Darahas kopparapu, Shubhankar Bhadra

Professor Name: Dr. Venkata ramana Badrala

Computer networking lab

15 September 2021

Multi-Client File transfer

We have created a prototype of a multi-client file transfer model where multiple clients can send text files. Our whole project is done over python using the socket library which is in-built in python.

There are 2 main files in the application, server.py and client.py. Each connection to the server by a client is through a socket which is launched as a new independent thread. On initial connection to the server by the client, a dictionary is maintained for storing the IP address of the client as key along with the instance as value.

2 independent clients ca ncommunicate to each other via this dictionary by accessing the instance of the connection through the IP address as the key. There are some commands which the user needs to input into the terminal such as “list”, “upload”, “receive”, “stop” for the client to perform necessary actions.

Commands and their functionality

list - To list out all the active clients in the form of IP address and port number connected to the server.

upload - For uploading the specified file, user needs to enter, specified file path, target client IP address and Port number.

receive - To start the client listening for any incoming files

stop - To disconnect from the server

File sending Protocol

This protocol uses sockets of python which in its core uses the TCP protocol. For the purpose of performing very crude actions such as “list”, we only need to send a header which has the content of:-

- Command
- IP address and Port
- Data

For sending files, a header is initially sent, to initialize the process of creation of a file from the receiver side. Proceeding that, Data is sent in the form of bytes and a final header is sent to close the file transferring process. The header is in the form of JSON and this JSON object is serialized using pickle library of python. We have set our chunk of each packet to be of 2 KB.

In case of “list” a header is sent to the server which in return responds with the number of clients and their respective IP addresses and Port Numbers.

Whereas, for “recieve”, the client itself goes into a listening mode where it receives the respective header with the target file name followed by the data of the file.

In case of “upload”, client sends the header to the client via the server, by using the data from the header and in return the target client responds with am “OK” on successful receiving of

the header. The content of the file is then sent in chunks and the receiver reassembles these chunks on their side. The end of the chunk is followed by another header indicating successful file transfer, and the client then stops listening.

Screenshots

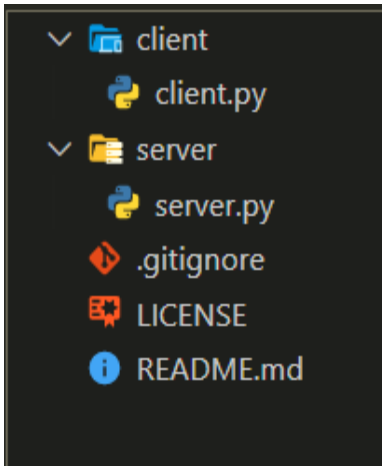

Fig 1. File structure

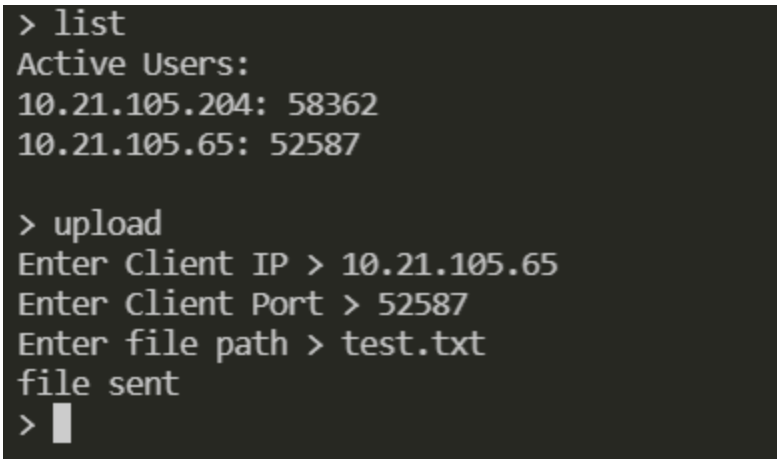

Fig 2. Client 1 sending via upload command

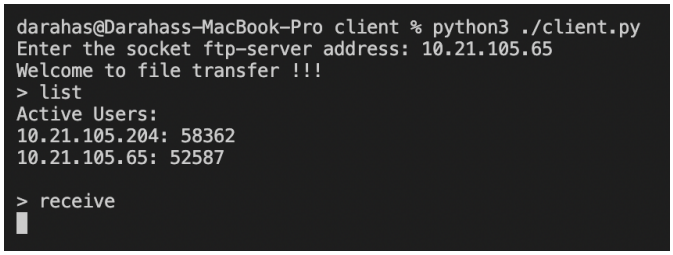

Fig 3. Client 2 in listening mode via receive command

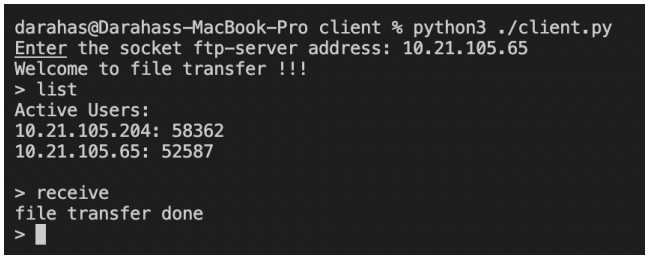

Fig 4. Client 2 received the file completely

Conclusion

We are able to successfully send a text file of any size directly from one client to the other without storing it in the server using utf-8 encoding type and only one file at a time. For sending images, we need to use base64 encoding and put conditional statements, when an image of aby format is sent.

The limitations from our side as of now are sending image files, and multiple files at a time.