Jordan Rowley
Computer Science
University of St. Andrews

Mini-Project 1 Report

Simple File Sharing Application in Java

My file sharing program takes advantage of java.net sockets. Sockets provide the communication mechanism between two nodes on a peer-to-peer network using the transmission control protocol (TCP). Using this, any file of size less than 10MB (approx.) can be sent from a host machine to a client. The application doesn't actually send a copy of the target file, instead it transmits the data from a file into a new instance of the file type. Sharing can be achieved between any two participating users as long as the IP address of the sender is known by the receiver; this is made easier as the host's IP address is printed to console. The user receiving the file can choose the directory to which it is saved and also must choose a filename for it to be saved under. Likewise, the sender must specify the path of the file being sent. Unfortunately, I could not figure out a way to store files in a medium; a data hold which could be accessed by both participating users which would hold available files. For this reason, my application cannot provide a list of available files to a user. In my program, the sender opens a server on a given port and awaits a response from a client. Once the client (file recipient) connects to the host server, the sender specifics the file to be sent and the recipient specifies where the file should be saved. If the sender tried to send a file that doesn't exist, an exception will be thrown and the method will return 'false' indicating that the transfer wasn't successful. Similarly, if the recipient declares a non-existent directory for the file to be saved to, the same exception will be thrown and 'false' will be returned.

The protocol used is TCP which relies upon an established connection. The server opens and listens to a predetermined port (0202). Then, using the accept() method the server will wait for a client to push a request to the server host IP address on the same port. The host doesn't have to worry about the port and is told the IP address they need to give the recipient. When a connection is made, the host is prompted for a file and the sender is then prompted for a directory in which to save the file. Once the transfer has been made, the server closes. A

sort of while loop could perhaps be implemented into the main method to re-run the program in the case that users want to share multiple files. This would require closing the server and reopening it at some point which I would integrate into the send() and receive() methods.

My program only consists of one class and so the UML diagram is simple and looks like this:

There are three methods: main, send, and receive. The main method will ask the user if they're sending or receiving a file and then run either send() or receive() in accordance to the user's response. Both send() and receive() return a boolean. If the file transaction is successful, 'true' is returned, otherwise 'false' is returned.

Since my program stores the data from a file as a File type instead of a String, any file type can be sent. Another useful feature is that the host gets a readout of their IP address using the InetAddress.getLocalHost() method as part of the java.net library. Although I did not implement a loop so that a user can share multiple files during a single run of the application, a do-while loop is used in the main method to catch undefined user inputs when asked if they're sending or receiving a file. As well as the file, my program sends an integer value from sender to recipient telling the recipient how large the file is and hence what memory to allocate.

I tested my application by sending both a plain text file and a jpg image on a localhost; once using 'localhost' as the IP address and once using the IP address printed to the sender terminal window. Screenshots: