CSE 283 Spring 2015 - Data Communications and Networks
Project 3: (100 points) Due on Sunday, April 19 by 11:59 pm

Learning Outcomes

- Implement single threaded Server applications which communicate with clients sequentially using the TCP and UDP transport layer protocols.
- Implement multithreaded server application which communicate with clients in parallel using the TCP transport layer protocols.
- Implement a web server that can compress and uncompress files.

Instructions

Write your code in two files WebServer.java and CompressionServer.java. Upload your Java code to Niihka. You need to ensure that your code runs in the virtual machine that we have been using in class. The teaching assistant will use the same virtual machine environment to test your code. **If your code does not compile, you will get an automatic zero**. Therefore, you need to ensure that your code compiles with the following commands:

```
javac Webserver.java
java WebServer 6789

javac CompressionServer.java
java CompressionServer 5000
```

Description

In this project, you will modify your WebServer.java and CompressionServer.java code to use a UDP socket to communicate instead of a TCP socket. Moreover, the CompressionServer will now be responsible for creating the two files (uncompressed and compressed).

Part A: Compression Server using UDP

In this part of the project, you will write a UDP server that takes input from the client and creates an uncompressed and compressed files in the file system. You will need to look at various code samples and put them together to create a UDP compression server called CompressionServer.java. The commands below assume that you will be using your CSE 283 virtual machine.

1. To learn how to compress files in java, take a look at the code sample at this link (http://www.users.miamioh.edu/stjustpt/cse283/examples/Zip.java), download it, compile it and test it

   ```
   wget http://www.users.miamioh.edu/stjustpt/cse283/examples/Zip.java
   javac Zip.java
   java Zip inputfilename outputfilename.zip
   ```

2. Download the UDPEchoServer code (http://www.users.miamioh.edu/stjustpt/cse283/examples/UDPEchoServer.java) and build it and test it. For this, you will need two Putty windows, Remember that you can

use Control-C to stop your server.

in Putty window 1
```
wget http://www.users.miamioh.edu/stjustpt/cse283/examples/UDPEchoServer.java
javac UDPEchoServer.java
java UDPEchoServer 5000
```

in Putty window 2
```
nc -u 127.0.0.1 5000
```

3. Now whatever you type will show up in both Putty windows. Take a look at both your Zip.java and UDPEchoServer.java files to understand how the code works. Merge the two files into a single class called CompressionServer.java which will read bytes from the DatagramSocket instead of the FileInputStream. Your code should then create two files: 1) an uncompressed file and 2) a compressed file.

4. In order for the CompressionServer to create the two files (uncompressed and compressed), you have to send the filename first, over the UDP socket, then send the actual bytes that need to be stored, finally, you need to send a special "magic" string when you are done sending the data bytes to indicate the end of the content. For example, your magic string could be "--------MagicStringCSE283Miami", once you receive that String, then you know you have reached the end of the content. If you send a filename called "test.txt" then your CompressionServer will create two files, one called "test.txt" and another called "test.txt.zip".

5. Once you have properly implemented your CompressionServer, you can test it with the following commands in the virtual machine

   in Putty window 1
   ```
   javac CompressionServer.java
   java CompressionServer 5000
   ```

   in Putty window 2, assuming you have created a simple text file called "text.txt" in the directly
   ```
   nc -u 127.0.0.1 5000
   test.txt
   Hello world, this is a simple test
   --------MagicStringCSE283Miami
   ```

6. If you implement this correctly, your CompressionServer will create two files, the first one will be called "test.txt" and it will contain the text "Hello world, this is a simple test". The zipped file will be called "test.txt.zip" and when you extract it, it will contain a file called "test.txt" (not proj2.bin).

Part B: Connecting the Servers
The final part of this project requires that you connect the two servers together. To accomplish this, you need to take a look at the link below

http://www.users.miamioh.edu/stjustpt/cse283/examples/UDPEchoClient.java

This code shows you how to create a UDP client that connects to a UDP server. Since both servers will be running on the same virtual machine, you can use the 127.0.0.1 as the server IP address for connecting the WebServer to the CompressionServer. You have to extend your WebServer.java to connect the CompressionServer.java via a UDP socket (DatagramSocket). You will then use the DatagramSocket to send bytes from the WebServer to the CompressionServer. Your WebServer will no longer be creating the two files. As a result, your CompressionServer will create the two files. If you have implemented the whole project successfully, you should be able to perform the following steps

1. Use VirtualBox to run your virtual machine and use WinSCP to upload your WebServer.java, CompressionServer.java, and index.html files to your virtual machine. Then create two Putty windows that connects to your virtual machine

2. Compile and run both servers. Remember to remove the line that says "package …" in your java files, it will be at the top of your code.

   in Putty window 2
   javac CompressionServer.java
   java CompressionServer 5000

   in Putty window 1
   javac WebServer.java
   java WebServer 6789

3. Now open your web browser and navigate to http://192.168.56.102:6789/index.html, the web page asking you to upload a file will show up. Find a file (pdf, txt, or jpg) from your computer to upload. Fill in the name that you want it to be called (e.g. "proj2-test.txt"), after you press upload, your browser should display a message saying "File uploaded successfully". Once you see that message, go to http://192.168.56.102:6789/proj2-test.txt to verify that your file uploaded properly. Then go to http://192.168.56.102:6789/proj2-test.txt.zip and your browser will download the zip file. Extract the zip file and verify that it contains proj2-test.txt and that the contents match the original file. Test a few different file types (txt, pdf, jpg, and png). If everything works (or if you are out of time), upload your two java files to Niihka. Good luck.