Mohiuddin Mondal
Roll- 001910501043 (A2)
Computer Networks
3rd year 1st Sem BCSE UG
Assignment 7

Problem statement: Implement any two protocols using TCP/UDP Socket as suitable.
1.BOOTP 2.FTP 3.DHCP 4.BGP 5.RIP

Description: FTP is implemented here.

Design: A server and a client is implemented following FTP. FTP server hosts few files. Client can login and see the list of files. Then request for any of them and download it. Two connection is implemented to establish FTP. One is control connection, this lasts through out the session and using this client initiates commands to the FTP server.before requesting a file, client must set a port number for data connection and send it to server using PORT command. Following list of command is implemented here:

- USER
- PASSWORD
- PORT
- RETRIVE
- QUIT
- LIST
- NoSuchCommand

N.B. No channel is implemented here.

Input/Output : For control conection, data is sent as following structure:

```
class UserInput{
   FTPCommands command;
   char args[ARG_LENGTH];
}
```

Here ARG_LENGTH is a constant defined as a preprocessor (value 100). FTPCommands is an enum which contains above mentioned command as value. Args is the additional arguement with command like file name or user name etc..

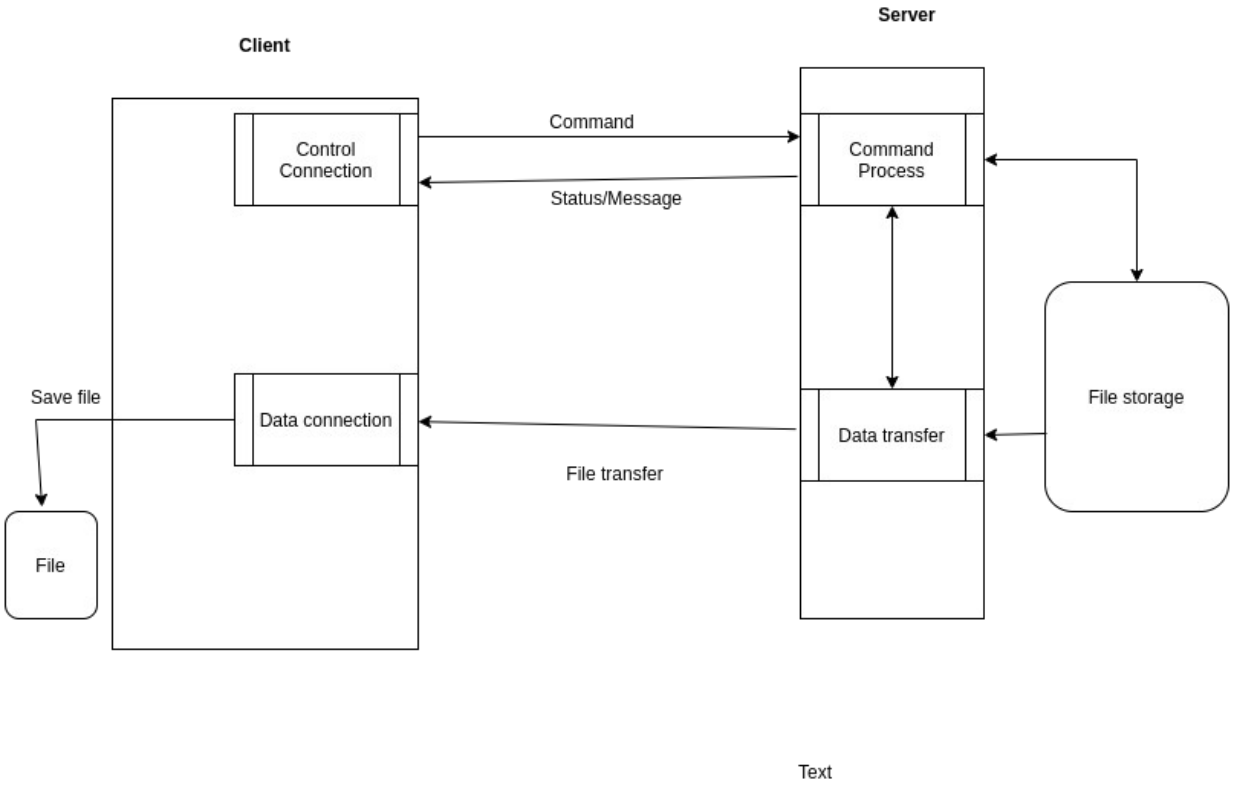
Server sends response in two way,

- It sends either a status code or a string of message. Status codes are defined as below:
  - FTPStatusCodeMessage = {
  - {220, "Service ready."},
  - {331, "Username Ok. Send password."},

- {230, "Login successfull, Kon'nichiwa "},
- {150, "Data connection opens shortly "},
- {200, "OK"},
- {250, "Request file action OK"},
- {226, "Closing Data connection"},
- {221, "Service closing. Goodbye senpai "},
- {404, "Unauthorized access "},
- {401, "Bad arguements recieved "},
- {500, "Internal server error, please try again "}
- };

- In case of data connection, whole file is sent as binary object using TCP.

Diagram:



Implementation:

FTPServer:

```
class FTPServer{
    const string dataPath="./DATA/";
    ComputerNode self;
    ConnectionSocket clientControl; // for control connection
    ConnectionSocket dataConnection; // for data connection
    bool isAuthorized;

    string getAvailbleDataList(){
        system("ls DATA > temp.txt");

        std::ifstream inFile("temp.txt");

        std::stringstream strStream;
        strStream << inFile.rdbuf();
        return strStream.str();
    }

    UserInput getClientInput(){
        vector<char> raw;
        self.recieveData(raw,clientControl);
        UserInput inp;
        memcpy((char*)&inp,raw.data(),sizeof(inp));
        cout<<"Clinet: "<<inp.getCommand()<<"  "<<inp.getArg()<<"\n\n";

        return inp;
    }
    void sendStatus(int status){
        vector<char> raw;
        raw.resize(sizeof(status));
