Comparison of TCP and UDP Client-Server APIs

Goal of the project

The goal of this project is to compare Java API for client-server development. Specifically, you will compare Java Socket API and the Java Datagram API. These API's will be compared from a programmer's standpoint as well as an architectural standpoint. Thus, you will need to compare the performance obtained in each case (quantitative comparison), as well as make a qualitative comparison. While discussing the quantitative results obtained, back your conclusions with data to support your conclusions. Try to answer why you feel technique A is faster than technique B etc.

In addition, you will compare these APIs, based on your observations, in relation to the ease of understanding the technology; time required for development, difficulties encountered, stability of code etc. This list is not meant to be exhaustive. Be sure to include any pertinent conclusions based on your experiences.

Description of the Client-server interaction

You will implement a reliable FTP client-server system.

Provide a simple menu to the user, e.g.

1. GET
2. PUT
3. CD
4. QUIT

Your code must be robust and handle any incorrect user input. Since there can be multiple clients querying the server at the same time with each client being serviced by a different thread on the server, the server must ensure concurrency control on the data file while it is being updated. Thus, only one thread must gain access to the data file during writes (hint: use *synchronized* keyword for the write method in your code).

Be sure to terminate each thread cleanly after each client request has been serviced.

Project 1

Implement the project as described using Java Sockets (TCP). This will be a connection-oriented client-server system with reliability built-in since TCP provides guaranteed delivery.

Points: 50

Project 2

Implement the project as described using Java Datagrams (UDP). This will be a connectionless client-server system since UDP is connectionless. However, you will need to provide reliability in your client-side application code since UDP does not guarantee

delivery. You may use the CRC32 checksum class available in the java.util.zip package in the JDK for this purpose. Refer to:
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/zip/CRC32.html for a description of this API. Sample code is at:
https://codingexplained.com/coding/java/generate-crc32-checksum-in-java

Points: 50

In each project, be sure to measure the mean response time for the server to service the client request. To graph this, have your client make requests in the order of the following data/file sizes: 1 MB, 25 MB, 50 MB, 100 MB. First do this for the PUT command and then do this for the GET command. Determine the response time in each case, then plot the response time vs. offered load (file size) graph.

Next, plot the throughout versus offered load graph using the data from the GET graph/command. Throughput in this case is bytes delivered per second (bytes/second). Plot the bytes/second on the y-axis and offered load on the x-axis. So, at the end of each project, you will have 3 graphs: one graph for the response time for the GET command, one graph for the response time for the PUT command, and the throughput graph for the GET command.

Your code must be well documented with adequate comments, suitable use of variable names, and follow good programming practices.

You will provide a demo at the end of each project to the instructor.
Have your server print out diagnostic messages about what it is doing (e.g., "accepting a new connection", etc.) Your code will be expected to deal with invalid user commands. Upload your source code and graphs to Canvas by the due date for each project. You should have the graphs on your laptop to show the instructor at the time of the demo.

Paper
Points: 50
The paper must be well written and must have the sections listed below.
It is good idea to keep writing your paper as each project ends, rather than waiting till the end of the semester. This way your experience with the technology will still be fresh in your mind.

Abstract
Describe what your project is about; what are its goals (i.e., comparison of various Java based client-server APIs for TCP and UDP); what experiments were conducted. The abstract must be concise. Try not to exceed 150 words.

Introduction
First describe your project in greater depth, i.e., explain your goals, and include an explanation of the client-server set-up.

Client-Server Computing APIs

You will describe each of the TCP and UDP sockets API in this section. Be sure to provide references when you refer to texts, papers, and websites etc. to describe these paradigms.

Results and Comparisons

First specify the test bed for your experiments (your laptop configuration). Then specify the studies carried out and the metrics you measured. Then provide the results by referring the reader to your graph(s). Explain your results. Then create a table where you compare the two APIs qualitatively with reference to the issues pointed out above.

Conclusions

Provide your conclusions, insights that you have gained in this section. E.g., which technology shows better performance and why, which technology is easier to learn, which technology has a richer set of services to offer, which technology is better used in which circumstance.

References

Use the following IEEE standard when listing references:

[1] Ahuja, Sanjay, P., and Collier, Nicole, *An Assessment of WiMax Security*, Communications and Network Journal, 2010, 2, pp. 134-137, June 2010.

All references listed in the reference section must be cited within the body of the paper using square brackets. E.g., The author refers to the TCP socket ….. [1].