COMP3331 Assignment Report
Nian Li z5241097

1. Reference Acknowledgement

A small piece of code is cited from https://stackoverflow.com/a/20007570
The author is “freakish”.
The function below is the modified version of the original function written by “freakish”. I made some change to combine “byte to number” and “number to byte” together in the same function. The reason I cited this function is because I need to convert the size into a fixed numbers of size, so socket.revc() can always receive exact amount of data.

```
def byte_number_converter(parameter, mode):
    if mode == 0:
        number = 0
        i = 0
        while i < 4:
            number += parameter[i] << (i*8)
            i += 1
        return number
    elif mode == 1:
        byte = bytearray()
        byte.append(255 & parameter)
        i = 0
        while i < 3:
            parameter = parameter >> 8
            byte.append(255 & parameter)
            i += 1
        return byte
```

the code below is also cited from “freakish” from the same thread. I modify it so instead of writing file, it will store the data in a string. The purpose of this citation is to make sure the receiver knows when to stop receiving data.

```
while curr_size < file_size:
                line = clientSocket.recv(1024)
                if not line:
                    break
                if file_size < len(line) + curr_size:
                    line = line[:file_size - curr_size]
                curr_size += len(line)
                thread += line.decode("utf-8")
```

```
thread = thread.split("\n")
```

The idea from the code below are repeatedly used a few times in my assignment, and in-code citation are used where it appears.

2. The design of the program:

Server: use multithreading to handle multiple clients, create a new thread for each new client. The main thread is in responsible for accepting new collection. Functionalities are distributed to different functions in server. The while loop will read the command from client and forward the command to the function that can handle the situation.

Client: client will check the arguments of each command locally before sending it to server. The client has two sockets and two threads, one for sending the command to the server, one for continuously connecting and closing in order to check if server is alive.

When the hosts are transferring some important data such as file, or protocol instruction, the size of the data will be told to the other end before being sent.

3, Possible improvement:

There are some duplicates in the code, which can be written assembly in a function. In some situation, the receiver doesn't know how much data it will receive. It might raise error if the server is extensively visited by clients rapidly.

4.Good Things In the assignment

1.Lock() is used to prevent possible conflict between threads, although the chance is very low.
2.File transfer is very stable (as the size of the file will be told to the receiver be it starts to receive file). And it has been tested by some big size files.
3. most of the edge cases are covered, even some of the assumptions are made in the spec.
4.successfully handled the SHT functionality, but creating extra socket and thread in client, and make it try to connect to the server every 0.5 second and disconnect right after the establishment of the connection.

5.Applocation layer format

Client and server communicate by sending messages that are 2 bytes long between each other. For example before sending a file, b"OK" will be send to acknowledge the receiver to get ready. Messages like b"NE" stand for "not exist", which will tell the other end that the thing you are asking for doesn't exist.

6.Overall comment on the assignment

All of the functionalities are implemented and successfully tested including the error handling. The style could have been done better, it's mainly because I didn't think

enough before I started. And I could have started the assignment earlier, so I will have more time to learn socket programming and multithreading as I found they are very interesting.