        memcpy(raw.data(),(char*)&status,sizeof(status));
        self.sendData(raw,clientControl);
    }
public:
    FTPServer(int port):self(port){
        cout<<"FTP server started\n";
        isAuthorized = false;
    }

    void run(){
        clientControl = ComputerNode::acceptConnection(self.getSelfListenerSocket());
        sendStatus(220);

        UserInput input;
        do{
            input = getClientInput();
            switch(input.getCommand()){
                case USER:{
                    int cmp = input.getArg().compare(userName);
                    if(cmp == 0){
                        sendStatus(331);
                    }else{
                        sendStatus(401);
                    }
                } break;

                case PASSWORD:{
                    if (input.getArg() == password){
                        isAuthorized = true;
                        sendStatus(230);
                    }
                    else{
                        sendStatus(401);
                    }
                } break;

                case PORT:{
                    if(!isAuthorized){
                        sendStatus(404);
                        break;
                    }

                    int p = stoi(input.getArg());
                    dataConnection = ComputerNode::connectToSocket(p);
                    if(dataConnection.id!=-1){
                        sendStatus(150);
                    }else{
                        sendStatus(500);
                    }
                } break;
```

```
                case RETRIVE:{
                    if (!isAuthorized){
                        sendStatus(404);
                        break;
                    }

                    sendStatus(250);

                    vector<char> raw = getFileContent(dataPath+input.getArg());
                    self.sendData(raw,dataConnection);
                } break;

                case LIST:{
                    if (!isAuthorized){
                        sendStatus(404);
                        break;
                    }

                    sendStatus(200);

                    string dataList = getAvailbleDataList();
                    vector<char> raw;
                    raw.resize(dataList.length());
                    memcpy(raw.data(),dataList.c_str(),dataList.length());
                    self.sendData(raw,clientControl);
                } break;

                case QUIT:{
                    sendStatus(221);
                    close(clientControl.id);
                    close(dataConnection.id);
                    exit(0);
                }

                default:{
                    sendStatus(401);
                }

            }



        }while(input.getCommand() != QUIT);


    }

};
```

