Khulna University of Engineering & Technology

Department of Electronics and Communication Engineering

Report on ECE 4110 Computer Networking Lab

Project Title

File Transfer Program Using Socket Programming

Submitted by

Pritom Mojumder

Roll : 1509053

Date of Submission

19.06.2019

File Transfer Program Using Socket Programming

Introduction

This project has been done for the purpose of **ECE 4110 Computer Networks Lab** course. Initially the project proposal was to build an FTP program. The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server. The specification was written by Abhay Bhushan and published as RFC 114[1], later replaced by a TCP/IP version, RFC 959 [2]. To follow the protocol RFC and build such a program is beyond the time limit for this course. So, a simple file transfer program has been created using some basic ideas of the standard protocol.

Motivation

FTP is one of the popular ways for file/data exchange. Working on this project will provides us some basic understanding in computer network programming. We have learnt socket programming for the purpose of this project.

Contribution

This project work has been performed by roll 1509037 and 1509053.

Contribution by 1509037: Client-side programming.

Contribution by 1509053: Server-Side programming.

Tools Used

i. Python Programming Language
ii. Socket Programming
iii. Visual Studio Code

Methodology

Sockets and the socket API have been used to send messages across a network. Socket API provides a form of inter-process communication (IPC). We have used Python's socket module to write our own client-server applications. We have used the Object-Oriented Programming (OOP) concept to create a module for the file transfer system so that, same program can be used as server or client. We have named our program **Quantum.**

This project follows connection-oriented communications where communication parties have different roles. One party waits for incoming connections; this party is referred to as "server". Another party initiates connection; this party is usually referred to as "client".

Basically, we have implemented the TCP socket flow showed in Fig (1).

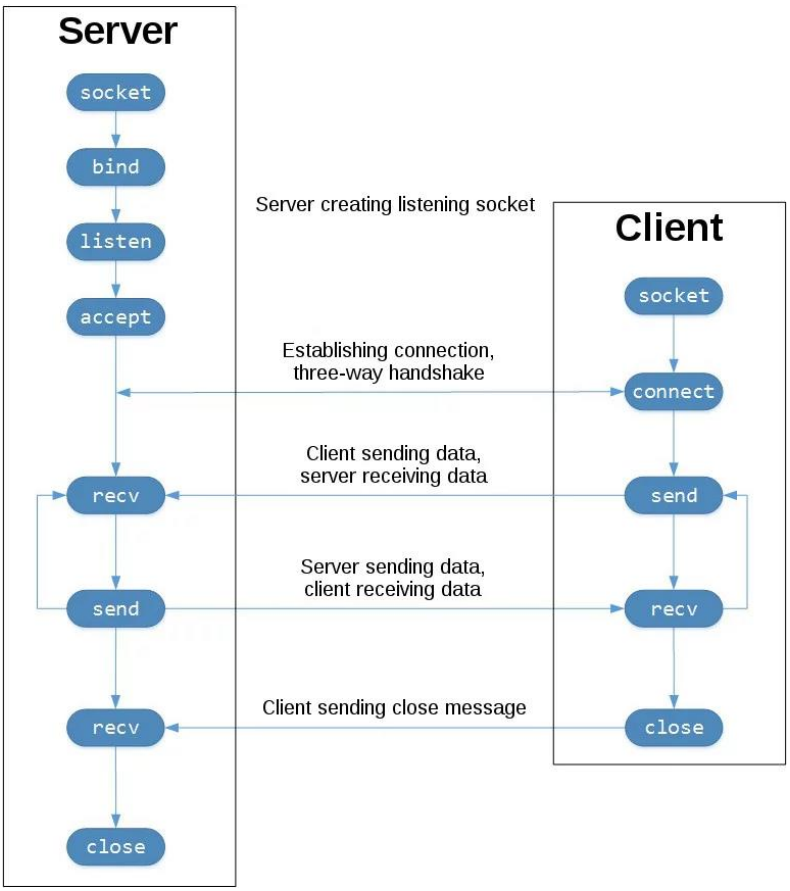

Fig (1): TCP Socket Flow

We have added some features also by exploiting the functionalities of Python's OS module [3], which provides a portable way of using operating system dependent functionality. Those features are:

i. Facility to know the valid commands for specific responses from server.
ii. Facility to know downloadable files from server.
iii. Facility to select specific file for downloading.

