# COMP3331- Assignment Report

BitTrickle By- Devaansh Kumar (z5474396)

## 1) Language and Code Organization

The whole project was completed using Python. The code includes two main files, which are server.py and client.py. The files handle the server and the clients respectively. The directory has these two files, and other than this it has a folder named server which holds the credentials.txt. This text file holds all the data needed for authentication, that is the usernames and passwords. The directory also has folders of all the active users and the folders contain all the data which the users own.

A sample directory structure would include (used for testing as given in the spec):

```
assign
  - client.py
  - server.py
  - hans
    |--  Assignment_v1.0.pdf 17
  - server
    |-- credentials.txt
  - vader
    |-- BitTrickle.mp4
    |-- rfc768.txt
  - yoda
```

The directory doesn't contain any other complex files or additional helper files, moreover, it doesn't contain any makefiles also since python is straightforward and can be executed directly from the command line.

## 2) Overall Program Design

The overall design of the program includes two main components:

1. **server.py:** Manages all the clients by keeping track of active users, managing published files and handling all the incoming messages from the clients using UDP.
2. **client.py:** Handles user input and communicates with the server to perform different operations such as authentication, checking active users and published files, publishing and unpublishing files, searching and more. The client also includes a TCP listener to handle incoming file requests.

The server is responsible for maintaining the state, authenticating users and handling various commands sent by the clients, however, the clients interact with the server over UDP and establish peer to peer connection to exchange files over TCP.

## 3) Data Structure Design

1. **active_users:** The server maintains a dictionary to track the currently logged-in users. This dictionary stores each user's information, such as their address, TCP port, and the timestamp of their last heartbeat.
2. **credentials:** User credentials are stored in this dictionary. This dictionary is populated by reading from a text file 'credentials.txt' inside the server folder when the server starts.
3. **published_files:** The dictionary is used to track the published files along with their owners. Each file maps to another dictionary that keeps track of the users who own the file and its active status.

## 4) Application Layer Protocol Message Format(s)

The server and clients communicate using a custom application layer protocol over UDP for control messages and TCP for file transfers.

Authentication: 'auth <username> <password> <tcp_port>'
Heartbeat: 'heartbeat <client_address>'
Get File: 'get <filename>'
List Active Peers: 'lap'
List Published Files: 'lpf'
Publish File: 'pub <filename>'
Search Files: 'sch <substring>'

Unpublish File: ‘unp <filename>’
Exit: ‘xit’

The responses from the server include appropriate success or error messages, as well as detailed information (e.g., available peers, file owners, etc.).

## 5) Known Limitations

1. If a user tries to download a file from another peer and the peer does not have the file, an appropriate message "File not found" is sent back. This is handled at both the client and server side, but this might lead to unexpected behavior if the file is renamed after being published.
2. The server host is hard coded to ‘127.0.0.1’, which restricts the solution to a local network setup. Hence, modifications are needed for deployment in broader network scenarios.
3. The file transfer we have implemented is conducted using plain TCP with no encryption, making them vulnerable to interception. So, this implementation is not a real-life application and is only limited to the purpose of this assignment.

## 6) Code Borrowing and References

https://docs.python.org/3/library/socket.html
https://docs.python.org/3/library/threading.html
https://beej.us/guide/bgnet/
https://www.geeksforgeeks.org/socket-programming-python/

Used all these online references for learning sockets and threading, and using all the functions necessary in the code like socket, bind, accept, listen, etc. Also used course material for learning TCP protocol implementation for peer-peer connection.