**DISTRIBUTED COMPUTING PROJECT (PART I)**

Team members:

* Leslie Jasmine Nyabeyeu Bamou
* WASIB
* SELINA

Document version: 1.0

Date: 10/13/2014

**Supervisor: JUAN GUZMAN**

INTRODUCTION:

A distributed system is a collection of independent computers which are interconnected via a network and they are also capable of collaborating on tasks. Computers are considered independent when they don’t share any resources, but distributed computers are the opposite of that. The type of computing these distributed computers perform is called distributed computing. Our distributed system is designed in such a way that the client can send request to the router, then the router passes the request to the server, then the server sends the response back through the router, and the client receives the response.

SYSTEM ARCHITECTURE AND DESIGN:

The programming language, Java, was used to design this distributed system. Netbeans was the Integrated Development Environment which was used to code the system. The overall network consists of 3 computers; 1 client, 1 router, and 1 server. The connected system was made possible by the usage of Sockets. The program was tested in our CS lab multiple times. We have also tested it on virtual machine.

CODE AND COMPILATION:

TCP protocol was used to make the communication between the server, client, and the router. Different aspects of the code ran on different machines. On the Client side, it is “ClientGui,” on the server side, it is “TCPServer,” and on the router side, it is “TCPRouter.” The Client Side is consisted of 3 different classes, and they are, ClientGui.java, TCPClient.java, and Time.java. The Router side has 1 class, and it is TCPServerRouter.java. The Server side has 3 classes, and they are, FileManager.java, TCPServer.java, and UpdateFiles.java.

**UML Diagram:**

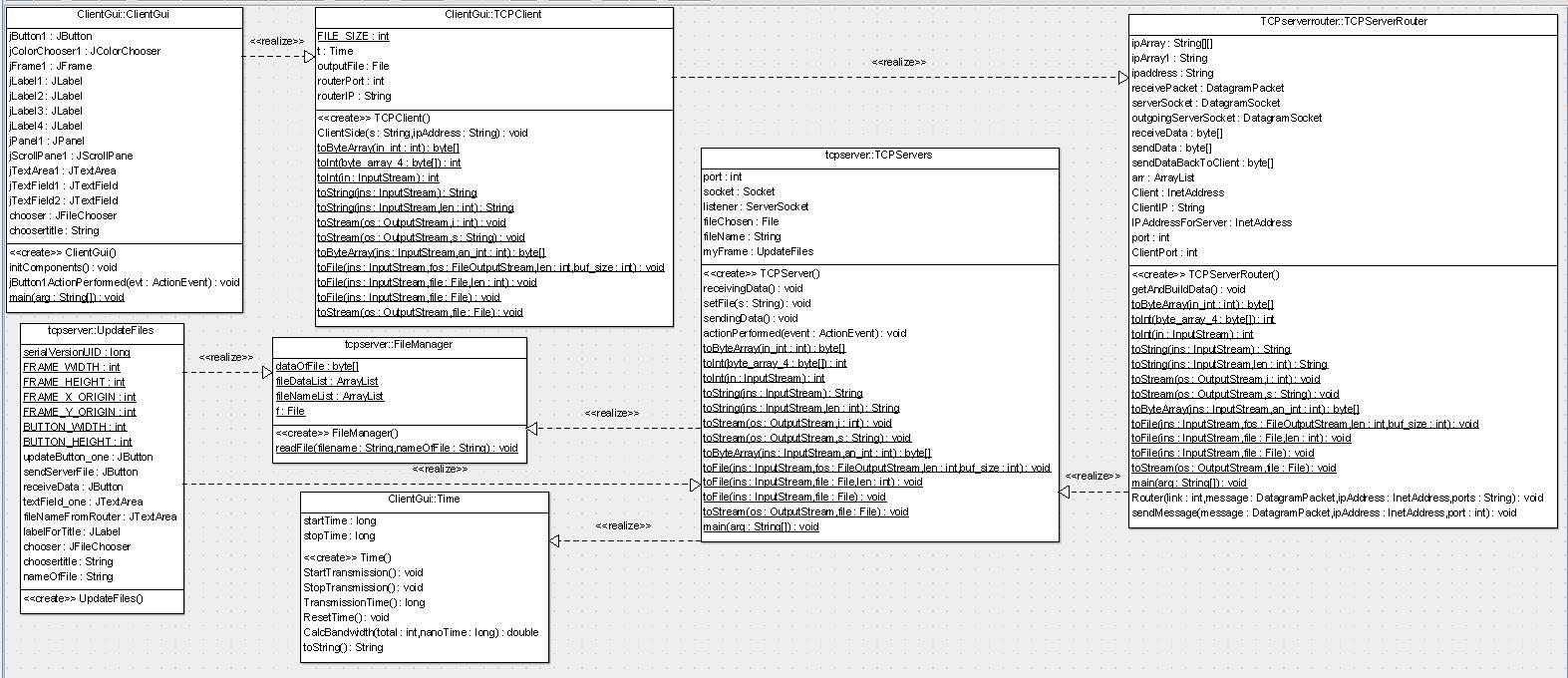


Figure 1: UML diagram

EXECUTION:

