TCP File Transfer

Due on or before 11.00am October 12th, 2023 (Thursday). Points: 20

Write a Java program for transferring a file from the client to the server using TCP. Modify the TCP client and server code already posted in Canvas as required for this program.

The following are **additional** requirements:

1. The server runs on port 9999 on the localhost. When server starts it should display: "Waiting for client on port 9999..." (without quotes).
2. The client should send a file that is provided as the *command line* argument to the server. Before sending the file, the client should compute the SHA256 hash code for the file. Also, after sending the file, the client should start a timer.
3. When the server receives the file, the server should compute the SHA256 hash code for the file and should return that hash code to the client. It should print the file size in bits and the computed hash code. This is shown in the right column of the top row in Table 1 and Table 2.
4. When the client receives the hash code from the server, it should stop the timer. The client should then check if the hash code from the server is same as its own computed hash code – if so, it should print "Successfully sent!" else it should print "Error!".
5. The client should then calculate the throughput for the transmission based on the file size and the time taken for the receipt of the hash code from server and print this value in Mbps (*rounded* to two decimals). The client should print the file name, its hash, file size in bits, time taken for server response in milliseconds (*rounded* to two decimals), and the throughput. Output from your client should be similar to the right-side column of bottom row of Table 1 (in case of correct transmission) or Table 2 (in case of erroneous transmission) below – your data will be different for the parameters, of course.

Table 1. Outputs Expected from Server and Client Programs For Correct Transmission

Server	Waiting for client on port 9999... Received file size in bits = 3524248 Received file SHA256 hash: 0Wn0jBnbViFc5RT9XPhlXSKOdxXmBW0SAtHorpkruyI=
Client	Connecting to localhost on port 9999 Just connected to localhost/127.0.0.1:9999 Successfully sent! File name: myImage.png SHA256 hash: 0Wn0jBnbViFc5RT9XPhlXSKOdxXmBW0SAtHorpkruyI= File size in bits = 3524248 Time taken = 2.16 ms Throughput = 1627985.33 Mbps

Table 2. Outputs Expected from Server and Client Programs In Case of Error

Server	Waiting for client on port 9999... Received file size in bits = 35677840 Received file SHA256 hash: p3MdNpSNiBXeb0dsD/TwfP87RErdFsY8jGkAcEvvSRE=
Client	Connecting to localhost on port 9999 Just connected to localhost/127.0.0.1:9999 Error! File name: test.pdf SHA256 hash: 0Wn0jBnbViFc5RT9XPhlXSKOdxXmBW0SAtHorpkruyI= File size in bits = 35677840

	Time taken = 26.56 ms Throughput = 1343374.36 Mbps

6. Create *.java* files – one for your client and one for your server. The client java file should be named *[first name initial][last name]TCPClient.java* and your server java file should be named *[first name initial][last name]TCPServer.java.* For example, if your first name is *John* and the last name is *Smith,* your files should be named: *JSmithTCPClient.java* and *JSmithTCPServer.java.*
7. Make sure your Java class names inside your files match your file names.
8. We will be grading by first compiling the two files using *javac*. Then we will open two terminal windows: on one terminal window we will start the server using the command: *java JSmithTCPServer* – this should start up the server listening on port 9999. On the other terminal window, we will run the client as: *java JSmithTCPClient xyz.abc* where *xyz.abc* is the file being uploaded to the server (in Table 1 above it is *myImage.png* and in Table 2 it is *test.pdf*).
9. You can assume that our test files will not exceed 5 MB in size.
10. If your program does not compile or does not run or gives an error, your grade will be zero.
11. Do **not** upload your java file as a txt file. Do **not** upload zip or rar files. Do **not** create a java package.
12. Upload your java files (one for the client and one for the server) to Canvas before the due date/time.