CSE5306
Distributed Systems
Fall 2020

Project 1

Vishnu Gopal Rajan

1001755911

Chinmaya Basavanahally Venkatesh

1001778014

I have neither given nor received unauthorized assistance on this work

Signed: Date: 10/05/2020

Vishnu Gopal Rajan

Chinmaya Basavanahally Venkatesh

Project 1: Single-Server, Single-Client.

We have implemented this project to build a simple single client server using python on Linux system. This single server can communicate with a single client at a time. This file server can perform operations like Uploading, Downloading, Renaming, and Deleting files. This is implemented using simple connection orientated file transfer using socket programming. The file is being transferred from server to client or vice versa on the same connection. Since this server can handle only one client at a time and will not function with multiple clients requesting service from one server. It will wait for one client to finish and then communicate with the other client after.

We also implemented the acknowledgement system where the server sends acknowledgement to the client on the status of the connection. Once the client sends the request, we use conditional cases to check the request at the server side.

Client-Side: Client connects to the host and port using socket which we initialize at the beginning of the program.

Data Structures:

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) // initialization of socket sock.connect((HOST, PORT))// connecting to host and port

Functions:

upload(file, link): This function handles the process of uploading a file from client to server by specifying the name of the file we want to upload. The server first checks if the file exists in the client directory and then proceeds with the upload.

download(file, link): This function handles the process of downloading the specified file from server to client.

rename(old_name , new_name , link): This function renames the file from its old name to new name. Both these parameters are sent to the server.

delete(file, link): This function is used to deleted the specified file. If the file exists, it is deleted.

Server-side:

Host IP is set to default 127.0.0.1

Port set to 8080

Functions:

upload(filename , link , addrs): On server side this is to check If the process selected is upload and if it is, it proceeds with verifying the file if it exists and then starts the uploading process.

download(filename , link, addrs): This sends the requested file from server to client. The client downloads this file.

rename(old_name , new_name, link, addrs): This renames the existing file using its oldname and new name.

delete(filename, link, addrs): This function is used to deleted the file if it exists, by taking the filename from client.

Project 2: Multi Client – Single Server

To solve the problem of the server being able to communicate with multiple clients at once, we have made use of threaded server which can handle multiple clients at once. For this project we have made use of threads. The server has multiple thread creation in order to handle multiple clients simultaneously. When new client is requesting connection, the server creates a thread for it and starts communicating with it immediately without waiting for the other clients to finish. The only drawback on this project is if more than one client is working on the same file simultaneously. For this we made use of multiple clients requesting server time simultaneously.

Thread creation in server: start_new_thread(thread, (link, addrs))

The data structures and functions of the server and client remain the same. Only thread creation is added for server.

Project 3: RPC Client and RCP Server

This project consists of these operations: calculate_pi(), add(i, j), sort(arrayA), matrix_multiply(matrixA, matrixB, matrixC). This process involves the client sending the parameters it needs to send to the server by packing it as a client message. The server receives this, unpacks the message to get the parameters.

Calculate_pi(): Using the Math function of Python to find the value of pi. The Client sends a request to the server to calculate the value of pi and the server computes the value and sends it to the client.

add(i,j): Implementation of add Remote Procedure call is used to get the sum of two numbers. The Client sends the server two numbers to be added and the server returns the sum of the two numbers it receives by the client.

Sort(array) : Our first approach was to send each element of the array individually through the message but faced problems as it is an array. We serialized the array using pickle.dumps() before sending it to the server. We use pickle.loads() to get data from Server to the Client. Array is sorted using the sort function at the server.

Matrix_multiply(matrixA,matrix,matrixC): We first check if the matrix multiplication conditions are satisfied. We pass the two dimensional array or matrix to the Server the same way as in

Sorting the array. We use numpy.dot function to calculate matrix multiplication. The result is sent to the client by serializing using pickle.loads() is used to get the result.

Project 4: Asynchronous and Deferred RPCs

For Asynchronous, we do not immediately send the computed result as in the synchronous part. It saves the result in a dictionary based key-value pair.

Eg: dict = {'add':0,'pi':0,'sort':[],'mat':[]}

Server computes the result and saves them in the key-value dictionary. When the Client requests the result of the computing task based on the request parameter.

For Deferred Synchronous we need 2 threads. One is needed for client to send the request and data, and another one is needed for receiving data. One thread is needed for server to compare the output and other for sending output. In deferred connection if connection is same the server will refuse one thread.

Result

Getting a hands-on experience in implementing server-client model based on different communication modes. We have learnt how to build a computational server and how to handle arrays, matrices as input and output data. We have learnt the importance of using locks in multithreaded servers to reduce data corruption and race around condition. After implementing project, we have a notion of how messaging apps work.