PROJECT1 REPORT

Sohil L. Shrestha and Akash Lohani

1001556964, 1001661458

October 5, 2018

I have neither given or received unauthorized assistance on this work.
Signed: Sohil L. Shrestha, Akash Lohani   Date: October 5, 2018

1 Abstract

In this programming assignment, we implemented a client server environment based on message oriented protocol where the communication is connection oriented meaning client and server have to agree and acknowledge before the actual task occurs.
Based on above mentioned protocols, we implemented single and multi-threaded environment which can deal with file processing between two nodes, client and server. In addition to this, we also implemented a computation server which can proceed 4 remote procedure call (RPC) where client can use those function by just passing the RPC name and the parameters.
The remainder of this report is structured as follows. We discuss related Understandings of theory in Section 2, followed by an explanation of our implementation in Section 3. We evaluate our programming by elaborating the encountered issue in Section 4. Finally, we will conclude our assignment work in Section 5.

2 What We have learnt

Socket programming
Socket is an end point of devices which can be routed using IPAddress and Port number. In Socket connection, We try to establish two way socket connection between Client and Server using this shared information. WWW, FTP, P2P are good example based upon Socket Connection.

Transmission Control Protocol (TCP)
TCP is connection oriented protocol where Server and Client has to make connection before sending and receiving the data. Connection establish by three acknowledgement among nodes so it is known as three way handshaking.

Message oriented protocol
Message oriented protocol is a protocol that communicates by putting the message in the queue. Therefore, the order of the message will be guaranteed but not the timing of the message. Message oriented protocol is typically compared with Stream oriented protocol which can deal with video and audio streaming.

Multi-thread Programming
For the purpose of increasing processing efficiency, multi-thread programming is used to perform multiple processes in parallel which means multiple threads will be executed simultaneously in one program(or process). In client server architecture for file transfer , it is important to synchronize multiple clients such that dependency such as read after write (RAW), write after write(WAW) and write after read (WAR) is maintained consistently.
To maintain such dependency avoiding any corruption of file, we have to make sure that the critical section(part of program that access shared resource) is accessed by only one thread at a time. To maintain the accessibility, distributed locks are used such that each thread(client) have to first acquire a lock before accessing the resource. In our case, client have to first acquire a lock tied to a file name before trying to upload a file. No two files can upload the same file or rename the same file at a time. When using locks, it is very important to avoid race condition.

Remote Procedure Call
Remote procedure call is a method for calling and executing a program on another computer connected via network. In the remote procedure call, a remote machine call procedures to execute a program on a machine through the network, so that the remote machine can process the subroutine. By using the remote procedure call, it is possible for other machines to process their desired calculation through the network and receive only the processing result.

3 Implementation

Based on the theory we talked about in Section[2], we'll discuss about our implementation we have done using java language. We established TCP connection and built socket connection between Client and Server and let them communicate based on message oriented method.

Basic environment
Establishing TCP connection
We established the socket connection by using *Socket* class which uses IP Address and Port number as a parameter. IP Address is set to 'localhost'(or 127.0.0.1) and Port number is set as 6666 respectively. TCP connection will be established with following step.

1. Server creates the socket and binds it with Port number.

```
Serversocket server = new ServerSocket(port);
```

2. Server will be in passive mode and listen to port for client to connect.

```
Server clientsocket = server.accept();
```

3. Client sets its own socket in port 6666 and routes it to Server socket.

```
Socket clientSocket = new Socket(IPAddress, PORT number);
```

4. Socket connection established.

Message oriented communication
The communication between Client and Server is proceeded with *java.io.** library which contains system input and output classes for data stream and file system. We used *DataInputStream* and *DataOutputStream* class to catch and send the serialized data. By defining,

```
//Client Side
c_output = new DataOutputStream(clientSocket.getOutputStream());

//Server Side
s_input = new DataOutputStream(clientSocket.getInputStream());
```

We can send and receive the data as follows.

```
//Client Side
c_output.writeUTF("HELLO");

//Server Side
s_input.readUTF();
```

Note that *DataInputStream* catches the data from queue according to the order it was sent.

Assignment1
We created a single-threaded file server which supports 4 file operations(UPLOAD, DOWNLOAD, DELETE, RENAME) in response to the clients request. The detail for this each function will be like below.
1. UPLOAD : The file in the client's directory will be sent to the server's directory.
2. DOWNLOAD : The file in the server's directory will be sent to the client's directory.
3. DELETE : The file in the server's directory will be deleted.
4. RENAME : The file in the server's directory will be renamed.

