COE 4DN4 LAB 2 REPORT

A MUTI-THREADED FTP SHARING APPLICATION "MCNAPSTER"

Jingnan Chen, 1073196, Wei Zhang, 0951321

Introduction

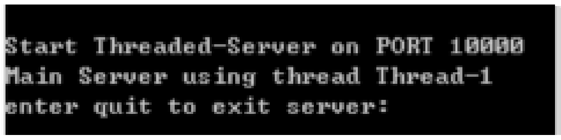
This lab is motivated by “Napster”, which is a popular file sharing system back in 1999. Our main objective for this lab is to create a server to client file sharing application alike Napster, where there is one central server and many clients. The server is required to create different threads for each connection that arrives, and is required to have a quit function to shut down the server for upgrades. This lab will give us an idea of relatively how early FTP server-client application for sharing files operated.

Experimental Results

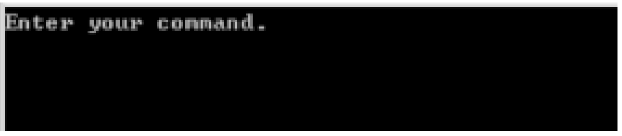
Server Side

Client Side

1. Start and sever and the client, the server is waiting for QUIT command and the client is waiting for any user input.

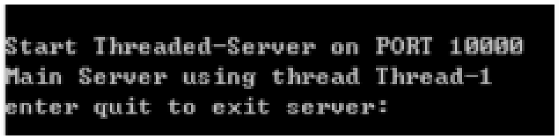


```
Start Threaded-Server on PORT 10000
Main Server using thread Thread-1
enter quit to exit server:
```

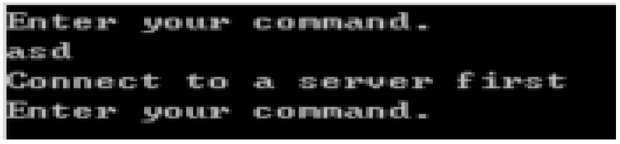


```
Enter your command.
```

2. Initially the client is not connected to the server. The client enters any input and the server will have no response.

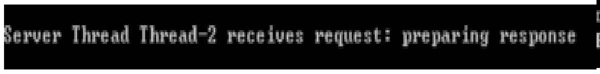


```
Start Threaded-Server on PORT 10000
Main Server using thread Thread-1
enter quit to exit server:
```

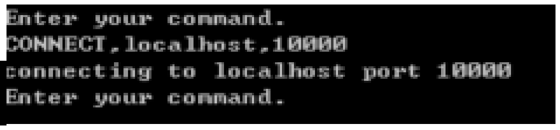


```
Enter your command.
asd
Connect to a server first
Enter your command.
```

3. Client connects to the server

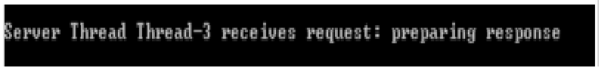


```
Server Thread Thread-2 receives request: preparing response
```

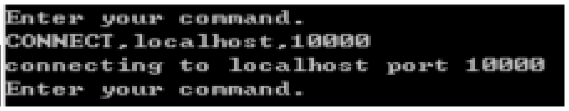


```
Enter your command.
CONNECT,localhost,10000
connecting to localhost port 10000
Enter your command.
```

4. Another Client connects to the server, server is indeed threaded.

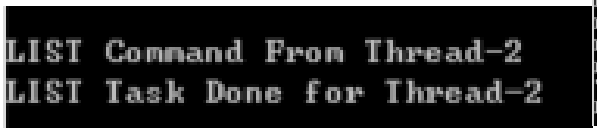


```
Server Thread Thread-3 receives request: preparing response
```