The client side and the server side programs have GUI interface for being visually pleasant and user friendly to the users. On the other hand, the router doesn’t have any graphical interface. Our system is designed in such a way that the server needs to be up and running first. The next step is to run the router. Both the router and the server is asynchronous, so they need to run before the client. The final step is to run the client. In the welcome window of the client program, the user is allowed to enter the IP address of the router, and the name of the file which he/she wants. The request gets passed to the router whose IP address was entered during the request. The router is known of the IP address of the Server, so the router transfers the request to the server. The server takes the request and sends the response to the router, and the router forwards the response back to the client. The desired file gets saved on the specified location.

DATA COLLECTION AND ANALYSIS:

A few different data structures were used to make our distributed system work as expected. We used an array to store the information and the size of the files. An arraylist was used to differentiate between the file types. While testing, we had issues with different port numbers, and we are not sure about that issue. When byte is used, we figured out that an array can hold only up to 50,000 bytes.

|  |  |  |
| --- | --- | --- |
| **Transmission Time** | **Data size** | **Bandwidth** |
| 2.0917E+10 | 509 | 24.334617 |
| 3.0675E+10 | 9934 | 323.849152 |
| 2.2695E+10 | 137864 | 6074.60265 |
| 1.7755E+10 | 467677 | 26341.1171 |
| 3.0367E+10 | 17933643 | 590563.031 |
| 1.6755E+10 | 111204982 | 6637154.24 |
| 1.2228E+10 | 113571065 | 9287792.18 |
| **AVERAGES: 34760810.57** | **2364039.05** | **21627314893** |

Figure 2: Table

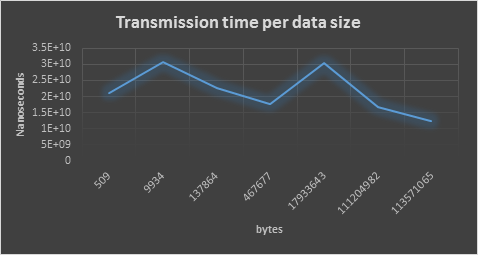


Figure 3: transmission time per data size

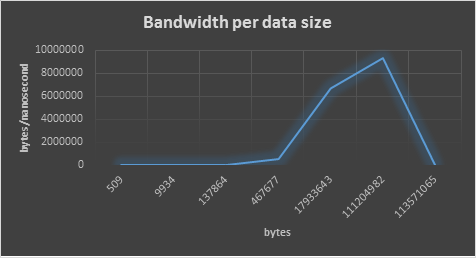
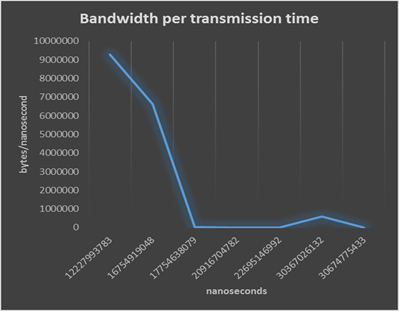
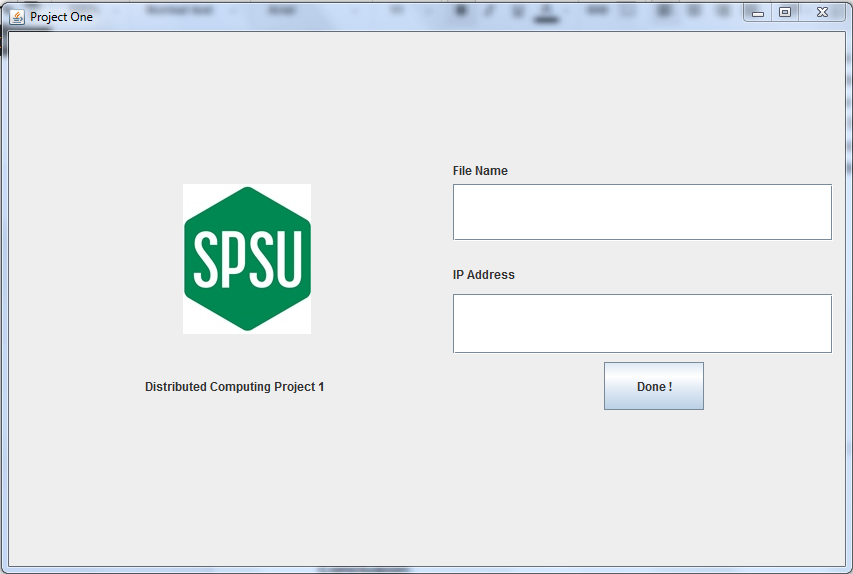