FTPClient:

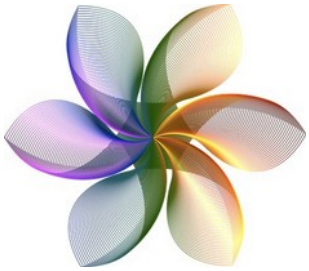
```
class FTPClient{
    ComputerNode self;
    ConnectionSocket clientControl; // for control connection
    ConnectionSocket dataConnection; // for data connection
    string fileName;

    void saveData(vector<char>& data){
        ofstream output(fileName, ios::out | ios::binary);
        output.write(data.data(),data.size());
        output.close();
        cout << "\tDATA_Connection> File saved to "<<fileName<<"\n";
    }

    vector<char> makeRawData(UserInput ui){
        vector<char> raw;
        raw.resize(sizeof(ui));
        memcpy(raw.data(),(char*)&ui,sizeof(ui));
        return raw;
    }

    string getStatus(vector<char>& response){
        int status;
```

```
        memcpy((char*)&status,response.data(),sizeof(status));
        return FTPStatusCodeMessage.at(status);
    }
    void startDataConnectionThread(){
        // spawn data recieving socket
        thread datahandling(&FTPClient::handleDataConnection, this);
        datahandling.detach();
    }
    string getMessage(vector<char>& response){
        return string(response.begin(),response.end());
    }
public:
    FTPClient(int port):self(port){
        cout<<"FTP client initiated\n\n";
    }

    void handleDataConnection(){
        while(true){
            dataConnection =
       ComputerNode::acceptConnection(self.getSelfListenerSocket());
            vector<char> data;
            self.recieveData(data,dataConnection);
            saveData(data);
            break;
        }
    }



    void run(){
        clientControl =  ComputerNode::connectToSocket(FTP_PORT);
        vector<char> raw;
        self.recieveData(raw, clientControl);
        cout << getStatus(raw) << "\n\n";

        while(true){
            cout<<"ftp> ";
            string inputString;
            getline(cin,inputString);
            UserInput ui(inputString);

            if (ui.getCommand() == RETRIVE) fileName = "recieved_" + ui.getArg();

            raw = makeRawData(ui);
            self.sendData(raw,clientControl);
            self.recieveData(raw,clientControl);
            cout<<getStatus(raw)<<"\n\n";

            switch(ui.getCommand()){
                case LIST:{
                    self.recieveData(raw, clientControl);
                    cout << "Server files:\n";
                    cout << getMessage(raw) << "\n\n";
                } break;

                case QUIT:{
                    cout << "Shutting down client   \n";
                    exit(0);
                } break;

                case RETRIVE:{
                    startDataConnectionThread();
                }

            }
        }
    }

};
```

Test cases:

Server has data folder consisting of few image files. These are sent to client on request.

These are saved in ftp:

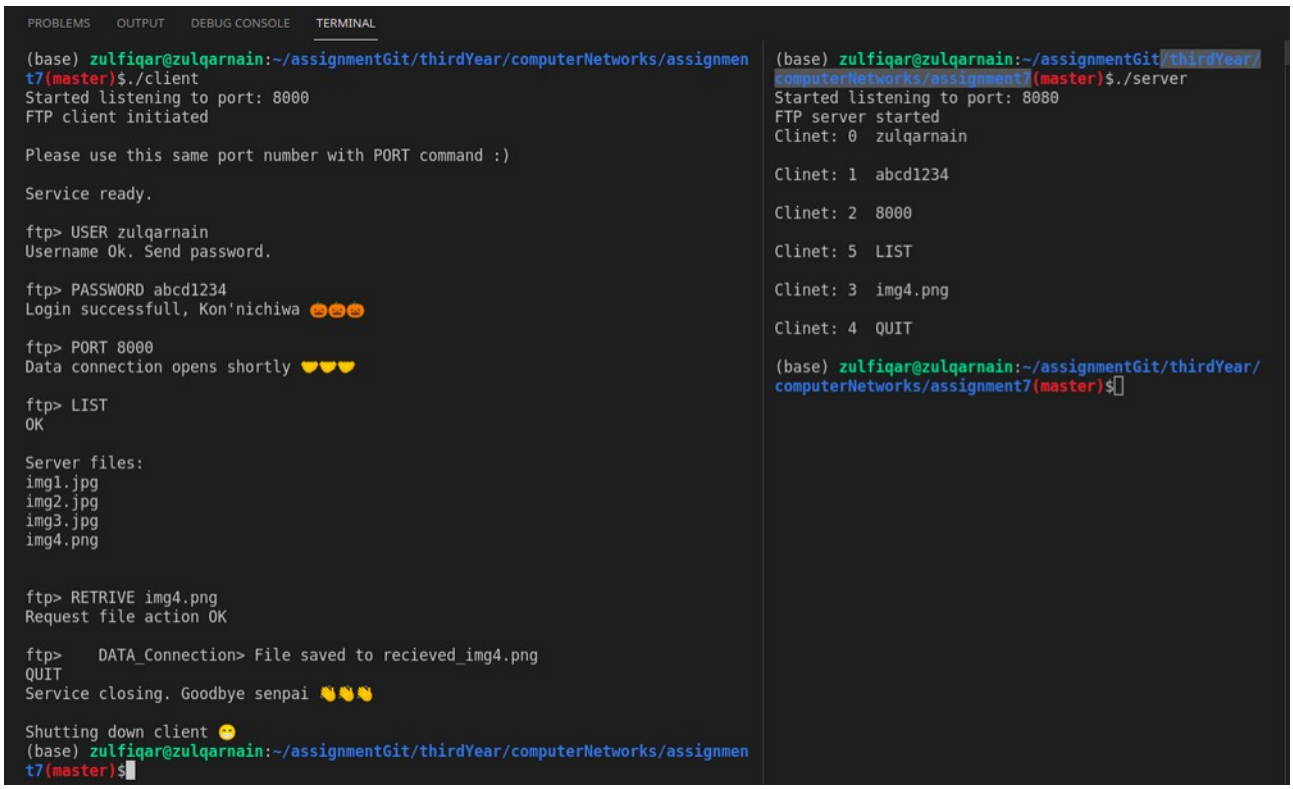


Following one is recieved and saved in client:

Reuslts and Analysis:

Results:

Following screenshot shows the interaction between client and server. Left terminal shows the client side:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
(base) zulfiqar@zulqarnain:~/assignmentGit/thirdYear/computerNetworks/assignment7(master)$./client
Started listening to port: 8000
FTP client initiated
Please use this same port number with PORT command :)
Service ready.
ftp> USER zulqarnain
Username Ok. Send password.
ftp> PASSWORD abcd1234
Login successfull, Kon'nichiwa
ftp> PORT 8000
Data connection opens shortly
ftp> LIST
OK
Server files:
img1.jpg
img2.jpg
img3.jpg
img4.png
ftp> RETRIVE img4.png
Request file action OK
ftp> DATA_Connection> File saved to recieved_img4.png
QUIT
Service closing. Goodbye senpai
Shutting down client
(base) zulfiqar@zulqarnain:~/assignmentGit/thirdYear/computerNetworks/assignment7(master)$
(base) zulfiqar@zulqarnain:~/assignmentGit/thirdYear/computerNetworks/assignment7(master)$./server
Started listening to port: 8080
FTP server started
Clinet: 0 zulqarnain
Clinet: 1 abcd1234
Clinet: 2 8000
Clinet: 5 LIST
Clinet: 3 img4.png
Clinet: 4 QUIT
(base) zulfiqar@zulqarnain:~/assignmentGit/thirdYear/computerNetworks/assignment7(master)$