Distributed Systems, Project1

CSE 5306, Fall 2021

Contributors:

PAVAN KUMAR CHAPARLA – 1001865675, pxc5675@mavs.uta.edu

POLA MANOHAR – 1001980283, mxp0284@mavs.uta.edu

07th October, 2021.

REQUIREMENTS:

- Assignment-0 (0pts)Create a virtual machine on your laptop or desktop and install a Linux operating system in the VM. You can use any Linux distributions. Ubuntu is recommended. Use the Linux environment to test your programs.

- Assignment-1(20pts)Implement a basic single-threaded file server that supports the four operations listed above. You willuse the message-oriented communicationprotocol. You can assume that the client and the server reside on the same machinebut communicate with each other using different ports. Use different folders to hold files downloaded to the client or uploaded to the server.

- Assignment-2(20pts)Based on the single-threaded server, implement a multi-threaded file server. The client-sidesoftware does not need any changes but the server should be able to support multiple concurrent operations. Note that you do not need to consider locking in your multi-threaded server at this stage. Use multiple clients to test your server.Remote procedure call (RPC) based communication

- Assignment-3 (35pts)Use the following design of the synchronous RPCs to implement a computation server. The server supports four RPCs: calculate_pi(), add(i, j), sort(arrayA), matrix_multiply(matrixA, matrixB, matrixC). The RPCs represent different ways to pass the parameters to the server. You need to implement a client stub to pack parameters into a message sent to the server and a server stub to unpack the parameters. You are NOTallowed to useJava remote method invocation (RMI) or RPC in other programming languages to implement the RPC-like communication.

- Assignment-4(25pts)Re-implement the computation server using asynchronous and deferred synchronous RPCs. For asynchronous RPC, the server immediately acknowledges anRPC call before it actually performs a computation. The result of the computation is saved in a table on the server, which can be looked up by the client for the RPC result. The design of the client will be slightly different from that in the synchronous RPC. Instead of waiting for a synchronous RPC to return, the client using an asynchronous RPC switches to other computations and queries the server for the RPC result at a later time. For deferred synchronous RPC, you need to devise a mechanism to interrupt the client to return the RPC result to the client when the server has completed its local computation.

CONCEPTS COVERED IN ASSIGNMENT ‘0’ and ‘1’:

- We have analysed the project and worked together to start with the file transfer mechanisms between the client and server using Socket Programming.

-We went through certain course materials to get the in-depth knowledge of the socket-programming before starting with the implementation and shared the tasks among us.

-Have downloaded the Oracle VM VIRTUAL BOX from ‘https://www.virtualbox.org/wiki/Downloads’ and also the ubuntu iso to start the virtualBox.

-Have implemented the socket programming to define client and server and also to perform file transfer operations between them.

-In Assignment 1 we have started a client machine and specified it with a port number of the server machine to connect to the server.

```
Using socket = new Socket(addr,port);
```

-Once our client socket is connected to the server socket the server receives the acknowledgment about the connection.

-Using the DataInputStream and the DataOutputStream, we do the communication between the client and server using methods of these objects like **readUTF**(), **writeUTF**().

-Then we take the first command from the terminal about the operation that we would like to perform i.e 1.)UPLOAD, 2.) DOWNLOAD 3.) RENAME 4.) DELETE.

-This command is then directed to the server from the client using writeUTF() of the DataOutputStream obj and thus the server performs the operation based on the command.

-For the files on which we perform operations we have created two folders called clientFiles and serverFiles and also stored client.txt and server.txt in both of the files respectively.

-For UPLOAD process The FileInputStream reads the file that need to be sent to the client and the stream is written in to the file of the server directory after transfer using OutputStream.

-Similarly we have implemented the download functionality with the InputStream reading the incoming stream of the file from the server and then writing it to the client's file using FileOutputStream. It will create a file if there is no such file in the directory.

-The Rename and Delete functionalities are implemented for the files on the client machine with the server performing the action.

CONCEPTS COVERED IN ASSIGNMENT '2':

-In this part we have implemented the client server communication using multi-threading concept.

-We build the architecture that supports multiple clients with the server.

-We have added a class called MultiClientServer that creates a thread to the server functionalitites written in MultiCLientHandler. Each client when asks for a connection to the server will be connected to a thread which runs and performs the actions of the server.

-We have done the same operations i.e. 1.) UPLOAD 2.) DOWNLOAD 3.) RENAME 4.) DELETE.

```
- MultiClientHandler st = new MultiClientHandler(s,clientNo);

              st.start();
```

-The above will start the thread instead of the single server itself from the MultiClientServer.java.

-All the functionalities implemented in the assignment 1 have been implemented in the same way in assignment 2 along with the multi-threading concepts.

CHALLENGES DISCUSSED WHILE DOING ASSIGNMENT 1 AND 2:

-Converting from single threaded to multi-threaded can create problems like giving write access to the outputStreams of multiple clients at a time on a single file.

-Here the distributed lock comes in to picture by which we give the access to the client on a file by locking it and holding other clients so that they donot perform any unwanted operations on the file at a given point of time while the other client is performing.

-Consider the above condition where we perform certain operation like rename or delete. These operations require distributed lock and the clients have to coordinate with each other and work accordingly.

CONCEPTS COVERED IN ASSIGNMENT ‘3’ and ‘4’:

-We have implemented the RPC with client and server stubs individually defined for each operation.

-The client requests for a certain operation from the terminal and the inputs are gathered from the client and sent to the client stub.

-These inputs are packed together by a client stub i.e. for every operation and sent to the serverstub i.e where the operations are performed (calculate_pi, Add, Matrix_multiply, Sort_array).

-The serverstub unpacks the parameters from the clientstub, performs the operations and returns the results to the client from the server.

-Every communication or data passage we send it along with the DataOutputStream or DataInputStream i.e(packed and unpacked.)

From Client to ThreadHandler.java

```
- new CalculatePi(dataOutput); in threadHandler.java to calculatePi.java

.

.

.

      -dataOutput.writeDouble(returnValue); in calculatePi.java
```

-In the above format we implement clientSTub and ServerStub sequentially.

ISSUES ENCOUNTERED WHILE IMPLEMENTING:

- The failure of the VirtualBox at times due to compatibility issues.
- The Remote procedure calls as everytime needs to be referenced with respect to objects we have faced issues in resolving the exceptions and the edge cases everytime.
- The client and server stub development consumed most of the time as we had to make sure that every functionality of the operation specified from the client satisfies the definition of RPC.