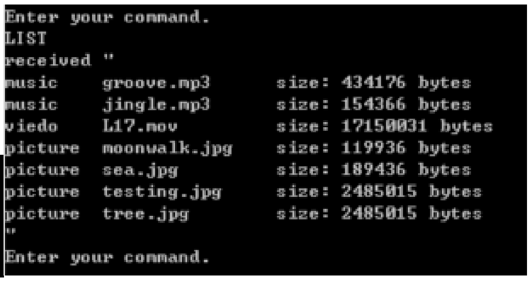


```
Enter your command.
CONNECT,localhost,10000
connecting to localhost port 10000
Enter your command.
```

5. Client requests “LIST”

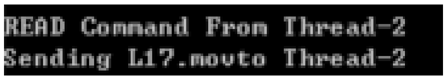


```
LIST Command From Thread-2
LIST Task Done for Thread-2
```

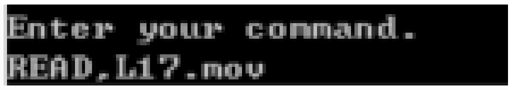


```
Enter your command.
LIST
received "
music    groove.mp3       size: 434176 bytes
music    jingle.mp3       size: 154366 bytes
viedo    L17.mov          size: 17150031 bytes
picture  moonwalk.jpg     size: 119936 bytes
picture  sea.jpg          size: 189436 bytes
picture  testing.jpg      size: 2485015 bytes
picture  tree.jpg         size: 2485015 bytes
"
Enter your command.
```

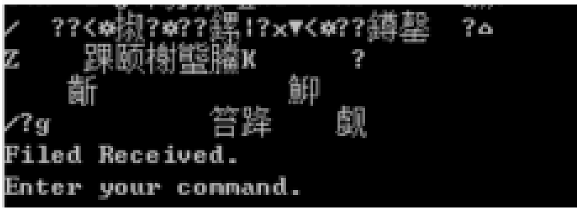
6. Client requests “Read,L17.mov”. The second picture for the CLIENT shows some of the data received in bytes and the user feedback of the file being successfully received.



```
READ Command From Thread-2
Sending L17.movto Thread-2
```

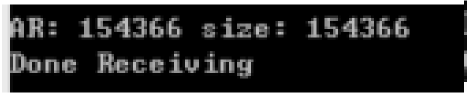


```
Enter your command.
READ,L17.mov
```

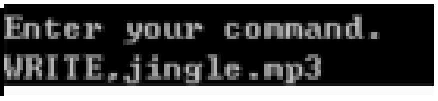


```
⁄  ??<◎楓?◎??鍱!?x▼<◎??鏄壂  ?⌂
Z    踝颐樹蹵臟K      ?
   斱          鉚
⁄?g          笞跭    觍
Filed Received.
Enter your command.
```

7. Client requests “WRITE,Jiggle.mp3”. The picture for the SERVER shows amount received “AR” and the expected file size. When they are equal, file transfer is finished and “Done Receiving is printed”.

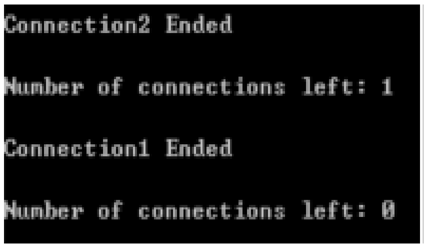


```
AR: 154366 size: 154366
Done Receiving
```

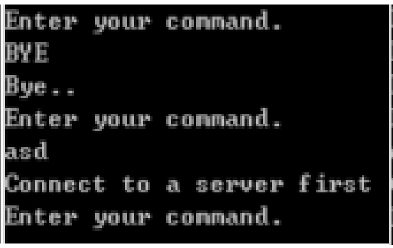


```
Enter your command.
WRITE,jingle.mp3
```

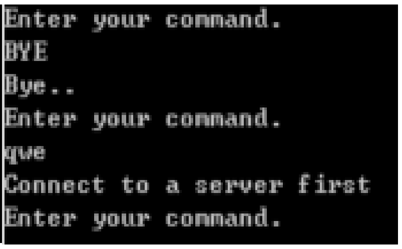
8. Client closes the connection with command “Bye”. The Server receives the “Bye” command and decreases counter for the number of connections.



```
Connection2 Ended

Number of connections left: 1

Connection1 Ended

Number of connections left: 0
```