Figure 3: Bandwidth per data size

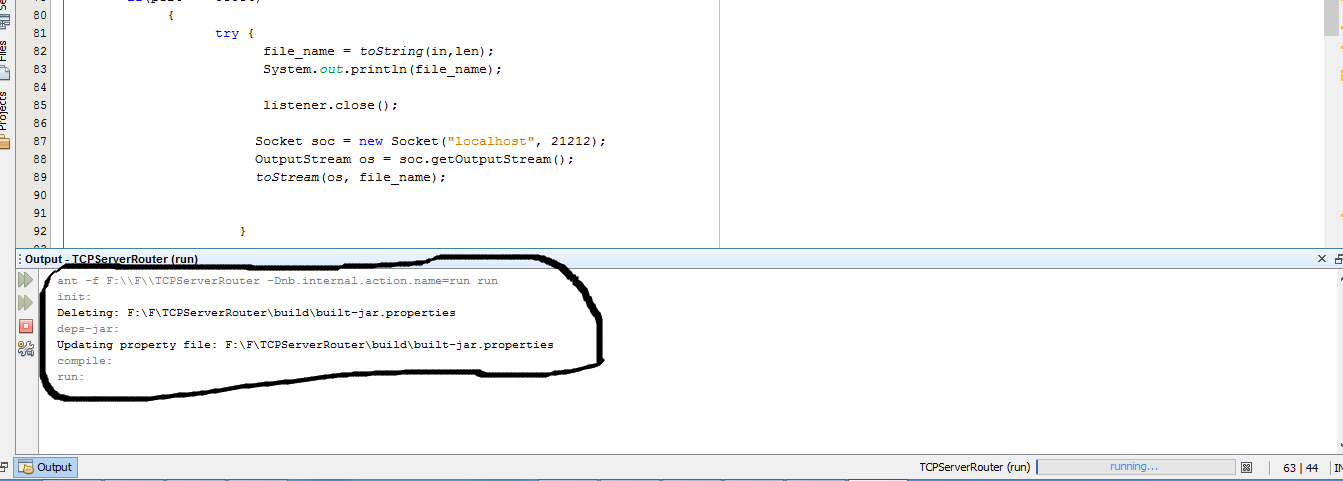


**Figure 4: Bandwidth per transmission time**

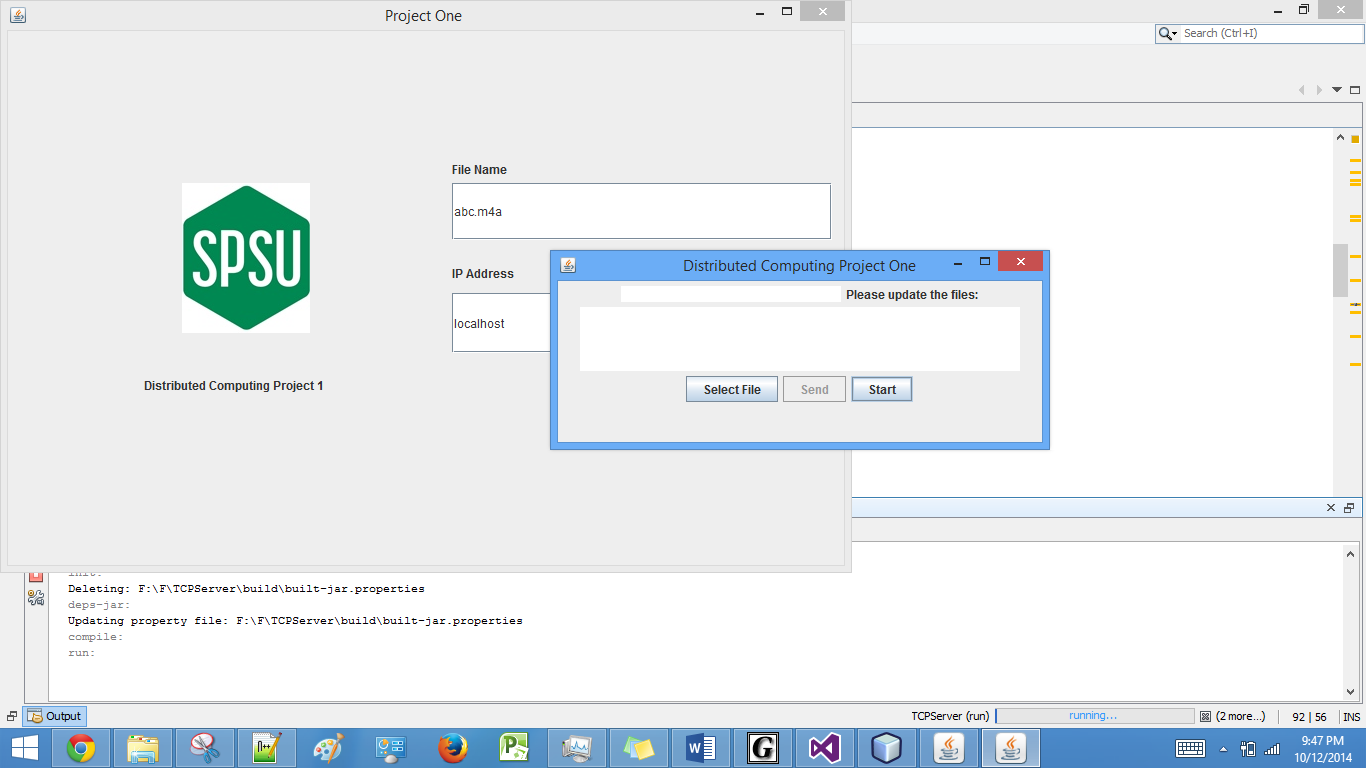
SCREENSHOTS:



