Part 2 – User Guide

Introduction

The second part of this project is meant to implement a scalable peer-to-peer file transfer system that utilizes a pair of routers and two node groups to transfer files from one Node to another. For the sake of this user guide, we will utilize a standard setup that consists of one Router1 instance, one Router2 instance, one N1 client node, and one M1 client node.

In this configuration, Router2 is connected to Router1 as a Router Pair to facilitate communication between two node groups. Each node group is in direct communication with a dedicated Router. The N node group will communicate with Router1 and, in this case, will only consist of one N1 Node. The M node group will communicate with Router 2 and, in this case, will only consist of one M1 Node.

File Transfer is initiated through an in-console menu on the client node. Once the sending Node is connected to the receiving Node through IP-based addressing and the intended file to send is defined in-console, we initiate the file transfer. Following a successful transfer, a statistical summary, including sent file size and sending time, is generated and stored in N1's executing directory as a text file. Both Routers also store statistical summaries of the time required to connect two clients through routing as text files in Router1's and Router2's executing directories, respectively.

Configuration

In order to correctly run the simulation, Router1, Router2, N1, and M1 must be run on separate machines with unique IP addresses. Router1 requires the RouterOne.Java file, the "routers" folder, and the "exceptions" Folder. Router2 requires the RouterTwo.Java file, the "routers" folder, and the "exceptions" Folder. N1 clients require the N1.java file, the "nodes" folder, and the "exceptions" folder. M1 clients require the M1.java file, the "nodes" folder, and the "exceptions" folder. The required folders and Java files for the type of Router or client intending to be executed must all be in the same directory on the host machine selected for that chosen process. N1 and M1 clients cannot be run on the same machine as the Router instances they are attempting to connect to due to the client and Router utilizing the same IP address.

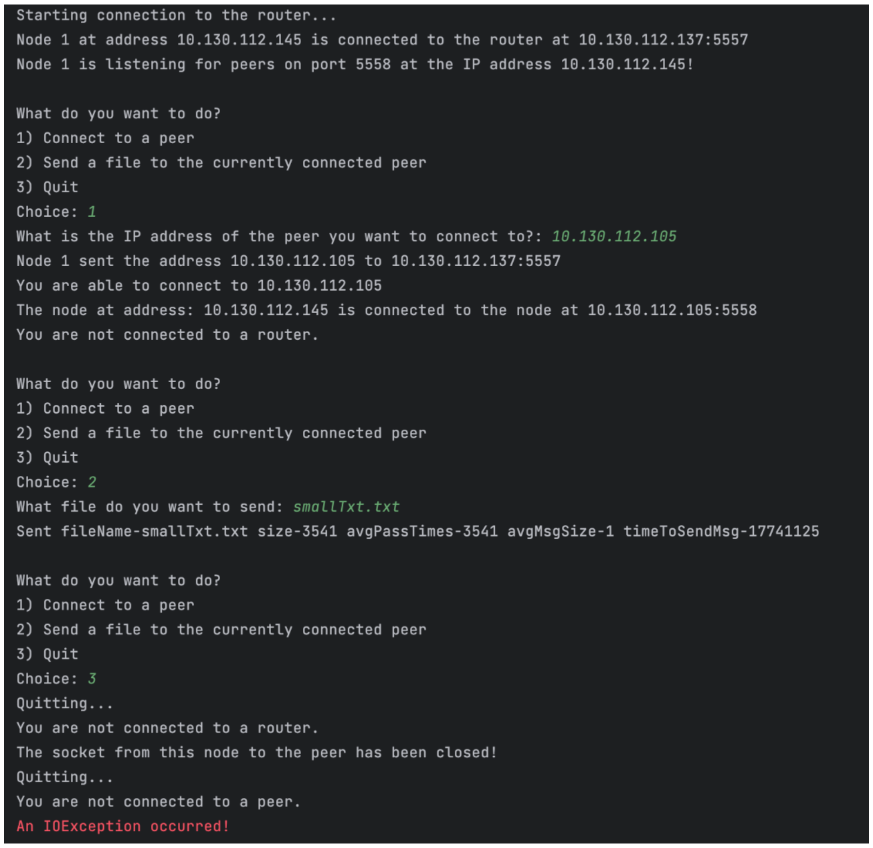
For a Router2 instance to connect to a Router1 instance, line 18 in RouterTwo.java needs to be modified such that the string argument in Router.connectToRouter() contains the IP address of the Router1 host machine to which it will connect. N1 and M1 clients also need to be modified in order for them to be directed to the correct routers. N1 clients need to modify line 16 in N1.java so that the new Nodes router address and port reflect the IP address and port used in Router1. For example, if Router1 is being run on a machine with the IP address 10.130.112.137 and the 5557 port is reserved for communication, line 18 should show: Node = new Node("10.130.112.137", 5557);. A similar process needs to be completed for M1 clients such that line 14 in M1.java reflects the IP and port number for Router2. If you are unsure of the IP address being used on Router1, Router2, N1, and M1, you can use the "ipconfig" command in the command prompt on Windows and the "ifconfig" on MacOS and Linux machines.

Compilation

All files need to be successfully compiled into JVM code before execution. If each client and Router is executed on a separate computer, each system's process execution will be handled independently as the communication configuration requires.

Execution

Router and client services must be executed in exact order for the network to operate correctly. Router1, through RouterOne.java needs to be executed first, followed closely by executing Router2 through RouterTwo.java. Once the two routers are connected, we can add clients to the network. For N1 to M1 communication, we must first execute the N1 client code through N1.java. Once we have confirmed that this client successfully connected to Router1, we can connect M1 to Router2 through M1.java. To initiate a file transfer, we must select option one from the menu presented in N1's console output. Then we insert M1's IP address into the console when prompted. Once the connection is confirmed and the menu reappears, we select option 2 to transfer a file. From here, we type the name of the file we intend to transfer, including its filename extension. The file transfer will begin if the file exists in the same directory as N1.java. Once the transfer is complete, we can view the file in the same directory as M1.java on the M1 client machine. An example of a successful connection and file transmission from the perspective of N1's command prompt execution output can be viewed below for reference:



```
Starting connection to the router...
Node 1 at address 10.130.112.145 is connected to the router at 10.130.112.137:5557
Node 1 is listening for peers on port 5558 at the IP address 10.130.112.145!

What do you want to do?
1) Connect to a peer
2) Send a file to the currently connected peer
3) Quit
Choice: 1
What is the IP address of the peer you want to connect to?: 10.130.112.105
Node 1 sent the address 10.130.112.105 to 10.130.112.137:5557
You are able to connect to 10.130.112.105
The node at address: 10.130.112.145 is connected to the node at 10.130.112.105:5558
You are not connected to a router.

What do you want to do?
1) Connect to a peer
2) Send a file to the currently connected peer
3) Quit
Choice: 2
What file do you want to send: smallTxt.txt
Sent fileName-smallTxt.txt size-3541 avgPassTimes-3541 avgMsgSize-1 timeToSendMsg-17741125

What do you want to do?
1) Connect to a peer
2) Send a file to the currently connected peer
3) Quit
Choice: 3
Quitting...
You are not connected to a router.
The socket from this node to the peer has been closed!
Quitting...
You are not connected to a peer.
An IOException occurred!
```

Figure 1. Screenshot from an N1 client terminal for a successful file transfer from an N1 client to an M1 client