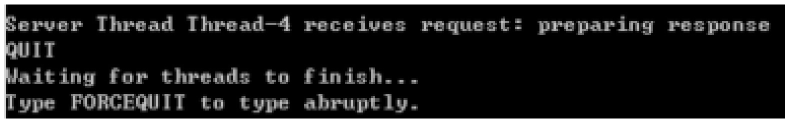


```
Enter your command.
BYE
Bye..
Enter your command.
asd
Connect to a server first
Enter your command.
```



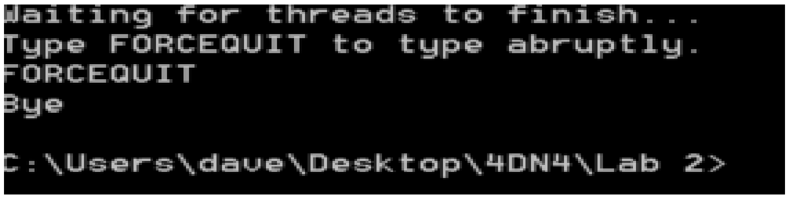
```
Enter your command.
BYE
Bye..
Enter your command.
qwe
Connect to a server first
Enter your command.
```

9. Server requests “Quit” to shut down the server with connections



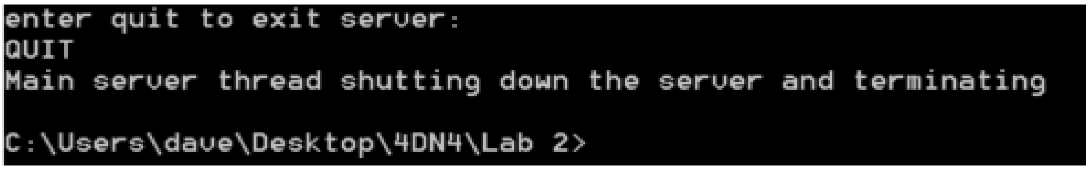
```
Server Thread Thread-4 receives request: preparing response
QUIT
Waiting for threads to finish...
Type FORCEQUIT to type abruptly.
```

At this point typing “FORCQUIT” will shut down the server at shown below.



```
Waiting for threads to finish...
Type FORCEQUIT to type abruptly.
FORCEQUIT
Bye

C:\Users\dave\Desktop\4DN4\Lab 2>
```

10. Server requests “Quit” without any connections.



```
enter quit to exit server:
QUIT
Main server thread shutting down the server and terminating

C:\Users\dave\Desktop\4DN4\Lab 2>
```

Issues and problems

We countered a few problems developing this application. First problem was the issue with files of the same name and type, the solution we use is to overwrite the old file.

Secondly, if the connection is suddenly terminated, the file becomes broken or corrupted, we implemented a “try and except block” to handle this issue. If the connection terminates before transfer finishes, the broken file is deleted.

Thirdly, we had to do some research about how to get the current path of the folder that the server and client is in. Using “OS” library, “os.getcwd” for current directory, and “os.listdir” for listing all the files, “os.path.getsize(path)” to get the size of the file, we are able to find all the necessary information we needed for the file transfer.

TA’s name for demonstration

We demonstrated to **MARYAM REZAEE** on **Thursday March 12, 2015**.

Conclusion

In Summary, This lab demonstrated the basic function of a FTP server to client application. We got a sense of the concept and methodologies used in a server program to handle file transfer requests and we learned about some of the problems that might occur during a file transfer and solutions to those problems. Additionally, we got to understand threaded and forked server to client connections, which provides a more reliable solutions, since it puts less stress on the “Main” thread or process of the server application. Furthermore, this lab gave us the idea of how the internet file transferring system worked during the early stages of booming in the internet industry.