The program run in the following way:

- The client and server will create its own folder when program runs for the first time:
  Client Folder Name : ClientFileStorage
  Server Folder Name : ServerFileStorage
  NOTE: The program is tested on MacOSX . Due to OS dependent file system, it may not work well with windows.
- The program will then ask Client input . Each input for each operation should be option choice seperated by filename/s. Examples are as follows :
  1 UPLOAD: 1 LargeClient.txt
  2 DOWNLOAD: 2 LargeClient.txt
  3 DELETE: 3 LargeClient.txt
  4 RENAME: 4 LargeClient.txt newLargeClientName.txt
- Connection is made to the server by the client as discussed earlier. Client then sends command to the server.
- Server sends the acknowledgement.
- According to the command, client read/write to the stream and server read/write from the stream or process the command(rename and delete)
- Finally server sends the completion message and client displays it on the console.

Assignment2
We augmented the operation in Assignment1 by creating multi-threaded file server. In this part, the server can handle multiple clients request at once using threads.

To maintain consistency, we can implemented our own read write lock in the server such as file which is being uploaded or renamed cant be downloaded or deleted at the same time. Each client thread have to acquire a lock before attempting to upload or delete a file. The lock prohibits any other client to upload , delete, rename or download the same file at the same time. Contrary to that, multiple file can download the file at the same time.

To test our program, we have created multiple client threads in a loop and asked it to randomly issue command to upload,download, delete and rename large text files(52 MB).

Assignment3
We created the Remote Procedure Call which enables client to calculate 4 procedure in the remote server.

1. CALCULATE PI :
Client calls the procedure by sending the function name "calculate_pi" to the Server. Server will calculate the value of π by using Archimedes method and return the result back to Client. The number of iteration which affects the granularity of the value π is set as 15.

2. ADD :
Client calls the procedure by sending the function name "add" and sends two numbers to the Server. Server will calculate the sum of the value and return the result back to Client.

3. SORT ARRAY :
Client calls the procedure by sending the function name "sort" and sends the Array to the Server. In this case we used ObjectInputStream / ObjectOutputStream to send and receive the chunk of data at once. Server will sort this Array using Insertion sort and return the result back to Client.

4. MULTIPLY MATRIX :
Client calls the procedure by sending the string "matrix multiply" and sends the Matrix to the Server. Server proceeds the matrix multiplication and return the result back to Client.

Bonus Question
We have implemented a helper thread which checks in client folder every 10 seconds for change. For this question, following folder is used:
Client Folder Name : ClientFileStorage_Test
Server Folder Name : ServerFileStorage_Test

Initially, helper thread will try to sync both the client and server folder by uploading from client to server and downloading from server to client. It maintains a hashmap of file name and last modified date of the files.
Whenever client makes any changes to the files in its folder, helper thread will check the list of files in the hashmap as well as compare the last modified date of the files. Helper thread will either upload the file on server if new file is created on the client side or current file as been modified. OR it will delete the file on server if the file on client has been deleted.

NOTE : The program has been tested in the following environment:

Device: 1

- macOS high Sierra
- JAVA version 1.8.0_144
- Files are tested with txt file only. Sample files are in the project itself

Device: 2

- Windows 10 Home
- JAVA version 1.8.0_181
- Files are tested with txt file only. Sample files are in the project itself

It might not run on different operating system because the file path structure is different.

4 Issues encountered

Converting the single threaded file transfer client server program to multi-threaded server was a challenge. We first make use of synchronized keyword from JAVA to maintain consistency. However synchronized block can only be accessed by one thread at a time. Thus it did not meet our propose of downloading same file by multiple client at a time. Distributed locking is indeed necessary in multi-threaded environment because any data that is shared between threads or processes should be locked before altering or reading. If multiple client thread changed the same files, file could get corrupt resulting loss of information. Hence we implemented our own read write lock such that the requirement specified by the question is met. The details are discussed on the Implementation section.

Issue was also encountered when trying to send a array or matrix object to the server via socket. Package the function name and its respective object was tackled using Java objectstreams.

5 Conclusion

Implementing the programming assignment, we learnt a lot regarding the server client architecture and the importance of locking in multithreaded environment. We were able to implement our own dropbox like file synchronizaiton system . The project helped us explore the language features of JAVA and enable to implement those features on our own using JAVA library.

6 References

[1] https://stackoverflow.com/questions/12715321/java-networking-explain-inputstream-and-outputstream-in-socket
[2]https://www.geeksforgeeks.org/socket-programming-in-java/