To create a server an user need to import Quantum as module and server can be initiated by single line like:

server = Quantum()

To connect with a server this the line will be from client side:

cl = Quantum(host, port, server=False)

So this program can be developed and used as API.

Outcome

We have tested our program both locally and remotely. For each case it has worked successfully. The results we have achieved from the project has shown in Fig(2). We have implemented three commands for client : help, file and download. A client can easily ask the server for existing downloadable files with 'file' command. Client can send a download request with 'download' command. With 'help' command client can easily know all valid commands.

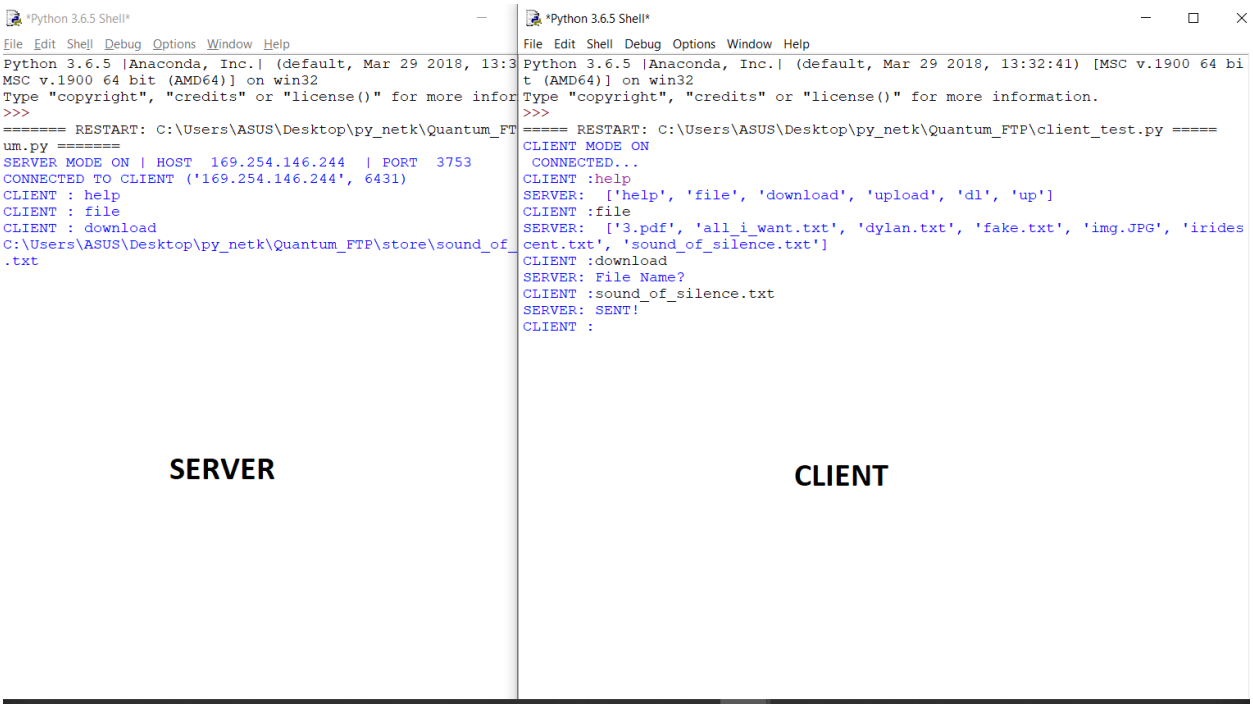

Fig (2): Program Operation Example

Conclusion

From this project work we have been introduced with socket programming. We have checked the program for different two conditions. But sometimes *TimeoutError: [WinError 10060]* which means a connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond. The probable reason for this error is firewall is prohibiting port access. But we can conclude that we have finished our project successfully and acquired knowledge on socket programming and computer networking.

