/MrJoe/C++Projects/Assignemt2/server$ ./server
Server is start listening on port 8080
```

```
mrjoe@MrJoe:~/MrJoe/C++Projects/Assignemt2/client$ ./client
The Packet number 0 is received 0
data size = 499
The Packet number 1 is received 1
data size = 500
The Packet number 2 is received 2
data size = 500
The Packet number 3 is received 3
data size = 500
The Packet number 4 is received 4
data size = 500
The Packet number 5 is received 5
data size = 500
The Packet number 6 is received 6
data size = 500
The Packet number 7 is received 7
data size = 500
The Packet number 8 is received 8
data size = 500
The Packet number 9 is received 9
data size = 500
The Packet number 10 is received 10
data size = 500
The Packet number 11 is received 11
```

```
mrjoe@MrJoe: ~/MrJoe/C++Projects/Assignemt2/client
data size = 500
The Packet number 66 is received 66
data size = 500
The Packet number 67 is received 67
data size = 500
The Packet number 68 is received 68
data size = 500
The Packet number 69 is received 69
data size = 500
The Packet number 70 is received 70
data size = 500
The Packet number 71 is received 71
data size = 500
The Packet number 72 is received 72
data size = 500
The Packet number 73 is received 73
data size = 500
The Packet number 74 is received 74
data size = 500
The Packet number 75 is received 75
data size = 366
No response from the server!!. XxXxXxTerminatingxXxXxX
file size: 76 packets
mrjoe@MrJoe:~/MrJoe/C++Projects/Assignemt2/client$
```

```
mrjoe@MrJoe: ~/MrJoe/C++Projects/Assignemt2/server
mrjoe@MrJoe:~/MrJoe/C++Projects/Assignemt2/server$ ./server
Server is start listening on port 8080
file name: input.txt
data size = 499
83---0
ack received for packet 0
data size = 500
86---0
ack received for packet 1
data size = 500
77---0
ack received for packet 2
data size = 500
15---0
ack received for packet 3
data size = 500
93---0
ack received for packet 4
data size = 500
35---0
ack received for packet 5
data size = 500
86---0
ack received for packet 6
```

```
mrjoe@MrJoe: ~/MrJoe/C++Projects/Assignemt2/server
13---0
ack received for packet 68
data size = 500
57---0
ack received for packet 69
data size = 500
24---0
ack received for packet 70
data size = 500
95---0
ack received for packet 71
data size = 500
82---0
ack received for packet 72
data size = 500
45---0
ack received for packet 73
data size = 500
14---0
ack received for packet 74
data size = 366
67---0
ack received for packet 75
```


bin
obj
client
client.cbp
client.cpp
client.
depend
client.
layout
input.txt