CS371 Project Report

Abstract:

As cloud computing develops, a tool called remote folder has become a very common and convenient. User is able to upload file(s) and folder(s) to a remote location, and able to perform all the operations like a local folder, such as view all the details of the file(s), download and delete files. The purpose of my project is to build a remote folder by establishing a client and server, connecting them using socket.

Introduction:

The remote folder was built using python with python Socket. The first step of my design is to recognize the functionality difference between the server and client. While they will be both sending and receiving files, they function quite differently. Client must enable the user to type in command in shell and execute it, and user-friendly environment must be implemented such as error messages and process updates. While server don't have these concerns, it must send client various ack to support all the features.

Project design and implementation:

```
Pseudo-code:
    Client:
        Specify host and portal
        Instantiate new socket
        Connect to the socket with the host and portal
        Printing user instructions
        Taking input of command
        Send command to server
        If(upload):
            Upload operation
        If(download):
            Download operation
        If(view all files)
            View files operation
        If(delete)
            Delete operation
        Asking user command input
```

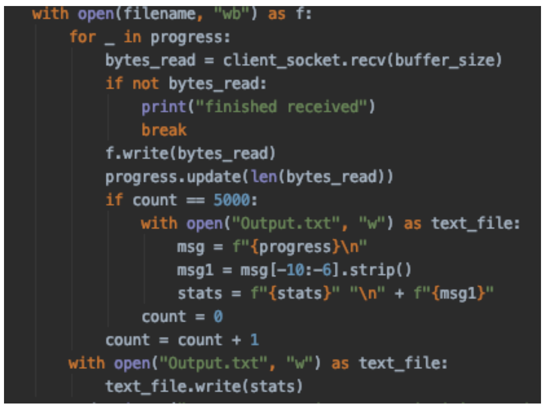
Server:

```
Specify host and portal
Instantiate new socket
Bind the socket with the host and portal
Listens to connection
Waiting for command input by client
If(upload):
    Upload operation
If(download):
    Download operation
If(view all files)
    View files operation
If(delete)
    Delete operation
```

In the project I used a library call tqdm, which gives you a progress bar and use algorithm to predict remaining file transfer time base on the download speed and filesize. The tqdm will out put a progress bar alone with the download speed, I was able to convert the progress bar into string, extract the download speed and output it to a file for network performance evaluation. For the 4 core features I divided them into individual block of code, even though they interact with the server using similar method. I believe it is better to separate them for better implementation and debugging.

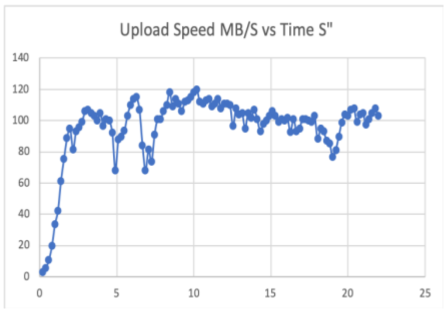
Experiemnt:

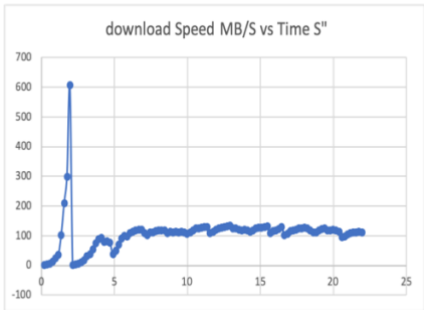
The project started with very simple functionality, such as connecting to the server, sending simple ack, and gradually developed to live chat server, then file transfer, lastly implement the core functionalities one by one. One problem that I had was the tqdm progress bar updates 60 times/sec, and the output was too big for network performance evaluation. I was able to record correct amount by outputting data every 5000 iterations, which is close to 1 output/sec.



```python
with open(filename, "wb") as f:
    for _ in progress:
        bytes_read = client_socket.recv(buffer_size)
        if not bytes_read:
            print("finished received")
            break
        f.write(bytes_read)
        progress.update(len(bytes_read))
        if count == 5000:
            with open("Output.txt", "w") as text_file:
                msg = f"{progress}\n"
                msg1 = msg[-10:-6].strip()
                stats = f"{stats}" "\n" + f"{msg1}"
            count = 0
        count = count + 1
    with open("Output.txt", "w") as text_file:
        text_file.write(stats)
```

Below are the network performance measured using method above





Conclusion:

As I proceeded to finish up this project, I realized all the theory we learned in class has very practical usage in real life, and computer networking is a fundamental cornerstone of many existing technology today. And these technologies such as internet has drastic influence on the development on mankind. I believe as computer networking develops, there will be more revolutional changes occurs to our society.