References:

1. A. Bhushan, “A FILE TRANSFER PROTOCOL”, RFC 114, April 1971
2. J. Postel , “FILE TRANSFER PROTOCOL (FTP)”, RFC 959, October 1985
3. https://docs.python.org/3/library/os.html

Code: **Main Program**

```
1.  import socket
2.  import os
3.
4.  class Quantum:
5.      def __init__(self,host=socket.gethostbyname(socket.gethostname()),port=5000,sok=socket.sock
    et(),server=True):
6.          self.host = host
7.          self.port = port
8.          self.sok = sok
9.
10.         if server==False:
11.             self.client()
12.         else:
13.             self.server()
14.
15.     def server(self):
16.         '''server program'''
17.         print("SERVER MODE ON | HOST ",self.host,' | PORT ',self.port)
18.         self.sok.bind((self.host,self.port))
19.         self.sok.listen(10)
20.
21.         ops = ['help','file','download','upload','dl','up']
22.         conn,addr = self.sok.accept()
23.         print('CONNECTED TO CLIENT',addr)
24.
25.         while True:
26.             ###communication...
27.             r = conn.recv(1024).decode() #rx
28.             print('CLIENT :',r)
29.
30.             if r in ops:
31.                 if r == 'help':
32.                     msg = str(ops).encode()
33.                     conn.send(msg)
34.                 elif r == 'file':
35.                     top = str(os.getcwd())+'\\store'
36.                     file_name = ''
37.                     for root, dirs, files in os.walk(top):
38.                         file_name = str(files).encode()
39.                     conn.send(file_name)
40.                 elif r == 'download' or 'dl':
41.                     '''download operation'''
42.                     buffer = 0 #buffer size variable
43.                     msg = str('dl_ack').encode() #download ack
44.                     conn.send(msg)               #tx
45.                     r = conn.recv(1024).decode() #rx file name
46.                     path_var = str(os.getcwd())+'\\store'+'\\'+r
47.                     print(path_var)
48.                     if os.path.exists(path_var):
49.                         msg = str('OK').encode()
50.                         conn.send(msg)          #tx 'OK'
51.                         r = conn.recv(1024).decode() #rx 'OK'
52.                         size = os.path.getsize(path_var)
53.                         msg = str(size).encode()
54.                         conn.send(msg)   #tx
55.                         r = conn.recv(1024).decode() #rx 'OK'
56.                         #file sending...
57.                         file = open(path_var,'rb')
```

```
                        file_data = file.read(size)
                        conn.send(file_data) #tx


                    else:
                        msg = "FILE DOES NOT EXIST".encode()
                        conn.send(msg)
                else:
                    print("SERVICE CURRENTLY UNAVAILABLE")


            else:
                conn.send(str('INVALID COMMAND').encode())

    def client(self):
        '''client program'''
        self.sok.connect((self.host,self.port))
        print("CLIENT MODE ON\n CONNECTED...")
        while True:
            msg = input(str('CLIENT :')).encode()
            self.sok.send(msg)                  #tx
            r = self.sok.recv(1024).decode()    #rx

            if r == 'dl_ack':
                '''download operation'''
                print('SERVER: File Name?')
                msg_file = input(str('CLIENT :')).encode()
                self.sok.send(msg_file)          #tx file name
                r = self.sok.recv(1024).decode() #rx  'OK'
                if r == 'OK':
                    msg = str('OK').encode()
                    self.sok.send(msg)  #tx 'OK'
                    r_size = int(self.sok.recv(1024).decode())#rx
                    self.sok.send(msg) #tx ok
                    #file receive
                    path_var = str(os.getcwd())+'\\download'+'\\'+msg_file.decode()
                    file = open(path_var,'wb')
                    file_data = self.sok.recv(r_size+100)  #rx file
                    file.write(file_data)
                    file.close()
                    print('SERVER: SENT!')

                        else:
                            pass

                    else:
                        print('SERVER: ',r)


        if __name__ == '__main__':
            ex = Quantum()
```

Creating Server: server_test.py

```
from quantum import Quantum

server = Quantum(host='192.168.43.126')
```

Creating Client:client_test.py

```
from quantum import Quantum

cl = Quantum(host='192.168.43.126',port=3753,server=False)
```