4C03 Programming Assignment 2

Justin Staples – 001052815

1 Test Case Selection

The rationale for test cases is given only for methods that I had to add my own code to. More detail about my implementation can be found in my file synchronizer Python comments.

1.1 get_file_info()

The main function of this method is to search the current directory and find all the files that are relevent for the tracker. For this method, I am mainly trying to test that it grabs all of the correct files while not grabbing any of the incorrect ones.

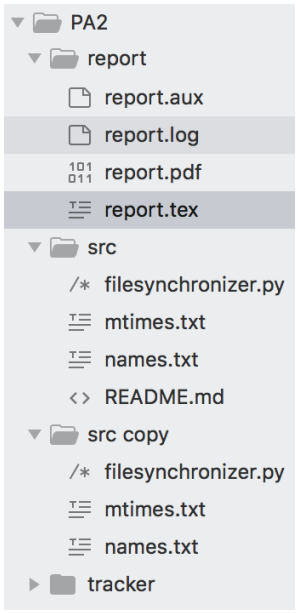

Figure 1: My directory structure shows that there is one instance of tracker and two instances of file synchronizer.

Shown above is my setup for a test. Inside the `src` folder, the only valid file is `readme.md`, because the python file is not allowed and the text files are automatically generated by the program. So we would expect there to be only file info on the `readme.md` document in this case. Figure 1 shows the desired output.

1.2 get_next_available_port()

For this method, I would expect it to return the successor of the input port in the event that I am working with what is known to be a contigious block of free ports. That is to say that if I know that port 8000 - 8080 are free, then I should expect to get 8003 when I use port 8002 as the initial port. The output of such a test is given below in Figure 3.

```
[piola:src staplejw$ python filesynchronizer.py 127.0.0
.1 8020
--------------------
[{'name': 'README.md', 'mtime': 1520699220}]
--------------------
Waiting for connections on port 8000
connect to:127.0.0.1 8020
--------------------
[{'name': 'README.md', 'mtime': 1520699220}]
--------------------
```

Figure 2: The result of the test.

```
[piola:src staplejw$ python filesynchronizer.py 127.0.0
.1 8000
Initial port: 8000
synchronizer port: 8001
--------------------
```

Figure 3: The result of the test.

1.3 process_message()

The main function for this method is to properly receive the filename and then retrieve the correct file contents. To test that this works, I focus on a simple case. Again, as before, the `readme.md` file is being transferred from the file synchronizer instance inside of the `src` folder to the one in the `src copy` folder. I have told this method to print the file contents that it is sending to the other peer. The desired output is shown in the terminal printout.

```
--------------------
connect to:127.0.0.1 8004
--------------------
[{'name': 'README.md', 'mtime': 1520699220}]
--------------------
--------------------
[{'name': 'README.md', 'mtime': 1520699220}]
--------------------
--------------------
Add your code in this directory
--------------------
connect to:127.0.0.1 8004
--------------------
[{'name': 'README.md', 'mtime': 1520699220}]
--------------------
```

Figure 4: The result of the test.

1.4 sync()

For the `sync` method, we want to test that a new user becomes properly synchronized when they sign on. Right away, they should receive copies of files from other active peers. The test case scenario is as before. In this particular test, we are interested in seeing what happens when the second peer signs on. So, right from the start of the test, I ask peer 2 to print their file info. From the terminal output, it can

be clearly seen that in a few seconds, peer 2 had no files. Then, just only after a few seconds, its file list indicates that it now contains the `readme.md` as intended. This means that the transfer and sync were successful.

```
[piola:src copy staplejw$ python filesynchronizer.py 12]
7.0.0.1 8004
--------------------
[]
--------------------
Waiting for connections on port 8002
connect to:127.0.0.1 8004
--------------------
[]
--------------------
Add your code in this directory
--------------------
[{'name': 'README.md', 'mtime': 1521640740}]
--------------------
connect to:127.0.0.1 8004
--------------------
[{'name': 'README.md', 'mtime': 1521640740}]
--------------------
```

Figure 5: The result of the test.