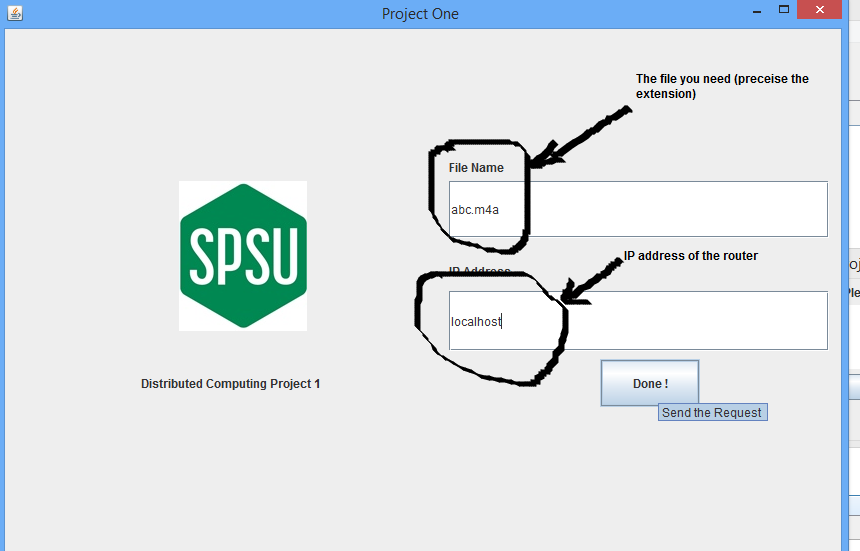
**Figure 5: Welcome Screen (Client Side)**

