BLG337E - Homework 2

Hakan Duran 150200091

November 12, 2023

Abstract

The project about client and server communications and socket programming.

1 Introduction

This project was about socket programming to achieve successful client-server communication. I've learnt about using socket library in Python and different usages of socket functions in client and server.

2 Code

I wrote two Python code.

2.1 Client code

```
import socket

# Client configuration
SERVER_HOST = "127.0.0.1"
UPLOAD_PORT = 12345
DOWNLOAD_PORT = 12346
BUFFER_SIZE = 1024

def show_menu():
    print("\n1. Create a text file")
    print("2. Upload the file to the server")
    print("3. Download the file from the server")
    print("4. Compare files")
    print("5. Exit")
    choice = input("Enter your choice (1-4): ")
    return choice

def menu_upload():
    print("\n1. Use created file")
    print("2. Select another file")
    choice = input("Enter your choice (1-2): ")
    return choice

def upload_file(client_socket, cfile):

    choice = menu_upload()

    if choice == "1":
        if cfile.name != "0.fail":
            file_name = cfile.name
            with open(file_name, "rb") as file:
                file_data = file.read()
        else:
            print("Please create a file!")
            return cfile
    elif choice == "2":
        file_name = input("Enter the path of the file to upload: ")
        with open(file_name, "rb") as file:
```

```
            file_data = file.read()

    #Created file will be sent to the server
    client_socket.sendall(file_data)
    print(f"\nFile '{file_name}' sent to the server\n")

    return file

def download_file(client_socket):
    file_name = "student_info.txt"

    #File name is sending to the server
    client_socket.sendall(file_name.encode())

    #Server sends the requested file to the client
    file_data = client_socket.recv(BUFFER_SIZE)
    with open(file_name, "wb") as file:
        file.write(file_data)
    print(f"\nFile '{file_name}' downloaded from the server\n")
    return file

def create_file():
    file_name = input("Enter the name of the text file: ")
    content = input("Enter the content of the text file: ")
    with open(file_name, "w") as file:
        file.write(content)
    print(f"\nFile '{file_name}' created.\n")
    return file

def compare(ufile, dfile):
    with open(ufile.name) as file_1:
        file_1_text = file_1.read()

    with open(dfile.name) as file_2:
        file_2_text = file_2.read()

    if file_1_text==file_2_text:
        print("\nThe files are same.\n")
    else:
        print("\nThe files are not same.\n")



def main():

    file = open('0.fail', 'w')

    # Create sockets for upload and download
    upload_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    download_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Connect to the server for upload and download
    upload_socket.connect((SERVER_HOST, UPLOAD_PORT))
    download_socket.connect((SERVER_HOST, DOWNLOAD_PORT))

    print(f"Connected for file upload to {SERVER_HOST}:{UPLOAD_PORT}")
    print(f"Connected for file download to {SERVER_HOST}:{DOWNLOAD_PORT}\n")

    while True:
        choice = show_menu()

        if choice == "1":
            file = create_file()
        elif choice == "2":
            ufile = upload_file(upload_socket, file)
        elif choice == "3":
            dfile = download_file(download_socket)
        elif choice == "4":
            compare(ufile, dfile)
        elif choice == "5":
            print("Exiting")
```

```
110             break
111         else:
112             print("Invalid choice. Please enter a number between 1 and 4.")
113 
114     # Close the connections
115     upload_socket.close()
116     download_socket.close()
117 
118 if __name__ == "__main__":
119     main()
```

I only used socket library for client side. Configurations of server IP, upload/download ports and buffer size can be found in first lines.

In main function, I have created a file named 0.fail and this file is going to be used for a lot of purposes later. Then, I have created two sockets at line 87-88. AF_INET and SOCK_STREAM indicates TCP and IPv4 is gonna be used for these sockets. At lines 94-95, sockets are connected to the server.

In the loop of line 97, there are options for creating file, upload, download and comparing files. We can start with the creating the file.

Creating a file is simple and the created file can be used in upload function. In upload function, we have two choices which indicates if you want to use pre-created file or create/select a new file. In both cases, the last operation is file_data = file.read(), and file_data will be uploaded to the server with the usage of sendall(file_data) function.

After uploading the file, server will store the file and waits for download process which will be initialized by the client. In download_file function, file_name is student_info.txt, because it is how server stores them. That txt file's name is sending to the server by encoding. Server sends file and client download it by using file_data = client_socket.recv(BUFFER_SIZE) and stores it in a file.

In the final, we can compare files, in that way, we can understand if successful transmission has happened.

At line 115-116, sockets are closed.

2.2 Server code

```
1  import socket
2  
3  # Server configuration
4  SERVER_HOST = "127.0.0.1"
5  UPLOAD_PORT = 12345
6  DOWNLOAD_PORT = 12346
7  BUFFER_SIZE = 1024
8  
9  # Function to handle file download
10 def download_file(client_socket):
11     # Received file name which comes from client
12     file_name = client_socket.recv(BUFFER_SIZE).decode()
13     try:
14         with open(file_name, "rb") as file:
15             file_data = file.read()
16             #Requested file is sending to the client
17             client_socket.sendall(file_data)
18             print(f"File '{file_name}' sent to the client")
19     except FileNotFoundError:
20         print(f"File '{file_name}' not found on the server")
21 
22 # Create socket for file upload
23 upload_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
24 upload_socket.bind((SERVER_HOST, UPLOAD_PORT))
25 upload_socket.listen(1)  # Listen for one incoming connection
26 
27 # Create socket for file download
28 download_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
29 download_socket.bind((SERVER_HOST, DOWNLOAD_PORT))
30 download_socket.listen(1)
31 
32 print(f"[*] Listening for file upload on {SERVER_HOST}:{UPLOAD_PORT}")
```

```
33 print(f"[*] Listening for file download on {SERVER_HOST}:{DOWNLOAD_PORT}\n")
34
35 # Accept connections for upload and download
36 upload_client_socket, upload_client_address = upload_socket.accept()
37 download_client_socket, download_client_address = download_socket.accept()
38
39 # Upload process
40 file_data = upload_client_socket.recv(BUFFER_SIZE)
41 file_name = "student_info.txt"
42 with open(file_name, "wb") as file:
43     file.write(file_data)
44 print(f"File received and saved as {file_name}")
45
46 # Download process
47 print("Server has waiting for client to start download process\n")
48 download_file(download_client_socket)
49
50 # Close the connections
51 upload_client_socket.close()
52 download_client_socket.close()
53 upload_socket.close()
54 download_socket.close()
```

I only used socket library for client side. Configurations of server IP, upload/download ports and buffer size can be found in first lines.

I have created two sockets at line 23 and 27. AF_INET and SOCK_STREAM indicates TCP and IPv4 is gonna be used for these sockets. At line 24 and 29, we are binding sockets with specific network address. At line 25 and 30, sockets are waiting for incoming connections.

Line 36 and 37 is for accepting incoming connections. Also, new sockets are created and client addresses has stored.

At line 40, "upload process" for client has started, of course it is download process for server. Received file saved as student_info.txt. Then server starts to wait for client to start download process.

Download function waits for the name of requested file and then, requested file is sent to the client by using sendall() function. Between the line of 51 and 54, sockets are closed.

3 Comments

I have recorded a video about how my client and server codes work and what happens when uploaded and downloaded files differ. Socket programming is crucial for networking and cybersecurity fields, i am glad that type of homework is given.