****

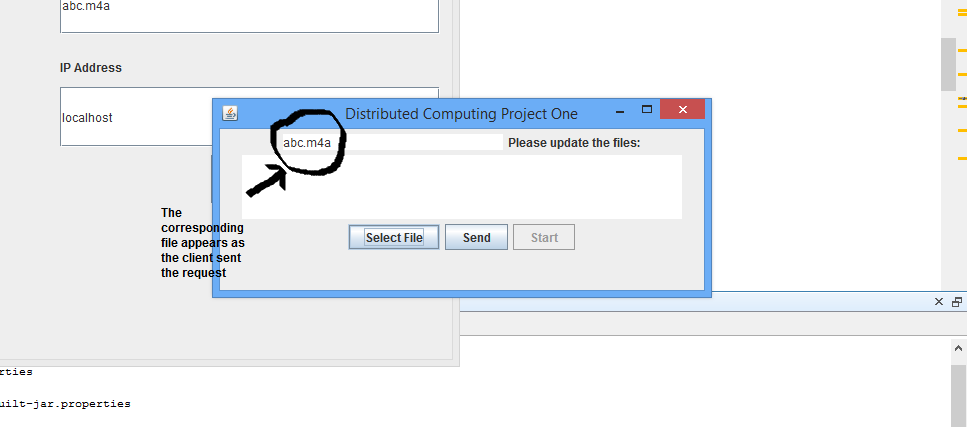
**Figure 6: Launch the router**

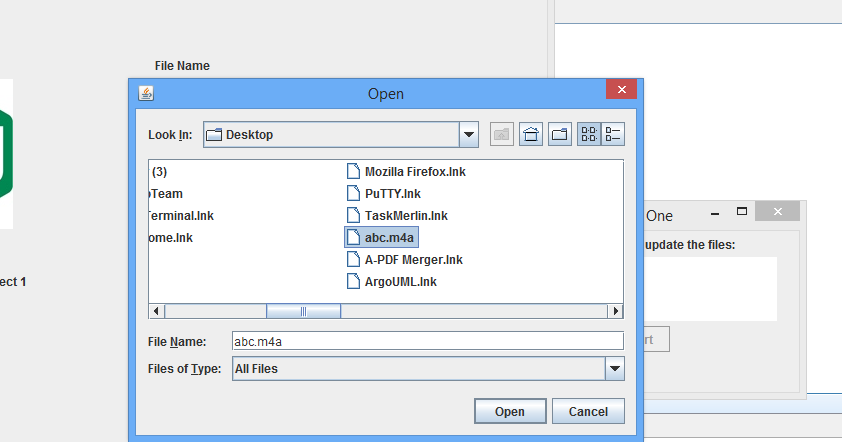


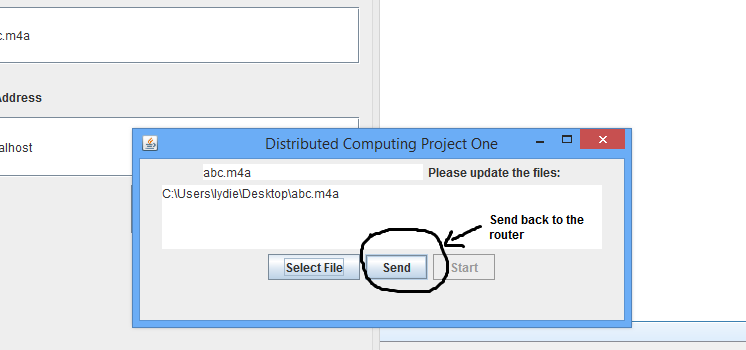
**Figure 7: launch the server and start his socket listener**



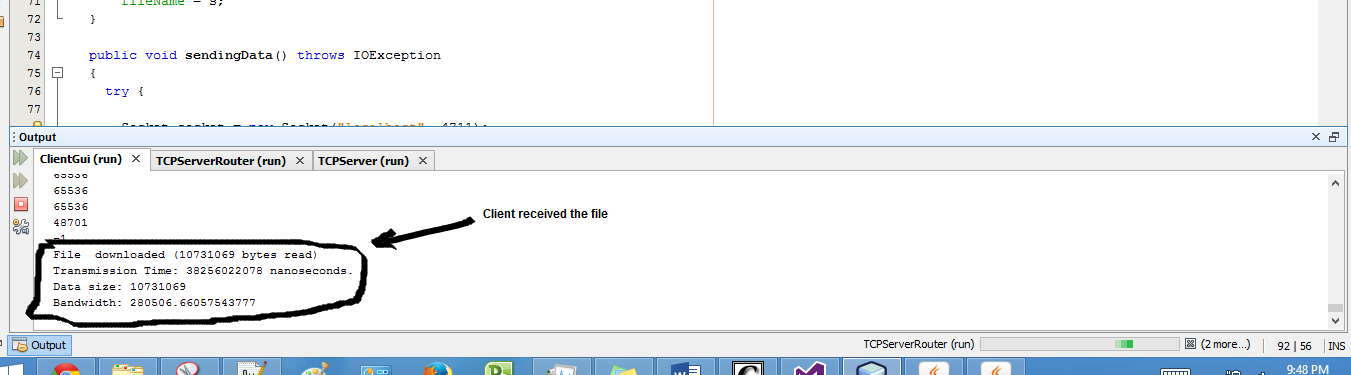
**Figure 8: Request the file from the server**





****

**Figure 9: (server side) Choose the corresponding file and send back**



**Figure 10: The client receives from the router**

CONCLUSION:

The project was a success since it was able to do what it was supposed to do. The system had three connected computers which were used to represent the client, the server, and the router. The request from the client goes through the router and goes to the server. The server processes the request and sends a response through the router to the client. The server can see the file name and knows which file to upload. After the server sends the file, the file appears on the location which was specified by the client. The code was tested in the CSE lab with the wired computers and also on our machines, and it worked as expected.